IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2020, No. 3, pp. 483–507. DOI:10.13154/tches.v2020.i3.483-507

High-Speed Masking for Polynomial Comparison in Lattice-based KEMs

Florian Bache[1], Clara Paglialonga[2], Tobias Oder[1], Tobias Schneider[3] and Tim Güneysu[1,4]

[1] Ruhr-Universität Bochum, Germany, {florian.bache,tobias.oder,tim.gueneysu}@rub.de
[2] Technische Universität Darmstadt, Germany, clara.paglialonga@tu-darmstadt.de
[3] NXP Semiconductors Austria, Gratkorn, Austria[†], tobias.schneider-a7a@rub.de
[4] Deutsches Forschungszentrum für Künstliche Intelligenz, Germany

Abstract. With the NIST post-quantum standardization competition entering the second round, the interest in practical implementation results of the remaining NIST candidates is steadily growing. Especially implementations on embedded devices are often not protected against side-channel attacks, such as differential power analysis. In this regard, the application of countermeasures against side-channel attacks to candidates of the NIST standardization process is still an understudied topic. Our work aims to contribute to the NIST competition by enabling a more realistic judgment of the overhead cost introduced by side-channel countermeasures that are applied to lattice-based KEMs that achieve CCA-security based on the Fujisaki-Okamoto transform. We present a novel higher-order masking scheme that enables an efficient comparison of polynomials as previous techniques based on arithmetic-to-Boolean conversions renders this (generally inexpensive) component extremely expensive in the masked case. Our approach has linear complexity in the number of shares compared to quadratic complexity of previous contributions and it applies to lattice based schemes with prime modulus. It comes with a proof in the probing model and an efficient implementation on an ARM Cortex-M4F microcontroller which was defined as a preferred evaluation platform for embedded implementations by NIST. Our algorithm can be executed in only 1.5-2.2 milliseconds on the target platform (depending on the masking order) and is therefore well suited even for lightweight applications. While in previous work, practical side-channel experiments were conducted using only 5,000 - 100,000 power traces, we confirm the absence of first-order leakage in this work by collecting 1 million power traces and applying the *t*-test methodology.

Keywords: Ideal Lattices · NewHope · Masking · Implementation · ARM Cortex-M

1 Introduction

Implementations on embedded devices often have to consider countermeasures against side-channel attacks such as timing attacks, power analysis, or fault injections. Remedies against timing attacks are usually simpler to realize, however, there are exceptions, like Gaussian samplers) [AJS16,SBG+18,KMRV,KRVV19]. A common countermeasure against power analysis is masking that splits sensitive data into random shares which are assumed difficult to be accessed given a limited attacker and probing model. Splitting the sensitive data into more shares means protection against more powerful attackers.

[†]The majority of the author's contribution was performed while he was with UC Louvain.

Received: 2020-01-15 Accepted: 2019-03-15 Published: 2020-06-19

In response to the looming threat of quantum computers rendering most contemporary public key crypto obsolete, NIST started a standardization competition for post-quantum cryptography that has entered the second round in early 2019. NIST explicitly mentions the side-channel security of schemes as one of their evaluation criteria [NIS16]. Among the remaining 17 encryption schemes and key encapsulation mechanisms (KEMs), there are 9 lattice-based KEMs, which constitute the largest group of KEMs. Many of these schemes use the Fujisaki-Okamoto transform [FO99] or a variant [TU16] to achieve CCA-security. One crucial component of the transform is the comparison of outputs. The comparison component has been widely disregarded in side-channel analysis of post-quantum cryptography so far as the performance cost for this component is negligible in the unmasked case. In this work, we show that when masking is applied using a conventional approach, the comparison step actually amounts for a sizable number of clock cycles in the execution of the KEM. We therefore propose a novel and highly efficient higher-order masked algorithm for the comparison component that solves the aforementioned problem and works on lattice based schemes with prime modulus. Our solution significantly outperforms the previous approach by one to two orders of magnitude, depending on the masking order. Our masking scheme is applicable for the NIST post-quantum standardization candidates `NewHope` [ADPS16], `Kyber` [BDK+18], and `LAC` [LLZ+].

1.1 Related Work

Applying masking to schemes based on the Ring-Learning with Errors (Ring-LWE) problem as countermeasure to power analysis has been first analyzed by Reparaz et al. in [RRVV15, RdCR+16, RRdC+16]. Their masking proposals however only protect schemes providing security against chosen-plaintext attackers (CPA). In a CPA-secure Ring-LWE-based scheme, no comparison is required. Previous approaches for masked polynomial comparison for lattice-based cryptography have been proposed by Oder et al. [OSPG18], Barthe et al. [BBE+18], and Migliore et al. [MGTF19]. The approach from [OSPG18] also targets Ring-LWE-based schemes and explicitly takes CCA-security into account, but is optimized for first-order masking only and cannot be adapted to higher orders. The approaches from [BBE+18] and [MGTF19] are masking schemes for lattice-based digital signatures. The comparison algorithms in both works are alike and both are based on conversion algorithms that transform arithmetic shares to Boolean shares. These transformations are computationally extremely expensive and have a quadratic complexity regarding the number of shares [SPOG19]. This makes the polynomial comparison an unnecessary overhead in masked implementations of lattice-based cryptography. Several contributions performed practical experiments to verify the side-channel security of their proposals. However, the extensiveness of these experiments is lower compared to this paper with only 5k traces in [RdCR+16], 10k traces in [MGTF19], and 100k in [OSPG18].

Our approach relies on reducing the comparison of multiple values to one comparison of a function these values. Similar approaches have been applied in other contexts than masking, e.g., batch verification of the equality of logarithms [APB+04]. However, to the best of our knowledge, our approach is the first to utilize this basic principle to speed up the secure comparison of masked values.

1.2 Contribution

In this work, we present the first higher-order masking algorithm for polynomial comparison that is specifically optimized for usage in lattice-based KEMs. The contributions of this paper are as follows:

- We show that the adaption of other approaches (from lattice-based signatures) introduces a significant computational overhead. That is because the A2B conversions used in this naive approach are expensive and have a quadratic complexity in the masking order.

- We present a new algorithm that has a simpler structure, better asymptotic run time, less constant overhead, and is still provable secure for higher masking orders. Our approach exploits that the comparison steps reduces the entire information contained in a polynomial to just a single bit.
- A theoretical proof in the $t-$probing model of the new scheme is provided. In particular we show that it satisfies the stronger property of $t-\mathsf{NI}$.
- Our developed ARM Cortex-M4F assembly implementation shows the superior practical performance of our approach as it is able to reduce the computational cost of the comparison step by a factor of 16 for first-order masking security and even more for higher orders.
- The side-channel security of our approach is evaluated in practical experiments with 1 million power traces that is significantly more than the number of power traces used in previous work on power analysis attacks on lattice-based cryptography.

Our algorithm can be applied to multiple lattice-based KEMs that are submitted to the NIST post-quantum standardization competition and with the publication of this work, we will make the source code of our microcontroller implementation publicly available.

1.3 Outline

This paper is organized as follows: In Section 2, the necessary theoretical background that is crucial for the understanding of this paper is introduced. After that, we present our novel approach for higher-order masked polynomial comparison in Section 3. The implementation of this algorithm is discussed in Section 4. In Section 5, we present the results of our evaluation of the algorithm. Finally, we draw a conclusion in Section 6.

2 Preliminaries

In this section, we introduce the necessary theoretical background. This includes a description of the `NewHope` KEM, a definition of side-channel security, and the masking countermeasure.

2.1 Notation

In the rest of the paper, we denote with q the modulus in the lattice-based scheme, which in this work is always a prime number. We will indicate with k the number of coefficients in a polynomial. Moreover, the lower case will be used for Boolean encoding and the upper one for arithmetic encoding. To refer to the i-th coefficient of a polynomial A, we will write A_i, while we write A_i^j to refer to the j-th share of the i-th coefficient of the polynomial.

2.2 The Basic `NewHope` Scheme

We chose `NewHope` as case study to evaluate our comparison algorithm. There are two variants of `NewHope`, one that is secure against chosen-plaintext attackers (CPA-secure) and one that is secure against chosen-ciphertext attackers (CCA-secure). Even though the comparison is only necessary in the CCA-secure scheme, we still review the basic construction of the CPA-secure `NewHope` as it is necessary for the understanding of how our comparison algorithm works. For the sake of simplicity, we omit a lot of details concerning efficiency, like the application of the number-theoretic transform or compression of polynomials, in the description of the key generation, encryption, and decryption in Algorithms 1,2, and 3. The most important parameters of the scheme are the modulus q, the lattice dimension k, and the sampling parameter κ.

Algorithm 1 `NewHope CPA.Keygen`

```
Input: Public constant polynomial $a$
Output: Public key $pk$ and secret key $sk$
1: $seed \xleftarrow{\$} \{0,\dots,255\}^{32}$
2: $r_1, r_2 \leftarrow \texttt{SampleBinomial}(seed)$
3: $p \leftarrow r_1 + ar_2$
4: return $pk = (a, p), sk = r_2$
```

Algorithm 2 `NewHope CPA.Encryption`

```
Input: Public key $pk$, message $\mu \in \{0,\dots,255\}^{32}, coin \in \{0,\dots,255\}^{32}$
Output: Ciphertext $A = (c_1, c_2)$
1: $e_1, e_2, e_3 \leftarrow \texttt{SampleBinomial}(coin)$
2: $c_1 \leftarrow ae_1 + e_2$
3: $c_2 \leftarrow pe_1 + e_3 + \texttt{Encode}(\mu)$
4: return $(c_1, c_2)$
```

Algorithm 3 `NewHope CPA.Decryption`

```
Input: Secret key $sk = r_2$, ciphertext $A = (c_1, c_2)$
Output: Message $\mu$
1: return $\texttt{Decode}(c_2 - c_1 r_2)$
```

The algorithm `SampleBinomial()` uses a PRNG that is seeded by a random bit string. The PRNG sends two random bit strings of length κ bits to the sampling algorithm. The sampler then calculates the Hamming weight of both bit strings and subtracts the Hamming weights. The result is a binomial distributed random number. `SampleBinomial()` outputs an entire polynomial with binomially distributed coefficients. The algorithm `Encode()` transforms the input message into a polynomial. Each bit of the message is encoded into four coefficients. This is done by setting these four coefficients to $\{0,0,0,0\}$ if the message bit is 0. If the message bit is 1, the respective coefficients are set to $\{\lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor, \lfloor \frac{q}{2} \rfloor\}$.

Many lattice-based KEMs use a very similar base construction. The main difference to `Kyber` is that the security of `Kyber` is based on the Module-LWE problem while the security of `NewHope` is based on the Ring-LWE problem. The security of `Frodo` is based on the plain LWE problem. This implies that the parameters of `Frodo` are much bigger, but the underlying lattice is actually random and not structured as in the case of the Ring-LWE and the Module-LWE problem. `Saber` and `Round5` are based on the the (Ring-) Learning with Rounding (LWR) problem that is related to LWE.

2.3 Fujisaki-Okamoto Transform as Applied to `NewHope`

As many lattice-based KEMs that base their security on the LWE or LWR problems, the original `NewHope` proposal [ADPS16] is only secure against chosen-plaintext attackers. The Fujisaki-Okamoto transform [FO99] is a standard conversion algorithm that many designers of post-quantum cryptography use to turn a CPA-secure scheme into a CCA-secure one. The idea behind the Fujisaki-Okamoto transform is that a re-encryption in the decryption

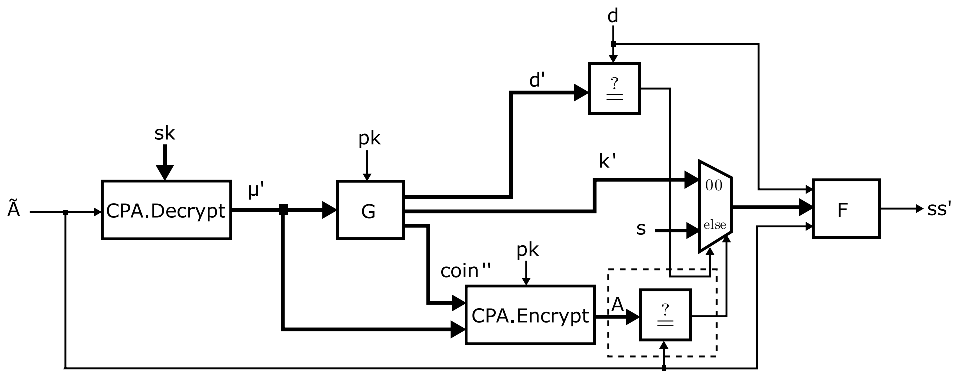


Figure 1: IND-CCA-secure variant of the `NewHope` KEM. The dashed line highlights the comparison component that is subject of this work. Bold lines indicate masked data.

detects whether the input ciphertext was valid or not. Therefore the re-encrypted ciphertext will be compared to the original input and if this comparison fails, a random value will be output by the algorithm. Figure 1 depicts how the Fujisaki-Okamoto transform is applied to `NewHope` in the specification of the NIST post-quantum standardization [AAB+]. The ciphertext $\tilde{A}$ is initially decrypted to μ' using the secret key sk and then used as input to the random oracle G. The random oracle outputs a) the seed for the PRNG of the re-encryption coin", b) the (symmetric) key material k', and c) and additional 256-bit value d'. After the re-encryption of μ' using the public key and the PRNG seed, the resulting ciphertext A is compared to the input $\tilde{A}$. Only if this comparison and the comparison of the bit string d' with the input d is true, the key material k' is hashed together with d and $\tilde{A}$ to receive the final symmetric key. Otherwise, k' is replaced by a random value.

As noted in [OSPG18], all intermediate values depending on the output of the initial decryption are sensitive and need to be masked as indicated in Figure 1 by bold lines. In particular, this also includes the output of the re-encryption A which is compared to the initial ciphertext $\tilde{A}$. This comparison step is linear in the number of coefficients k and therefore negligible regarding the performance for the unmasked case. When masking is applied to the comparison of these polynomials, the performance cost gets significantly larger. Therefore, this work analyzes how an efficient masking scheme can be applied to this comparison step.

2.4 Security against Side Channel Attacks

In order to secure our scheme against side-channel attacks, we adopt the countermeasure of masking, which consists in randomizing the computation of the targeted circuit such that if a bounded amount of information is leaked during the execution, this is statistically independent of the sensitive variable. To this scope, every sensitive variable S is encoded into n shares S^j, such that the sum of all of them gives the original masked variable S but a collection of less than n shares does not allow to reconstruct S. Ishai et al. in [ISW03] formalized for the first time the concept of masking and introduced the so-called *t-probing model*, where an adversary is allowed to access up to t intermediate values in a circuit.

The definition of $t-$probing security requires the existence of a simulator, which can simulate the adversary's view without having access to the sensitive variables, but using only a subset of cardinality at most t of their shares. The first security definition has been enriched later on in [BBD+15] in order to additionally guarantee security also when an algorithm is part of a bigger circuit and ensuing that using the output of a gadget as input to another one does not add sensitive information to the adversary's knowledge. The new security definitions are called $t-$non interference (abbreviated with $t-\mathsf{NI}$) and $t-$strong non interference (abbreviated with $t-\mathsf{SNI}$). Under some circumstances, $t-\mathsf{NI}$ gadgets and $t-\mathsf{SNI}$ ones can be securely composed. Instead, $t-\mathsf{SNI}$ gadgets can always be securely composed with each other. The formal definitions are given below.

Definition 1 ($t-\mathsf{NI}$)**.** A given gadget $\mathcal{G}$ is t-Non-Interfering ($t-\mathsf{NI}$), if every set of t probes on the internal and output values can be simulated by using at most t shares of each input.

Definition 2 ($t-\mathsf{SNI}$)**.** A given gadget $\mathcal{G}$ is t-Strong-Non-Interfering ($t-\mathsf{SNI}$), if every set of t_1 probes on the internal values and t_2 probes on the output values, with $t_1+t_2 \leq t$, can be simulated by using at most t_1 shares of each input.

In particular, the last definition requires that the number of probes on the outputs are independent from the number of the shares needed by the simulation.

In the rest of the paper, n will always refer to the number of shares and t to the security order.

2.5 Arithmetic and Boolean Masking

Masking schemes for cryptographic algorithms can apply different forms of masking. When Boolean masking is applied, the shares have to be combined via the XOR operation to reconstruct the secret value.

$$A = \bigoplus_j A^j = A^1 \oplus A^2 \oplus \ldots \oplus A^n$$

In arithmetic masking, this operation is replaced by modular addition

$$A = \sum_j A^j = A^1 + A^2 + \ldots + A^n \bmod q$$

Some components of lattice-based KEMs are more efficient when masked arithmetically, like polynomial multiplication. Other parts, especially symmetric components as used for PRNGs and XOFs, are more efficient when using Boolean masking. Therefore, we need to be able to switch between both forms of masking. Special conversion algorithms have been developed for Arithmetic-to-boolean (A2B) and Boolean-to-arithmetic (B2A) conversion. For this work, A2B conversions are especially relevant. We therefore review in Algorithm 4 the A2B conversion algorithm from [SPOG19] that is specifically designed for arbitrary moduli. The subroutines `Expand`() and `SecAdd`() are given in Appendix A.

3 Higher-Order Masking of Comparison of Polynomials

In this section, we first explain why previous approaches from lattice-based signature schemes are not suitable for application in lattice-based KEMs. Then, we present a more efficient solution and provide a security proof for the side-channel security of our algorithm.

3.1 Evaluation of Previous Approaches

In [OSPG18], Oder et al. explain why it is necessary to apply masking to the comparison step. In short, a CCA-attacker is able to make predictions about the input to the comparison

Algorithm 4 ConvertA2B [CGV14]

```
Input: $(A^i)_{1\leq i\leq n} \in \mathbb{F}_q$ such that $\sum_i A^i = A \mod q$
Output: $(z^i)_{1\leq i\leq n} \in \mathbb{F}_2$ such that $\bigoplus_i z^i = A$
1: if n=1 then return $A_1$
2: end if
3: $(x^i)_{1\leq i\leq n/2} \leftarrow$ ConvertA2B($(A^i)_{1\leq i\leq n/2}$)
4: $(x'^i)_{1\leq i\leq n/2} \leftarrow$ Expand($(x_i)_{1\leq i\leq n/2}$)
5: $(y^i)_{1\leq i\leq n/2} \leftarrow$ ConvertA2B($(A^i)_{n/2+1\leq i\leq n}$)
6: $(y'^i)_{1\leq i\leq n/2} \leftarrow$ Expand($(y^i)_{1\leq i\leq n/2}$)
7: $(z^i)_{1\leq i\leq n} \leftarrow$ SecAdd($(x'^i)_{1\leq i\leq n}, (y'^i)_{1\leq i\leq n}$)
8: return $(z^i)_{1\leq i\leq n}$
```

and by applying masking to the comparison we prevent the attacker from verifying these predictions. The approach for the comparison of arithmetically masked polynomials from [OSPG18] however only works for first-order masking schemes and is not adaptable to higher-order masking. In [OSPG18] the reported cycle counts also do not explicitly mention the cost of the comparison. The approaches from [BBE+18] and [MGTF19] rely on A2B conversions. They have been proposed for application in lattice-based signature schemes, but an adaptation to lattice-based KEMs is possible and trivial to realize. Due to costly conversions, these comparisons can become a major slow-down factor in implementations of higher-order masked lattice-based schemes [SPOG19]. Therefore, we first analyze the impact that a comparison algorithm based on A2B conversions on the overall performance would have. We give a lower bound for the cost in terms of cycle counts on Cortex-M4 by measuring the necessary cycle counts of an A2B conversion as the most expensive component of the algorithm. Due to the lack of code availability for the implementation described in [BBE+18] and [MGTF19], we developed an A2B implementation ourselves. For our assembly-optimized implementation, we follow the quadratic A2B approach from [SPOG19] that improves upon the cubic A2B algorithm from [BBE+18]. Adjusting the algorithms from [BBE+18] and [MGTF19] to the specific NewHope parameters ($n = 1024$ and $q = 12289$), we measure 4,031 cycles for one first-order masked A2B conversion. In one execution of the CCA-secure NewHope decapsulation, 2048 A2B conversions are necessary for the comparison of both ciphertext polynomials. These 2048 conversions therefore take 8.3 million cycles and would introduce a significant performance overhead. For reference, in [OSPG18] a first-order-only masked implementation of NewHope takes 25.3 million cycles in total. This means that using A2B conversions for a higher-order masked comparison would introduce an unexpected performance overhead. In the following section, we describe a more efficient way that significantly reduces this overhead.

3.2 Our Proposal

The inputs to the new comparison algorithm are one unshared polynomial $\tilde{A} \in \mathbb{F}_q[X]$, that is also the input to the CCA.Decryption algorithm, and one shared polynomial $\mathbf{A} \in \mathbb{F}_q[X]$ that is the result from the re-encryption in CPA.Encrypt. For k coefficients and n shares we have

$$0 \leq i < k : A_i = \sum_{j=1}^{n} A_i^j \mod q.$$

The core idea of our efficient algorithm is to perform the comparison directly on a large set of the coefficients of the two polynomials, instead on each of them individually as in previous schemes. The set of coefficients is chosen in such a way that an attacker could pass the comparison step using a malformed ciphertext only with negligible probability. The k coefficients are distributed into x sets of cardinality $l = \frac{k}{x}$ using index sets I_m (for the sake of simplicity, we assume that $x|k$). The sets I_m are constructed in the following way:

$$0 \leq m < x : I_m := \{m \cdot \frac{k}{x}, \ldots, (m+1) \cdot \frac{k}{x} - 1\}.$$

After the construction of the sets I_m the following x shared sums B_m^j over subsets of coefficients are calculated (all operations are *mod q* and performed share-wise):

$$0 \leq j < n, 0 \leq m < x : B_m^j = \sum_{i \in I_m} \left(A_i^j + r_{1,i}\right) r_{2,i}$$

with $r_{1,i}$ and $r_{2,i}$ being uniformly random numbers $\in_r \mathbb{F}_q$. Then each of the shares is summed up:

$$B_m = \sum_{j=0}^{n-1} B_m^j$$

By computing the sum we get

$$\begin{aligned} B_m &= \sum_{j=0}^{n-1} \left(\sum_{i \in I_m} \left((A_i^j + r_{1,i}) r_{2,i} \right) \right) \\ &= \sum_{i \in I_m} \left(\left(\sum_{j=0}^{n-1} A_i^j + n \cdot r_{1,i} \right) r_{2,i} \right) \\ &= \sum_{i \in I_m} \left((A_i + n \cdot r_{1,i}) \, r_{2,i} \right). \end{aligned}$$

Therefore we can now calculate the sums $\tilde{B}_m$ of the coefficients of the same index sets of the unshared polynomial $\tilde{A}$

$$\forall m \in [0, x-1] : \tilde{B}_m = \sum_{i \in I_m} \left(\left(\tilde{A}_i + n \cdot r_{1,i} \right) r_{2,i} \right)$$

and compare it to the sums B_m. This comparison is performed unmasked using any method, e.g., subtracting the B_m from $\tilde{B}_m$ and checking the zero-Bit.

The comparison returns success if $\tilde{B}_m = B_m$ for every $m \in [0, x-1]$ and fails otherwise. This approach needs $2k$ q-sized words randomness and the performance is linear in k and n. Algorithm 5 shows how to compute the masked sums B_m and $\tilde{B}_m$. In the following, our analysis will focus on this algorithm.

Algorithm 5 `MaskedSum` of m-th set

```
Input: $\mathbf{A_1}, \ldots, \mathbf{A_l} \in \mathbb{F}_q^n$ such that $\sum_j A_i^j = A_i \mod q$, $\tilde{A}_1, \ldots, \tilde{A}_l \in \mathbb{F}_q$
Output: $B_m, \tilde{B}_m \in \mathbb{F}_q$
1: $(B_m^i)_{1 \le i \le n} \leftarrow \mathbf{0}$
2: $B_m \leftarrow 0$
3: $\tilde{B}_m \leftarrow 0$
4: for $i = 1$ to $l$ do
5:     $R_1 \xleftarrow{\$} \mathbb{F}_q$
6:     $R_2 \xleftarrow{\$} \mathbb{F}_q$
7:     for $j = 1$ to $n$ do
8:         $B_m^j \leftarrow B_m^j + (A_i^j + R_1) \cdot R_2 \mod q$
9:     end for
10:    $\tilde{B}_m \leftarrow \tilde{B}_m + (\tilde{A}_i + n \cdot R_1) \cdot R_2 \mod q$
11: end for
12: for $j = 1$ to $n$ do
13:    $B_m \leftarrow B_m + B_m^j \mod q$
14: end for
```

3.2.1 Correctness

The correctness of the algorithm is given if a valid ciphertext input successfully passes the comparison and the probability that a malformed ciphertext successfully passes the comparison is negligible. It is trivial to see that the first condition is fulfilled. For the second condition, we analyze the upper bound Y for the probability that an attacker can create a collision, indicated with P_{coll}, i.e., that all m masked sums B_m and $\tilde{B}_m$ are equal even though $A \neq \tilde{A}$.

$$P_{coll} = P(\text{compare}(\mathbf{A}, \tilde{A}) = \text{ true } | \exists i : A_i \neq \tilde{A}_i) \leq Y$$

In order to determine P_{coll}, we point out that

$$P_{coll} \quad = \quad (P_{single-coll})^x$$

where $P_{single-coll}$ is the probability that one pair of masked sums B_m and $\tilde{B}_m$ is equal when $A \neq \tilde{A}$. Assuming random input A, $P_{single-coll}$ is given by:

$$\begin{aligned}
P_{single-coll} &= P(B_m = \tilde{B}_m) - P(\forall i \in I_m : A_i = \tilde{A}_i) \\
&= P\left(\sum_{i=1}^{l} \left(A_i - \tilde{A}_i\right) \cdot r_{2,i} = 0\right) - q^{-l} \\
&= P\left((A_l - \tilde{A}_l) \cdot r_{2,l} = 0\right) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = 0\right) \\
&+ \sum_{c=1}^{q-1} \left(P((A_l - \tilde{A}_l) \cdot r_{2,l} = c) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = q - c\right)\right) - q^{-l} \\
&= \left(1 - \left(\frac{q-1}{q}\right)^2\right) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = 0\right) \\
&+ \sum_{c=1}^{q-1} \left(\left(\frac{q-1}{q^2}\right) \cdot P\left(\sum_{i=1}^{l-1} (A_i - \tilde{A}_i) \cdot r_{2,i} = q - c\right)\right) - q^{-l} \\
&= \frac{1 - q^{-l}}{q} \approx \frac{1}{q}.
\end{aligned}$$

If this assumption holds, the overall collision probability is then

$$P_{coll} \approx \frac{1}{q^x}.$$

As shown in Fig 1, the attacker has full control over $\tilde{A}$, but cannot directly influence A. In the following, we want to show that for some input $\tilde{A}$ the resulting A is indistinguishable from a deterministic but random vector and therefore the equation above holds. As Figure 1 shows, A is the output of the `CPA.Encryption` and as such also depends on the output of the `CPA.Decryption` that in turn depends on the secret key. With his choice of the $\tilde{A}$, the attacker can single out single coefficients of the secret key and therefore minimize the influence of the secret key. By doing so, the number of possible outputs of the `CPA.Decryption` is q. The output of the `CPA.Decryption` is also the input to the random oracle G, which outputs the seed for the PRNG that is used in the `CPA.Encryption`. The PRNG is then used to generate the noise polynomials in the `CPA.Encryption`. This means that, while the attacker can reduce his uncertainty about the output of the `CPA.Decryption` to a single coefficient, this uncertainty will spread through the PRNG and influence all other coefficients in the output of the `CPA.Encryption`. Therefore A will look random and our equation for P_{coll} holds.

The number of sets is therefore calculated as $x = \lceil -\log_q Y \rceil$. For $q = 12289$ and $Y < 2^{-128}$, we have $x = 10$. In our implementation, we will set $x = 16$ since $16 | (k = 1024)$ and there is no difference in performance. The only downside of increasing the number of sets is that the maximum masking order is reduced. But for $x = 16$ sets and $k = 1024$ coefficients, the possible number of shares is still $n \leq l = 64$, which should be much more than needed in practice. For $x = 16$, the collision probability is $P_{coll} \approx 2^{-217}$.

3.2.2 Probing Security Proof

In this section we provide the formal proof that our comparison algorithm is t probing secure, with $t \leq \min(n - 1, l)$, with probability $1 - q^{-l}$. We proceed by first showing, in the following proposition, that the algorithm `MaskedSum` is $t - \mathsf{NI}$.

Proposition 1. *Algorithm 5 is $t-\mathsf{NI}$ at any order $t \leq \min(n-1, l)$, unless $\forall i \in I_m$: $A_i = \tilde{A}_i$.*

Proof. Let $\mathcal{P} = (\mathcal{I}, \mathcal{O})$ be the set of t adversarial probes on Algorithm 5, with $\mathcal{I}$ the ones on the internals and $\mathcal{O}$ on the output. The elements of $\mathcal{I}$ belong to the following possible groups:

(0) A_i^j

(1) $R_1^{(i)}, R_2^{(i)}$, the values of R_1 and R_2 at the i^{th} iteration of the loop at line 4

(2) $A_i^j + R_1^{(i)}$

(3) $(A_i^j + R_1^{(i)}) \cdot R_2^{(i)}$

(4) $(A_1^j + R_1^{(1)}) \cdot R_2^{(1)} + \cdots + (A_k^j + R_1^{(k)}) \cdot R_2^{(k)}$ with $k \leq l$, for $j = 1, \ldots, n$

(5) $B_m^1 + \cdots + B_m^h$ with $h < n$

(6) $\tilde{A}_i$

(7) $(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$

(8) $\sum_{i=1}^{k} (\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ with $k < l$

with, where not differently stated, $i = 1, \ldots, l$ and $j = 1, \ldots, n$. The set $\mathcal{O}$, instead, is constituted by B_m and $\tilde{B}_m$.

We start by constructing l sets of indexes $I^{(1)}, \ldots, I^{(l)}$ in the following way: for each probe in group (0), (2) or (3) add the index j to the set $I^{(i)}$ and for each probe in group (4) add the index j to each $I^{(i)}$ with $i \leq k$. Since for each adversarial observation at most one index is added to the $I^{(1)}, \ldots, I^{(l)}$ and no index is added when there is a probe on the output, then each of the sets has cardinality at most $|\mathcal{P}|$.

We now simulate the set of probes $\mathcal{P}$ by using only the A_i^j with $j \in I^{(i)}$. Since the $\tilde{A}_i$ are already public, group (6) does not need to be simulated.

Step 1 For each element in group (0), by construction $j \in I^{(i)}$ and therefore the element can be simulated as in the real algorithm.

Step 2 Each element in group (1) is simulated uniformly at random, i.e., by picking $R_1^{(i)} \in \mathbb{F}_q$ and $R_2^{(i)} \in \mathbb{F}_q$.

Step 3 For each element in group (2), we distinguish two cases. If $R_1^{(i)}$ was already observed, we take the value simulated in the previous step. Otherwise we pick $R_1^{(i)}$ uniformly at random from $\mathbb{F}_q$. In both cases, since $j \in I^{(i)}$ we can simulate A_i^j as in the real algorithm.

Step 4 For each element in group (3), we distinguish the following cases. If $(A_i^j + R_1^{(i)})$ was probed, we take the already simulated value, otherwise we simulate it according to the previous step. Additionally, we simulate the value $R_2^{(i)}$ with his value from Step 2, if it was previously observed, and by picking it uniformly at random from $\mathbb{F}_q$ otherwise. We finally calculate the product $(A_i^j + R_1^{(i)}) \cdot R_2^{(i)}$.

Step 5 For each element in group (4), we distinguish the following cases. For any sum $\sum_{i=1}^{h} (A_i^j + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ with $h < k$ that has already been probed, use the probed value for its simulation. For the rest of the addends, if one of the sums $(A_i^j + R_1^{(i)}) \cdot R_2^{(i)}$ was probed, we take the simulated value, otherwise we simulate it as in the previous step. We finally calculate the sum of the values as in the real algorithm. Note that by construction, even if all the random bits are probed, the simulation needs at most one share A_i^j of each input.

Step 6 If the probe is in group (5), we point out that, due to the common random bits among the addends, the sum can be rewritten as $\sum_{i=1}^{l}(A_i^1 + \cdots + A_i^h + hR_1^{(i)})R_2^{(i)}$. Despite the simplification, since $t \leq l$, then there exists at least one pair $R_1^{(\hat{i})}$ and $R_2^{(\hat{i})}$ that has not been probed. We pick such $R_1^{(\hat{i})}$ and $R_2^{(\hat{i})}$ uniformly at random from $\mathbb{F}_q$, and therefore there exists at least one of the sums that will be simulated at random, and as a consequence the entire sum is simulated randomly and independently from each A_i.

Step 7 In the case the probe is in group (7), if $R_1^{(i)}$ (resp. $R_2^{(i)}$) has not already been simulated during one of the steps above, pick uniformly at random $R_1^{(i)} \in \mathbb{F}_q$ (resp. $R_2^{(i)} \in \mathbb{F}_q$), otherwise take the value already assigned and in both cases compute the probe as in the algorithm, using the public value $\tilde{A}_i$.

Step 8 In the case the probe is in group (8), for any sum $\sum_{i=1}^{h}(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ with $h < k$ that has already been probed, use the probed value for its simulation. For the rest of the sums, for each $(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ already observed, take the value already simulated. For the remaining addends, simulate them as in Step 7. Finally sum up the $(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ as in the real algorithm.

Step 9 For the output $B_m = B_m^1 + \cdots + B_m^n$, since this corresponds to an element in group (5) with $h = n$, the simulations follows the same procedure as Step 6. This time the sum reduces to $\sum_{i=1}^{l}(A_i + nR_1^{(i)})R_2^{(i)}$. We point out again that, since $t \leq l$, then there exists at least one pair of elements $R_1^{(i)}$ and $R_2^{(i)}$ that has not been probed.

Step 10 Finally, for the simulation of the output $\tilde{B}_m$, i.e., $\sum_{i=1}^{l}(\tilde{A}_i + n \cdot R_1^{(i)}) \cdot R_2^{(i)}$ we notice that since $t \leq l$, there exists at least one pair $R_1^{(i)}$ and $R_2^{(i)}$ that has not been probed, and therefore there exists at least one of the sums that will be simulated at random, and as a consequence the entire sum is simulated randomly and independently from $\tilde{A}$. If $\forall i \in I_m : A_i = \tilde{A}_i$, this simulation is not consistent because $\tilde{B}_m$ must be equal to B_m in this case and it therefore depends on the A_i.

From the simulation above and Definition 1 we can conclude that Algorithm 5 is $t - \mathsf{NI}$, unless $\forall i \in I_m : A_i = \tilde{A}_i$. □

In the following we show, that the probabilistic nature of our security proof is of no consequence and the comparison is secure when using practical parameters.
Note that Proposition 1 implies that the outputs B_m and $\tilde{B}_m$ of Alg. 5 and, by extension, the result of the comparison of these values can be simulated without knowledge of any coefficient A_i unless $\forall i \in I_m : A_i = \tilde{A}_i$. In this case the proof fails which results in possible leakage of the A_i. This type of collision can happen in one of the following two cases:

(1) $\forall m, \forall i \in I_m : A_i = \tilde{A}_i$, i.e., all coefficients are equal.

(2) $\exists m, \forall i \in I_m : A_i = \tilde{A}_i$ and $\exists m, \exists i \in I_m : A_i \neq \tilde{A}_i$, i.e, only the coefficients used in some sums are equal.

In case (1) the A_i are not sensitive as they are already known by the attacker.
As noted in Sect. 3.2.1 any change in a coefficient of $\tilde{A}$ is propagated to all coefficients of A through the random oracle G. Therefore, if the output $coin''$ of $\mathtt{G}$ is collision free, the probability of case (2) is $P_{B-coll} = q^{-l}$. When using practical values for q and l, P_{B-coll}

is always smaller than the collision probability of `G`[1] and the algorithm is secure against t-order attackers. For example, with our parameter choice for `NewHope` ($q = 12289$, $l = 64$) $P_{B-coll} < 2^{-869}$.

3.3 Application to `NewHope` and Other Schemes

While our masking scheme is relevant for a large variety of schemes, we specifically pick `NewHope` [ADPS16] as case study to be consistent with previous work [RRVV15, RdCR+16, RRdC+16, OSPG18, SPOG19]. The relevant parameters of the scheme are the lattice dimension (i.e., the number of coefficients in the polynomials) of $k = 1024$ and the modulus $q = 12289$. We expect similar results when our algorithm is applied to other schemes. Generally, higher-order masked lattice-based KEM implementations that rely on the Fujisaki-Okamoto transform to achieve CCA security can benefit from our comparison approach as long as the parameters are compatible to the requirements described in Sect. 3.2. For example, in `Kyber` [BDK+18] the lattice dimension is only $k = 256$, but depending on the parameters set there are more (up to four) polynomials in the ciphertext. As the CCA security of `Kyber` depends on the Fujisaki-Okamoto transform and as shown in Alg. 9 in the `Kyber` specification [SAB+19], the comparison is the same as in `NewHope`. Consequentially, the input to the comparison in line 6 also depends on the output of a random oracle G, therefore our proof in Sect. 3.2.2 holds. The modulus of LAC [LLZ+] is only $q = 251$. This modulus would lead to reduced memory requirements as each coefficient can be stored in a single byte.

4 Microcontroller Implementation

In this section, we discuss our microcontroller implementation in detail and the methodology for our side-channel measurements.

4.1 Microcontroller Implementation

Our evaluation platform is the STM32F4-DISCOVERY board that is based on the STM32F407VGT6 ARM Cortex-M4F microcontroller. NIST recommends to use the Cortex-M4F as target platform for microcontroller evaluations of post-quantum standardization candidates [Moo19]. Furthermore, the concrete processor and board we used in our performance and side-channel evaluation was suggested as a reference platform for PQC algorithms in [KRSS19]. It runs with a clock frequency of up to 168 MHz. The board offers 192 kB of RAM as well as 1 MB of flash memory. Furthermore, it features a true random number generator (TRNG) based on analog circuitry and a floating-point unit (FPU). The Cortex-M4F has 13 general purpose registers and ($R_0 - R_{12}$), one register reserved for the stack pointer, a link register, one register reserved for the program counter, and special-purpose program status registers. When mixing C with assembly it is important to note that the calling convention requires parameters to be in $R_0 - R_3$ and the result to be in $R_0 - R_1$. The link register can be used as general purpose register as well, if the assembly function does not call any other function and its original value is restored before leaving the function.

To prevent timing leakages, implementations of cryptographic schemes are usually expected to be protected against timing attacks, this is usually referred to as *constant-time* property of implementations. However, we need to distinguish between two different notions of *constant-time*. In the following, we will use the expression *constant-time* in case the execution time of an implementation is actually constant. Furthermore, we will refer

[1]More precisely, the relevant metric is second-preimage resistance. In the case of `NewHope` *coin*$''$ is $\in \{0, ..., 255\}^{32}$.

to an implementation as *timing-independent* if the implementation has a non-constant execution time but is still protected against timing side channels because the execution is independent from the input data.

The implementation of our comparison algorithm requires the generation of random numbers in $[0, q-1]$, where q is an arbitrary integer and in many cases (like `NewHope`) a prime. To ensure a uniform distribution of these numbers, we apply the rejection sampling method from [SPOG19] that takes as input random bit vectors from the on-board TRNG and only accepts the input if the value is in $[0, q-1]$ and rejects otherwise. As this method works with rejections, it is *timing-independent*, but not *constant-time*. We therefore implemented two variants of the algorithm - one *constant-time* implementation that is suitable for side-channel evaluation and one performance-optimized variant for practical use. The main difference between these two implementations is that the side-channel measurement-friendly variant first fills a buffer with random values in $[0, q-1]$. The implementation of the actual comparison then just accesses this buffer to load the necessary random numbers. By doing so, side-channel measurements can be triggered after the (*non-constant-time*) generation of random numbers is completed avoiding the necessity of trace alignment in the side-channel experiments. However, as the on-board TRNG needs to sample sufficient thermal noise in the background, requesting random numbers from the TRNG in rapid succession is quite slow as the program will be halted until the TRNG is ready. In our second performance-optimized approach, we therefore spread out the TRNG calls throughout the algorithm to minimize the TRNG waiting time.

```
MOV TMP,#0x3001
ADD INPUT_A, INPUT_B
SUB INPUT_A, TMP
SXTB TMP2, INPUT_A, ROR #24
AND TMP, TMP2
ADD INPUT_A, TMP
```

Listing 1: Combined modular addition and subtraction in assembly.

For the modular reduction, we use the constant-time Barrett reduction from [OSPG18] that uses the floating point unit of the Cortex-M4F. We furthermore use a special method to combine the addition and modular reduction of two values *mod q* as shown in Listing 1. The idea is to perform a conditional subtraction of the modulus in constant-time. We first load the modulus into a temporary registers tmp_1. Then we add the two inputs and subtract the modulus from the sum. With the help of the `SXTB` instruction and the internal barrel shifter of the Cortex-M4F we create a bit mask that is either $1...1_2$ in case the result of the subtraction was negative or $0...0_2$ if the result was positive and store the bit mask in tmp_2. We then compute the `AND` of tmp_1 and tmp_2. The register tmp_1 now contains either the modulus if the result of the subtraction was negative or $0...0_2$ if the result was positive. Finally, tmp_1 is added to the result of the subtraction to receive the output. This approach takes only six cycles for combined addition and modular reduction and needs two temporary registers for intermediate results.

We try to minimize the load and store memory accesses by efficiently using the available registers of the Cortex-M4F. While doing so, it is important to keep in mind that we first sum up all coefficients within a set of a share and then sum up the sums of each share. It might be tempting to switch the order of these summation because it would save a big number of loads and stores of the random r values. However, this would also introduce a side-channel leakage as the secret shares would be combined without sufficient randomness. Apart from the r values, no value is loaded twice and no store instruction is used. We therefore argue that our memory access scheme is optimal.

4.2 Side-channel Measurements

In order to practically evaluate the resistance of the proposed comparison algorithm against side-channel attacks, we performed a Test Vector Leakage Assessment [GJJR11] of the *constant-time* implementation described in Section 4.1. With the goal of reducing the computational complexity of the evaluation, we set the number of coefficients in the measured implementation to four and only considered $k = 1$ sets. Note, that this does not weaken the evaluation results due to the construction of our algorithm. By using the non-specific fixed-versus-random t-test, the analysis can show possible leakage points independent of specific sensitive variables. In this evaluation methodology, the target device is supplied with either a fixed or random input in a random order. During the computation of the target, the side channel (e.g. power consumption or EM emanation) is measured on which the test metric is applied to decide if the consumption is distinguishable depending on the input. For the first-order univariate case, the test statistic to evaluate if the mean of a sample point of the two sets of traces F and R is different can be computed as

$$t_{F,R} = \frac{\bar{F} - \bar{R}}{s_n}$$

with

$$s_n = \sqrt{\frac{s_F^2}{n_F} + \frac{s_R^2}{n_R}}$$

where n_X, $\bar{X}$, and s_X^2 are the number of collected samples, the estimated means and the estimated variance of the respective point. The magnitude of this test-statistic can then be compared to a threshold which is required to be reached to confirm an input-dependent mean of the analyzed sample point. For the evaluation of complete power- or EM-traces, the statistic can be computed point-wise. As pointed out in [DZD$^+$17], this simultaneous application of multiple tests can artificially skew the outcome towards a positive result. In order to obtain a result with a given confidence, the detection threshold must therefore be adjusted depending on the number of samples in a trace. We use the Šidák Correction as suggested in [BPG18] to calculate the threshold for a confidence level of α given a trace length L and n measurements:

$$t_{th} = Q_t(1 - \sqrt[L]{1-\alpha}, v),$$

where $Q_t(\cdot)$ is the quantile-function of the t-distribution and $v \approx n/2$ the degree of freedom. Therefore a threshold for multivariate leakage (with $L * L$ points) is always higher when compared to the univariate case. As our experiment targets a software implementation where different shares of a sensitive variable are manipulated at different points in time, a multivariate analysis is necessary for achieving a meaningful evaluation of higher-order leakage. We restricted the experiments to first- and second-order analysis of the two- and three-share variants of the comparison algorithm, as higher-order multivariate leakage assessment is computationally prohibitive. This is because the effort for the required sample-point combination is proportional to $n \cdot L^E$ where E is the evaluation order. The evaluation of second-order multivariate leakage was performed following [SM15] by combining all pairs of points in a trace using the optimal centered product according to [SVO$^+$10].

We sent a fixed or random challenge to the target microcontroller in a random order. The power consumption was measured while the microcontroller was evaluating the comparison algorithm on either a fixed or a random input. The device generated a trigger pulse for optimal trace alignment and performed the comparison algorithm on the provided input and a fixed set of coefficients stored at compile-time. When generating the random input, we rejected values that were equal to the reference coefficients to avoid producing high false-positive (non-exploitable) t-test results in the evaluation.

The measurements were conducted on the same ARM Cortex-M4F board that was used for the performance evaluation.

In order to show the side-channel resistance of our implementation in a worst-case scenario, we increased the signal-to-noise ratio as much as possible by modifying the PCB as well as adapting the firmware accordingly. For data transfer between a host computer and the target board, we made use of a UART-core of the microcontroller which eliminates noise introduced by the on-board USB interface. In addition to disabling data and instruction caches available in the STM32F407VG, the SysTick-timer and all interrupt source in the controller were disabled to ensure *constant-time* behavior of the measured code. For further noise reduction in the measurement system, all non-essential clock paths in the target controller were disabled. In an effort to reduce other noise sources as much as possible, all non-essential peripheral devices on the board, such as the MEMS accelerometer and the audio DAC, were de-soldered.

As power measurements appeared to not contain a sufficient signal-to-noise ratio for a successful evaluation on our microcontroller board, we collected EM traces with a near-field probe and amplified them before feeding them into an oscilloscope. The EM traces were acquired with a sample rate of 156.25 MS/s at 8 bit resolution using a 50 MHz-bandwidth H-field probe. We kept the position of the probe relative to the microcontroller fixed between measurements in order to produce comparable results.

The findings of the practical SCA-evaluation are provided in Sect. 5.3.

5 Results

In this section, we present the results of our practical evaluations. This includes performances evaluations as well as side-channel evaluations.

5.1 Performance Evaluation

We evaluated our work by using the *OpenSTM32 System Workbench* (version 2.6), which is based on the development environment `Eclipse` and has specifically been designed to support the development for ARM-based STM32 boards. The IDE uses the GNU ARM Embedded Toolchain (version 7.2) and we set the optimization level to `-O3`. Determining the performance of our implementation was done by using the cycle count register `DWT_CYCCNT` of the *Data Watchpoint and Trace* unit that the Cortex-M4F offers. We set the clock frequency of the microcontroller to 24 MHz to avoid any wait cycles at the memory. All cycles counts were obtained by measuring our performance-optimized implementation. With the publication of this work, we will make our implementation publicly available to allow independent verification of our results.

In Table 1, we show the cycle counts for the comparison algorithm of one polynomial with $k = 1024$ coefficients. We compare our approach with the cost of 1024 A2B conversions as explained in Section 3.1. Both implementations benefit from assembly optimization. The direct comparison of both approaches shows that for two shares already, our approach is at least 14 times faster than an A2B-based approach. It is also obvious from the table that our algorithm has a better asymptotic complexity as it is only linear in the number of shares while A2B conversions are at best quadratic. Therefore the speed-up factor gets even bigger when the number of shares is increased which makes our algorithm two orders of magnitude faster for five shares already. Extrapolating these numbers we expect the cycle counts for higher orders to be around $165,000 + 41,000 \cdot n$, where n is the number of shares.

At our measurement frequency of 24 MHz, 250,991 cycles are executed in 10 milliseconds. However, the maximum clock frequency of the microcontroller is 168 MHz. For reference, we also measured our implementation at 168 MHz, to get a realistic impression of the

Table 1: Clock cycle counts for our ARM implementations of the masked comparison at 24 MHz for $k = 1024$ including randomness generation. All results are averaged over 100 runs.

Shares	2	3	4	5
Comparison in [OSPG18]	480,227	-	-	-
`NewHope` CCA-DEC [OSPG18]	25,334,493	-	-	-
1024 A2B conversions	4,127,744	11,875,328	21,027,840	35,353,600
Our comparison algorithm	250,991	284,989	329,053	373,860
Speed-up factor	**x16**	**x42**	**x64**	**x95**

time needed to execute the algorithm. We observe only a minor increase in the number of cycles at 168 MHz, namely to 258,695 cycles. Since the algorithm does not load any constants values from flash memory, we assume that this difference is mainly caused by the variable timing behavior of our implementation (see Section 4.1 for a discussion about why our implementation is secure against timing attacks). We therefore conclude that the cycle counts obtained at 24 MHz also give a realistic impression of the performance of the implementation at 168 MHz. Therefore, at 168 MHz, the masked comparison of one polynomial takes 1.5 ms for two shares, 1.7 ms for three shares, 2.0 ms for four shares, and 2.2 ms for five shares. This makes our approach a low-overhead component for the construction of higher-order masking schemes lattice-based cryptography.

For polynomials with $k = 1024$, the maximum masking order that our implementation supports is 64 as we decided to implement 16 subsets of coefficients. We expect this to be sufficient for practical uses in the foreseeable future. One downside of our approach is the relatively high dynamic memory consumption as we need to store $2k$ random values *mod* q. This is equal to the memory consumption of two polynomials. However, embedded into a lattice-based KEM, we do not expect our algorithm to increase the dynamic memory consumption at all as the peak memory usage is expected to be in the `CPA.Encryption` or `CPA.Decryption` and the comparison algorithm is executed after these two components. The static Flash memory consumption of our algorithm is also very low as it needs only about 200 lines of assembly code.

To our knowledge, currently there exists no higher-order CCA-secure implementation of any lattice-based KEM. However, we also implemented the approach from [OSPG18], even though this algorithm is only first-order secure and cannot be extended to higher orders. Table 1 includes the cycle count for the complete CCA2-secure decryption as well as for only the comparison step. The board used to measure these performance values was identical to the one we used in our evaluation. The idea from [OSPG18] is to subtract $\tilde{A}$ from one share of A, individually hash the result of this subtraction and the other share and compare the outputs of the hash calls. Our implementation achieves similar performance even though our source code can generically support higher orders too and the implementation from [OSPG18] is optimized for the first-order case.

Table 2: Clock cycle counts for our ARM implementations of the masked comparison at 24 MHz for different parameter sets including randomness generation. All results are averaged over 100 runs.

Shares		2	3	4	5
`KYBER-768`	A2B conversions	3,095,040	8,906,496	15,770,880	26,515,968
	Our algorithm	185,338	216,945	248,455	279,973
	Speed-up factor	**x17**	**x41**	**x63**	**x95**
`LAC-192`	A2B conversions	2,267,136	6,360,064	11,131,904	18,551,808
	Our algorithm	230,432	272,489	314,558	356,621
	Speed-up factor	**x10**	**x23**	**x35**	**x52**

In Table 2 we also exemplarily evaluated the performance of our implementation for the parameter sets of `KYBER-768` and `LAC-192` to show the impact of the choice of n on the performance of the comparison. `KYBER-768` uses $k = 768$ coefficients and a modulus $q = 3329$. In this case, we observe very similar speed-up factors in comparison with an A2B approach. This is expected since both approaches are linear in k. Therefore the cycle counts for $k = 768$ are also roughly equal to three quarters of the cycle counts for $k = 1024$. In this scenario the cycle count for higher orders can be expected to be around $122,000 + 32,000 \cdot n$ with n shares.
While `LAC-192` operates on the same number of $k = 1024$ coefficients as `NewHope` the modulus is $q = 251$ in this case. This results in lower randomness requirements of our algorithm and therefore slightly increased performance when compared to `NewHope`. As the A2B-based approach benefits even more from this, the speed-up gained by our approach is lower. For `LAC-192`, the cycle count for higher orders can be expected to be around $145,000 + 42,000 \cdot n$ with n shares.

As stated before, complete higher-order masked lattice-based KEMs are currently not available for comparison. However, we can roughly estimate the impact of our comparison algorithm at higher orders. For obvious reasons, the cycle count of any masked implementation must be at least linear in the number of shares n.[2] As our algorithm has linear complexity in n the relative overhead of the algorithm does not (asymptotically) grow with increasing n.

5.2 Randomness Consumption

In this section, we analyze how much randomness our implementation needs for the masking scheme. For our specific choice of $q = 12289$, the rejection-based sampling of uniform random numbers *mod q* has an acceptance rate of 75% since the acceptance rate a_r is calculated as $a_r = \frac{12289}{2^{14}}$. For efficiency reasons, we use chunks of 16 bits of randomness for one sampling attempt of which two bits are simply ignored. On our evaluation platform the randomness can be efficiently generated by the on-board TRNG. On platforms with lower randomness generation capabilities the randomness requirements can be reduced to 87.5% of the values below by using 14- instead of 16-bit chunks. This approach requires more memory to store the randomness and more processor cycles to extract it. As we need $2k$ random numbers *mod q*, the expected number of required random bits for our approach is

$$r_{bits} = 2k\frac{16 \text{ bits}}{a_r} = 2 \cdot 1024 \cdot \frac{16 \text{ bits}}{0.75} = 43,688 \text{ bits}$$

This calculation is valid for any number of shares $n < 64$. This theoretical amount of randomness is confirmed by our practical experiments. We compare the randomness consumption of our approach to the A2B-based approach in Table 3 as measured by our implementation. The first-order only approach from [OSPG18] does not need any additional randomness. While our algorithm needs less randomness than the A2B-based approach even for low masking orders this advantage increases for higher orders because the randomness requirement is independent of the masking order.

5.3 Leakage Evaluation

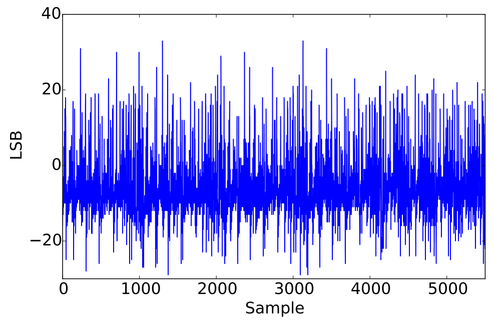
This section details the results of our experimental side-channel evaluation. We used a significance level of $\alpha = 0.01$ for all assessments in this section. For reference, Figure 2a

[2]Otherwise it could just use less shares.

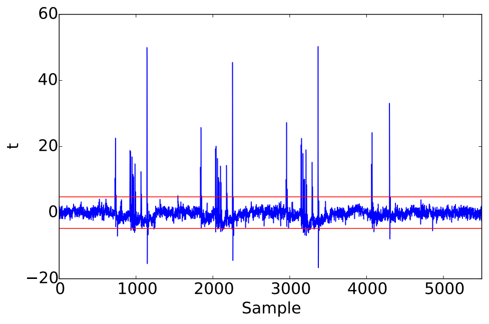
Table 3: Random bit consumption for our ARM implementations of the masked comparison for different parameter sets. All results are averaged over 100 runs.

Shares		2	3	4	5
NewHope	A2B conversions	655,360	2,392,064	4,653,056	8,519,680
	Our algorithm	43,648	43,680	43,584	43,712
	Improvement factor	**x15**	**x55**	**x107**	**x195**
KYBER-768	A2B conversions	491,520	1,794,048	3,489,792	6,389,760
	Our algorithm	30,240	30,208	30,176	30,272
	Improvement factor	**x16**	**x59**	**x116**	**x211**
LAC-192	A2B conversions	393,216	1,343,488	2,555,904	4,587,520
	Our algorithm	16,704	16,711	16,706	16,705
	Improvement factor	**x24**	**x80**	**x153**	**x275**

shows an example trace of the measured EM-emanation. When the masking countermeasure is deactivated by setting the masks to zero, large first-order leakage can be observed as shown in Figure 2b even using only 5000 traces. This behavior is expected as the algorithm operates on unmasked values in this case.



(a) Example trace of two-share version with four coefficients.



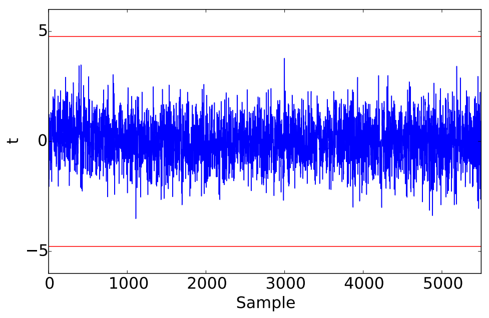
(b) First-order leakage for two-share version with four coefficients (masks disabled, 5 k measurements, $t_{th} = 4.77$).

Figure 2: Sample trace and reference measurement.

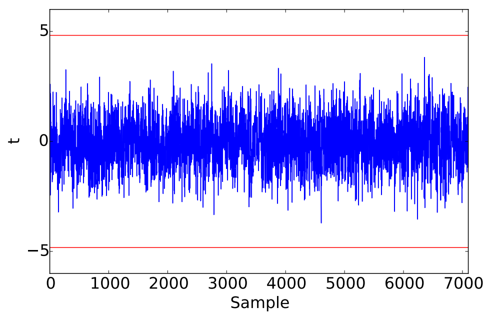
When random masks are used, an evaluation using 1 million traces does not show detectable first-order leakage (Fig. 3a). The two-dimensional plot resulting from the bivariate TVLA is shown in Figure 4. Each pixel is colored according to the absolute t-value present in the respective combinations of points in time. In order to ensure readability of Figures 4 and 5 we binned the sample points into a 100 times 100 pixel-grid and plotted the maximum t-value of each bin. The multivariate analysis of the second-order leakage allows to clearly identify points at which different shares of coefficients are handled, as the 2-share implementation only protects against first-order attacks.

Figure 3b shows the result of a first-order evaluation on traces collected from the three-share implementation with four coefficients and activated masking after 1 million measurements. As expected, no first-order leakage can be detected. The results of the second-order multivariate t-test is shown in Fig. 5. The leakage detection threshold of $t_{th} = 6.36$ is not reached at any point in time.

In summary, we were not able to detect first-order leakage in the two-share *constant-time* implementation or second-order multivariate leakage in the three-share implementation even using 1 million measurements.



(a) First-order leakage for two-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 4.77$).



(b) First-order leakage for three-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 4.82$).

Figure 3: First-order SCA analysis of 2- and 3-share implementation.

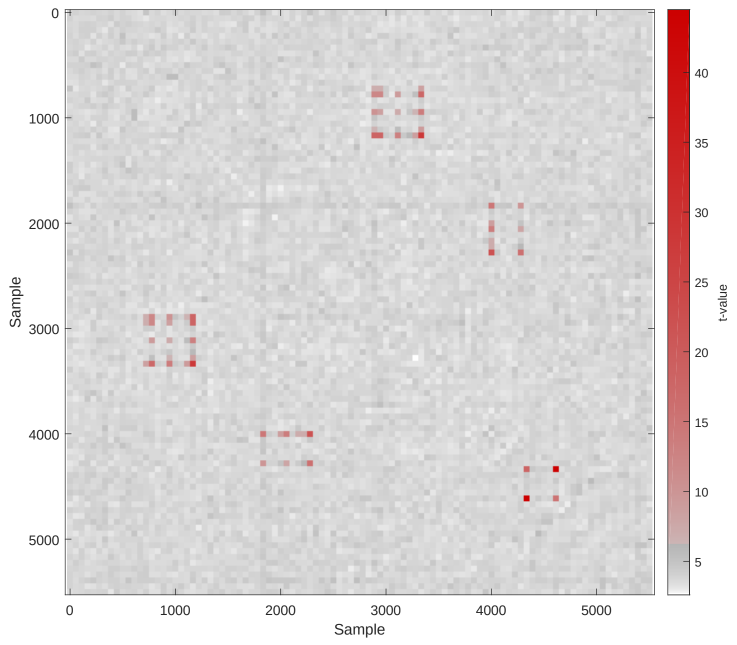


Figure 4: Second-order leakage for two-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 6.28$). Points with t-values above the threshold are highlighted red.

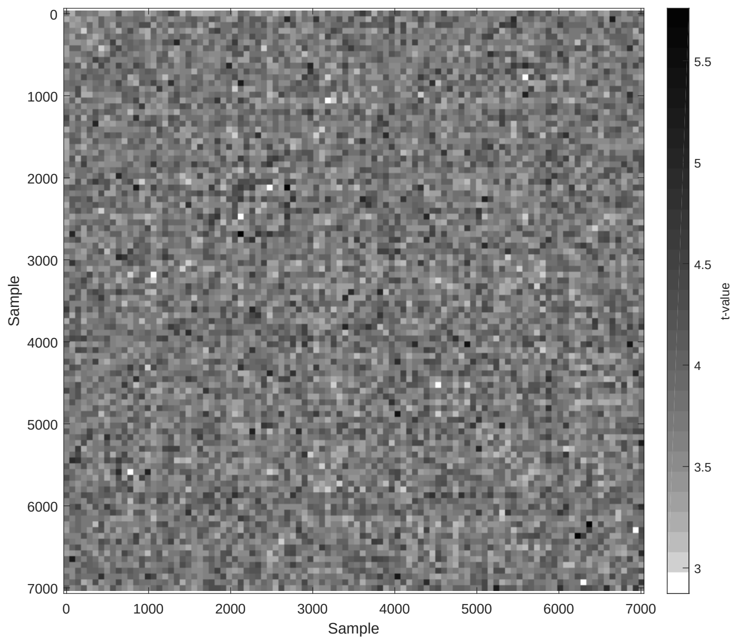


Figure 5: Second-order leakage for three-share version with four coefficients (masks enabled, 1 M measurements, $t_{th} = 6.36$). Points with t-values above the threshold are highlighted red (none present).

6 Conclusions

In this work, we identify the comparison step of the Fujisaki-Okamoto transform as, a so far overlooked, bottleneck in higher-order masking of lattice-based cryptography. We present a novel higher-order masking scheme for the comparison, that outperforms the naive approach by at least one order of magnitude and it is applicable to constructions with prime modulus. The naive approach based on A2B conversions has a complexity of $O(n^2k)$, while the asymptotic complexity of our algorithm is only $O(nk)$, i.e., it is linear in the number of shares and in the number of coefficients of the polynomial. Furthermore, the probability for an attacker to forge an invalid ciphertext that is still accepted by our comparison is negligible (2^{-217}). We give a theoretical proof of the side-channel security of our algorithm and confirm with practical measurements that our highly efficient microcontroller implementation does not show side-channel leakage, even for significantly more power traces than in previous work on masking for lattice-based cryptography. In the ongoing NIST post-quantum standardization, our work is an important step towards understanding the overhead cost of side-channel countermeasures applied to the NIST candidates.

6.1 Future Work

As future work, we advise extending the comparison algorithm to work for power of two moduli, which at the moment are not considered in our scheme. Furthermore it would be interesting to see how our masking countermeasure can be combined with countermeasures against other attacks, like fault injection attacks, since it was already analyzed in [OSPG18] that the comparison step could be a primary target for fault attacks.

Acknowledgment

The research in this work was supported in part by the German Research Foundation (DFG) under Germany's Excellence Strategy - EXC 2092 CASA - 390781972, the Federal Ministry of Education and Research of Germany through the QuantumRISC project (16KIS1038) and VeriSec project (16KIS0603), by the European Unions Horizon 2020 program under project number 780701 PROMETHEUS and the ERC Project 724725 (SWORD). We would also like to thank the anonymous reviewers for their very valuable and helpful feedback.

References

[AAB+] Erdem Alkim, Roberto Avanzi, Joppe Bos, Léo Ducas, Antonio de la Piedra, Thomas Pöppelmann, Peter Schwabe, and Douglas Stebila. NewHope Algorithm Specifications and Supporting Documentation. https://newhopecrypto.org/data/NewHope_2018_12_02.pdf.

[ADPS16] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Post-quantum key exchange - A new hope. In *25th USENIX Security Symposium*, pages 327–343, 2016.

[AJS16] Erdem Alkim, Philipp Jakubeit, and Peter Schwabe. Newhope on ARM Cortex-M. In *Security, Privacy, and Applied Cryptography Engineering - 6th International Conference, SPACE*, pages 332–349, 2016.

[APB+04] Riza Aditya, Kun Peng, Colin Boyd, Ed Dawson, and Byoungcheon Lee. Batch verification for equality of discrete logarithms and threshold decryptions. In Markus Jakobsson, Moti Yung, and Jianying Zhou, editors, *Applied Cryptography and Network Security, Second International Conference, ACNS 2004, Yellow Mountain, China, June 8-11, 2004, Proceedings*, volume 3089 of *Lecture Notes in Computer Science*, pages 494–508. Springer, 2004.

[BBD+15] Gilles Barthe, Sonia Belaïd, François Dupressoir, Pierre-Alain Fouque, and Benjamin Grégoire. Compositional verification of higher-order masking: Application to a verifying masking compiler. *IACR Cryptology ePrint Archive*, 2015.

[BBE+18] Gilles Barthe, Sonia Belaïd, Thomas Espitau, Pierre-Alain Fouque, Benjamin Grégoire, Mélissa Rossi, and Mehdi Tibouchi. Masking the GLP lattice-based signature scheme at any order. In *Advances in Cryptology - EUROCRYPT*, pages 354–384, 2018.

[BDK+18] Joppe W. Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS - Kyber: A CCA-secure module-lattice-based KEM. In *IEEE European Symposium on Security and Privacy, EuroS&P*, pages 353–367, 2018.

[BPG18] Florian Bache, Christina Plump, and Tim Güneysu. Confident leakage assessment - A side-channel evaluation framework based on confidence intervals. In *DATE*, pages 1117–1122. IEEE, 2018.

[CGV14] Jean-Sébastien Coron, Johann Großschädl, and Praveen Kumar Vadnala. Secure conversion between boolean and arithmetic masking of any order. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 188–205. Springer, 2014.

[DZD$^+$17] A. Adam Ding, Liwei Zhang, François Durvaux, François-Xavier Standaert, and Yunsi Fei. Towards sound and optimal leakage detection procedure. In *CARDIS*, volume 10728 of *Lecture Notes in Computer Science*, pages 105–122. Springer, 2017.

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. How to enhance the security of public-key encryption at minimum cost. In Hideki Imai and Yuliang Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 1999.

[GJJR11] 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.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Advances in Cryptology - CRYPTO 2003*, pages 463–481, 2003.

[KMRV] Angshuman Karmakar, Jose M. Bermudo Mera, Sujoy Sinha Roy, and Ingrid Verbauwhede. Saber on ARM CCA-secure module lattice-based key encapsulation on ARM. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(3):243–266.

[KRSS19] Matthias J. Kannwischer, Joost Rijneveld, Peter Schwabe, and Ko Stoffelen. pqm4: Testing and benchmarking NIST PQC on ARM cortex-m4. *IACR Cryptology ePrint Archive*, 2019:844, 2019.

[KRVV19] Angshuman Karmakar, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Pushing the speed limit of constant-time discrete gaussian sampling. A case study on the falcon signature scheme. In *Proceedings of the 56th Annual Design Automation Conference, DAC*, pages 88:1–88:6, 2019.

[LLZ$^+$] Xianhui Lu, Yamin Liu, Zhenfei Zhang, Dingding Jia, Haiyang Xue, Jingnan He, and Bao Li. LAC: practical ring-LWE based public-key encryption with byte-level modulus. *IACR Cryptology ePrint Archive*, 2018:1009.

[MGTF19] Vincent Migliore, Benoît Gérard, Mehdi Tibouchi, and Pierre-Alain Fouque. Masking dilithium - efficient implementation and side-channel evaluation. In *Applied Cryptography and Network Security, ACNS*, pages 344–362, 2019.

[Moo19] Dustin Moody. Round 2 of NIST PQC competition. *Talk at PQCrypto 2019, Chongqing, China, 2019*, 2019.

[NIS16] NIST. Submission requirements and evaluation criteria for the post-quantum cryptography standardization process. *National Institute of Standards and Technology*, December 2016. See https://csrc.nist.gov/csrc/media/projects/post-quantum-cryptography/documents/call-for-proposals-final-dec-2016.pdf.

[OSPG18] Tobias Oder, Tobias Schneider, Thomas Pöppelmann, and Tim Güneysu. Practical CCA2-secure and masked Ring-LWE implementation. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, (1):142–174, 2018.

[RdCR$^+$16] Oscar Reparaz, Ruan de Clercq, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. Additively homomorphic ring-LWE masking. In *Post-Quantum Cryptography - PQCrypto*, pages 233–244, 2016.

[RRdC$^+$16] Oscar Reparaz, Sujoy Sinha Roy, Ruan de Clercq, Frederik Vercauteren, and Ingrid Verbauwhede. Masking ring-LWE. *J. Cryptographic Engineering*, 6(2):139–153, 2016.

[RRVV15] Oscar Reparaz, Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. A masked ring-LWE implementation. In *Cryptographic Hardware and Embedded Systems - CHES*, pages 683–702, 2015.

[SAB$^+$19] P Schwabe, R Avanzi, J Bos, L Ducas, E Kiltz, T Lepoint, V Lyubashevsky, JM Schanck, G Seiler, and D Stehle. Crystals-kyber–algorithm specifications and supporting documentation. *NIST Technical Report*, 2019.

[SBG$^+$18] Markku-Juhani O. Saarinen, Sauvik Bhattacharya, Óscar García-Morchón, Ronald Rietman, Ludo Tolhuizen, and Zhenfei Zhang. Shorter messages and faster post-quantum encryption with Round5 on Cortex M. In *Smart Card Research and Advanced Applications, CARDIS*, pages 95–110, 2018.

[SM15] 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.

[SPOG19] Tobias Schneider, Clara Paglialonga, Tobias Oder, and Tim Güneysu. Efficiently masking binomial sampling at arbitrary orders for lattice-based crypto. In *Public-Key Cryptography - PKC*, pages 534–564, 2019.

[SVO$^+$10] 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.

[TU16] Ehsan Ebrahimi Targhi and Dominique Unruh. Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In *Theory of Cryptography - TCC*, pages 192–216, 2016.

A Subroutines of A2B Conversion

For more details on the algorithms and a definition of the subroutines, we refer to the original publications [CGV14] and [BBE+18].

Algorithm 6 `Expand` [CGV14]

```
Input: $(x_i)_{1\leq i\leq n} \in \mathbb{F}_2$
Output: $(y_i)_{1\leq i\leq 2n} \in \mathbb{F}_2$ such that $\bigoplus_{i=1}^{2n} y_i = \bigoplus_{i=1}^{n} x_i$
1: $(r_i)_{1\leq i\leq n} \overset{\$}{\leftarrow} \mathbb{F}_2$
2: $(y_i)_{1\leq i\leq n} \leftarrow (x_i \oplus r_i)_{1\leq i\leq n}$
3: $(y_{2i+1})_{1\leq i\leq n} \leftarrow (r_i)_{1\leq i\leq n}$ return $(y_i)_{1\leq i\leq 2n}$
```

Algorithm 7 `SecAdd` [BBE+18]

```
Input: $\mathbf{x} = (x_i)_{1\leq i\leq n} \in \mathbb{F}_{2^k}$, $\mathbf{y} = (y_i)_{1\leq i\leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i x_i = x$, $\bigoplus_i y_i = y$
Output: $\mathbf{z} = (z_i)_{1\leq i\leq n} \in \mathbb{F}_{2^k}$ such that $\bigoplus_i z_i = x + y \mod 2^k$
1: $\mathbf{p} \leftarrow \mathbf{x} \oplus \mathbf{y}$
2: $\mathbf{g} \leftarrow \texttt{SecAnd}(\mathbf{x}, \mathbf{y})$
3: for $j = 1$ to $W = \lceil \log_2(k-1) \rceil - 1$ do
4:     $\text{pow} \leftarrow 2^{j-1}$
5:     $\mathbf{a} \leftarrow \mathbf{g} << (\text{pow})$
6:     $\mathbf{a} \leftarrow \texttt{SecAnd}(\mathbf{a}, \mathbf{p})$
7:     $\mathbf{g} \leftarrow \mathbf{g} \oplus \mathbf{a}$
8:     $\mathbf{a}' \leftarrow \mathbf{p} << (\text{pow})$
9:     $\mathbf{a}' \leftarrow \texttt{RefreshXOR}(\mathbf{a}', k)$
10:    $\mathbf{p} \leftarrow \texttt{SecAnd}(\mathbf{p}, \mathbf{a}')$
11: end for
12: $\mathbf{a} \leftarrow \mathbf{g} << (2^W)$
13: $\mathbf{a} \leftarrow \texttt{SecAnd}(\mathbf{a}, \mathbf{p})$
14: $\mathbf{g} \leftarrow \mathbf{g} \oplus \mathbf{a}$
15: $\mathbf{z} \leftarrow \mathbf{x} \oplus \mathbf{y} \oplus (\mathbf{g} << 1)$
```