# Ring-LWE Ciphertext Compression and Error Correction

## Tools for Lightweight Post-Quantum Cryptography

Markku-Juhani O. Saarinen
DarkMatter LLC
P.O. Box 27655, Abu Dhabi, UAE
markku.saarinen@darkmatter.ae

## ABSTRACT

Some lattice-based public key cryptosystems allow one to transform ciphertext from one lattice or ring representation to another efficiently and without knowledge of public and private keys. In this work we explore this lattice transformation property from cryptographic engineering viewpoint.

We apply ciphertext transformation to compress Ring-LWE ciphertexts and to enable efficient decryption on an ultra-lightweight implementation targets such as Internet of Things, Smart Cards, and RFID applications. Significantly, this can be done without modifying the original encryption procedure or its security parameters. Such flexibility is unique to lattice-based cryptography and may find additional, unique real-life applications.

Ciphertext compression can significantly increase the probability of decryption errors. We show that the frequency of such errors can be analyzed, measured and used to derive precise failure bounds for $n$-bit error correction. We introduce XECC, a fast multi-error correcting code that allows constant time implementation in software.

We use these tools to construct and explore `trunc8`, a concrete Ring-LWE encryption and authentication system. We analyze its implementation, security, and performance. We show that our lattice compression technique reduces ciphertext size by more than 40% at equivalent security level, while also enabling public key cryptography on previously unreachable ultra-lightweight platforms.

The experimental public key encryption and authentication system has been implemented on an 8-bit AVR target, where it easily outperforms elliptic curve and RSA-based proposals at similar security level. Similar results have been obtained with a Cortex M0 implementation. The new decryption code requires only a fraction of the software footprint of previous Ring-LWE implementations with the same encryption parameters, and is well suited for hardware implementation.



ISBN 000-0-00-000000-0.
DOI: 0.0/0

## Keywords

Post-Quantum Cryptography, Lattice Cryptography, Ring-LWE Encryption, Lightweight Cryptography

## 1. INTRODUCTION

Polynomial-time Quantum algorithms for attacking cryptographic standards based on RSA [20] and elliptic curve discrete logarithms [16] have been known for many years [34, 38]. However, the continuing progress of quantum computing has only recently prompted the National Security Agency (NSA) to express an intention to transition to new, post-quantum algorithms [10, 30]. These algorithms run on classical computers but are expected to be resistant to cryptanalysis by quantum methods. U.K. CESG has also expressed their preference for post-quantum algorithms over "quantum technologies" such as Quantum Key Distribution to counter the threat of quantum computing [8].

We observe that the new post-quantum threat model applies equally to smart devices known as Internet-of-Things, RFID, smart cards, and to other common lightweight cryptography applications. New lightweight proposals should be designed to be resistant to quantum adversaries in addition to side channels and other implementation attacks. Current public key standards do not generally meet these criteria.

To meet the future asymmetric requirement, National Institute of Standards and Technology (NIST) has initiated an effort to standardize post-quantum cryptography (PQC) [9]. The algorithms should be implementable on a wide range of platforms, and should provide at least one of: signature, encryption, or key exchange / encapsulation (KEM) [29, 27].

In this work we show how ciphertext compression and error correcting codes can be used with the basic Lindert-Peikert Ring-LWE algorithm [22] to build a robust, and quantum resistant public key encryption, KEM, and authentication mechanism for Internet of Things and other lightweight targets. Our construct uses many of the same building blocks as previous well-studied encryption [12, 24, 32, 33, 36], signature [13, 37], and key exchange algorithms [2], and can be used in conjunction with them.

**Structure of this paper and our contributions.** We describe the basic "Lindner-Peikert" Ring-LWE public key encryption scheme in Section 2. A ciphertext compression technique is introduced in Section 3. Its lightweight implementation, application to authentication, and novel auxiliary algorithms such as a constant-time error correction are described in Section 4. This is followed by conclusions and further discussion in Section 5.

## 2. RING-LWE PUBLIC KEY ENCRYPTION

We use a variant of the well-established "Lindner-Peikert" Ring-LWE encryption algorithm first given in [22] and in the extended version of [26]. This algorithm has been implemented on numerous lightweight targets in both hardware and software in recent years [3, 12, 17, 31, 32, 36]. Our implemented scheme – comprising of a certain set of parameters and algorithms – is referred to as trunc8.

### 2.1 Notation and Conventions

Arithmetic is performed in the "anti-cyclic" ring formed of polynomials modulo $f(x) = x^n + 1$ with coefficients defined in the field $\mathbb{Z}_q$, where $q$ is a (small) prime. Coefficient of degree $i$ of ring element (polynomial) $v$ is denoted $v[i]$; we have $v = \sum_{i=0}^{n-1} v[i]x^i$. The ring element polynomials are algorithmically handled as zero-indexed vectors.

Let $U^n$ be a source of uniform random polynomials in $\mathbb{Z}_q^n$ and $B^n \in \{0,1\}^n$ a source of random polynomials with binary coefficients.

Let $D_\sigma$ be a source of integers with discrete Gaussian distribution defined by deviation parameter $\sigma$. $D_\sigma^n$ is a source of polynomials with coefficients randomly sampled from that distribution. The probability mass of $D_\sigma$ at $x \in \mathbb{Z}$ is

$$f_\sigma(x) \propto e^{-\frac{x^2}{2\sigma^2}} \text{ with } \sum f_\sigma(x) = 1. \quad (1)$$

We typically have a very close approximation

$$f_\sigma(x) \approx \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}. \quad (2)$$

We use $e_0, e_1, e_2, e_3 \leftarrow D_\sigma^n$ to denote random Gaussian noise parameters that are newly sampled from the distribution every time the corresponding algorithms are invoked.

### 2.2 Key Generation

Public key is $(a, p)$ and the private key is the $n$-bit binary string $s$.[1] The $a$ parameter may be "global" and shared between a large number of public keys.

| | |
|---|---|
| $s = B^n$ | *Binary private key.* |
| $a = U^n$ | *Public parameter (shared).* |
| $p = e_0 - a * s$ | *Public key.* |

### 2.3 Encryption

When encoding an $n$-bit message $z \in \{0,1\}^n$, we add either 0 or $\frac{q-1}{2}$ to each $v_i$ coefficient, depending corresponding on bit $z_i$. Ciphertext message consists of the pair $(u, v)$.

| | |
|---|---|
| $m = \lfloor q/2 \rfloor * z$ | *Message encoding.* |
| $u = a * e_1 + e_2$ | *Ciphertext part 1.* |
| $v = p * e_1 + e_3 + m$ | *Ciphertext part 2.* |

### 2.4 Decryption

Decrypting $(u, v)$ with secret key $s$ can be done via

$$m' = u * s + v. \quad (3)$$

Decoding the message $z$ can be achieved by normalizing $m$ into zero-centered range $-\frac{q+1}{2} \leq m'_i \leq \frac{q-1}{2}$ and outputting $z_i = 1$ if $\text{abs}(m'_i) > \frac{q}{4}$ and $z_i = 0$ otherwise.

[1]This is equivalent to the CHES 2014 variant of Roy et al. [36] (Section 6). Some variants sample the private key $s$ from $D_\sigma^n$. See [7] for analysis of the binary case.

**Algorithm 1** Simple ring multiplication for decryption.

**Input:** Ciphertext $(u, v)$ and private key $s$.

```
m ← v
for i = 0, 1, ... n − 1 do
  if s_i = 1 then
    for j = 0, 1, ..., n − i − 1 do
      m[i + j] ← m[i + j] + u[j]
    end for
    for j = n − i, ..., n − 1 do
      m[i + j − n] ← m[i + j − n] − u[j]
    end for
  end if
end for
```

**Output:** Undecoded plaintext vector $m$.

### 2.5 Parameter Selection

Göttert et al. has suggested $(n, q, \sigma) = (512, 12289, 4.8591)$ as a "high-security" Ring-LWE encryption parameter set [17]. We adopt these parameters for our work. This parameter set has been used in many subsequent Ring-LWE Encryption studies [12, 24, 32, 33, 36] and therefore works as a suitable benchmark when comparing implementations. We refer to these earlier works for detailed analysis of the given parameters. We further note recent work by Buchmann et al. on the "binary secret" case of Ring-LWE [5, 7].

Following the methodology adopted by Alkim et al. in [2], we can compare the security of this parameter set to other lattice-based encryption and key exchange proposals[2]. Table 1 offers estimates for the best known classical and quantum attacks against trunc8 and some other contemporary lattice-based cryptosystems. For a recent survey of LWE hardness, see [1]. We note that our use of a binary secret may affect the estimates in this table.

## 3. USING CIPHERTEXT COMPRESSION

Decryption operation (Equation 3) involves multiplication of $u$ by the secret binary vector $s$ in the anti-cyclic ring. This is often done using Number Theoretic Transforms (NTT), but can also be implemented with Algorithm 1. This simpler method has a higher (quadratic) complexity than the NTT method, but it is still very fast as it requires no field multiplications, just additions and subtractions.

THEOREM 1. *Any ciphertext produced with common Ring-LWE encryption parameters (Section 2.5) can be significantly compressed in a way that still allows correct decryption with high probability. The compression has no effect on confidentiality provided by the scheme.*

PROOF. The compression argument is shown by subsequent Theorems; it works. The security argument follows from the fact the transformation is "public" (uses no secret information) and only removes redundancy. Therefore it cannot negatively affect the confidentiality of the message against attacks. □

The arithmetic for decryption is performed in $\mathbb{Z}_q$ by default, with $q = 12289$ chosen in Section 2.5. We show that

[2]The script used in [2] for finding optimal attack parameters: https://github.com/tpoeppelmann/newhope/blob/master/scripts/PQsecurity.py

**Table 1: Comparing the hardness of our scheme against some other recent lattice-based encryption and key exchange proposals using the methodology from [2]. trunc8 (and other proposals using the "high-security" parameter set from [17]) exceeds 128-bit security against all known quantum attacks; attack improvements beyond $2^{102}$ are not plausibly expected according to this methdology.**

| Attack type | Parameters $(m, b)$ | Known Classical | Known Quantum | Plausible Quantum |
|---|---|---|---|---|
| trunc8 [this work] | | $q = 12289$, $n = 512$, $\sigma = 4.859$ | | |
| Primal | (660, 496) | 144 | 131 | 102 |
| Dual | (674, 494) | 144 | **131** | 102 |
| Medium-Security LWE [17] | | $q = 7681$, $n = 256$, $\sigma = 4.512$ | | |
| Primal | (345, 222) | 64 | 58 | 46 |
| Dual | (360, 222) | 64 | **58** | 46 |
| NTRU-743 [18] | | $q = 2^{12}$, $n = 743$, $\sigma = 0.8164$ | | |
| Primal | (613, 603) | 176 | 159 | 125 |
| Dual | (635, 600) | 175 | **159** | 124 |
| BCNS [4] | | $q = 2^{32}$, $n = 1024$, $\sigma = 3.192$ | | |
| Primal | (1062, 296) | 86 | 78 | 61 |
| Dual | (1055, 296) | 86 | **78** | 61 |
| NewHope [2] | | $q = 12289$, $n = 1024$, $\sigma = 2.828$ | | |
| Primal | (1100, 967) | 282 | 256 | 200 |
| Dual | (1100, 962) | 281 | **255** | 199 |
| JarJar [2] | | $q = 12289$, $n = 512$, $\sigma = 3.464$ | | |
| Primal | (623, 449) | 131 | 119 | 93 |
| Dual | (602, 448) | 130 | **118** | 92 |

decryption operations can be approximated by arithmetic in group $\mathbb{Z}_{p=256}$ which corresponds to fast byte operations.

Modulus reduction techniques have been considered in a theoretical setting in works such as [5]. However, our angle is simply to analyze the rounding artifacts that occur in our concrete trunc8 proposal to guarantee that approximate decryption works with sufficient probability.

We choose the group of integers mod $p = 256$ as a natural "compressed ideal lattice" base. For $q = 12289$ we define a rounding truncation of $0 \le x < q$:

$$\mathsf{T}(x) = \left\lfloor \frac{(x \bmod q) + 23}{48} \right\rfloor \mod 256. \tag{4}$$

Theorem 2. *Group $\mathbb{Z}_q$ addition $(x + y)$ for $q = 12289$, when approximated* mod $p = 256$ *by byte addition $\boxplus$ via mapping $\mathsf{T}(x)$ is characterized by*

$$\mathsf{T}(x) \boxplus \mathsf{T}(y) \equiv \begin{cases} \mathsf{T}(x+y) \boxplus 1 & \text{with } P = 18877440/q^2, \\ \mathsf{T}(x+y) & \text{with } P = 113264641/q^2, \\ \mathsf{T}(x+y) \boxminus 1 & \text{with } P = 18877440/q^2 \end{cases} \tag{5}$$

Proof. Result was obtained via exhaustive computation of all $q^2$ cases. The three cases sum up to 1; the approximation is never off by more than one. □

The probabilities are very close to 1/8, 3/4, and 1/8, respectively – accurate to 8 decimal places as these are the closest fractions with denominator $q^2$.

Theorem 3. *Let $m'$ be the output of Algorithm 1 when run in original group $\mathbb{Z}_q$ and $m$ when input is truncated to $(\mathsf{T}(u), \mathsf{T}(v))$ and byte arithmetic in* mod $p = 256$ *is used throughout. For random keys and inputs, the distance of compressed decoding in $\mathbb{Z}_p$ to decoding in $\mathbb{Z}_q$ for individual values of $m[i] - \mathsf{T}(m'[i])$ is closely approximated by Discrete Gaussian distribution $D_{\sigma=8}$.*

Proof. Let $a = \frac{18877440}{q^2}$ (from Equation 5) be the probability that an off-by-one error (in each direction) is introduced in each addition. Combinatorial analysis of a random $n = 512$ - bit secret key, of weight $w$, with $b$ nonzero steps, of which $c$ are positive gives the total probability of error of exactly $x \in \mathbb{Z}$ steps as

$$f(x) = \sum_{\substack{x = 2c - b, \\ 0 \le c \le b \le w \le n}} \frac{a^b (1-2a)^{w-b}}{2^n} \binom{n}{w} \binom{w}{b} \binom{b}{c}. \tag{6}$$

The statistical distance (total variation distance) between the exact combinatorial $f(x)$ of Equation 6 and the discrete Gaussian distribution of Equation 1 with deviation $\sigma = 8$ is less than 0.0006. Figure 1 illustrates Equation 6. □

Combinatorial analysis can be made for any approximation. However, one generally prefers to have an easy-to-analyze characterization such as the one in Equation 5.

## 3.1 Rough Statistical Analysis of Decryption

Expanding Equation 3 we obtain:

$$\begin{aligned} m' &= (a * e_1 + e_2) * s + (e_0 - a * s) * e_1 + e_3 + m \\ &= e_2 * s + e_0 * e_1 + e_3 + m. \end{aligned} \tag{7}$$

We see that the $a * s * e_1$ term cancels out. Decryption can work correctly only if $|m'_i - m_i| < q/4$ in the original mod $q$ operation. We first analyze this case.

Each ring multiplication $x * y = z$ evaluates $n$ inner products $z_i = \sum_{j=0}^{n-1} x_j y_{i-j}$ (we interpret indexes in anti-cyclic

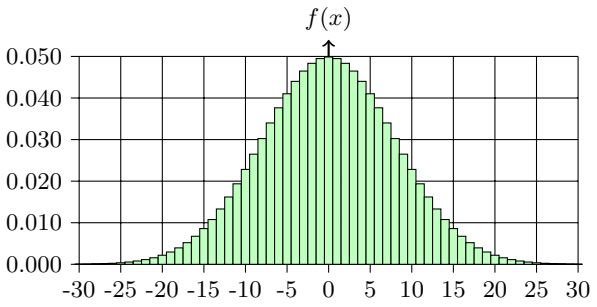


**Figure 1: Probability distribution of the approximation error given by Equation 6. Approximation error does not have to be 0 for successful decryption; generally $|x| < q/4$ still works with our scheme.**

wrap-around fashion: $v_i = -v_{i+n} = -v_{i-n}$.) Recall that the variance $\sigma^2$ of independent zero-centered variables is both additive and multiplicative: $\mathrm{Var}(X+Y) = \mathrm{Var}(X)+\mathrm{Var}(Y)$ and $\mathrm{Var}(XY) = \mathrm{Var}(X)\,\mathrm{Var}(Y)$ when $E[X] = E[Y] = 0$. Furthermore, the sum of $n$ equal-variance independent variables approaches a Gaussian distribution when $n$ grows. We may therefore estimate the coefficients of the polynomial ring product $z = x * y$ by Gaussian $D^n_{\sigma_z}$ with deviation

$$\sigma_z \approx \sqrt{n\sigma_x^2\sigma_y^2}. \tag{8}$$

In the case where both $x$ and $y$ in ring multiplication $z = x * y$ come from discrete Gaussian $D^n_\sigma$ with example parameters $\sigma = 4.8591$ and $n = 512$, the corresponding approximate Gaussian has $\sigma_z = \sqrt{512}\sigma^2 \approx 534.3$. The total variation distance of this $D_{\sigma_z}$ approximation to the true product distribution of $z_i$ is approximately 0.0007.

We further write the binary distribution $s \in B^n$ as a zero-centered distribution with variance $\frac{1}{2}$. This allows us to bound the error vector $m' - m$ with Gaussian $D^n_{\sigma_e}$ where

$$\sigma_e = \sqrt{n\left(\frac{1}{2}\sigma^2 + \sigma^4\right) + \sigma^2} \approx 539.9. \tag{9}$$

To estimate the error in the truncated mod 256 case, we must scale the "intrinsic" error $\sigma_e$ of $\mathbb{Z}_q$ rings to our target group and add the approximation error of Theorem 3. The scaled intrinsic error component $\frac{\sigma_e}{48} \approx 11.25$ dominates over the approximation deviation 8.000, bringing the total deviation to 13.80. Scaling the $q/4$ bound we obtain the condition $-64 < m_i - m'_i < 64$ for the error. From cumulative Gaussian distribution we obtain failure probability $P = 0.000004194$ for each bit and $1-(1-P)^{512} = 0.002145$ for at least one-bit failure in the $n$-bit vector. However, experimentally the failure probability was found to be smaller:

Observation 1. *Approximate decryption (Equation 3) in $\mathbb{Z}_{p=256}$ produces, on average, less than $10^{-4}$ single-bit errors per decrypted message in experiments.*

Hundreds of millions of key pair generations, encryptions, and decryptions were performed to verify Observation 1. The actual failure probability is roughly $0.94 * 10^{-4}$ per message, or $2^{-22.4}$ per bit. These measurements allow us to estimate failure bounds when error correction is used, as explained in Section 4.2.

## 3.2 Further Ciphertext Compression

We observe that even when truncated, not all bits of ciphertext are required for reasonably successful decryption. For example, we may choose to transmit only four bits of $v[i]$; this corresponds to ciphertext vector of $512 + 256 = 768$ bytes, consisting of $(u[i], \lfloor \frac{v[i]+8}{16} \rfloor)$ pairs. We found an average of 0.00040 bit errors per decrypted message. However, we do not use this method in our current proposal.

# 4. APPLICATION TO LIGHTWEIGHT AUTHENTICATION TOKENS

While there has been steady progress in proofs and theoretical constructs in this field, long-term experience has shown us that when cryptosystems fail, they often fail on implementation level. Therefore, all aspects of practical, efficient, and secure quantum-safe implementations must receive the same – or higher – level of attention that is the norm for more established algorithms. This includes protocols and auxiliary algorithms such as error correcting codes.

We designed trunc8 as a lightweight and quantum-resistant public key authentication scheme. The system may be used to authenticate users in physical access control systems, or as an upper layer (TLS [35] "post-handshake") authentication protocol to Internet services; here the server (once authenticated) interrogates the user's terminal-connected token or smart card to grant access.

## 4.1 One-Way Authentication Protocol

One of the main advantages of a public key authentication scheme over any symmetric scheme is that (public) key retrieval and validation can be performed using certificate-based Public Key Infrastructure (PKI) methods over untrusted channels, as is commonly done in various Internet security protocols such as TLS. Keys can be revoked with CLRs (Certificate Revocation Lists) [11, 40]. Naturally the PKI itself must be based on post-quantum signatures if quantum resistance is required.

For simple authentication tokens (such as smart cards and RFID key fobs), only the private key operation (Section 2.4) needs to be implemented. trunc8 decryption does not require non-uniform random sampling; such heavier lifting can be performed by the interrogator ("server"). The token must just be able to use its secret key to decrypt a challenge message and to generate an appropriate response. The ultra-lightweight public key authentication problem has been previously addressed with elliptic curves in works such as [39].

In our basic one-way authentication protocol Alice (the token) authenticates herself to Bob (the interrogator) by proving possession of private key $K_a^{-1}$. Bob must be able to derive the public key $K_a$ using $A$. The protocol itself is a one-sided variant Needham–Schroeder [28] (Lowe's fix is not needed in this case [25]).

*Alice's identity and nonce.*
1. $A \to B \quad : A, N_a$

*$K_{ab}$ is a random symmetric key.*
2. $B \to A \quad : \{K_{ab}, \{N_a\}_{K_{ab}}\}_{K_a}$

*Symmetric response to challenge.*
3. $A \to B \quad : \{-1\}_{K_{ab}}, \{-2\}_{K_{ab}}$

Here the symmetric random "session" key chosen by Bob, $K_{ab}$, plays the role of Bob's nonce ("$N_b$"). Message 1 may contain additional information such as a certificate chain for $A$. We encrypt the 128-bit nonce $N_a$ in step 2 to serve as an "integrity check" for the message. The ciphertext length for trunc8 is 1024 bytes. For symmetric encryption AES-256 [15] is used; $K_{ab}$ is 256 bits, as is the step 3 response $\mathsf{AES}_{K_{ab}}(\texttt{0xF..FF}) \mid \mathsf{AES}_{K_{ab}}(\texttt{0xF..FE})$. These constants were chosen to not overlap with CTR or AES-GCM constants [14] in case $K_{ab}$ is used to protect further communication from Bob, such as his authentication.

## 4.2 A Constant-time Error Correcting Code

As indicated by Observation 1, the decryption procedure can be expected to produce an erroneous bit in one of 10000 messages. However, the 512-bit payload has sufficient redundancy to accommodate an error correcting code.

Since error correction operates on confidential payload data, it must be constant-time to deter timing attacks. Our design focus is to accommodate this requirement with a lightweight algorithm that is easily implemented in both hardware and software. The algorithm is named $\mathsf{XECC}^4_{32}$.

Listings 1 and 2 contain functions `xecc_compute()` and `xecc_fixerr()` that are used to compute the error correcting code and then to fix possible bit errors. They operate on arrays of 32-bit words `v[16]`, where `v[0..7]` contains the 256-bit payload and `v[8..11]` the code itself. The last part `v[12..15]` is for a cryptographic integrity check in our application – see Section 4.1.

We view the payload as a binary polynomial of deg(256). It is reduced modulo four different redundancy polynomials, $x^{32}-1$, $x^{31}-1$, $x^{29}-1$, and $x^{27}-1$. This can be done with some fast shifts and XOR operations, as can be observed from the listings. A three-of-four Boolean rule is used to find congruent positions with at least three parity errors and to fix them in each word.

It is easy to verify that the code can correct all single- and double-bit errors in the payload and the code itself. It can also correct all but 718972 out of $\frac{384*383*382}{3!} = 22238720$ possible three-bit errors, a failure rate of 0.07678 in this case.

**Listing 1: Compute and set XECC**

```
void xecc_compute(uint32_t v[16])
{
    int i;
    uint32_t r32, r31, r29, r27, t;

    r32 = r31 = r29 = r27 = 0;

    for (i = 0; i < 8; i++) {
        t = v[i];
        r32 ^= t;
        r31 ^= t ^ (t >> 31);
        r31 &= 0x7FFFFFFF;
        r31 = (r31 >> 1) | (r31 << 30);
        r29 ^= t ^ (t >> 29);
        r29 &= 0x1FFFFFFF;
        r29 = (r29 >> 3) | (r29 << 26);
        r27 ^= t ^ (t >> 27);
        r27 &= 0x07FFFFFF;
        r27 = (r27 >> 5) | (r27 << 22);
    }
    v[8]  ^= r32;      // XOR code with original.
    v[9]  ^= r31;
    v[10] ^= r29;
    v[11] ^= r27;
}
```

**Listing 2: Correct Errors**

```
void xecc_fixerr(uint32_t v[16])
{
    int i;
    uint32_t r32, r31, r29, r27;

    xecc_compute(v);  // recompute

    r32 = v[8];       // get codes
    r31 = v[9];       // (now "diffs")
    r29 = v[10];
    r27 = v[11];

    for (i = 7; i >= 0; i--) {
        r31 &= 0x7FFFFFFF;
        r31 = (r31 << 1) | (r31 >> 30);
        r29 &= 0x1FFFFFFF;
        r29 = (r29 << 3) | (r29 >> 26);
        r27 &= 0x07FFFFFF;
        r27 = (r27 << 5) | (r27 >> 22);

        // 3-of-4 rule for correcting
        v[i] ^= (r32 & r31 & (r29 | r27))
             | ((r32 | r31) & r29 & r27);
    }
}
```

## 4.3 Side-Channel Security

Like all cryptographic algorithms, lattice cryptosystems may be breached via side-channel attacks [6]. This was the main design criteria the for $\mathsf{XECC}^4_{32}$ code (Section 4.2). General blinding countermeasures for NTT-based LWE implementations have been proposed in [37], but these are not directly applicable to trunc8 due to its different multiplication strategy.

In order to achieve constant time decryption, the secret key $s$ (Section 2.2) is chosen so that it has exactly $\frac{n}{4} = 128$ ones in even bit positions and at odd bit positions. This choice drops secret key entropy to $\log_2 \binom{256}{128}^2 \approx 503.3$ bits, but analysis of our decryption algorithm shows guaranteed constant-time operation. An alternative would be to balance the implementations' "zero" cases but that would essentially make the algorithm always run at its worst-case speed.

Another countermeasure is to store the secret key internally as a randomly shuffled list of $2 \times 128$ index bytes indicating positions of ones (at both even and odd positions). By avoiding "scanning" the secret key bit-by-bit, and by operating in random order, this greatly helps to mask emissions.

We note that these countermeasures may be insufficient against power analysis attacks [21] and are intended to counter only network-based attacks against the authenticator.

## 4.4 Security and Failure Rate Estimates

Since the basic encryption scheme and parameters are the same as in [12, 17, 32, 36], the proposed system should have equivalent, or better, security than these schemes. We simply compress the ciphertext after encryption. See Theorem 1 for the main security argument.

We ignore up to four-bit errors in the 128-bit integrity check $\mathsf{AES}_{K_{ab}}(N_a)$, leading to $2^{-128}\sum_{i=1}^{4}\binom{128}{i} \approx 2^{-104.6}$ probability of failure caused by that. The $\mathsf{XECC}^4_{32}$ code (Section 4.2) is able to catch all single- and double bit errors in $K_{ab}$ and the code itself. From Observation 1 and related discussion we know that the single-bit failure rate is $P = 2^{-22.4}$, leading to triple-bit failed-correction failure rate in first 384 bits of roughly $0.07678 * (1-(1-p)^{384})^3 \approx 2^{-45.1}$.

We claim that this is tolerable bound for a user authentication mechanism, especially since failures are detected and

**Table 2: Comparison of trunc8 AVR private key operation. We are not aware of RSA implementations beyond 1024-bit modulus size for AVR. For elliptic curves we use the Curve25519 scalar multiplication from AVR NaCl [19] for comparison. Two Ring-LWE implementations with comparable security parameters were reported in [24]; one optimized for high speed (HS-512) and another for memory efficiency (ME-512). We assume that $\mathbb{Z}_q$ elements are transmitted as 16-bit words and that shared parameter $a$ is used for RLWE encryption; otherwise the public key would be two times longer. Plaintext payload is $n = 512$ bits in all RLWE cases.**

| Scheme / Implementation | Cipher-text | Public Key | Work RAM | Flash ROM | Processor Cycles |
|---|---|---|---|---|---|
| RSA-1024 [23] | 128 | 128 | N/A | N/A | 75,680,000 |
| Curve25519 HS [19] | 32+32 | 32 | 681 | N/A | 22,954,657 |
| RLWEenc [33] | 2,048 | 1,024 | 2,144 | 9,258 | 600,351 |
| RLWE $n = 512$ HS [24] | 2,048 | 1,024 | 3,121 | 7,506 | 700,099 |
| RLWE $n = 512$ ME [24] | 2,048 | 1,024 | 2,737 | 8,500 | 1,450,713 |
| trunc8 [this work] | **1,024** | **1,024** | **1,088** | **282** | **1,387,824** |

unauthorized access is not granted upon failure. Authentication may be simply repeated upon protocol failure as it is randomized.

## 4.5 Related Hardware Considerations

In CHES 2014 Roy et al. [36] published a design for a Ring-LWE cryptoprocessor that implements essentially the same Ring-LWE scheme as our proposal using a number of dedicated optimizations, but without compressed decryption. The implementation of [36] uses NTT for ring arithmetic, and therefore requires the use of field multipliers, which are expensive in hardware. Our simpler decryption scheme has the same security parameters but does not need modular multiplication. Furthermore, it has about half of their RAM requirement.

It is easy to see that hardware implementation of the $\mathsf{XECC}^4_{32}$ parity-based error-correcting code (Section 4.2) needs only few hundred gates. Based on these observations, we expect hardware implementation of trunc8 to be smaller and more energy-efficient than NTT-based decryption cores by about an order of magnitude. This is for future work.

## 4.6 Lightweight Implementation

We implemented the authenticator (and decryption) code primarily on Atmel ATmega 328P, which is a popular and inexpensive 8-bit microcontroller implementing the AVR instruction set. The chip has only 2 kB of SRAM and 32 kB of Flash.[3] The platform was chosen due to its popularity and availability of benchmark data.

Highly optimized AVR implementations of Ring-LWE encryption schemes have been published by Liu et al. in CHES 2015 [24] and by Pöppelmann et. al in LATINCRYPT 2015 [33]. Prior work (e.g. [3]) has clearly inferior performance when compared to these. The [24, 33] "high security" implementations have essentially the same security parameters as trunc8, but use NTT for ring arithmetic. The private key and ciphertext are stored in the NTT domain, and therefore the private key multiplication of Equation 3 requires only one NTT transform instead of three for these implementations. Our ciphertext and implementation footprint are much smaller, however.

Examining Table 2 we note that even their "memory efficient" implementations would not be able to run on our ATmega 328P target as not enough SRAM is available. Only decryption code was included in these figures; the very large difference in implementation size between trunc8 and [24, 33] is mainly due to their complexity and need for NTT-related tables. We further observe that our method does not require multiplications and is therefore efficiently implementable on architectures (such as 8051) that do not have a multiplier.

**Cortex M0.** The entire authentication mechanism was also ported to Cortex M0 in plain C. Here the private key operation required only 44.5 ms on LPC11U24 running at 48 MHz; about 2.13M cycles. This speed-optimized decode function required 1252 bytes of flash, while AES-256 required 824 bytes, and XECC 175 bytes. Most of the rest of the 21kB binary were used up by standard libraries. Entire authentication protocol of Section 4.1 was implemented and tested over a 115200 baud serial line; the entire "handshake" required only a fraction of a second.

# 5. CONCLUSIONS

We have described a method for compressing Ring-LWE ciphertexts and the related decryption algorithm. The new algorithm enables realization on hardware and microcontroller targets which previously have not allowed implementation of private key operations.

The lattice compression technique reduces ciphertext size to about half (one kilobyte) when compared to previous implementations. Since neither the encryption process or the plaintext payload size needs to be changed, the compressed scheme has the same, or better, confidentiality properties as numerous previous Ring-LWE public key implementations with the same security parameters.

We have also proposed a full padding method that incorporates a fast, constant-time error correcting code, which allows us to tolerate increased decryption errors.

We view smart cards and RFID tags as an ideal application area for the compression method. For this purpose, we have proposed a concrete authenticator algorithm and protocol, trunc8. It is intended for both physical access control and network user authentication. In addition to an error cor-

[3]Atmel ATmega 328P (http://www.atmel.com/devices/atmega328p.aspx) has under $2 unit price in 2016. We used the widely available "stock" Arduino Uno (R3) board (https://www.arduino.cc/en/Main/ArduinoBoardUno) for our experiments.

rection code, the proposed message format also has a cryptographic integrity check, which we assume to be requirements for any future real-life Ring-LWE padding schemes.

The new scheme has been implemented on an 8-bit AVR microcontroller target, and shows comparable decryption speed to previous highly optimized implementations that use the Number Theoretic Transform. However, our implementation is much simpler and requires only a fraction of their implementation footprint. We have also discussed how to make the private key operation itself run in constant time by slightly reducing key space, and how to mask other key-dependent emissions.

Our work demonstrates that the ciphertext transformation property of lattice-based cryptosystems may have significant applications in cryptographic engineering. When faced with an application that has severe performance or bandwidth constraints, one can examine whether ciphertext transformation can help to overcome those limitations.

# 6. REFERENCES


[1] M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. *Journal of Mathematical Cryptology*, 9(3):169–203, October 2015. URL: https://eprint.iacr.org/2015/046, doi:10.1515/jmc-2015-0016.

[2] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Newhope without reconciliation. Preprint., 2016. URL: https://cryptojedi.org/papers/newhopesimple-20161217.pdf.

[3] A. Boorghany, S. B. Sarmadi, and R. Jalili. On constrained implementation of lattice-based cryptographic primitives and schemes on smart cards. *ACM Transactions on Embedded Computing Systems (TECS)*, 14(3):42:1–42:25, May 2015. URL: https://eprint.iacr.org/2014/514, doi:10.1145/2700078.

[4] J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the TLS protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 553–570. IEEE Computer Society, 2015. Extended version available as IACR ePrint 2014/599. URL: https://eprint.iacr.org/2014/599, doi:10.1109/SP.2015.40.

[5] Z. Brakerski, A. Langlois, C. Peikert, O. Regev, and D. Stehl'e. Classical hardness of learning with errors. In *STOC '13*, pages 575–584. ACM, 2013. doi:10.1145/2488608.2488680.

[6] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, Gauss, and reload – a cache attack on the BLISS lattice-based signature scheme. In B. Gierlichs and A. Y. Poschmann, editors, *CHES 2016*, volume 9813 of *LNCS*, pages 323–345. Springer, 2016. URL: https://eprint.iacr.org/2016/300, doi:10.1007/978-3-662-53140-2_16.

[7] J. Buchmann, F. Göpfert, R. Player, and T. Wunderer. On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack. In D. Pointcheval, A. Nitaj, and T. Rachidi, editors, *AFRICACRYPT 2016*, volume 9646 of *LNCS*, pages 24–43. Springer, 2016. doi:10.1007/978-3-319-31517-1_2.

[8] CESG. Quantum key distribution: A CESG white paper, February 2016. URL: https://www.cesg.gov.uk/white-papers/quantum-key-distribution.

[9] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. Report on post-quantum cryptography. NISTIR 8105, April 2016. URL: http://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8105.pdf, doi:10.6028/NIST.IR.8105.

[10] CNSS. Use of public standards for the secure sharing of information among national security systems. Committee on National Security Systems: CNSS Advisory Memorandum, Information Assurance 02-15, July 2015.

[11] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and T. Polk. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. IETF RFC 5280, May 2008. URL: https://www.rfc-editor.org/rfc/rfc5280.txt, doi:10.17487/RFC5280.

[12] R. de Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede. Efficient software implementation of Ring-LWE encryption. In *DATE 2015*, pages 339–344. IEEE, 2015. doi:10.7873/DATE.2015.0378.

[13] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal Gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013*, pages 40–56. Springer, 2013. Extended version available as IACR ePrint 2013/383. URL: https://eprint.iacr.org/2013/383, doi:10.1007/978-3-642-40041-4_3.

[14] M. Dworkin. Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. NIST Special Publication 800-38D, November 2007. URL: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38D.pdf, doi:10.6028/NIST.SP.800-38D.

[15] FIPS. Specification for the Advanced Encryption Standard (AES). Federal Information Processing Standards Publication 197, November 2001. URL: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[16] FIPS. (FIPS) 186-4, digital signature standard (DSS). Federal Information Processing Standards Publication, July 2013. doi:10.6028/NIST.FIPS.186-4.

[17] N. Göttert, T. Feller, M. Schneider, J. Buchmann, and S. Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In E. Prouff and P. Schaumont, editors, *CHES 2012*, volume 7428 of *LNCS*, pages 512–529. Springer, 2012. doi:10.1007/978-3-642-33027-8_30.

[18] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte, and Z. Zhang. Choosing parameters for NTRUEncrypt. IACR ePrint 2015/708, 2015. URL: https://eprint.iacr.org/2015/708.

[19] M. Hutter and P. Schwabe. NaCl on 8-bit AVR microcontrollers. In A. Youssef, A. Nitaj, and A. E. Hassanien, editors, *ASIACRYPT 2013*, volume 7918 of *LNCS*, pages 156–172. Springer, 2013. URL: https://eprint.iacr.org/2013/375, doi:10.1007/978-3-642-38553-7_9.

[20] J. Jonsson and B. Kaliski. Public-key cryptography standards (PKCS) #1: RSA cryptography

specifications version 2.1. IETF RFC 3447, February 2003. URL: https://www.rfc-editor.org/rfc/rfc3447.txt, doi:10.17487/RFC3447.

[21] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer, 1999. doi:10.1007/3-540-48405-1_25.

[22] R. Lindner and C. Peikert. Better key sizes (and attacks) for LWE-based encryption. In A. Kiayias, editor, *CT-RSA 2011*, volume 6558 of *LNCS*, pages 319–339. Springer, 2011. doi:10.1007/978-3-642-19074-2_21.

[23] Z. Liu, Großschädl, and I. Kizhvatov. Efficient and side-channel resistant RSA implementation for 8-bit AVR microcontrollers. In *Proc. SecIoT 2010*, 2010. URL: https://www.nics.uma.es/pub/seciot10/files/pdf/liu_seciot10_paper.pdf.

[24] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, H. Kim, and I. Verbauwhede. Efficient Ring-LWE encryption on 8-bit AVR processors. In T. Güneysu and H. Handschuh, editors, *CHES 2015*, volume 9293 of *LNCS*, pages 663–682. Springer, 2015. URL: https://eprint.iacr.org/2015/410, doi:10.1007/978-3-662-48324-4_33.

[25] G. Lowe. An attack on the Needham-Schroeder public-key authentication protocol. *Information Processing Letters*, 56(3):131–136, November 1995. doi:10.1016/0020-0190(95)00144-2.

[26] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In H. Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 1–23. Springer, 2010. doi:10.1007/978-3-642-13190-5_1.

[27] D. Moody. Post-quantum cryptography: NIST's plan for the future. Talk given at PQCrypto '16 Conference, 23-26 February 2016, Fukuoka, Japan, February 2016. URL: https://pqcrypto2016.jp/data/pqc2016_nist_announcement.pdf.

[28] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978. doi:10.1145/359657.359659.

[29] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. Official Call for Proposals, National Institute for Standards and Technology, December 2016. URL: http://csrc.nist.gov/groups/ST/post-quantum-crypto/documents/call-for-proposals-final-dec-2016.pdf.

[30] NSA/CSS. Information assurance directorate: Commercial national security algorithm suite and quantum computing FAQ, January 2016. URL: https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm.

[31] T. Pöppelmann and T. Güneysu. Area optimization of lightweight lattice-based encryption on reconfigurable hardware. In *ISCAS 2014*, pages 2796–2799. IEEE, 2014. doi:10.1109/ISCAS.2014.6865754.

[32] T. Pöppelmann and T. Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In T. Lange, K. Lauter, and P. Lisoněk, editors, *SAC 2013*, volume 8282 of *LNCS*, pages 68–85. Springer, 2014. doi:10.1007/978-3-662-43414-7_4.

[33] T. Pöppelmann, T. Oder, and T. Güneysu. High-performance ideal lattice-based cryptography on 8-bit ATxmega microcontrollers. In K. Lauter and F. Rodríguez-Henríquez, editors, *LATINCRYPT 2015*, volume 9230 of *LNCS*, pages 346–365. Springer, 2015. URL: http://eprint.iacr.org/2015/382, doi:10.1007/978-3-319-22174-8_19.

[34] J. Proos and C. Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quantum Information & Computation*, 3(4):317–344, July 2003. Updated version available on arXiv. URL: https://arxiv.org/abs/quant-ph/9508027.

[35] E. Rescorla. The Transport Layer Security (TLS) protocol version 1.3. IETF Internet Draft, September 2016. URL: https://tlswg.github.io/tls13-spec/.

[36] S. S. Roy, F. Vercauteren, N. Mentens, D. D. Chen, and I. Verbauwhede. Compact Ring-LWE cryptoprocessor. In L. Batina and M. Robshaw, editors, *CHES 2014*, volume 8731 of *LNCS*, pages 371–391. Springer, 2014. URL: https://eprint.iacr.org/2013/866, doi:10.1007/978-3-662-44709-3_21.

[37] M.-J. O. Saarinen. Arithmetic coding and blinding countermeasures for lattice signatures. *Journal of Cryptographic Engineering*, 2017. To appear. Also IACR ePrint 2016/276. URL: https://eprint.iacr.org/2016/276, doi:10.1007/s13389-017-0149-6.

[38] P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proc. FOCS '94*, pages 124–134. IEEE, 1994. Updated version available on arXiv. URL: https://arxiv.org/abs/quant-ph/9508027, doi:10.1109/SFCS.1994.365700.

[39] P. Tuyls and L. Batina. RFID-tags for anti-counterfeiting. In D. Pointcheval, editor, *CT-RSA 2006*, volume 3860 of *LNCS*, pages 115–131. Springer, 2006. doi:10.1007/11605805_8.

[40] P. E. Yee. Updates to the internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. IETF RFC 6818, January 2013. URL: https://www.rfc-editor.org/rfc/rfc6818.txt, doi:10.17487/RFC6818.