Practical CCA2-Secure and Masked Ring-LWE Implementation

Tobias Oder[1], Tobias Schneider[1], Thomas Pöppelmann[2], and Tim Güneysu[3]

[1]Horst Görtz Institute for IT Security, Ruhr-Universität Bochum, Germany
{tobias.oder,tobias.schneider-a7a}@rub.de
[2]Infineon Technologies AG, Germany
thomas.poeppelmann@infineon.com
[2]University of Bremen and DFKI, Germany
tim.gueneysu@uni-bremen.de

Abstract. In this work we provide the first practical instantiation of ring-LWE-based public-key encryption that is protected against active attacks (i.e., adaptive chosen-ciphertext attacks) and equipped with countermeasures against side-channel attacks (masking and hiding). We propose a novel provably first-order secure masking scheme that outperforms previous work and we combine this masking approach with blinding and shuffling techniques to further thwart higher-order attacks. Our work shows that extremely fast and secured implementations of post-quantum public-key encryption are possible on constrained devices and we give evidence that ring-LWE-based schemes are highly suitable for implementations on smart cards due to the large amount of linear operations. Even with conservative parameter choices ($n = 1024, q = 12289$) for ring-LWE encryption we obtain 243 bits of quantum security based on a recently established model. Our implementation requires 1,222,054 cycles for encryption and 2,372,242 cycles for decryption with masking and hiding countermeasures on a Cortex-M4F. Furthermore, the first-order security of our masked implementation is practically verified using the non-specific t-test evaluation methodology.

1 Introduction

Public-key encryption (PKE) is a fundamental asymmetric cryptographic primitive and plays an extremely important role in numerous applications and security protocols, where a prominent example is email encryption or key-transport for key-exchange. However, in case one of the numerous attempts and approaches to build a sufficiently large quantum computer turns out to be successful, Shor's algorithm [58] could be used to break RSA and ECC-based PKEs in polynomial time. This would be especially devastating as it would also jeopardize the security of RSA-encrypted ciphertexts that are sent today, in case they are stored and decrypted by a malicious entity when quantum computers become accessible[1].

[1] Note that this scenario is quite likely for encrypted emails as users (or their providers) might keep them encrypted for a long time on a server over which they do not have control.

Concerns over quantum computers have recently been fueled by an announcement of NIST to start the standardization process for post-quantum cryptography [14, 43] and by the statement of NSA's Information Assurance Directorate (IAD) to "initiate a transition to quantum resistant algorithms in the not too distant future" for Suite B cryptography [42].

Possible candidates to replace RSA and ECC-based public key encryption are cryptosystems based on the hardness of certain lattice problems – the most prominent example is NTRUEncrypt, which was proposed by Hoffstein, Pipher, and Silverman [31] more than 18 years ago. However, recently cryptosystems and various instantiations based on ideal lattices or more specifically the ring-learning with errors (ring-LWE) problem also gained popularity as a research topic. This happened presumably due to their simplicity, high efficiency, and scalability (see [10, 11, 17, 54]), as well as because of theoretical foundations and security reductions (see [37, 44]). A practical advantage of ring-LWE-based encryption over NTRU is fast key generation and relatively easy constant-time implementation, which is useful when constructing schemes for ephemeral key exchange (e.g., NewHope [1] and BNCS [13]).

However, there are several challenges that have to be solved before ring-LWE-based encryption can be considered as a serious replacement of RSA or ECC for public-key encryption and (authenticated) key exchange. The most pressing, and often overlooked issues are encryption errors (i.e., correctness), security against adaptive chosen-ciphertext attacks (CCA2), and the protection against side-channel attacks. In this context, a basic semantically secure encryption scheme with parameters leading to a negligible amount of decryption errors is a requirement to achieve CCA2-security as discussed by Dwork, Naor, and Reingold [22] when applying CCA2-transformations. This issue was also shown by practical attacks on NTRU [32]. Moreover, CCA2-security is a condition for most real world usage scenarios and has to be in place before side-channel protection can be considered. Otherwise, an attacker with physical access to a decryption oracle containing a secret key could simply create malformed ciphertexts to reveal a secret key, without the need to perform a side-channel attack at all[2]. The importance of CCA2-security is also reflected in the current NIST draft submission requirements for post-quantum public-key encryption and key-exchange [43] that explicitly ask for CCA2-security.

Contribution. In this work we address the aforementioned issues with practical usage of ring-LWE PKE schemes as post-quantum alternative to RSA and ECC. We provide a parametrization of ring-LWE that achieves negligible decryption errors and security against chosen-ciphertext attacks using the conversion described by Peikert [44] that is basically the well-established Fujisaki-Okamoto [26] transformation. For parameter selection we follow the conservative approach of [1] and show that it is still possible to achieve long-term security. Moreover, we show how the CCA2-conversion interacts with an enhanced masking scheme for protection against differential power analysis (DPA). Our imple-

[2] Note that a chosen-ciphertext attack on CPA-secure ring-LWE encryption [36, 37] is trivial; see [25].

mentation and measurements were carried out on a Cortex-M4F, which is similar to some extend to architectures of popular smart cards[3]. In the end we provide the first implementation of ring-LWE encryption with enhanced side-channel countermeasures (provably secure masking combined with hiding) that could be run on an embedded microcontroller and that does not suffer the aforementioned limitations of previous schemes (i.e., decryption errors and no CCA2-conversion). Our implementation achieves 781,423 cycles for key generation, 1,222,054 cycles for encryption, 2,372,242 cycles for decryption with masking and hiding countermeasures for a conservative parameter choice with dimension $n = 1024$, modulus $q = 12289$, and noise std. deviation $\varsigma = \sqrt{11/2}$. The supposed security level against currently known quantum adversaries in the model of [1] is 243-bits. It offers protection against timing side-channels and even after 100,000 traces no first order leakage was detectable by the non-specific t-test [27]. In a fair comparison, our masking scheme outperforms previous masking approaches for ring-LWE by at least a factor of 1.4.

2 Preliminaries

In this section we cover preliminaries on ring-LWE-based public-key encryption, discuss previous attempts to mask ring-LWE-based PKE schemes, and cover related work on protected NTRU implementations.

2.1 Notation

Unless explicitly stated we denote addition (resp. substraction) modulo q with $+$ (resp. $-$). We denote multiplication by $\cdot$ and point-wise multiplication by $\circ$. We use $\oplus$ as operator for addition modulo 2. Polynomials are labeled by bold lower case letters.

2.2 Ring-LWE Encryption

The plain ring-LWE-based public-key encryption scheme we are using was previously proposed in [36, 38, 39] and key generation is described in Algorithm 1, encryption in Algorithm 2, and decryption in Algorithm 3. All elements are polynomials over $\mathcal{R}_q = \mathbb{Z}_q[\mathbf{x}]/\langle x^n + 1\rangle$ where we always assume implicit reduction modulo q and reduction modulo $x^n + 1$ where the authors make explicit use of the number theoretic transform[4]. The application of NTT techniques to RLWE.CPA has been previously described in [47, 54] and the same approach (besides others) was used for efficiency optimization of the NewHope key exchange scheme in [1].

[3] See http://www.arm.com/products/processors/securcore/sc300.php.

[4] The NTT basically allows to efficiently compute a polynomial multiplication $\mathbf{a} \cdot \mathbf{b}$ as $\mathbf{a} \cdot \mathbf{b} = \text{INTT}(\text{NTT}(\mathbf{a}) \circ \text{NTT}(\mathbf{b}))$.

Algorithm 1: RLWE.CPA Key Generation

$\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{gen}}()$
  $\tilde{\mathbf{r}}_1 \leftarrow \mathsf{NTT}(\mathsf{SampleNoisePoly}())$
  $\tilde{\mathbf{r}}_2 \leftarrow \mathsf{NTT}(\mathsf{SampleNoisePoly}())$
  $\tilde{\mathbf{a}} \leftarrow \mathsf{NTT}(\mathsf{SampleUniformPoly}())$
  $\tilde{\mathbf{p}} = \tilde{\mathbf{r}}_1 - \tilde{\mathbf{a}} \circ \tilde{\mathbf{r}}_2$
  return $(\mathsf{pk}, \mathsf{sk}) = ((\tilde{\mathbf{p}}, \tilde{\mathbf{a}}), \tilde{\mathbf{r}}_2)$
end

Algorithm 2: RLWE.CPA Encryption

$\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}(\tilde{\mathbf{a}}, \tilde{\mathbf{p}}, m \in \{0,1\}^n;\ seed)$
  $InitPRNG(seed)$
  $\tilde{\mathbf{e}}_1 = \mathsf{NTT}(\mathsf{SampleNoisePoly}())$
  $\tilde{\mathbf{e}}_2 = \mathsf{NTT}(\mathsf{SampleNoisePoly}())$
  $\tilde{\mathbf{c}}_1 = \tilde{\mathbf{a}} \circ \tilde{\mathbf{e}}_1 + \tilde{\mathbf{e}}_2$
  $\tilde{\mathbf{h}}_2 = \tilde{\mathbf{p}} \circ \tilde{\mathbf{e}}_1$
  $\mathbf{e}_3 \leftarrow \mathsf{SampleNoisePoly}()$
  $\mathbf{c}_2 = \mathsf{INTT}(\tilde{\mathbf{h}}_2) + \mathbf{e}_3 + \mathsf{LWEEncode}(\mu)$
  return $c = (\tilde{\mathbf{c}}_1, \mathbf{c}_2)$
end

Algorithm 3: RLWE.CPA Decryption

$\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}(\tilde{\mathbf{r}}_2, c = [\tilde{\mathbf{c}}_1, \mathbf{c}_2])$
  return $\mathsf{LWEDecode}(\mathsf{INTT}(\tilde{\mathbf{c}}_1 \circ \tilde{\mathbf{r}}_2) + \mathbf{c}_2)$.
end

Efficient algorithms for the computation of the NTT exist when $1 \equiv q \bmod 2n$ for q being a prime and n being a power-of-two. The NTT is especially helpful when coefficients are fixed or needed twice as it is then possible to directly store them in NTT representation to save subsequent transformations. By $\mathsf{SampleNoisePoly}$ we denote a function that samples a polynomial in $\mathcal{R}_q$ with coefficients coming from a noise distribution. In earlier works [28,36] this distribution was usually a (high-precision) discrete Gaussian distribution with parameter σ. However, newer results show that security can also be achieved with distributions that are close to a discrete Gaussian (e.g, the binomial distribution [1] or some fixed distribution [12]). A uniformly random polynomial is sampled by $\mathsf{SampleUniformPoly}()$ and we decided to include $\mathbf{a}$ in the public key for simplification (see [1,12] for a discussion of on-the-fly generation or choice of $\mathbf{a}$ as global constant). Note that RLWE.CPA requires (as a minimum) a simple message encoding ($\mathsf{LWEEncode}$ and $\mathsf{LWEDecode}$) to allow the extraction of the message from remaining noise (i.e., $\mathbf{e}^* = \mathbf{e}_1\mathbf{r}_1 + \mathbf{e}_2\mathbf{r}_2 + \mathbf{e}_3$) during decryption. Therefore, Lindner and Peikert [36] proposed threshold encoding functions for individual coefficients which we implicitly also apply to polynomials. For a bit of the message $m \in \{0,1\}$ they define

$$\mathsf{LWEEncode}(m) = m \cdot \left\lfloor \frac{q}{2} \right\rfloor$$

and the decoding of a corresponding coefficient $\alpha \in \mathbb{Z}_q$ as

$$\mathsf{LWEDecode}(\alpha) = \left\{ \begin{array}{l} \text{return 1 iff } \alpha \in \left[-\left\lfloor \frac{q}{4} \right\rfloor, \left\lfloor \frac{q}{4} \right\rfloor \right) \\ \text{return 0 otherwise.} \end{array} \right\}$$

Thus the error tolerance is $t = \left\lfloor \frac{q}{4} \right\rfloor$ and decryption correctness is obtained as long as $e_i^* \in [-t, t)$ for each coefficient e_i^* of $\mathbf{e}^*$. For more information on parameters we refer to Table 1 and the discussion in Section 3.1.

2.3 Related Work on Masked ring-LWE

Masking schemes for the ring-LWE encryption scheme have already been investigated by Reparaz *et al.* in [51, 52] and subsequently in [49]. However, both approaches suffer from drawbacks that make them difficult to apply in practical applications. The main idea of [51,52] is to split the secret key $\mathbf{r}_2$ into two shares and to compute the multiplication $\mathbf{r}_2 \cdot \mathbf{c}_1$ separately on both shares and add $\mathbf{c}_2$ to one of the shares. The authors construct a masked decoder that takes both shares as input and checks whether certain pre-defined rules are satisfied or not. Note that for half of all inputs no rule applies and thus, the value cannot be decoded immediately. This problem is solved by adding a certain δ to the shares and restarting the decoding process. The decoding process can be re-started up to 16 times until it eventually fails to decode. Thus, the decoding failure rate rises. This is highly undesirable for our application, since a negligible failure probability is required for the CCA2-conversions.

In their follow-up work [49], a different masking scheme is applied. The authors exploit that the ring-LWE decryption is *almost* additively homomorphic. Instead of dividing the secret key into two shares, they split the ciphertext into two shares and compute $\mathtt{decrypt}(\mathbf{c}_1 + \mathbf{c}_1', \mathbf{c}_2 + \mathbf{c}_2')$ to receive $(m \oplus m')$ as output. Notice that this procedure includes an additional encryption of m' during the decryption. Unfortunately, the addition of two ciphertexts implies that also the including error vectors are added and this again raises the failure probability of the scheme and lowers performance.

In both masking schemes, the output of the decryption is split into two parts, $(m \oplus m')$ and m'. In Appendix B of [53] the authors state that they were able to simulate a DPA attack targeting the pre-decoded value α. As the output of the decryption is $m = \mathsf{LWEDecode}(\alpha)$, we expect a DPA attack on m to be feasible as well. Thus, the shares of m must not be combined on the device that performs the decryption. Transmitting the message in two shares could lead to compatibility issues as the receiver of the decrypted message has to be aware of the masking scheme. Another more severe issue is that the simulated attack on α from [53] requires an attacker to be able to choose arbitrary ciphertexts. Such an attacker is even able to find the secret key without DPA as ring-LWE itself does not provide CCA2-security (see [25]) but only security against chosen plaintext attacks (CPA). Thus, we draw two conclusions for the implementation of practically secure ring-LWE encryption:

– Assuming a CPA-only attacker, the DPA attack on ring-LWE without masked decoding is not feasible and thus no masked decoder is required.
– Assuming a CCA2 attacker, a CCA2-conversion has to be applied to ring-LWE. Otherwise, an attacker would be able to break the system without performing a DPA and thus rendering any side-channel countermeasures useless. The message m must not be stored unmasked in this setting.

2.4 Related Work on NTRU

In this section we review works on implementation attacks on NTRU. This is relevant as NTRU and ring-LWE have a similar structure (especially from the perspective of an implementer). Thus, (older) works on protecting NTRU are a natural reference for countermeasure to protect ideal (or even standard) lattice-based scheme that should not get overlooked.

In [3] a hardware implementation of NTRU and a first study regarding DPA attacks is provided. The attack allows to recover secret coefficients one-by-one using a Hamming distance model and Pearson's correlation coefficient. In [62] a correlation power analysis of an NTRU implementation equipped with the blinding countermeasures proposed in [41] is attacked. These countermeasures are addition of a random integer before convolution that can easily be removed, blinding using a random value, and randomization of the order of which coefficients are processed. As additional countermeasures in [62] random delays are proposed, masking, as well as dummy operations. A first order collision attack on NTRU is given in [63] and as countermeasure, besides random delays, a mathematical randomization is proposed where two inputs $\mathbf{a}$ and $\mathbf{b}$ to a convolution are randomly rotated as $\mathbf{a}' = \mathbf{a} \cdot x^i$ and $\mathbf{b}' = \mathbf{b} \cdot x^{n-i}$ for a random i so that the result $\mathbf{a}' \cdot \mathbf{b}' = \mathbf{a} \cdot x^i \cdot \mathbf{b} \cdot x^{n-i} = \mathbf{ab}$. The same countermeasure has recently also been proposed by Saarinen in [56] for lattice-based signatures with the observation that the shifting can be integrated into the NTT. Additionally, Saarinen proposes the multiplication with random constants that could also be integrated into the NTT computation. Timing attacks on NTRU have been investigated in [59]. Fault attacks are given in [34] and countermeasures against fault attacks are given in [35], mainly using spatial and temporal duplication.

3 CCA2-Conversion and Masking

In this section we describe our approach for parameter selection, how we deal with decryption errors, and how we apply the generic Fujisaki-Okamoto [26] (FO) CCA2-transformation to the concrete parametrized ring-LWE-based encryption scheme. Moreover, we describe side-channel countermeasures and analyze fault resistance.

3.1 Parameter Selection

For parameter selection we roughly follow the conservative approach from [1]. The scheme should provide long-term security and withstand possible dimension-halving attacks. Moving to larger dimensions also simplifies the security analysis

Table 1: Security levels and failure probability of previously proposed ring-LWE-based public-key encryption or key-exchange schemes. The security level was computed based on the model in [1]. Note that C-Sec = classical bit-level security and Q-Sec = known quantum bit-level security.

Set	Parameter (n, q, ς)	C-Sec	Q-Sec	Failure
RLWE.CPA [28]	$(256, 7681, \approx 4.5)$	64	58	$\approx 2^{-11}$
RLWE.CPA [28]	$(512, 12289, \approx 4.9)$	144	131	$\approx 2^{-10}$
BCNS [13]	$(1024, 2^{32}-1, \approx 3.2)$	86	78	$2^{-2^{17}}$
NewHope [1]	$(1024, 12289, \approx 2.8)$	282	256	2^{-60}
This work	$(1024, 12289, \approx 2.3)$	268	243	2^{-128}

as attacks that might work in lower dimensions do not have to be considered anymore. Moreover, our goal is to transfer a 256-bit symmetric key to account for quantum acceleration of brute-force attacks [29]. With $n = 1024$ we have four coefficients to encode one bit of a 256-bit message (similar as in NewHope and exactly as described in [47]) and can thus tolerate noise levels $4 \cdot \frac{q}{4} = q$. However, to obtain a decryption error probability lower than 2^{-128} for the basic CPA-secure scheme we have to lower the parameter of the binomial distribution used in [1] to $k = 11$ (contrary to $k = 16$ in NewHope). This way the noise has a standard deviation of $\varsigma = \sqrt{11/2} \approx 2.3452$. This slightly lowers the security level but still gives us more than 128-bits of security against a known-quantum adversary. Our final parameter set is compared with previous proposals in Table 1 (see Table 5 and Appendix A for more details and the parameters used for the scripts). However, the bit security level compared to NewHope is only slightly smaller and the security is still better than BCNS or previously proposed RLWE.CPA parameters. It is also worth mentioning that the security estimation in [1] uses several worst-case estimations/simplifications so that the concrete security of the instances might be higher (i.e., there is currently no known algorithm that breaks, e.g., BCNS with 2^{86} steps). We would also like to note that we do not cut any bits of the second component of the ciphertext as described in [47]. This way we can sustain a larger error than NewHope (it has tolerance of $\frac{3q}{4}$) but at the cost of a larger ciphertext.

3.2 CCA2 Conversion for RLWE.CPA

In this work we use the second Fujisaki-Okamoto [26] transformation to achieve security against adaptive chosen ciphertext attacks (CCA2). For this transformation, Peikert came to the conclusion [44] that it is the best choice to convert a passively secure encryption scheme into an actively secure one (in the random oracle model; CCA2 due to security against adaptive attacks). For this transformation, two random oracles $G : \{0,1\}^L \rightarrow \{0,1\}^l$ and $H : \{0,1\}^{L+l} \rightarrow \{0,1\}^\lambda$ are required. The parameter L determines the size of message to be encrypted, l the length of the input to ring-LWE encryption, and λ the length of the seed for the PRNG. In our implementation, the parameters L, l, and λ are set to 256.

We define $\mathsf{RLWE.CCA}^{\mathsf{NTT}}_{\mathsf{enc}}$ and $\mathsf{RLWE.CCA}^{\mathsf{NTT}}_{\mathsf{dec}}$ as follows:

- $\mathsf{RLWE.CCA}^{\mathsf{NTT}}_{\mathsf{enc}}(\tilde{\mathbf{a}}, \tilde{\mathbf{p}}, M)$: let $(c_1, c_2) = \mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}(\tilde{\mathbf{a}}, \tilde{\mathbf{p}}, \nu; H(\nu||M))$ where ν is a nonce $\in \{0,1\}^L$ and $H(\nu||M)$ seeds the PRNG of $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}$ and $c_3 = G(\nu) \oplus M$ and output (c_1, c_2, c_3).
- $\mathsf{RLWE.CCA}^{\mathsf{NTT}}_{\mathsf{dec}}(\tilde{\mathbf{r}}_2, \tilde{\mathbf{a}}, \tilde{\mathbf{p}}, c_1, c_2, c_3)$: compute $\nu = \mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}(\tilde{\mathbf{r}}_2, c_1, c_2)$ and $M = G(\nu) \oplus c_3$, and check whether $(c_1, c_2) \stackrel{?}{=} \mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}(\tilde{\mathbf{a}}, \tilde{\mathbf{p}}, \nu; H(\nu||M))$. If so, output M, otherwise output $fail$.

Using this transformation and our chosen parameters we obtain a theoretical public-key size of $|(\mathbf{a}, \mathbf{p})| = 2n\lceil \log_2(q) \rceil = 2 \cdot 1024 \cdot 14 = 28672$ bits (3584 bytes) and a theoretical ciphertext size of $|(c_1, c_2, c_3)| = 2n\lceil \log_2(q) \rceil + 256 = 28928$ bits (3616 bytes). The secret key is $|\mathbf{r}_2| = n\lceil \log_2(q) \rceil = 14336$ bits (1792 bytes). Note that it would be possible to generate the secret key or $\mathbf{a}$ from a seed of 256-bits (or to choose $\mathbf{a}$ as a global constant; see [1] for a discussion). Additionally, the secret key $\mathbf{r}_2$ could be encoded as it is not distributed uniformly but roughly follows a discrete Gaussian (see [46, 56]). However, for simplicity, comparability, and maintainability, we leave these state-of-the art optimizations as future work. To obtain negligible error we replace the encoding and decoding functions ($\mathsf{LWEEncode}$ and $\mathsf{LWEDecode}$) with variants that encode one message bit into four coefficients [47] (as mentioned earlier). The encoding function used in $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}$ is defined as

$$\mathsf{Encode}(m) = \sum_{i=0}^{n-1} m[\lfloor i/4 \rfloor] \cdot q \cdot x^i$$

(where $m[i]$ denotes the i-th bit of m). The decoding function $\mathsf{Decode}(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ used in $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}$ that takes four coefficients $\alpha_1, \alpha_2, \alpha_3, \alpha_4 \in [-\lfloor q/2 \rfloor, \lfloor q/2 \rfloor]$ as input (that carry one bit of the message) is defined as

$$\mathsf{Decode}(\alpha_1, \alpha_2, \alpha_3, \alpha_4) = \left\{ \begin{array}{l} \text{return 0 iff } |\alpha_1| + |\alpha_2| + |\alpha_3| + |\alpha_4| < q \\ \text{return 1 otherwise.} \end{array} \right\}$$

3.3 Masked CCA2-Secure ring-LWE Decryption

To achieve side-channel resistance, it is necessary to mask all vulnerable modules of the CCA2-secure decryption. As depicted in Figure 2, these modules are $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}$, G, H, and $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}$. Note that it is not sufficient to only protect $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}$, because in a chosen-ciphertext setting an adversary can target the unmasked output of $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}$ (see Appendix B of [53]) to recover the secret key. Therefore, all modules that receive the masked output of $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}$ as input need to be masked as well. Relying on the preimage resistance of G, we do not need to mask $c_3 \oplus G$.

In the following, we analyze the first-order security of each module separately in the common probing model [33]. To this end, we show that an attacker, which can probe one intermediate variable of the computation, cannot derive any secret

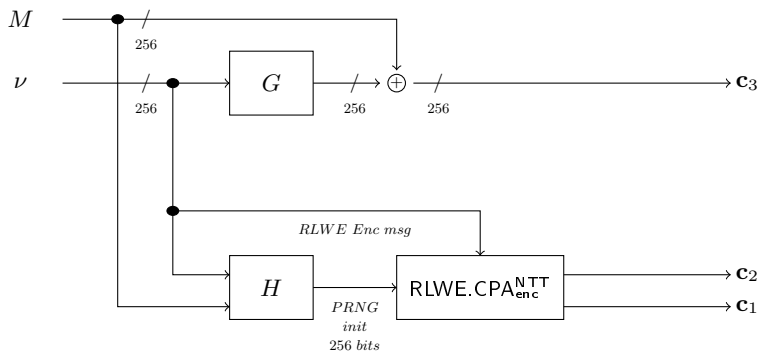


Fig. 1: CCA2-secure encryption.

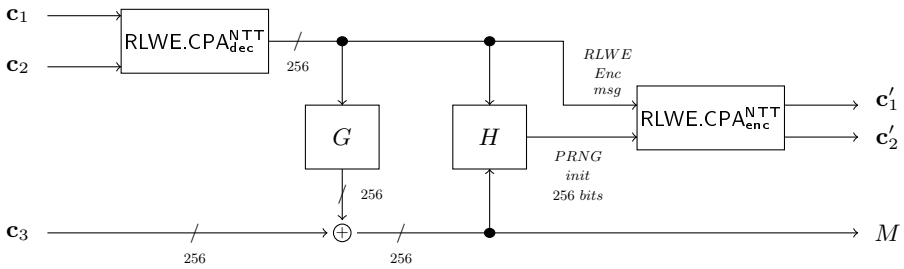


Fig. 2: CCA2-secure decryption.

information. This notion is equivalent to showing that each intermediate variable follows a distribution independent of any sensitive variable. For one probe it is indeed sufficient to analyze each module separately, if the input and output distributions between the modules are consistent. Therefore, 1-probing security with correct input distributions for each module implies 1-probing security of the complete masked CCA2-secure decryption. However, for more probes (i.e., 2-probing security) this approach would not cover every possible attack vector and more sophisticated analysis techniques need to be utilized.

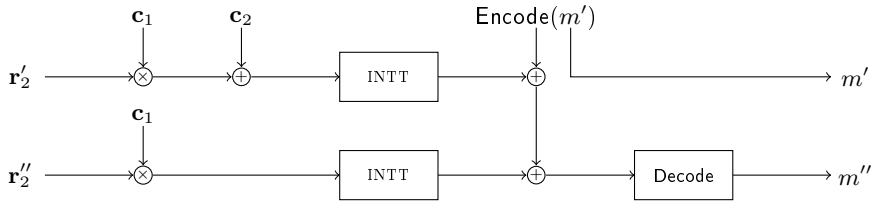


Fig. 3: Our masking scheme for the ring-LWE decryption.

Ring-LWE Decryption. As mentioned in Section 2.3, the masking schemes of the ring-LWE decryption from [51, 52] and [49] suffer from a higher failure probability and slower performance. Therefore, we present a new approach which avoids the aforementioned problems and still provides side-channel protection. Figure 3 shows the basic structure of our masked ring-LWE decryption. For the initial multiplications, additions, and INTTs we rely on a simple sharing of $\mathbf{r}_2 = \mathbf{r}'_2 + \mathbf{r}''_2$ similar to [51, 52]. Given the linearity of the operations, it is easily possible to perform these computations on each share separately. However, this approach does not work for the final Decode. In [51, 52], the authors proposed to use a rather complex masked decoder instead. To increase efficiency, we avoid a specific masked decoder and remask the intermediate values. To this end, first a random bitstring $m' \xleftarrow{\$} \{0,1\}^{256}$ is sampled and encoded. The encoded bitstring is added to the first share, before adding the result to the second share. Then Decode is performed once on this new share to compute m'' with $m'' \oplus m' = m$. With this approach, we can avoid the costly masked decoder and the additional error of the scheme from [49].

Correctness. To show the correctness of this scheme, we first denote the outputs of the INTT operations as α' and α'' with $\alpha = \alpha' + \alpha''$. Showing that this relation holds is trivial, since the INTT is linear and the scheme is identical to [51, 52] up to this point. Nevertheless, α' and α'' are computed as

$$\begin{aligned}\alpha' &= \mathrm{INTT}(\tilde{\mathbf{r}}'_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2)\\ \alpha'' &= \mathrm{INTT}(\tilde{\mathbf{r}}''_2 \circ \tilde{\mathbf{c}}_1)\end{aligned}$$

and the unshared output α is defined as

$$\begin{aligned}\alpha &= \mathrm{INTT}(\tilde{\mathbf{r}}_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2)\\ &= \mathrm{INTT}((\tilde{\mathbf{r}}'_2 + \tilde{\mathbf{r}}''_2) \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2)\\ &= \mathrm{INTT}(\tilde{\mathbf{r}}'_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{r}}''_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2)\\ &= \mathrm{INTT}(\tilde{\mathbf{r}}'_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2) + \mathrm{INTT}(\tilde{\mathbf{r}}''_2 \circ \tilde{\mathbf{c}}_1)\\ &= \alpha' + \alpha''.\end{aligned}$$

Next, we show that $m = m' \oplus m''$. To this end, we use that Decode is linear when the ring-LWE Gaussian noise term $\mathbf{r}_2\mathbf{e}_2 + \mathbf{r}_1\mathbf{e}_1 + \mathbf{e}_3$ is less than q for every four coefficients. The noise term is denoted in the following example as *noise*. Then the following relation holds

$$\begin{aligned}&\mathsf{Decode}(\mathsf{Encode}(x) + \mathsf{Encode}(y) + \mathit{noise})\\ &= \mathsf{Decode}(\mathsf{Encode}(x)) \oplus \mathsf{Decode}(\mathsf{Encode}(y) + \mathit{noise}))\\ &= x \oplus y\end{aligned}$$

where *noise* is small enough to not cause a decoding error. This property is not satisfied for the outputs of the INTTs α' and α'' as the uniform masking results

in too much additional noise. Therefore, we remask to reduce the noise level and compute the output of $\mathsf{RLWE.CPA}_{\mathsf{dec}}^{\mathsf{NTT}}$ as

$$\begin{aligned}
&\mathsf{Decode}(\alpha' + \mathsf{Encode}(m') + \alpha'') \\
&= \mathsf{Decode}(\alpha + \mathsf{Encode}(m')) \\
&= \mathsf{Decode}(\alpha) \oplus \mathsf{Decode}(\mathsf{Encode}(m')) \\
&= m \oplus m' = m''
\end{aligned}$$

with $\alpha = \mathsf{Encode}(m) + noise$. Thus the decrypted ciphertext is Boolean-masked by the random bitstring m'.

Security Analysis. The masked ring-LWE decryption as described above does indeed compute the correct results. However, in its current form it does not provide SCA security. This is due to the rounding applied on the ciphertext. In particular, the intermediate value $\alpha + \mathsf{Encode}(m') \bmod q$ leaks information about the sensitive variable α even though m' is a fresh random mask. The issue becomes clear by closer inspection of the distribution of α which is the encoded message m. For simplicity we assume in the following example that α and m are only one coefficient meaning that m is a uniformly distributed bit. Thus, α follows one of two distributions depending on the value of m.

$$\begin{aligned}
\alpha &\sim \mathcal{N}(0, \sigma), \text{ if } m = 0, \\
\alpha &\sim \mathcal{N}\left(\left\lfloor \frac{q}{2} \right\rfloor, \sigma\right), \text{ if } m = 1,
\end{aligned}$$

where σ denotes the standard deviation of the noise terms $\mathbf{r}_2\mathbf{e}_2 + \mathbf{r}_1\mathbf{e}_1 + \mathbf{e}_3$. Since m is only one bit, it is sufficient to mask m (and thus α) with one fresh random bit m'. If we randomly switch the distribution of α between the two cases, an adversary that can probe the masked α will not be able to distinguish between $m = 0$ and $m = 1$. However, due to rounding the mean of α for $m = 1$ is $\left\lfloor \frac{q}{2} \right\rfloor$ instead of exactly $\frac{q}{2}$. Therefore, to switch from $0 \to 1$ it is required to add $\left\lfloor \frac{q}{2} \right\rfloor$ to α, while in the case $0 \to 1$ $\left\lfloor \frac{q}{2} \right\rfloor + 1$ are required to reach a mean of 0. This asymmetry makes it impossible to mask α with addition modulo q, since depending on the secret value m a different term needs to be added to α. To solve this issue, we reduce the modulus of two operations (i.e., $\alpha' + \mathsf{Encode}(m')$ and $(\alpha' + \mathsf{Encode}(m')) + \alpha''$) to $q-1$. Now, the switch for both cases requires the addition of $\left\lfloor \frac{q}{2} \right\rfloor$ enabling us to use our remasking scheme securely. It should be noted, that this approach introduces an additional error. However, as there are only two additions performed mod $(q-1)$ the error is at most 2 for one coefficient, resp. 8 for four coefficients. Since our decoding is capable of correcting an error up to q per four coefficients this is negligible. The final version of the masked ring-LWE decryption algorithm is shown in Algorithm 4.

In [44], Peikert deals differently with the odd modulus (as the context is different as well). His solution is to multiply the input by two and subtract a random bit *mod* $2q$ to ensure that the result is not biased. This leads to the following distributions of α

$$\alpha \sim \mathcal{N}(2q - 0.5, \sigma), \text{ if } m = 0,$$
$$\alpha \sim \mathcal{N}(q - 1.5, \sigma), \text{ if } m = 1,$$

In this case, the same argumentation as above holds and thus we cannot apply this approach.

Algorithm 4: Masked ring-LWE Decryption

Input : $\tilde{\mathbf{r}}'_2, \tilde{\mathbf{r}}''_2, \tilde{\mathbf{c}}_1, \tilde{\mathbf{c}}_2, m'$
Output: m''
1: $\alpha' \leftarrow \tilde{\mathbf{r}}'_2 \circ \tilde{\mathbf{c}}_1$
2: $\alpha' \leftarrow \alpha' + \tilde{\mathbf{c}}_2$
3: $\alpha' \leftarrow \mathrm{INTT}(\alpha')$
4: $\alpha'' \leftarrow \tilde{\mathbf{r}}''_2 \circ \tilde{\mathbf{c}}_1$
5: $\alpha'' \leftarrow \mathrm{INTT}(\alpha'')$
6: $\mathbf{t} \leftarrow \mathsf{Encode}(m')$
7: $\alpha' \leftarrow \alpha' + \mathbf{t} \bmod (q-1)$
8: $\alpha'' \leftarrow \alpha' + \alpha'' \bmod (q-1)$
9: $m'' \leftarrow \mathsf{Decode}(\alpha'')$

Lemma 1. *When $\tilde{\mathbf{r}}'_2$ and m' are uniformly and independently distributed in $\mathcal{R}_q$ and $\{0,1\}^{n/4}$, all intermediate variables in Algorithm 4 have a distribution independent of the sensitive variables $\mathbf{r}_2$ and m.*

Proof. For the proof, we analyze the distributions of the variables of each line from Algorithm 4 and show that their distributions are independent of the sensitive variables $\mathbf{r}_2$ and m.

- *Lines 1,2,4*: $\tilde{\mathbf{r}}_2$ is masked by $\tilde{\mathbf{r}}'_2$. Therefore, $\tilde{\mathbf{r}}'_2$ and $\tilde{\mathbf{r}}''_2$ follow each a uniform distribution in $\mathcal{R}_q$ which is independent of $\tilde{\mathbf{r}}_2$ and each result of these lines cannot reveal any information about $\tilde{\mathbf{r}}_2$. In fact, since $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ are part of the ciphertext, an adversary can simulate the intermediate values perfectly without knowledge of any sensitive variable. Indeed, $\tilde{\mathbf{c}}_2$ depends on $\tilde{\mathbf{r}}_2$ but according to the ring-LWE assumption an attacker does not gain any information about $\tilde{\mathbf{r}}_2$ by observing $\tilde{\mathbf{c}}_2$.
- *Lines 3,5*: The result of an INTT of a variable which is independent of any sensitive variable is still independent of any sensitive variable. In other words, the attacker can perfectly simulate this variable with only $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$.
- *Line 6*: m' is a uniform random variable. Therefore, the encoding of it can be perfectly simulated without additional knowledge and its distribution does not leak any sensitive information.
- *Line 7* ($\mathrm{INTT}(\tilde{\mathbf{r}}'_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2) + \mathsf{Encode}(m') \bmod (q-1)$): $\tilde{\mathbf{r}}'_2$ and m' are independent random variables and $\tilde{\mathbf{c}}_1$ and $\tilde{\mathbf{c}}_2$ are part of the ciphertext. Therefore, the result can be simulated and does not leak sensitive information.

- *Line 8* ($\mathrm{INTT}(\tilde{\mathbf{r}}_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2) + \mathsf{Encode}(m') \bmod (q-1)$): Here the first term ($\mathrm{INTT}(\tilde{\mathbf{r}}_2 \circ \tilde{\mathbf{c}}_1 + \tilde{\mathbf{c}}_2)$) can be written as $\mathsf{Encode}(m) + \mathcal{N}(0, \sigma)$. Therefore, the complete equation is simplified to $\mathsf{Encode}(m) + \mathsf{Encode}(m') + \mathcal{N}(0, \sigma) \bmod (q-1)$. Since the added noise is independent of the sensitive variable m, we can further reduce the equation to $\mathsf{Encode}(m) + \mathsf{Encode}(m') \bmod (q-1)$. The result of this addition is equally likely 0 or $\lfloor \frac{q}{2} \rfloor$ and does not depend on leaked information about m (resp. $\tilde{\mathbf{r}}_2$), since it is perfectly masked by m'. As mentioned before, this would not hold for an odd modulus, e.g., q.
- *Line 9*: The input to Decode is masked and therefore the output does not leak information.

As shown above, the distribution of every intermediate variable of Algorithm 4 is independent of the sensitive variables $\tilde{\mathbf{r}}_2$ and m. Therefore, it is not possible for an attacker, which can probe one value, to derive sensitive information. The output shares m' and m'' with $m = m' \oplus m''$ are both uniformly distributed in $\{0,1\}^{n/4}$.

***G* and *H* (SHAKE).** We choose to instantiate G and H with the commonly-used extendable-output function SHAKE that is based on the Keccak algorithm [7] and apply the masking scheme presented in [8]. Therefore, we do not include the security analysis of this module and instead refer the reader to the original publications.

Ring-LWE Encryption. While in the original Ring-LWE encryption scheme only one encoded value is added to the ciphertext polynomial $\mathbf{c}_2$, our masking scheme requires the addition of two shares during the re-encryption.

$$\mathbf{c}_2 = \mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3 + \mathsf{Encode}(m') + \mathsf{Encode}(m'')$$

Correctness. In this equation, $m' \oplus m'' = m$. Since our modulus q is odd and therefore $2\lfloor \frac{q}{2} \rfloor \neq q$, we have to adjust this operation so that the correct result is computed, i.e., the result of the re-encryption has to be exactly the same as the result of the original encryption. The naive approach would be to multiply the intermediate result of $\mathbf{c}_2$ (without the message) by 2, encode the shares of m as $\{0, q\}$, perform the two additions modulo $2q$, and divide the result by 2. While this approach indeed yields the correct result, it introduces an easily detectable side-channel leakage as the last bit of the intermediate results before the division is always set to 1 if and only if the unshared message bit is 1, i.e. q has been added exactly one time. Similarly, the last bit is always set to 0 if and only if the unshared message bit is 0. Similarly as in the decryption, we cannot apply the technique described in [44] as adding a random bit yields a different result if the value that bit is added to is odd. In the CCA2 setting, we must ensure that both, the original encryption and the re-encryption output exactly the same result and thus even a single bit error is not tolerable.

We therefore propose a different approach to securely compute this addition. The computation only outputs a false result in case both shares, m' and m'',

have the value 1. In this case, the `floor` operation cuts off $\frac{1}{2}$ two times and thus the result is off by one. To get the correct result, we have to add m' `AND` m''. Obviously, we cannot compute this multiplication of the shares directly without leakage. Thus we split the shares into subshares.

$$m' = \mathbf{m'_1} + \mathbf{m'_2}$$
$$m'' = \mathbf{m''_1} + \mathbf{m''_2}$$

Notice that for this calculation m' and m'' are implicitly treated as polynomials in $\mathcal{R}_q$ and not as bit vectors. For simplicity we assume in this description that one bit is encoded into one coefficient but this approach trivially generalizes to multi-coefficient encodings as well. As a consequence of the splitting into shares, we have to compute $(\mathbf{m'_1} + \mathbf{m'_2}) \circ (\mathbf{m''_1} + \mathbf{m''_2})$ instead of m' `AND` m''. To obtain the correct result, we compute:

$$\mathbf{c}_2^{pre} = ((((\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3) + \mathbf{m'_1}\mathbf{m''_1}) + \mathbf{m'_1}\mathbf{m''_2}) + \mathbf{m'_2}\mathbf{m''_1}) + \mathbf{m'_2}\mathbf{m''_2}$$
$$\mathbf{c}_2 = \mathbf{c}_2^{pre} + \mathsf{Encode}(m') + \mathsf{Encode}(m'')$$

Note that the terms $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3$ provide the randomness to secure the masked `AND` computation akin to Trichina's masked `AND` [61]. Therefore, the order of operations in the computation of $\mathbf{c}_2^{pre}$ is important for the security. Our complete masked re-encryption is shown in Algorithm 5. It should be noted that an attacker with knowledge of $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3 + \mathsf{Encode}(m)$ is able to conduct a side-channel attack on the intermediate value $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3$ to recover $\mathsf{Encode}(m)$. However, this approach is only applicable for valid ciphertexts, i.e., the input (c_1, c_2) of the CCA2-secure decryption has been generated using the CCA2-secure encryption. Otherwise, m ist anders also original m For valid ciphertexts, this attack does not pose a threat to the secret key r_2 since m can be known by the attacker, i.e., the attacker uses the public key to generate valid ciphertexts.

Lemma 2. *When each coefficient of $\mathbf{e}_1$, $\mathbf{e}_3$ is independent of the others and follows a Gaussian distribution $\sim \mathcal{N}(0, \varsigma)$, m' is uniformly distributed in $\{0,1\}^{n/4}$, m'_1 and m'_2 are uniformly and independently distributed in $\mathcal{R}_q^n$, and all these masks are independent from each other, all intermediate variables in Algorithm 5 have a distribution independent of the sensitive variable m.*

Proof. For the proof, we analyze the distributions of the variables of each line from Algorithm 5 and show that they are independent of the sensitive variable m.

- *Lines 1,2*: The terms are completely independent of m.
- *Lines 3,4*: m'_1 (resp. m''_1) are new random masks that are used to mask the shares of m. Since only one share of m is involved in each line, the result is still independent of m.
- *Lines 5,6,7,8*: Both terms of each line are uniformly and independently distributed in $\mathcal{R}_q$. Therefore, the multiplication of these terms does not create a new dependency on m and the results can be easily simulated.

Algorithm 5: Masked Ring-LWE Encryption

Input : $\mathbf{p}, \mathbf{e}_1, \mathbf{e}_3, m', m'', \mathbf{m'_1}, \mathbf{m''_1}$
Output: $\mathbf{c}_2$

1: $\mathbf{c}_2 \leftarrow \mathbf{p} \cdot \mathbf{e}_1$
2: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \mathbf{e}_3$
3: $\mathbf{m'_2} \leftarrow m' - \mathbf{m'_1}$
4: $\mathbf{m''_2} \leftarrow m'' - \mathbf{m''_1}$
5: $\mathbf{t}_{11} \leftarrow \mathbf{m'_1} \circ \mathbf{m''_1}$
6: $\mathbf{t}_{12} \leftarrow \mathbf{m'_1} \circ \mathbf{m''_2}$
7: $\mathbf{t}_{21} \leftarrow \mathbf{m'_2} \circ \mathbf{m''_1}$
8: $\mathbf{t}_{22} \leftarrow \mathbf{m'_2} \circ \mathbf{m''_2}$
9: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \mathbf{t}_{11}$
10: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \mathbf{t}_{12}$
11: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \mathbf{t}_{21}$
12: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \mathbf{t}_{22}$
13: $\ell' \leftarrow \mathsf{Encode}(m')$
14: $\ell'' \leftarrow \mathsf{Encode}(m'')$
15: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \ell'$
16: $\mathbf{c}_2 \leftarrow \mathbf{c}_2 + \ell''$

- *Lines 9,10,11,12*: The final result can be written as $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3 + m' \cdot m''$ in which the distribution of the term $m' \cdot m''$ depends on m. However, we rely on the security of ring-LWE encryption (i.e., m cannot be recovered from $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3 + \mathsf{Encode}(m)$) to mask it. In particular, $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3$ provides enough randomness to hide $m' \cdot m''$. The security properties of Lines 9-11 are similar given that they only include parts of the total product $(m'_1 + m'_2) \cdot (m''_1 + m''_2)$.
- *Lines 13,14*: Each of the shares m', m'' is uniform distributed in $\{0,1\}^{n/4}$ and independent of m. Therefore, the encoding can be simulated by an adversary without knowledge of a sensitive variable.
- *Lines 15,16*: The final result $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3 + \mathsf{Encode}(m)$ does not leak information about m based on the security properties of the ring-LWE encryption. Furthermore, $m' \cdot m'' + \mathsf{Encode}(m')$ does not leak more information about m than $\mathsf{Encode}(m)$. Thus, the intermediate result of Line 15 also does not leak information about m.

As shown above, the distribution of every intermediate variable of Algorithm 5 is independent of the sensitive variables $\mathbf{r}_2$ and m. Therefore, it is not possible for an attacker, who can probe one value, to derive sensitive information.

3.4 Hiding

To increase the level of noise and make higher-order attacks harder, we do not only rely masking to thwart side-channel analysis but also include hiding schemes. We therefore applied the aforementioned blinding technique from [56] to our implementation by multiplying the coefficients of $\tilde{\mathbf{c}}_1$ by a random value $a \in [0, q-1]$ and the coefficients of $\tilde{\mathbf{r}}_2$ by a different random value $b \in [0, q-1]$.

The coefficients of $\tilde{\mathbf{c}}_2$ are multiplied by $ab \mod q$ as they get added to the product $(a \cdot \tilde{\mathbf{c}}_1) \cdot (b \cdot \tilde{\mathbf{r}}_2)$. The mask is then removed by multiplying all coefficients by $(ab)^{-1} \mod q$. Due to the linearity of the NTT it is possible to remove the mask after the result has been transformed back to the time domain. To introduce even more noise we used shuffling to execute linear operations during the decryption in a randomized order. To achieve this we shuffled the list of coefficients by using the Fisher-Yates algorithm [24]. A similar countermeasure has been implemented by Pessl [45] to avoid cache-timing attacks. For every run of the decryption, the list of coefficients gets shuffled again.

3.5 Fault Resistance

Fault injection is an additional physical threat for embedded systems. Previous publications have analysed the vulnerability of lattice-based signature schemes against fault attacks [9,23] and found several attacks. In the following, we present the – to our knowledge – first vulnerability analysis of ring-LWE.

In our analysis, we assume that the adversary targets the secret key $\mathbf{r}_2$ during the CCA2-secure decryption. Without CCA2-security, a fault attack is not necessary to recover the secret key as described before. Given that the CCA2-conversion includes a validity check at the end, it inherently includes resistance against certain faults. In particular, any fault injected in the ring-LWE decryption that changes the output of the ring-LWE decryption module will be detected by the re-encryption at the end[5]. Therefore, to perform any type of fault attack, it is required to inject another fault into the input of H (to change the seed) or the ring-LWE encryption with the goal of passing the validity check. Depending on the capabilities of the fault attacker, this approach can be very complex. Therefore, in most cases it is easier to directly inject the fault in the validity check itself, e.g., by skipping instructions.

Furthermore, due to the construction of our scheme the attacker does not have direct access to the output of the ring-LWE decryption module. Instead the output is defined as

$$\begin{aligned} out &= c_3 \oplus G(m) \\ &= (M \oplus G(m)) \oplus G(m) = M \end{aligned}$$

where M is the message and m the output of the ring-LWE decryption. Assuming that the attacker knows M and $G(m)$, the faulty output is

$$out_F = c_3 \oplus G(m_F)$$

and therefore the only novel information the attack can access is $G(m_F)$. Based on the pre-image resistance of G, it is not easily possible to compute m_F from $G(m_F)$ for arbitrary m_F. This poses another difficulty for the fault attacker, as

[5] There are no two distinct outputs of the decryption that can be valid at the same time

it is necessary to skip the computation of G to perform attacks that target the output of the decryption assuming m_F is uniformly distributed in $\{0,1\}^{n/4}$.

However, it is possible to overcome this limitation. A much simpler attack relies on the basic vulnerability of ring-LWE decryption to chosen ciphertexts. By skipping the validity check at the end, the attacker effectively removes the CCA2-security. Meaning an attacker can send chosen ciphertexts and receive the output

$$out = c_3 \oplus G(m_C)$$

where m_C denotes the output of the decryption for the chosen ciphertext. Even though, we noted above that G provides pre-image resistance, this does not apply when m_C has only a very limited value space. Then it is possible to compute $G(m_C)$ for all possible m_C and use *out* to check for the correct one. For our implementation, an attacker can target each coefficient of the secret key polynomial separately by choosing $\tilde{\mathbf{c}}_1$ as a polynomial with all coefficients but one set to zero. Therefore, an attacker needs to compute only q different values for m_C. Overall, this attack only requires the injection of one fault at the end to skip the validity check of the CCA2-conversion.

To sum up, our implementation provides basic resistance against simple faults in the ring-LWE decryption. However, if the attacker can skip the validity check, it becomes very easy to extract the secret key. Therefore, to increase the resistance against physical attacks, additional countermeasures need to be included to protect this final check.

3.6 Higher-Order Masking

As mentioned before, it is not sufficient to analyze the security against d probes for each modules separately to show the security of the full decryption. Nevertheless, we now briefly discuss the possible extension of our masked modules to higher-orders.

For the first part of $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{dec}}$, each share is processed separately and therefore extending the security to more probes is trivial. A designer only needs to increase the number of shares and process them similar to α''. However, the addition of the shares in the second part requires special care, e.g., order of operation, to obtain higher-order security. Nevertheless, both extensions especially the first half are very efficient due to the linear increase with the order in the number of operations.

For G and H we refer to [8] for a discussion of higher-order resistance. We want to note that for this module the efficiency strongly depends on the chosen function. KECCAK is efficient for first-order security. However, for higher orders a different function might be better suited.

The module $\mathsf{RLWE.CPA}^{\mathsf{NTT}}_{\mathsf{enc}}$requires additional care. With two probes, it is easily possible to probe $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3$ and the final result $\mathbf{p} \cdot \mathbf{e}_1 + \mathbf{e}_3 + \mathsf{Encode}(m)$ to derive the sensitive value $\mathsf{Encode}(m)$. Thus, our current approach cannot be simply adapted to higher orders. Nevertheless, the inclusion of additional

randomness should enable higher-order masked encryption rather easily which we leave as part of future work.

Maintaining an additional share means that we need extra temporary storage for one polynomial that stores the third share of the key (2048 bytes) and one additional KECCAK state (200 bytes). Our target platform provides 192 kbytes of RAM and therefore we expect that a higher-order masking scheme still fits onto the microcontroller.

4 Implementation

To evaluate the performance of the CCA2-conversion and our masking scheme, we implemented the constructions on an ARM Cortex-M4F. Our evaluation platform is an STM32F4 DISCOVERY board with 1 Mbyte of flash memory, 192 Kbyte of RAM, a floating-point unit (FPU), and a true random number generator (TRNG). To make sure our implementations has a constant or secret-independent running time, we implemented critical components in assembly language. Furthermore, to prevent cache timing attacks we disabled the cache of the on-board flash memory by setting the `DCEN` bit of the `FLASH_ACR` register to zero.

We use SHAKE-128 as instantiation for both hash functions H and G. SHAKE-128 is standardized in FIPS-202 [18] and an extendable-output function that is based on the KECCAK algorithm [7]. As the hashing plays a minor role in terms of performance, we selected the readable KECCAK implementation by Saarinen [55] as basis for our implementation as it allowed us to easily implement side-channel countermeasures.

There are a lot of algorithms that sample Gaussian distributed values from uniformly distributed ones. But most of them do not have a constant running time. For instance, the Bernoulli approach from [19] leaks timing information and is thus irrelevant for our implementation. Therefore, we decided to implement the binomial sampler from [1] with $k = 11$ as it is simple and has a constant execution time. It basically counts the Hamming weight of two 11-bit vectors. We realize the computation of the Hamming weight via look-up tables. To this end, we store a table with 64 entries that stores the Hamming weight of all 6-bit values. By looking up the upper five bit and the lower six bit of the 11-bit vector, we can compute the Hamming weight in a fast and efficient way. Using only one table look-up for the entire bit vector would require $2^{11} = 2048$ table entries what we decided would be excessive for a microcontroller application.

To sample the necessary randomness, we implemented a pseudo-random number generator (PRNG) that gets initialized by 256 bits. For encryption we generate a secret seed from the on-board TRNG and then use a PRNG to generate Gaussian noise. As we have to perform a re-encryption during the decryption that must sample the exact same values, we cannot use the TRNG for this purpose but have to initialize the PRNG with the same seed. As our evaluation board does not feature a crypto-coprocessor, we use a software implementation of the ChaCha8 stream cipher by Daniel J. Bernstein as PRNG [6].

For the implementation of polynomial arithmetic we need fast and constant-time modular reduction to rule out SPA and remote timing attacks. As a consequence, the implementation of the NTT and especially the three-instruction modular reduction from [16] is not suitable. It uses the `DIV` instruction, which has a data-dependent variable execution time that can reach from 2 to 12 clock cycles.

Therefore, we implemented a Barrett reduction [5] using the FPU of the Cortex-M4F. Instead of using the `DIV` instruction, we perform a floating-point multiplication of the input by $1/q$ to find the number of necessary subtractions of q. As the floating-point multiplication takes only one clock cycle, this reduction technique is timing-independent and takes 6 clock cycles, of which 4 are only used for the transition between integer and floating point representation. The downside of this approach is that the result might not be fully reduced as the floating point representation is limited to a certain precision. However, this does not pose a problem as we are placing 14-bit numbers in 32-bit registers and subsequent reductions can easily handle numbers that are greater than q. De Clercq et al. [16] also present an optimized implementation of the NTT. They implement the NTT in assembly and also proposed an optimized memory access scheme. Their idea is to store two coefficients in one data word and being able to load/store both coefficients with the same instruction. Alkim et al. implemented the NTT as well as reported in [2]. By combining a Montgomery reduction with Barrett reduction, their NTT is considerably faster than the one from [16] and most importantly also has a constant execution time. We therefore embedded the NTT from [2] into our implementation.

For our constant-time implementation of the decoder, we need to compute the absolute of a coefficient as the input is some $x \in [-q/2, q/2]$. We realize this computation by first extracting the sign of the coefficient using the `SXTH` instruction that outputs 0 for positive values and -1 for negative ones. Then we multiply this value by 2 and add 1 to get -1 for negative values and 1 for positive ones. Notice that the last two steps can be executed in a single cycle using the `MLA` instruction. We then multiply the value with its sign to get the absolute value in 3 clock cycles. After that we sum up four coefficients and check whether the result is greater or equal to q or not and return the single bit output of this function.

For our implementation of the blinding countermeasure, we need to compute the inverse of the product of the blinding values $(ab)^{-1} \mod q$. To realize this inversion efficiently, we used an addition chain to compute $(ab)^{q-2} \mod q$ what equals $(ab)^{-1} \mod q$ according to Fermats little theorem. To shuffle the list of indices of the polynomials and therefore change the order of the computations according to the Fisher-Yares algorithm [24], we run a loop with n steps and in each step, we exchange the index at the position of the loop counter with a random one. We apply this shuffling to all point-wise multiplications and additions since a side-channel attack will most likely target these operations for an attack.

5 Side-Channel Evaluation

Even though we provide proofs for most of our modules against one probe, practical first-order side-channel security is not automatically implied by that. Implementation errors can still negatively affect the resistance due to effects that are not included in the model [4]. Therefore, to completely evaluate the security of our masked implementation, we performed several side-channel experiments. Since our aim is to show first-order resistance, we rely on the commonly-used t-test leakage detection methodology initially proposed in [15,27]. In particular, we perform the non-specific *fixed vs. random* t-test. To this end, we take two types of measurements. One with fixed input and one with random input. The t-statistic t is computed as

$$t = \frac{\mu_F - \mu_R}{\sqrt{\frac{\sigma_F^2}{n_F} + \frac{\sigma_R^2}{n_R}}}$$

where μ_F, σ_F^2, and n_F (resp. μ_R, σ_R^2, and n_R) denote the mean, variance, and number of measurements set with fixed input (resp. random input). If the value exceeds the threshold $|t| > 4.5$, the test has detected leakage. In the following, we performed the test at first and second order. For bivariate second-order evaluation, we relied on the optimal centered product [48,60] as the combination function and tested every possible combination of all sample points of the traces. For more information, we refer the interested reader to further literature related to this side-channel evaluation methodology [57].

We use a PicoScope 5203 with a sample rate of 125 MS/s to measure the power consumption over the STM32F4 Discovery board. To increase the quality of the measurements, we reduce the internal clock to 12 MHz and remove some capacities. The communication with the board is done over USART as the on-board USB interface causes additional noise in the power traces. Since the whole masked decryption requires an extremely high number of clock cycles, we cannot easily perform a bivariate evaluation with our proposed method. Instead, we split the practical evaluation into the modules similar to the theoretical evaluation of Section 3.3. For first-order evaluation this does not affect the evaluation coverage, since each sample is considered separately. However, for the bivariate second-order test we do not cover the scenario of two probes in different modules. Nevertheless, our goal is to show the existence of second-order leakage to verify our measurement setup and we already found this for every module separately. Therefore, it is not necessary to further consider the special case of two probes in two different modules. For each module we took 100,000 measurements and performed the aforementioned tests. To further speed up the second-order evaluation, we adjusted the module to only process a small number of coefficients. This helps to limit the total number of sample points. We measured the computation of the butterfly during the NTT for two coefficients, the addition of the two shares during the masked re-encryption as described in 3.3 for one coefficient, the remasking and decoding as described in Section 3.3

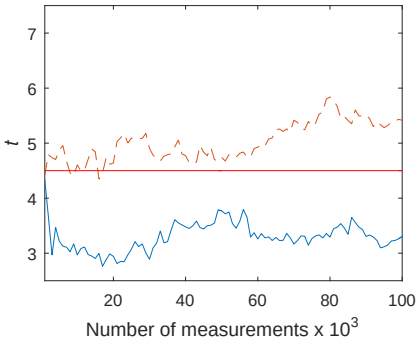
for four coefficients (that encode one bit), the masked χ-step of KECCAK for five bytes, and point-wise multiplication and addition for two coefficients.

Figure 4 depicts the results for each module. The lower (resp. upper) curve shows the maximum absolute value of the first-order (resp. second-order) test as a function of the total number of measurements considered in the evaluation. It is noticeable that there is indeed no first-order leakage up to 100,000 traces. There is also no obvious increase of the t-values. Thus, the implementation is first-order secure as expected. Additionally, the second-order evaluation shows leakage early on for every module and displays an upward trend with higher number of measurements. This is also expected given that we implemented first-order masking.

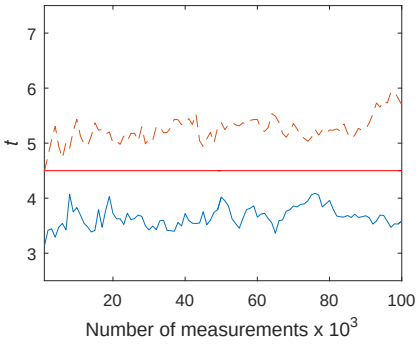
However, it should be noted that even though our evaluations clearly show second-order leakage, performing an actual second-order attack on the masked implementation might not be trivial. For one, there is the aforementioned issue of the extreme high number of sample points which makes our naive combination approach infeasible. Instead, more sophisticated point of interest detection mechanisms need to be utilized to reduce the number of considered sample pairs [21,50] which further increases the complexity of the attack. Another aspect which is briefly mentioned in Section 3.4 is the mixture of masking with hiding countermeasures. Higher-order attacks are very sensitive to the measurement noise. Therefore, to increase the higher-order security it is advised to include one of the discussed hiding countermeasures. In one recent example, this increased the practical resistance more than implementing a higher-order masking scheme [40]. Furthermore, if only the start of the ring-LWE decryption is targeted, a designer can rely on the linear masking property to increase the number of shares significantly. However, including hiding countermeasures prevents us from evaluating only simplified versions of the modules, since the efficiency strongly increases with the number of coefficients. Therefore, we did not measure a masked design with hiding, since the second-order evaluation would not be feasible (the first-order test would give similar results to Figure 4).

6 Results and Comparison

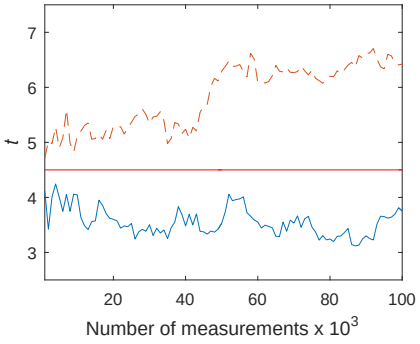
We evaluate the performance of our implementation using Keil μVision V5.17 and use `-O3` optimization for compiling. We took special care that no side-channel leakage was induced by compiler optimization, e.g. by overwriting one shared value in a register with the second share. Cycle counts are measured using the on-board cycle count register (`DWT_CYCCNT`). We present the cycle counts of our implementation in Table 2. The CCA2-secure encryption takes 1,222,054 cycles which translates to 7.3 milliseconds when operating at a clock frequency of 168 MHz. This is an overhead of 16.7% compared to the ring-LWE encryption without CCA2-conversion. The costs of this operation are divided into two forward NTTs (5.9% each) and one backward NTT (7.4%), three runs of the Gaussian sampler (15.2% each), the execution of both hash functions H and G (6.2%



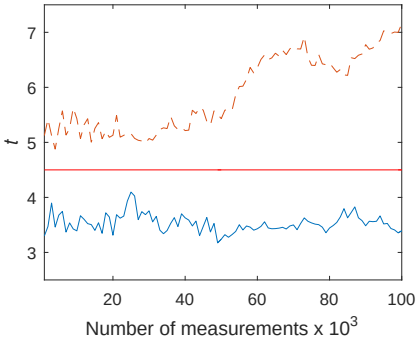
(a) Butterfly



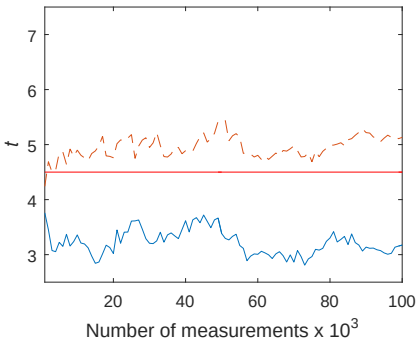
(b) Encryption



(c) Decoding



(d) Hash function



(e) Point-wise multiplication and addition

Fig. 4: Absolute maximum t-values for different modules of our masking scheme. The solid blue line marks the first-order t-values and the dashed red line marks the second-order t-values.

each), and other minor computations like encoding or polynomial addition. The key generation takes 4.7 ms at 168 MHz.

Applying the CCA2-conversion to the decryption causes a much higher overhead due to the necessary re-encryption. In the unmasked case, it requires 8.6 times more cycles and in the masked case 5.8 times more cycles. Thus the masked CCA2-decryption takes 12.3 milliseconds which is an overhead of 46.6% compared to the CCA2-secure decryption without masking. Applying masking to the (core) ring-LWE decryption is expected to increase the cost by at least a factor of two as the most time-consuming operation, the NTT, has to be executed twice in both masking schemes. Additionally, we have to generate 32 bytes of randomness for the re-masking using the TRNG what is also expensive. Masking the hash functions also costs twice as much since most operations have to be performed on both shares.

Table 2: Cycle counts of our implementation on an ARM Cortex-M4F. Cycle counts for sampling are given for a whole polynomial. Our parameters are n = 1024, q = 12289, and k = 11.

Operation	**Cycle counts**	
	unmasked	**masked**
Key Generation ($\mathsf{RLWE.CPA}_{\mathsf{gen}}^{\mathsf{NTT}}$)	781,423	-
CCA2-secure Encryption ($\mathsf{RLWE.CCA}_{\mathsf{enc}}^{\mathsf{NTT}}$)	1,222,054	-
CCA2-secure Decryption ($\mathsf{RLWE.CCA}_{\mathsf{dec}}^{\mathsf{NTT}}$)	1,412,839	2,070,952
CPA-RLWE Encryption ($\mathsf{RLWE.CPA}_{\mathsf{enc}}^{\mathsf{NTT}}$)	1,047,394	1,239,510
CPA-RLWE Decryption ($\mathsf{RLWE.CPA}_{\mathsf{dec}}^{\mathsf{NTT}}$)	163,882	355,444
Hash H (Shake-128)	87,463	201,721
Hash G (Shake-128)	86,943	201,206
NTT	83,906	-
INTT	104,010	-
Uniform Sampling (TRNG)	26,306	-
SampleNoisePoly (PRNG)	214,974	-
PRNG (64 bytes)	2,979	-

Table 3: Cycle counts of our CCA2-secure decryption.

	Masking	
Hiding	**unmasked**	**masked**
no hiding	1,412,839	2,070,952
blinding/shuffling	1,614,553	2,372,242

The results of combining our masking approach of the decryption with additional hiding countermeasures are given in Table 3. The overhead caused by shuffling and blinding is 14.3% without masking and 14.5% with masking. The reason that the overhead for both countermeasures does not simply add up is that the hiding countermeasures also is applied to the second share in the masked case. Thus, the overall running time for our protected decryption is 2,372,242 cycles which leads to 14.1 milliseconds runtime at 168 MHz.

The secret key is one polynomial and therefore requires 2,048 bytes of memory. As the public key consists of two polynomials, it needs twice of much memory (4,096 bytes). The ciphertext consists of two polynomials (c_1, c_2) and a bit string c_3 of 256 bits and therefore has a total size of 4,104 bytes.

6.1 Comparison

Notice that the masked implementation in [52] is a hardware implementation and that [49] does not give any performance numbers and thus we cannot directly compare our results to theirs. Thus we re-implemented their proposals together with a CCA2-conversion to allow a fair comparison. Our results can be seen in Table 4. By doing so, we are also able to give results independent of the speed of the NTT. In this fair comparison, our CCA2-secure decryption needs 43% less cycles than the masked decoder approach from [52] and 29% less cycles than additively homomorphic masking [49]. Additively homomorphic masking takes more than twice as many cycles in comparison with the unmasked implementation as there are now two encryption runs (one re-encryption for the CCA2 security and one additional encryption for the masking scheme) during the decryption. Additionally, the hashing and the re-encryption also have to be performed in a masked fashion. The [52] approach suffers from the masked decoder that has to be evaluated up to 16 times. It is also worth mentioning that encoding one message bit into four coefficients is much more complex when using the masked decoder approach as we no longer have $4^2 = 16$ possible combinations of values hitting quadrants but $4^{2 \cdot 4} = 256$ combinations. Thus, for the evaluation of the masked decoding approach, we encode one message bit into only one coefficient even though this leads to a higher failure probability of the scheme. Similarly, the additively homomorphic masking also inherently increases the failure probability. Thus, the security level of the scheme is actually lower when one of these masking schemes is applied.

To measure the dynamic memory consumption we used the callgraph feature of the Keil IDE. Our approach as well as the masked decoder approach from [52] needs seven temporary polynomials and each polynomials needs 2,048 bytes of memory. Because of the second encryption the additively homomorphic masking from [49] needs another two polynomials. Additionally, each approach requires 400 bytes to store the two shares of the state of the hash function.

Table 4: Cycle counts and dynamic memory consumption of our CCA2-secure decryption.

Masking Scheme	Cycle counts	Dynamic memory
Our scheme	2,070,952	15,284 bytes
Masked decoder [52]	3,661,431	15,412 bytes
Additively homomorphic masking [49]	2,931,411	19,380 bytes

7 Future Work

In this work we have used the straight-forward Fujisaki-Okamoto [26] CCA2 transformation as a tool to make the side-channel analysis meaningful and optimization would certainly improve the overall performance. In [44] Peikert already discusses some of the issues that arise from the instantiation with ring-LWE. Still, an avenue for future research is certainly the question whether this transformation can be tuned for the instantiation with RLWE.CPA or if different and more efficient transformations could be proven secure (maybe also considering the costs of masking). Certainly, a huge performance boost could be gained in case the re-encryption during decryption could be made obsolete (however, this seems to be a hard problem).

Regarding more security we explained in Section 3.6 countermeasures against higher-order attacks and what challenges arise when using the t-test to check for leakage in combination with hiding. Future work is to further explore protection and exploitation of higher-order leakages, their practicality, and the performance of implementations. Additionally, we plan to carry out actual attacks against our implementation as a complement of our leakage assessment based on the t-test. Moreover, our masking scheme (and also the CCA transformation) could also be applied to a standard lattice-based encryption scheme, e.g., the one given in [36] or Frodo [12]. For Frodo the challenge would be to obtain a negligible error probability to make the CCA2 conversions applicable. However, the performance of a side-channel protected standard lattice-based scheme on microcontrollers or FPGAs would be interesting for a comparison with our ideal lattice-based approach. In addition, in this work we have not protected the key generation but some use-cases would also require a key generation algorithm that is protected against side-channel and fault attacks. Moreover, we plan to explore FPGA implementations of our masking scheme and the corresponding evaluations.

Acknowledgement

The research in this work was supported in part by the DFG Research Training Group GRK 1817/1 and by the European Unions Horizon 2020 program under project number 645622 PQCRYPTO and 644729 SAFEcrypto.

References

1. Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 327–343. USENIX Association, 2016. 2, 3, 4, 6, 7, 8, 18, 32
2. Erdem Alkim, Philipp Jakubeit, and Peter Schwabe. A new hope on ARM Cortex-M. In Claude Carlet, Anwar Hasan, and Vishal Saraswat, editors, *Security, Privacy, and Advanced Cryptography Engineering*, Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2016 (to appear). Document ID: c7a82d41d39c535fd09ca1b032ebca1b, http://cryptojedi.org/papers/#newhopearm. 19
3. AC Atici, Lejla Batina, Benedikt Gierlichs, and Ingrid Verbauwhede. Power analysis on NTRU implementations for RFIDs: First results. In *The 4th Workshop on RFID Security, July 9th -11th, Budapest*, 2008. 6
4. Josep Balasch, Benedikt Gierlichs, Vincent Grosso, Oscar Reparaz, and François-Xavier Standaert. On the cost of lazy engineering for masked software implementations. In Marc Joye and Amir Moradi, editors, *Smart Card Research and Advanced Applications - 13th International Conference, CARDIS 2014, Paris, France, November 5-7, 2014. Revised Selected Papers*, volume 8968 of *Lecture Notes in Computer Science*, pages 64–81. Springer, 2014. 20
5. Paul Barrett. Implementing the Rivest Shamir and Adleman public key encryption algorithm on a standard digital signal processor. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 311–323. Springer, 1986. 19
6. Daniel J. Bernstein. The ChaCha family of stream ciphers. https://cr.yp.to/chacha.html, 2008. 18
7. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Keccak. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 313–314. Springer, 2013. 13, 18
8. Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. Building power analysis resistant implementations of Keccak. In *Second SHA-3 candidate conference*, volume 142. Citeseer, 2010. 13, 17
9. Nina Bindel, Johannes A. Buchmann, and Juliane Krämer. Lattice-based signature schemes and their sensitivity to fault attacks. *IACR Cryptology ePrint Archive*, 2016:415, 2016. to appear in FDTC'16. 16
10. Ahmad Boorghany and Rasool Jalili. Implementation and comparison of lattice-based identification protocols on smart cards and microcontrollers. *IACR Cryptology ePrint Archive*, 2014:78, 2014. 2
11. Ahmad Boorghany, Siavash Bayat Sarmadi, and Rasool Jalili. On constrained implementation of lattice-based cryptographic primitives and schemes on smart cards. *IACR Cryptology ePrint Archive*, 2014:514, 2014. 2
12. Joppe W. Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila. Frodo: Take off the ring! Practical, quantum-secure key exchange from LWE. *IACR Cryptology ePrint Archive*, 2016:659, 2016. to appear in CCS'16. 4, 25

13. Joppe W. Bos, Craig Costello, Michael Naehrig, and Douglas 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. 2, 7, 32
14. Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. DRAFT NISTIR 8105, report on post-quantum cryptography, 2015. http://csrc.nist.gov/publications/drafts/nistir-8105/nistir_8105_draft.pdf. 2
15. Jeremy Cooper, Elke Demulder, Gilbert Goodwill, Joshua Jaffe, Gary Kenworthy, and Pankaj Rohatgi. Test Vector Leakage Assessment (TVLA) Methodology in Practice. International Cryptographic Module Conference, 2013. 20
16. Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient software implementation of ring-LWE encryption. *IACR Cryptology ePrint Archive*, 2014:725, 2014. 19
17. Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient software implementation of ring-LWE encryption. In Wolfgang Nebel and David Atienza, editors, *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition, DATE 2015, Grenoble, France, March 9-13, 2015*, pages 339–344. ACM, 2015. 2
18. PUB DRAFT. Fips 202. SHA-3 standard: Permutation-based hash and extendable-output functions. *Information Technology Laboratory, National Institute of Standards and Technology*, 2015. See http://csrc.nist.gov/publications/drafts/fips-202/fips_202_draft.pdf. 18
19. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. *IACR Cryptology ePrint Archive*, 2013:383, 2013. Full version of [20]. 18
20. Léo Ducas, Alain Durmus, Tancrède Lepoint, and Vadim Lyubashevsky. Lattice signatures and bimodal Gaussians. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in Computer Science*, pages 40–56. Springer, 2013. Full version available at http://eprint.iacr.org/2013/383.pdf. 27
21. François Durvaux, François-Xavier Standaert, Nicolas Veyrat-Charvillon, Jean-Baptiste Mairy, and Yves Deville. Efficient selection of time samples for higher-order DPA with projection pursuits. In *Constructive Side-Channel Analysis and Secure Design - 6th International Workshop, COSADE 2015, Berlin, Germany, April 13-14, 2015. Revised Selected Papers*, pages 34–50, 2015. 21
22. Cynthia Dwork, Moni Naor, and Omer Reingold. Immunizing encryption schemes from decryption errors. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 342–360. Springer, 2004. 2
23. Thomas Espitau, Pierre-Alain Fouque, Benoît Gérard, and Mehdi Tibouchi. Loop abort faults on lattice-based Fiat-Shamir & hash'n sign signatures. *IACR Cryptology ePrint Archive*, 2016:449, 2016. 16
24. Ronald Aylmer Fisher, Frank Yates, et al. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.*, (5th rev. ed), 1957. 16, 19
25. Scott R. Fluhrer. Cryptanalysis of ring-LWE based key exchange with key share reuse. *IACR Cryptology ePrint Archive*, 2016:85, 2016. 2, 5

26. Eiichiro Fujisaki and Tatsuaki Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In Michael J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999. 2, 6, 7, 25
27. G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi. A testing methodology for side channel resistance validation. In *NIST non-invasive attack testing workshop*, 2011. 3, 20
28. Norman Göttert, Thomas Feller, Michael Schneider, Johannes A. Buchmann, and Sorin A. Huss. On the design of hardware building blocks for modern lattice-based encryption schemes. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 512–529. Springer, 2012. 4, 7, 32
29. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 212–219. ACM, 1996. 7
30. Jeff Hoffstein, Jill Pipher, John M. Schanck, Joseph H. Silverman, William Whyte, and Zhenfei Zhang. Choosing parameters for NTRUEncrypt. IACR Cryptology ePrint Archive report 2015/708, 2015. http://eprint.iacr.org/2015/708. 32
31. Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998. 2
32. Nick Howgrave-Graham, Phong Q. Nguyen, David Pointcheval, John Proos, Joseph H. Silverman, Ari Singer, and William Whyte. The impact of decryption failures on the security of NTRU encryption. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 226–246. Springer, 2003. 2
33. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, pages 463–481, 2003. 8
34. Abdel Alim Kamal and Amr M. Youssef. Fault analysis of the NTRUEncrypt cryptosystem. *IEICE Transactions*, 94-A(4):1156–1158, 2011. 6
35. Abdel Alim Kamal and Amr M. Youssef. Strengthening hardware implementations of NTRUEncrypt against fault analysis attacks. *J. Cryptographic Engineering*, 3(4):227–240, 2013. 6
36. Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011. 2, 3, 4, 25
37. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Henri Gilbert, editor, *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, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010. Presentation slides: http://crypto.rd.francetelecom.com/events/eurocrypt2010/talks/slides-ideal-lwe.pdf. 2, 29
38. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings, 2010. Presentation of [37] given by Chris Peikert at Eurocrypt'10. See http://www.cc.gatech.edu/~cpeikert/pubs/slides-ideal-lwe.pdf. 3
39. Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. *IACR Cryptology ePrint Archive*, 2012:230, 2012. Full version of [37]. 3
40. Amir Moradi and Alexander Wild. Assessment of hiding the higher-order leakages in hardware - what are the achievements versus overheads? In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 453–474, 2015. 21
41. LEE Mun-Kyu, Jeong Eun Song, and HAN Dong-Guk. Countermeasures against power analysis attacks for the NTRU public key cryptosystem. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 93(1):153–163, 2010. 6
42. National Security Agency. NSA suite B cryptography. https://www.nsa.gov/ia/programs/suiteb_cryptography/, Updated on August 19, 2015. 2
43. NIST. Draft: Proposed submission requirements and evaluation criteria for the post-quantum cryptography standardization process. *National Institute of Standards and Technology*, August 2016. See https://federalregister.gov/a/2016-18150. 2
44. Chris Peikert. Lattice cryptography for the Internet. In Michele Mosca, editor, *Post-Quantum Cryptography - 6th International Workshop, PQCrypto 2014, Waterloo, ON, Canada, October 1-3, 2014. Proceedings*, volume 8772 of *Lecture Notes in Computer Science*, pages 197–219. Springer, 2014. 2, 7, 11, 13, 25
45. Peter Pessl. Analyzing the shuffling side-channel countermeasure for lattice-based signatures. In *Progress in Cryptology–INDOCRYPT 2016: 17th International Conference on Cryptology in India, Kolkata, India, December 11-14, 2016, Proceedings*, pages 153–170. Springer, 2016. 16
46. Thomas Pöppelmann, Léo Ducas, and Tim Güneysu. Enhanced lattice-based signatures on reconfigurable hardware. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2014. 8
47. Thomas Pöppelmann and Tim Güneysu. Towards practical lattice-based public-key encryption on reconfigurable hardware. In Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors, *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*, pages 68–85. Springer, 2013. 3, 7, 8
48. Emmanuel Prouff, Matthieu Rivain, and Régis Bevan. Statistical analysis of second order differential power analysis. *IEEE Trans. Computers*, 58(6):799–811, 2009. 20
49. Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Additively homomorphic ring-LWE masking. In Tsuyoshi Takagi, editor, *Post-Quantum Cryptography - 7th International Workshop, PQCrypto 2016, Fukuoka, Japan, February 24-26, 2016, Proceedings*, volume 9606 of *Lecture Notes in Computer Science*, pages 233–244. Springer, 2016. 5, 10, 24, 25

50. Oscar Reparaz, Benedikt Gierlichs, and Ingrid Verbauwhede. Selecting time samples for multivariate DPA attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 155–174, 2012. 21
51. Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptographic Engineering*, 6(2):139–153, 2016. 5, 10
52. Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-LWE implementation. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 683–702. Springer, 2015. 5, 10, 24, 25
53. Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-LWE implementation. *IACR Cryptology ePrint Archive*, 2015:724, 2015. 5, 8
54. Sujoy Sinha Roy, Frederik Vercauteren, Nele Mentens, Donald Donglong Chen, and Ingrid Verbauwhede. Compact ring-lwe cryptoprocessor. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems - CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 371–391. Springer, 2014. 2, 3
55. Markku-Juhani O. Saarinen. tiny_sha3. https://github.com/mjosaarinen/tiny_sha3, 2011. 18
56. Markku-Juhani O. Saarinen. Arithmetic coding and blinding countermeasures for ring-LWE. *IACR Cryptology ePrint Archive*, 2016:276, 2016. 6, 8, 15
57. Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 495–513, 2015. 20
58. Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, pages 124–134. IEEE Computer Society, 1994. 1
59. Joseph H. Silverman and William Whyte. Timing attacks on NTRUEncrypt via variation in the number of hash calls. In Masayuki Abe, editor, *Topics in Cryptology - CT-RSA 2007, The Cryptographers' Track at the RSA Conference 2007, San Francisco, CA, USA, February 5-9, 2007, Proceedings*, volume 4377 of *Lecture Notes in Computer Science*, pages 208–224. Springer, 2007. 6
60. François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 112–129, 2010. 20
61. Elena Trichina. Combinational logic design for AES subbyte transformation on masked data. *IACR Cryptology ePrint Archive*, 2003:236, 2003. 14
62. An Wang, Xuexin Zheng, and Zongyue Wang. Power analysis attacks and countermeasures on NTRU-based wireless body area networks. *TIIS*, 7(5):1094–1107, 2013. 6

63. Xuexin Zheng, An Wang, and Wei Wei. First-order collision attack on protected NTRU cryptosystem. *Microprocessors and Microsystems - Embedded Hardware Design*, 37(6-7):601–609, 2013. 6

A Appendix

In Table 5 we provide details security estimations based on the approach and using the script provided in [1]. For the evaluation of the error rate we also used the script from [1] and set the parameters $dim = 1024, q = 12289, k = 11, bound\_CC = 94833.915727, t\_CC = 0.007911055, tau = 13.3208737785$.

Table 5: Security level of various parameters for ring-LWE encryption schemes

Attack	m	b	Known Classical	Known Quantum	Best Plausible
RLWE.CPA [28] $q = 7681$, $n = 256$, $\varsigma \approx 4.5160$					
Primal	347	222	64	58	46
Dual	369	222	64	58	46
RLWE.CPA [28] $q = 12289$, $n = 512$, $\varsigma \approx 4.8591$					
Primal	660	496	145	131	102
Dual	674	494	144	131	102
BCNS proposal [13]: $q = 2^{32} - 1$, $n = 1024$, $\varsigma = 3.192$					
Primal	1062	296	86	78	61
Dual	1055	296	86	78	61
NTRUEncrypt [30]: $q = 2^{12}$, $n = 743$, $\varsigma \approx \sqrt{2/3} \approx 0.8165$					
Primal	613	603	176	159	125
Dual	635	600	175	159	124
JarJar: $q = 12289$, $n = 512$, $\varsigma = \sqrt{12} \approx 3.4641$					
Primal	623	449	131	119	93
Dual	602	448	131	118	92
NewHope: $q = 12289$, $n = 1024$, $\varsigma = \sqrt{8} \approx 2.8284$					
Primal	1100	967	282	256	200
Dual	1099	962	281	255	199
Our Work: $q = 12289$, $n = 1024$, $\varsigma = \sqrt{11/2} \approx 2.3452$					
Primal	1046	922	269	244	191
Dual	1080	917	268	243	190