Analysis of Error-Correcting Codes for Lattice-Based Key Exchange

Tim Fritzmann[1], Thomas Pöppelmann[2], and Johanna Sepulveda[1]

[1] Technische Universität München, Germany
{tim.fritzmann,johanna.sepulveda}@tum.de
[2] Infineon Technologies AG, Munich, Germany
thomas.poeppelmann@infineon.com

Abstract. Lattice problems allow the construction of very efficient key exchange and public-key encryption schemes. When using the Learning with Errors (LWE) or Ring-LWE (RLWE) problem such schemes exhibit an interesting trade-off between decryption error rate and security. The reason is that secret and error distributions with a larger standard deviation lead to better security but also increase the chance of decryption failures. As a consequence, various message/key encoding or reconciliation techniques have been proposed that usually encode one payload bit into several coefficients. In this work, we analyze how error-correcting codes can be used to enhance the error resilience of protocols like NewHope, Frodo, or Kyber. For our case study, we focus on the recently introduced NewHope Simple and propose and analyze four different options for error correction: i) BCH code; ii) combination of BCH code and additive threshold encoding; iii) LDPC code; and iv) combination of BCH and LDPC code. We show that lattice-based cryptography can profit from classical and modern codes by combining BCH and LDPC codes. This way we achieve quasi-error-free communication and an increase of the estimated post-quantum bit-security level by 20.39 % and a decrease of the communication overhead by 12.8 %.

Keywords: Post-quantum key exchange · NewHope Simple · Error-correcting codes

1 Introduction

Recently, lattice-based key exchange [9,4,3], public-key encryption (PKE) [22,11] and signature schemes [13,6,7] have attracted great interest due to their performance, simplicity, and practicality. Aside from NTRU [19] and when focusing on ephemeral key exchange and PKE, the Learning with Errors (LWE) problem and the more structured Ring-LWE (RLWE) problem are the main tools to build state of the art schemes. An interesting property of LWE and RLWE is that the

The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-10970-7_17

security of the problem depends on the dimension of the underlying lattices but also on the size and shape of the distribution used to generate random secret and error elements. When constructing key exchange or PKE schemes this is critical as error elements cannot always be removed by the communicating parties and can lead to differences in the derived key (in key exchange) or differences in the message (in most PKE instances). Thus, small differences in the shared key or decrypted message have to be mitigated by encoding techniques or might finally cause a re-transmission or lead to the inability to decrypt a certain ciphertext.

A reduction of the failure probability by using a better encoding opens up the possibility to (a) increase the LWE/RLWE secret and error terms and thus to strengthen security or (b) to decrease the size of ciphertexts, or in general exchanged data, by removing more information. Moreover, it is important to distinguish between the requirements for ephemeral key exchange and PKE schemes. For ephemeral key exchange, a higher failure probability may be acceptable (e.g., around 2^{-40}) because key agreement errors do not affect the security of the scheme. In the presence of errors, the two parties can just repeat the key exchange process. The issue of decryption errors is more critical when using LWE or RLWE-based schemes to instantiate a PKE scheme. The basic LPR10 [24] scheme is only considered appropriately secured with respect to adaptive chosen plaintext attacks (CPA), which is usually not sufficient in a setting where an adversary has access to a decryption oracle. A commonly used tool for transforming a CPA-secured PKE into a scheme secured against chosen-ciphertext attacks (CCA) is the Fujisaki-Okamoto transformation [15,31]. However, a CCA secured cryptosystem using this transformation requires a decryption/decoding routine with a negligible error rate because an attacker could exploit decryption errors. To increase the resilience against attacks exploiting decryption errors, the failure rate is desired to be lower than 2^{-128}. As in Frodo [2] and Kyber [5], in this work we aim for a failure rate lower than 2^{-140} to have a sufficient margin on the error probability. Note that existing works, such as Hila5 [29] and LAC [23], use an independence assumption to calculate the protocol's failure rate. This assumption is related to the correlation between the coefficients of the error term in LWE/RLWE based schemes. The effect of this correlation on the failure rate is still an open research question and it is not in the scope of this work (see also Section 3.2 in [28] for a discussion). However, to decrease decryption errors without decreasing the security of the underlying lattice problem, the reconciliation and en-/decoding techniques are important.

Concurrent to our work, the lattice-based algorithms Hila5 [29], KCL [32], ThreeBears [18] and LAC [23] where developed. They explicitly use forward error correction to achieve better resilience against decryption errors. Except of LAC, which uses a powerful Bose-Chaudhuri-Hocquenghem (BCH) code, all aforementioned schemes apply an error correction that is only capable of correcting a few errors. In this work, we investigate the applicability of more elaborated and modern codes for lattice-based cryptography. Generally, error-correcting codes can be applied in LWE/RLWE schemes when the exchanged key (or message) is chosen by only one of the parties. For example, Frodo, Kyber and NewHope Simple can benefit from the application of powerful error-correcting codes[1]. For our case study, we focus on the RLWE-based NewHope Simple scheme [3], which was submitted with small changes to NIST's call for post-quantum proposals [25]. Compared to an earlier version (called just NewHope) [4], NewHope Simple features a simpler message encoding scheme that uses an additive threshold encoding algorithm and which exhibits a failure rate of less than 2^{-61}. Note that the version of NewHope submitted to the NIST process reaches a failure rate lower than 2^{-140}. This was achieved by reducing the variance of the error distribution. However, in this paper we analyzed different approaches to reduce the failure rate without decreasing the level of security.

Contribution. In this work, we perform an exploration of more powerful error-correcting codes for key exchange mechanisms in order to obtain a quasi-error-free communication and to improve important performance parameters. Our work intensively studies the behaviour of the failure rate when different error-correcting codes and security parameters are applied. For the first time, modern codes, more specifically low-density parity-check (LDPC) codes, are used in this context and compared with the performance of classical BCH codes. In general, the results of the exploration of the design space show that there are several design decisions that make it possible to decrease the failure rate to a value lower than 2^{-140}, increase the security and decrease the communication overhead between the two parties. The selection of a coding option is driven by the requirements of the application. In addition, regarding the protocol's failure rate calculation, we extend the works of [10], [9] and [28], to apply the approach to NewHope Simple. Additionally, we provide first benchmark results. However, we leave the optimization of the implementations with regard to cache and timing attacks to future work as we focus on the exploration of the large design space.

2 NewHope Simple

NewHope Simple, proposed by Alkim, Ducas, Pöppelmann and Schwabe in 2016 [3] as a simplification of NewHope [4], is a lattice-based key exchange, or more specifically a key encapsulation mechanism (KEM), that is built upon the RLWE problem. It allows two entities (Alice and Bob) to agree on a 256-bit shared key μ that is selected by Bob. In the following subsections, the description, security considerations and parameters of NewHope Simple are summarized.

2.1 Notation

Let $\mathcal{R} = \mathbb{Z}_q[x]/(x^n + 1)$ be a ring of integer polynomials. All elements of the ring $\mathcal{R}$ can be written in the form $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1}$, where the integer coefficients $a_0, a_1, \ldots, a_{n-1}$ are reduced modulo q. We write $a \xleftarrow{\$} S$

[1] In order to apply error-correcting codes, some changes in the protocol may be necessary, e.g. different parameter selection and/or encoding/decoding functions.

for sampling a value a from the distribution S, where sampling means to take a random value from a set S. Let Ψ_k be a binomial distribution with parameter k. The distribution is determined by $\Psi_k = \sum_{i=0}^{k-1} b_i - b'_i$, where $b_i, b'_i \in \{0,1\}$ are uniform independent bits. The binomial distribution is centered with a zero mean, approximates a discrete Gaussian, has variance $k/2$, and gives a standard deviation of $\psi = \sqrt{k/2}$.

2.2 Protocol

Protocol 1 shows the underlying algorithm of NewHope Simple. Eight steps are highlighted due to the relevance to the present work. For a more detailed description of the algorithm and for details about the application of the CCA transformation, we refer the reader to [3] and [1].

Alice (server):		**Bob (client):**
① seed $\xleftarrow{\$} \{0,1\}^{256}$ $a \leftarrow$ Parse(SHAKE(seed))		
② $s, e \xleftarrow{\$} \Psi_{16}^n$		② $s', e', e'' \xleftarrow{\$} \Psi_{16}^n$
③ $b \leftarrow as + e$	$\xrightarrow{m_a = \text{encodeA}(b,\text{seed})}$	③ (b,seed) $\leftarrow$ decodeA(m_a) $a \leftarrow$Parse(SHAKE(seed)) ④ $v \xleftarrow{\$} \{0,1\}^{256}$ $d \leftarrow$ NHSEncode(v) ⑤ $u \leftarrow as' + e'$ $c \leftarrow bs' + e'' + d$
⑥ $(u, \bar{c}) \leftarrow$ decodeB(m_b) $c' \leftarrow$ NHSDecompress($\bar{c}$) ⑦ $d' \leftarrow c' - us$ $v' \leftarrow$ NHSDecode(d') ⑧ $\mu \leftarrow$ SHA3-256(v')	$\xleftarrow{m_b = \text{encodeB}(u,\bar{c})}$	⑥ $\bar{c} \leftarrow$ NHSCompress(c) ⑧ $\mu \leftarrow$ SHA3-256(v)

Protocol 1. NewHope Simple protocol. All polynomials are elements of the ring $\mathcal{R} = \mathbb{Z}_q[x]/(x^n + 1)$, where $n = 1024$ and $q = 12289$ [3].

1. Alice samples the seed from a random number generator. The seed is expanded with the SHAKE-128 extendable-output function. The expanded seed is used to generate the public polynomial a.
2. Alice and Bob sample the coefficients of the secret polynomials s and s', and the error polynomials e, e' and e'' according to the error distribution Ψ_k.
3. Alice calculates $b = as + e$ and sends it together with the seed to Bob. Extraction of the secret s from b is hard due to the error term e and because b is exactly an RLWE instance. Similar to Alice, Bob can use the seed to generate the public polynomial a.
4. Bob samples 256 bits from a random number generator and assigns them to the secret key vector v. Then, Bob encodes v into the most significant bit of

the coefficients of polynomial $d = \mathrm{NHSEncode}(v)$. The function NHSEncode, which is given in Appendix A, maps one bit of v into four coefficients of d. This redundancy is used by the NHSDecode function in Step 7 to average out small errors.

5. Bob calculates $u = as' + e'$ and hides the secret key polynomial d in $c = bs' + e'' + d = ass' + es' + e'' + d$. The polynomials u and c are again instances of the RLWE problem.
6. Bob sends to Alice the polynomial u and the compressed polynomial $\bar{c}$. The goal of the compression of polynomial c is the reduction of the communication overhead between Alice and Bob.
7. Alice removes the large noise term ass' from the decompressed polynomial c' by calculating $d' = c' - us \approx bs' + e'' + d - (as' + e')s = ass' + es' + e'' + d - ass' - e's = (es' - e's) + e'' + d$. Alice obtains the term v' after decoding d', using the function NHSDecode, which is also provided in Appendix A.
8. After the decoding, Alice and Bob can use v' and v, respectively, as input for the SHA3-256 function to obtain the shared key.

The functions NHSEncode and NHSDecode of NewHope Simple build an error-correcting code, which is used to remove small errors and to increase the probability that Alice and Bob share a similar key. For the remainder of this paper, this error-correcting code is denoted as *additive threshold encoding algorithm*.

2.3 Security of NewHope Simple

The security level of NewHope Simple depends on three parameters: the dimension n of the ring, the modulus q, and the parameter k that determines the standard deviation of the noise distribution Ψ_k. In this work, the parameters n and q are not modified. Only k is used to improve the security of NewHope Simple as larger noise also leads to a higher security level. For determining the security level, the test script and methodology[2] from [4] can be used.

2.4 Noise Sources of the Protocol

NewHope Simple contains two noise sources: the *difference noise* and the *compression noise*. As noise we define all terms that have an influence on the correctness of the decryption/decoding or reconciliation mechanisms. The reason is that we can model the distortion caused by the convolutions of the secret and error polynomials as noise that is added to encoded data transmitted over a channel.

The *difference noise* emerges from the design of the protocol. Alice is able to remove the strongest noise term ass' from polynomial c (Step 7), but a small noise term remains. This noise term is called difference noise and is equal to $(es' - e's) + e''$. The coefficients of the secret and error polynomials are sampled from the error distribution Ψ_k. When k is increased, the probability of receiving a stronger difference noise increases as well.

[2] Script *PQsecurity.py* in `https://cryptojedi.org/crypto/#newhope`

The *compression noise* is introduced by the function NHSCompress (Step 6). It compresses the polynomial $c = ass' + es' + e'' + d$ to reduce the communication overhead between Alice and Bob. This is possible as lower-order bits carry a high amount of noise and have low information content. To remove such lower order bits, a coefficient-wise modulus switching between the security parameter q and 2^r is performed, where r is the number of remaining bits. To reduce the number of transmitted bytes between Alice and Bob, the transmitted polynomials b, c and u can be compressed. In the original implementation of NewHope Simple, the compression is only applied on polynomial c, where each coefficient of c is reduced from 14 bits to 3 bits. In this work, we further reduce the communication overhead by compressing polynomial u as in Kyber [10]. To obtain a moderate compression noise, a weaker compression on the coefficients of polynomial u has to be applied. As the uniformly distributed compression noise of u is multiplied with the binomially distributed secret s, the compression noise of u gets magnified.

3 Failure Rate of NewHope Simple

In the original implementation of NewHope Simple, the failure rate is bounded applying the Cramér-Chernoff inequality [3]. This approach provides a probability bound that can be far away from the real failure probability. Previous works, such as Frodo [9], Kyber [10] and Hila5 [28], use probability convolutions to determine the probability distribution of the difference between the keys of Alice and Bob. With the probability distribution of the difference, it is possible to derive the protocol's failure rate. In the following subsections, we shortly explain how to calculate the probability distributions of the two noise terms, *difference noise* and the *compression noise*, mentioned in Subsection 2.4, and how to calculate the failure rate by a given error distribution.

3.1 Mathematical Operations with Random Variables

In this subsection, the mathematical background for determining the probability distributions of the *difference noise* and the *compression noise* is given.

NewHope Simple uses a binomial distribution for sampling secret and error polynomials. The probability mass function of a binomial random variable (RV) X is $f(i) = Pr(X = i) = \binom{l}{i} p^i (1-p)^{l-i}$ for $i = 0, 1, \ldots, l$. For NewHope Simple, $p = 0.5$ and l is equal to the error distribution parameter k multiplied by two.

Let us define in Theorem 1 the probability distribution of the addition and in Theorem 2 the probability distribution of the multiplication of two independent RVs. Since in NewHope Simple polynomial instead of conventional multiplications are required, we define in Theorem 3 the polynomial product distribution. The proof for Theorem 3 can be found in Appendix B.

Theorem 1 (Addition of random variables). *Let $\Psi_X(x)$ and $\Psi_Y(y)$ be two probability distributions of the independent random variables X and Y. Then the*

Table 1. Calculating distribution of $d = (es' - e's) + e''$

Step	Action	Result
Step 1	product distribution of two RVs sampled from Ψ_k	Ψ_Z
Step 2	n-fold convolution of the product distribution	es'
Step 3	convolve distribution of es' with itself	$(es' - e's)$
Step 4	convolve distribution of $(es' - e's)$ with Ψ_k	$(es' - e's) + e''$

probability distribution of the sum of both random variables corresponds to the convolution of the individual probability distributions, which can be written as $\Psi_{X+Y} = \Psi_Z(z) = \Psi_X(x) \circledast \Psi_Y(y)$ *[17].*

Theorem 2 (Product distribution). *Let $\Psi_X(x)$ and $\Psi_Y(y)$ be two probability distributions of the independent random variables X and Y. Then the product distribution $\Psi_Z(XY = c) = \sum_{x \in X, y \in Y s.t. xy=c} \Psi_X(x)\Psi_Y(y)$.*

Theorem 3 (Polynomial product distribution). *Let a and b be two polynomials of a ring $\mathcal{R}_q$ with rank n and with independent random coefficients sampled from Ψ_k and let c be the result of the polynomial multiplication of a and b. Then the probability distribution of a random coefficient of c is equal to the n-fold convolution of the product distribution Ψ_Z of two random variables sampled from Ψ_k.*

3.2 Probability Distributions of Difference and Compression Noise

Difference Noise. The partial steps for calculating the probability distribution of the difference term are summarized in Table 1. Note that all calculated probability distributions are related to a single coefficient of a polynomial. The probability distribution of the polynomial product es' can be described as an n-fold convolution of the product distribution of two RVs sampled from Ψ_k. In our case, the probability distributions of an addition and subtraction of two RVs are equal because the RVs are sampled from a symmetrical distribution that is centered at zero. To obtain the probability distribution for $(es' - e's)$, we convolve the probability distribution of es' with itself. Finally, we convolve the distribution of e'' with the result to obtain the probability distribution of $(es' - e's) + e''$.

Compression Noise. The probability distribution of the compression noise can be calculated similar to the probability distribution of the difference noise. The polynomial $c = ass' + es' + e'' + d$ consists of the uniformly distributed public parameter a, some terms sampled from the error distribution and the secret key d. Depending on the respective key bit, the coefficients of polynomial d are either zero or $\lfloor q/2 \rfloor$. Both values, zero and $\lfloor q/2 \rfloor$, are not affected by the compression. They can be compressed and decompressed without any loss of information. Consequently, the compression noise is only dependent on the term

$c_{\text{uncompressed}} = ass' + es' + e''$. The coefficients of the secret and error polynomials are sampled from Ψ_k and the coefficients of a are sampled from a uniform distribution U_q with outcomes between 0 and $q-1$ (after modulus reduction). In Figure 11 (Appendix C) the probability distribution of the difference and compression noise is plotted.

3.3 From the Noise Distribution to the Failure Rate

Note that the coefficients of the product of two polynomial ring elements are correlated and not independent anymore. This correlation does not influence the validity of *Theorem 3* and the calculations done in Subsection 3.2 because there the calculations are related to a single coefficient. To determine the failure rate, we apply arithmetic operations on correlated coefficients and thus assume that the correlation between the coefficients has a negligible influence to the final result. The experiments discussed in Appendix D have shown that this assumption is valid at least for high failure rates. The independence assumption is also required in Equation 2 (Subsection 5.2) to calculate the failure rate of NewHope Simple with a t-bit error-correcting code. To have a safety margin, we aim for a failure rate of 2^{-140} instead of 2^{-128}.

In order to determine the failure rate, given a noise distribution, a closer look at the NHSDecode function must be taken. During the decoding, when one bit is mapped into four coefficients, the absolute values of the four coefficients that are subtracted by $\lfloor q/2 \rceil$ are summed up. This decoding step is done for all outcomes of the overall error distribution (convolution of difference and compression noise distribution). First, the values of all outcomes are subtracted by $\lfloor q/2 \rceil$ and the absolute values are built. Let us denote the resulting error distribution as Ψ_{dec}. In the next step, we convolve the distribution of four coefficients $\Psi'_{dec} = \Psi_{dec} \circledast \Psi_{dec} \circledast \Psi_{dec} \circledast \Psi_{dec}$. Note again that for this step we assume that the correlation between the coefficients is negligible. To obtain the bit error rate (BER) of NewHope Simple, all probabilities of the outcomes of Ψ'_{dec} that are lower than or equal to q are summed up. The BER can be multiplied with the secret key length, in our case 256, to compute the union bound. The union bound is the upper bound for the block error rate (BLER) or failure rate of the protocol.

4 Error-Correcting Codes

4.1 Modern and Classical Error-Correcting Codes

Error-correcting codes are an essential technique for realizing reliable data transmissions over a noisy channel. In this work, error-correcting codes are used to mitigate the influence of the difference and compression noise on the failure probability of RLWE based key exchange protocols. Instead of the additive threshold encoding, which is used in the original NewHope Simple scheme, in this work we explore the effect of using more powerful error-correcting codes. The design objectives for the error-correcting code are: i) good error-correcting capability, to increase the security or decrease the amount of exchanged data; ii) low failure rate, to avoid repetition of the protocol and to apply CCA transformation; and iii) reasonable time complexity. The additive threshold encoding has usually a weak error-correcting capability and cannot efficiently achieve low failure rates for certain noise levels. Therefore, more powerful classical[3] and modern[4] codes can be used. The drawback of using powerful error-correcting codes is the increase of computation time.

Modern codes have a strong error-correcting capability and can get close to the channel capacity for long code lengths. The most commonly used error-correcting codes belonging to the class of modern codes are LDPC and Turbo codes. In comparison to Turbo codes, LDPC codes usually have a lower time complexity since they do not require long interleavers and can abort the iteration loop when a correct codeword is found [14]. Moreover, their error floor occurs at lower failure rates [21]. The error floor is a phenomenon of some modern codes that limits the performance for low failure rates. That is, the channel capacity can only be very closely approached for moderate failure rates. Since the goal is to have a low (or even no) error floor and to keep the time complexity low, in this work we select LDPC instead of Turbo codes for obtaining a high error-correcting capability.

The advantages of classical error-correcting codes are the lack of error floor and that the number of correctable errors can be determined during the construction of the code. When the number of correctable errors is known, the performance of the code can be calculated, otherwise, simulations are required. In contrast to classical codes, where the number of correctable errors is known, for modern codes this value is unknown. However, it has been demonstrated by simulation that modern codes achieve a higher error-correcting capability, when compared to the classical approach.

There are a large number of classical error-correcting codes, e.g. Hamming, Reed Muller and BCH codes. Among this alternatives, BCH codes are widely spread in real world applications because of their good performance, the ability to correct multiple errors and their flexibility in terms of code length and code rate. These characteristics motivate us to use BCH codes in the protocol to achieve very low failure rates.

To reach both a high error-correcting performance and a very low failure rate, usually different codes are concatenated. The concatenation of BCH and LDPC codes is a common method, which is used, for example, in the second generation of the digital video broadcast standard for satellite (DVB-S2).

4.2 BCH Codes

BCH codes are a class of powerful classical error-correcting codes that were discovered in 1960. The code length of a BCH code must be $n = q^m - 1$, where $m \in \mathbb{Z}$ is greater or equal to three and q equal to two for the binary BCH codes.

[3] Classical codes are described by algebraic coding theory.

[4] Modern codes have a new approach based on probabilistic coding theory.

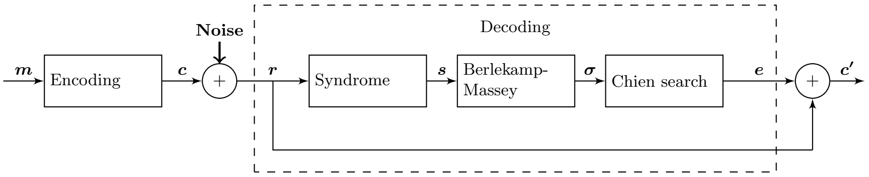

Fig. 1. BCH error correction

There exists a BCH code for any valid code length and any positive integer $t < 2^{m-1}$, where t denotes the number of correctable errors [21]. Figure 1 illustrates the encoding and decoding process of BCH codes. During the encoding, the codeword c is built out of a message m. In the noisy channel, noise is added to the transmitted codeword. At NewHope Simple, this would be the *difference* and *compression noise*. The decoder is used to correct multiple errors in the received codeword r. Generally, the decoding process consists of three parts: determining the syndrome s, error locator polynomial σ and the zeros of σ. Berlekamp's algorithm, which was proposed in 1966, is an efficient method for determining the error locator polynomial [8]. The error polynomial e can be determined by finding the zeros of the error locator polynomial with the Chien search algorithm [12]. The predicted codeword c' is calculated by taking r xor e.

4.3 LDPC Codes

LDPC codes were developed by Gallager in 1962 [16]. They have become attractive since the 90's, when the required computational power has been available. Figure 2 shows a block diagram of an LDPC code. LDPC codes are characterized by its parity check matrix H, which has, in case of LDPC codes, a low density, i.e. a low number of ones. For the encoding, usually, the systematic form of H is computed to derive the generator matrix. With the generator matrix it is possible to calculate the codeword c by a given message m. After transmitting c through the noisy channel, the receiver obtains a noisy codeword r.

The sum-product algorithm is a very efficient soft decision message-passing decoder. It takes as input a parity check matrix, the maximum number of iterations and the log-likelihood ratios (LLR) of the received codeword r. To visualize the decoding process, the Tanner Graph representation of the parity check matrix is used. This representation consists of a bipartite graph with check nodes (CN) and variable nodes (VN), which represent the rows and columns of H, respectively. The sum-product algorithm iteratively sends LLR messages from variable nodes to check nodes and vice versa until a correct codeword is found or the maximum number of iterations is reached. A full description of the algorithm can be found in works like [20] and [26].

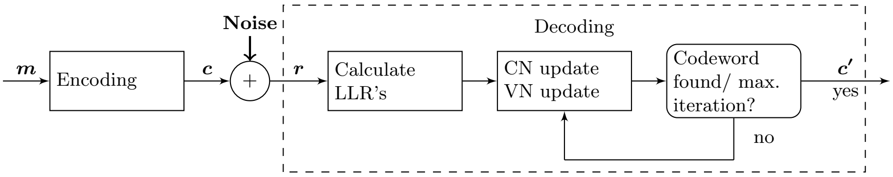

Fig. 2. LDPC error correction with sum-product algorithm

4.4 Error-Correcting Codes for NewHope Simple

To meet the requirements mentioned in Subsection 4.1, we use LDPC codes to maximize the error-correcting capability and BCH codes to achieve very low error rates. In the following paragraphs, we investigate four design options that make use of various combinations of these codes. The respective advantages and disadvantages are summarized in Table 2.

Table 2. Summary of explored coding options

Option	Coding technique	Advantages	Disadvantages
Option 1	BCH	Good error correction	Computationally expensive
Option 2	BCH and additive threshold enc.	Speed up of Option 1 (lower Galois field)	Weaker error correction compared to Option 1
Option 3	LDPC	Closer to channel capacity	Does not achieve very low error rates
Option 4	LDPC and BCH	Lower error rates than Option 3 achievable	Computationally expensive

Option 1. For Option 1, we use a BCH(1023,258) for the error correction. The BCH encoder builds the codeword out of 256 secret key bits, 765 redundancy bits and 2 padding bits. By using the NHSEncode function (Step 4 in Protocol 1), each of the 1023 code bits is mapped to one coefficient of d. Then, in the NHSDecode function (Step 7 in Protocol 1), the coefficients are mapped back to the received codeword with a hard threshold. Finally, the BCH decoder corrects up to 106 bit errors and returns the estimated secret key vector.

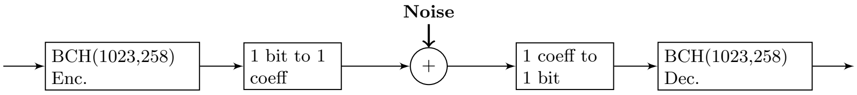

Fig. 3. Option 1, block diagram BCH(1023,258)

Option 2. For Option 2, we use a BCH(511,259) as outer code and the additive threshold encoding as inner code. In this case, the BCH code uses 252 bits of redundancy in order to correct up to 30 errors. The additive threshold encoding has as input 512 bits (BCH code length with one padding bit). These bits are mapped to 1024 coefficients, resulting into a redundancy of 512 bits. With the additive threshold encoding, it is expected that even more than 30 errors are correctable. In comparison to Option 1, this option is faster because it only requires calculations in GF(2^9). The drawback of this approach is a lower error-correcting capability at the target failure rate (2^{-140}), as shown in Subsection 5.2.

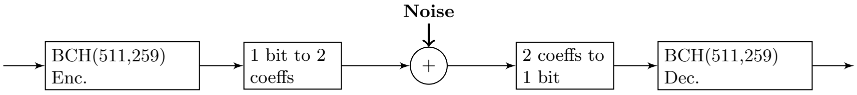

Fig. 4. Option 2, block diagram BCH(511,259) + additive threshold encoding

Option 3. For Option 3, we use an LDPC(1024,256). In this case, all available coefficients are used for the LDPC encoding. Similar to Option 1, one bit is mapped to one coefficient, but within the function NHSDecode, no hard threshold is used. Instead, we apply a transformation on the coefficients in order to allow the usage of the sum-product algorithm. Each received coefficient d_i' is transformed to

$$d_i'' = \frac{4|d_i' - \lfloor q/2 \rfloor|}{q} - 1 \ . \tag{1}$$

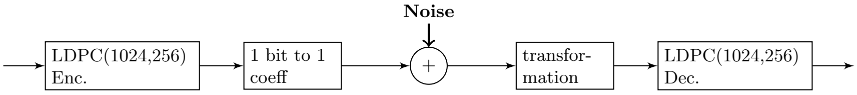

Fig. 5. Option 3, block diagram LDPC(1024,256)

Option 4. For Option 4, we build a concatenation of a BCH(511,259) and an LDPC(1024,512). In this approach, the advantages of BCH and LDPC codes are combined to achieve very low error rates and to get closer to the channel capacity. More specifically, the LDPC(1024,512) is used to remove the strong noise and the BCH(511,259), which can correct up to 30 errors, is applied to remove the remaining errors and thus achieve a very low error rate.

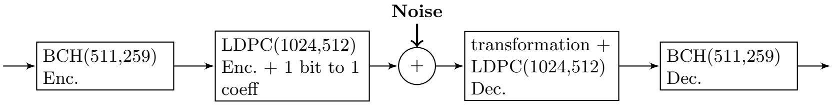
Noise
BCH(511,259) Enc.
LDPC(1024,512) Enc. + 1 bit to 1 coeff
+
transformation + LDPC(1024,512) Dec.
BCH(511,259) Dec.

Fig. 6. Option 4, block diagram BCH(511,259) + LDPC(1024,512)

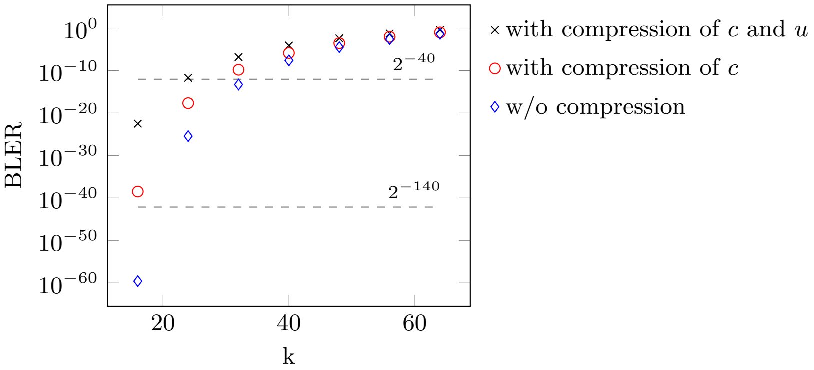

Fig. 7. Influence of compression noise on NewHope Simple's failure rate.

5 Experimental Results

5.1 NewHope Simple Compression Noise

Figure 7 illustrates the influence of the compression noise on the failure rate. The graph shows that the compression has a strong influence on the failure rate for low values of k. For higher values of k, the difference noise dominates. To improve both, security and bandwidth, a balance between difference noise and compression noise has to be found. When applying the error-correcting options described in Subsection 4.4, we found the optimum at a compression of c from 14 to 3 bits per coefficient and a compression of u from 14 to 10 bits per coefficient. Removing even more bits from the coefficients of c and u leads to a significantly higher compression noise.

The curve with compression of c corresponds to the original implementation of NewHope Simple. For a value of $k = 16$, a failure rate of $2^{-127.88} = 3.20 \cdot 10^{-39}$ is determined, whereas in [3] a failure probability lower than $2^{-61} = 4.34 \cdot 10^{-19}$ is claimed. This difference is not surprising because the Cramér-Chernoff bound is based on an exponential inequality. Due to the exponential behavior, even small changes can entail large differences.

5.2 NewHope Simple with BCH Code

When the failure rate of the protocol is known, the improvement using BCH codes can be calculated. The probability that a binary vector of S bits (in our analysis 256) has more than t errors is

$$\text{BLER} = \sum_{i=t+1}^{S} \binom{S}{i} p_b^i (1-p_b)^{S-i} = 1 - \sum_{i=0}^{t} \binom{S}{i} p_b^i (1-p_b)^{S-i} \ , \qquad (2)$$

where p_b denotes the probability of a bit error [30]. Figure 8 shows the improvements with BCH codes. The results show that both BCH variants (Option 1 and

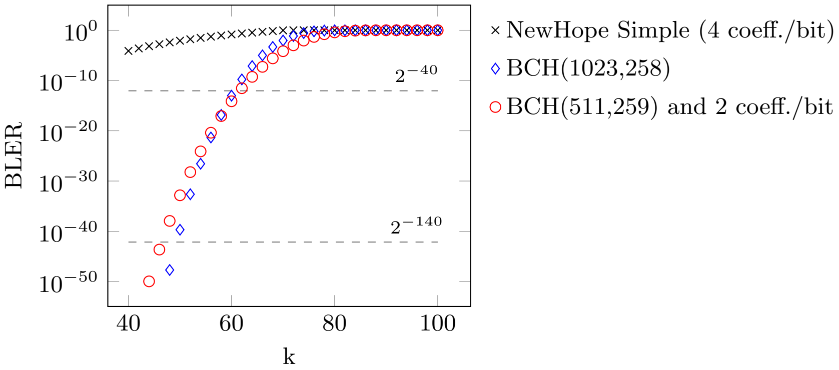

Fig. 8. Improvement of failure rate with Option 1, BCH(1023,258); and Option 2, concatenation of BCH(511,259) and additive threshold encoding. Compression on c and u applied.

Option 2) allow a quasi-error-free communication for k's lower than 46. While NewHope Simple with compression of c and u has a failure rate of $1.69 \cdot 10^{-3}$ for $k = 46$, Option 1 and Option 2 achieve a failure rate of $1.83 \cdot 10^{-57}$ and $2.30 \cdot 10^{-44}$, respectively. In comparison to the original implementation of NewHope Simple, we can choose a much higher k to obtain the same failure rate when BCH codes are used within the protocol.

5.3 NewHope Simple with LDPC Code

When a binary input additive white Gaussian noise channel (BI-AWGNC) is used as channel model and a code length of $n = 1024$ (1023) is chosen, the improvement of the applied LDPC code over the applied BCH code is for the rate 1/2 about 2.8 dB and for the rate 1/4 about 3.8 dB at a BER of 10^{-6}. As a consequence, LDPC codes can get closer to the channel capacity when compared to BCH codes, even with a moderate code length.

Figure 9 compares the original implementation of NewHope Simple (with additional compression of u) with the implementations using an LDPC code (Option 3) and a BCH code (Option 1). The graph shows that LDPC codes can be used to further improve the error-correcting performance. While the BCH(1023,258) begins to operate in the *waterfall region* for k's smaller than 76, the *waterfall region* for the LDPC(1024,256) begins for k's smaller than 92. However, the error floor is expected to limit the performance of the LDPC code for error rates smaller than about 10^{-10} (see analysis in [27]) so that BCH codes perform better in this region. Interesting is also that the *waterfall region* of the additive threshold encoding is less distinct (lower gradient).

5.4 NewHope Simple with Concatenation of BCH and LDPC Code

To achieve very low error rates and get closer to the channel capacity as a pure BCH implementation, the BCH code is combined with an LDPC code

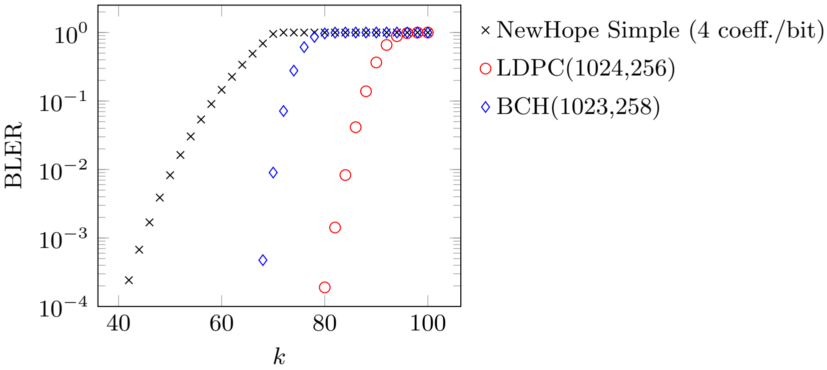

Fig. 9. Improvement of failure rate with Option 3, LDPC(1024,256). Compression on c and u applied.

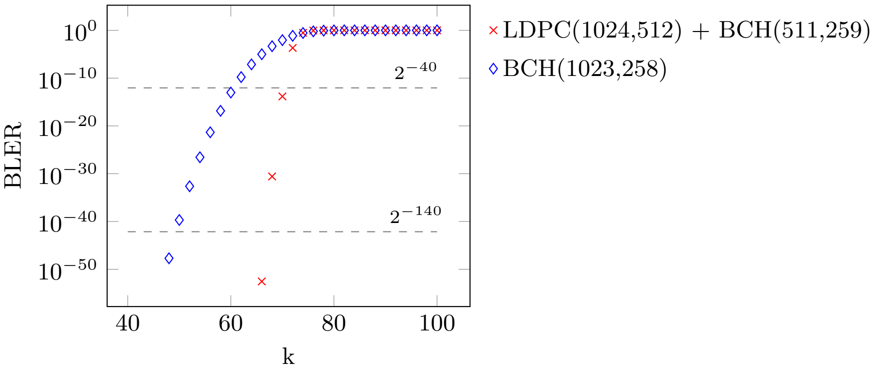

Fig. 10. Improvement of failure rate with Option 4, concatenation of LDPC(1024,512) and BCH(511,259). Compression on c and u applied.

(Option 4). Figure 10 illustrates the performance of the concatenation of the LDPC(1024,512) and the BCH(511,259).

5.5 Comparison Coding Options

Table 3 summarizes the results of the different coding options. Our analysis shows that NewHope Simple, with the original parameter set, has a much lower failure rate than expected. However, to increase the security and decrease the bandwidth, stronger error-correcting codes have to be applied. To achieve a failure rate of 2^{-140}, parameter k is set for Option 1, Option 2 and Option 4 to 48, 46 and 66, respectively. Since we cannot prove such an error rate for the pure LDPC implementation, we chose a higher failure rate for Option 3. Although Option 1 has a better security strength, we recommend Option 2 because it requires calculations in GF(2^9) instead of GF(2^{10}). This reduces the time complexity. For moderate failure rates, Option 3 achieves the best error-correcting capability, but for failure rates lower than about 10^{-10} the error floor

Table 3. Comparison error correction options

Coding option	Failure rate	k	Security classical/ quantum	Exchanged bytes
NewHope Simple [3]	$2^{-127.88}$ [a)]	16	281/255 bits	4,000
Option 1, BCH(1023,258)	$< 2^{-140}$	48	324/294 bits	3,488
Option 2, BCH(511,259) + 1 bit to 2 coeffs.	$< 2^{-140}$	46	323/292 bits	3,488
Option 3, LDPC(1024,256)	$< 2^{-12}$ [b)]	80	348/315 bits	3,488
Option 4, LDPC(1024,512) + BCH(511,259)	$< 2^{-140}$	66	338/307 bits	3,488

[a)] In the reference, NewHope Simple provides a failure rate of lower than 2^{-61}. This bound was determined using the Cramér-Chernoff inequality. With our approach, we determine a failure rate of $3.20 \cdot 10^{-39} = 2^{-127.88}$.
[b)] With Option 3, a failure rate of $\approx 10^{-10} = 2^{-33.22}$ can be efficiently reached.

limits the performance. Option 4 cannot get as close to the channel capacity as Option 3, but it achieves extremely low error rates. With Option 4, we can realize an error rate of 2^{-140}, an increase of the post quantum security by 20.39 % and a decrease of the communication overhead by 12.80 %. If k and thus the security level is left unchanged and only the compression on u is increased, the communication overhead can be reduced with Option 4 by 19.20 %.

5.6 Benchmark

This section summarizes the run times of the applied algorithms. Table 4 provides an overview of the determined results. All tests were performed on an Intel Core i7-6700HQ (Skylake), which runs at 2.6 GHz (turbo boost disabled). The C-code was compiled with gcc (version 5.4.0) and flags *-O3 -fomit-frame-pointer -march=corei7-avx -msse2avx.* In comparison to NewHope Simple, the time complexity increases for Option 1 by 238 %; for Option 2 by 40 %; for Option 3 by 6462 % (when $k = 80$); and for Option 4 by 4455 % (when $k = 66$). Option 2 has a relatively small overhead, thus being suitable for applications that require a low time complexity. The costs for the other options are quite high. However, as NewHope Simple is implemented very efficiently and is already very fast, the time overhead can be acceptable. The decoding complexity of LDPC codes depends on the parameter k. To decrease the run time, k can be decreased. Moreover, the min-sum algorithm can be used instead of the sum-product algorithm. Thus, the complexity is reduced at the cost of a lower decoding performance.

Table 4. Benchmark: median and average clock cycles with 1000 test rounds

	Function	**Cycles median/average**
NewHope Simple:	KeyGen (server)	223952/ 225452
	KeyGen+shared key (client)	353201/ 358821
	Shared key (server)	78216/ 78614
BHC(511,259):	Encoding	104520/ 108738
	Decoding	157704/ 154652
BCH(1023,258):	Encoding	298043/ 302021
	Decoding	1259554/ 1206814
LDPC(1024,512):	Encoding	2069582/ 2073136
	Decoding ($k = 66$)	26862391/ 27464623
LDPC(1024,256):	Encoding	2068959/ 2071198
	Decoding ($k = 80$)	40282855/ 41912347

6 Conclusion and Future Work

Our analysis has shown that powerful error-correcting codes within lattice-based key exchange protocols can lead to a significant improvement of important performance parameters, such as failure rate, security level and bandwidth. Modern codes, e.g. LDPC codes, can be used to get a high error-correcting capability. However, to obtain very low error rates, classical codes, e.g. BCH codes, should be employed. The concatenation of LDPC and BCH codes combines their advantages to achieve a quasi-error-free key exchange with a high error-correcting capability. With quasi-error-free communication, the CCA transformation can be applied in order to allow protocols, like NewHope Simple, to be also used for encryption. Before LDPC and BCH codes are used in encryption schemes, it is necessary to investigate these codes with respect to the vulnerability to attackers. For instance, constant-time implementations may be challenging. The selection of the encoding technique is driven by the application characteristics. Many applications may not require or may not be able to integrate powerful error-correcting codes. Different application may benefit from the reduction of data transmission by using strong error-correcting codes, even if the computation time increases. Examples are battery-powered wireless devices, where the radio module usually represents a substantial portion of the overall energy consumption.

Acknowledgments. We thank the anonymous reviewers for their valuable comments and suggestions. This work was partly funded by the Fraunhofer High Performance Center for Secure Connected Systems of Munich.

References

1. Alkim, E., Avanzi, R., Bos, J., Ducas, L., de la Piedra, A., Pöppelmann, T., Schwabe, P., Stebila, D.: NewHope: Algorithm Specifcations and Supporting Doc-

umentation (2017), https://newhopecrypto.org/data/NewHope_2017_12_21.pdf

2. Alkim, E., Bos, J.W., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., Stebila, D., Easterbrook, K., LaMacchia, B.: FrodoKEM - Learning With Errors Key Encapsulation: Algorithm Specifications And Supporting Documentation (2017), https://frodokem.org/files/FrodoKEM-specification-20171130.pdf
3. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: NewHope without reconciliation. IACR Cryptology ePrint Archive 2016, 1157 (2016)
4. Alkim, E., Ducas, L., Pöppelmann, T., Schwabe, P.: Post-quantum key exchange - A new hope. In: 25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016. pp. 327–343 (2016)
5. Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS-Kyber: Algorithm Specifications And Supporting Documentation (2017), https://www.pq-crystals.org/kyber/data/kyber-specification.pdf
6. Bai, S., Galbraith, S.D.: An Improved Compression Technique for Signatures Based on Learning with Errors, pp. 28–47. Springer International Publishing, Cham (2014), doi:10.1007/978-3-319-04852-9_2
7. Barreto, P.S., Longa, P., Naehrig, M., Ricardini, J.E., Zanon, G.: Sharper Ring-LWE signatures. IACR Cryptology ePrint Archive 2016, 1026 (2016)
8. Berlekamp, E.R.: Nonbinary BCH decoding. In: International Symposium on Information Theory. San Remo, Italy (1966)
9. Bos, J.W., Costello, C., Ducas, L., Mironov, I., Naehrig, M., Nikolaenko, V., Raghunathan, A., Stebila, D.: Frodo: Take off the ring! practical, quantum-secure key exchange from LWE. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016. pp. 1006–1018 (2016), doi:10.1145/2976749.2978425
10. Bos, J.W., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J.M., Schwabe, P., Stehlé, D.: CRYSTALS - Kyber: a CCA-secure module-lattice-based KEM. IACR Cryptology ePrint Archive 2017, 634 (2017)
11. Cheon, J.H., Kim, D., Lee, J., Song, Y.S.: Lizard: Cut off the tail! // practical post-quantum public-key encryption from LWE and LWR. IACR Cryptology ePrint Archive 2016, 1126 (2016)
12. Chien, R.T.: Cyclic decoding procedures for Bose-Chaudhuri-Hocquenghem codes. IEEE Trans. Information Theory 10(4), 357–363 (1964), doi:10.1109/TIT.1964.1053699
13. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal Gaussians. Cryptology ePrint Archive, Report 2013/383 (2013)
14. Fan, J.: Constrained Coding and Soft Iterative Decoding. The Springer International Series in Engineering and Computer Science, Springer US (2012)
15. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings. pp. 537–554 (1999), doi:10.1007/3-540-48405-1_34
16. Gallager, R.G.: Low-density parity-check codes. IRE Trans. Information Theory 8(1), 21–28 (1962), doi:10.1109/TIT.1962.1057683
17. Gitlin, R., Hayes, J., Weinstein, S.: Data Communications Principles. Applications of Communications Theory, Springer US (2012)
18. Hamburg, M.: Supporting documentation: ThreeBears (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions

19. Hoffstein, J., Pipher, J., Silverman, J.H.: NTRU: A ring-based public key cryptosystem. In: Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings. pp. 267–288 (1998), doi:10.1007/BFb0054868
20. Hu, X., Eleftheriou, E., Arnold, D., Dholakia, A.: Efficient implementations of the sum-product algorithm for decoding LDPC codes. In: Proceedings of the Global Telecommunications Conference, 2001. GLOBECOM '01, San Antonio, TX, USA, 25-29 November, 2001. p. 1036 (2001), doi:10.1109/GLOCOM.2001.965575
21. Lin, S., Costello, D.J.: Error Control Coding, Second Edition. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (2004)
22. Lindner, R., Peikert, C.: Better Key Sizes (and Attacks) for LWE-Based Encryption, pp. 319–339. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), doi:10.1007/978-3-642-19074-2_21
23. Lu, X., Liu, Y., Jia, D., Xue, H., He, J., Zhang, Z.: Supporting documentation: LAC (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions
24. Lyubashevsky, V., Peikert, C., Regev, O.: On ideal lattices and learning with errors over rings. In: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings. pp. 1–23 (2010), doi:10.1007/978-3-642-13190-5_1
25. National Institute of Standards and Technology: Announcing request for nominations for public-key post-quantum cryptographic algorithms (2016), https://csrc.nist.gov/news/2016/public-key-post-quantum-cryptographic-algorithms
26. Qian, C., Lei, W., Wang, Z.: Low complexity LDPC decoder with modified Sum-Product algorithm. Tsinghua Science and Technology 18(1), 57–61 (Feb 2013), doi:10.1109/TST.2013.6449408
27. Richardson, T.: Error floors of LDPC codes. In: Proceedings of the annual Allerton conference on communication control and computing. pp. 1426–1435. The University; 1998 (2003)
28. Saarinen, M.O.: HILA5: On reliability, reconciliation, and error correction for Ring-LWE encryption. IACR Cryptology ePrint Archive 2017, 424 (2017)
29. Saarinen, M.J.O.: Supporting documentation: HILA5 (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions
30. Safak, M.: Digital Communications. Wiley (2017)
31. Targhi, E.E., Unruh, D.: Post-quantum security of the Fujisaki-Okamoto and OAEP transforms. In: Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part II. pp. 192–216 (2016), doi:10.1007/978-3-662-53644-5_8
32. Zhao, Y., Jin, Z., Gong, B., Sui, G.: Supporting documentation: KCL (2017), https://csrc.nist.gov/Projects/Post-Quantum-Cryptography/Round-1-Submissions

A NewHope Simple Algorithms NHSEncode and NHSDecode

Algorithm 1: NHSEncode [3]

```
Input: Randomized vector $v \in \{0,1\}^{256}$
Result: Polynomial $d \in R_q$
for $i$ from 0 to 255 do
    $d_i \leftarrow v_i \lfloor q/2 \rfloor$
    $d_{i+256} \leftarrow v_i \lfloor q/2 \rfloor$
    $d_{i+512} \leftarrow v_i \lfloor q/2 \rfloor$
    $d_{i+768} \leftarrow v_i \lfloor q/2 \rfloor$
end
```

Algorithm 2: NHSDecode [3]

```
Input: Polynomial $d \in R_q$
Result: Bit vector $v_i \in \{0,1\}^{256}$
for $i$ from 0 to 255 do
    $t \leftarrow \sum_{j=0}^{3} |d_{i+256j} - \lfloor q/2 \rfloor|$
    if $t < q$ then
        $v_i \leftarrow 1$
    else
        $v_i \leftarrow 0$
    end
end
```

B Proof Theorem 3

Proof. Suppose that a and b are polynomials of a ring with coefficients sampled from the probability distribution Ψ_k and let n be the rank of the polynomials. Then the polynomials can be written as $a = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $b = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$. If we multiply a with b, we can write $c = (a_0 + a_1x + \cdots + a_{n-1}x^{n-1})(b_0 + b_1x + \cdots + b_{n-1}x^{n-1})$. By using the distributive law and grouping all terms with the same rank together, it can be obtained $c = (a_0b_0 + \cdots - a_{n-2}b_2 - a_{n-1}b_1) + (a_0b_1 + \cdots - a_{n-2}b_3 - a_{n-1}b_2)x + \cdots + (a_0b_{n-1} + \cdots + a_{n-2}b_1 + a_{n-1}b_0)x^{n-1}$. Where each coefficient of polynomial c is determined by a sum of n products. Since all coefficients of a and b are independently sampled from the probability distribution Ψ_k, the probability distribution of the coefficients of c is an n-fold convolution of the product distribution of two RVs sampled from Ψ_k.

C Noise Distribution

Figure 11 illustrates the probability distribution of the difference and compression noise for an error distribution parameter of $k = 16$ and a compression from

14 to 3 bits. In the graph, the RV X of a coefficient of a noise polynomial has the outcomes $x = 0, 1, \ldots, q-1$. The compression noise is uniformly distributed between zero and $q/16$, and between $q - q/16$ and q. All values in between are not affected by the compression.

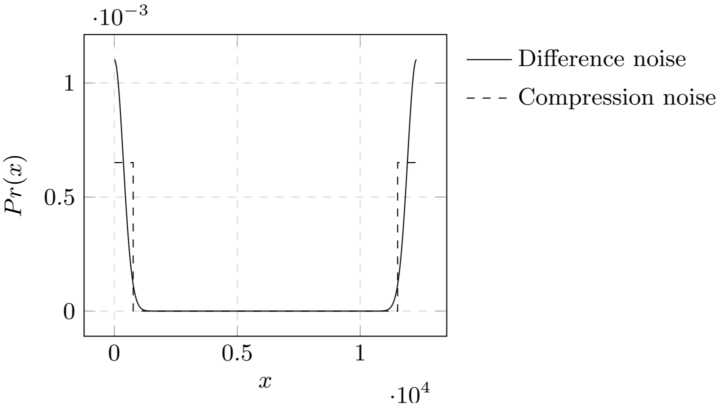

Fig. 11. Noise distributions for $k = 16$ and compression from 14 to 3 bits, where $0 \leq x \leq q-1$

D Validation of Failure Rate Analysis

In Fig. 12, the calculated difference and compression noise distribution discussed in Section 3 are compared with test measurements. For better visibility, this figure illustrates only values from zero to 1,500. Unlike in Fig. 11, the logarithmic scale is used. For the experiment, the original parameters of NewHope Simple were used. For the tested noise distribution, we used 100,000 test rounds. With $n = 1024$ this leads to 102,400,000 samples. The test measurements match the calculated values. Only for probabilities lower than 10^{-5} the difference noise shows some inaccuracies. With more test samples, the curve is expected to flatten in this region as well.

Figure 13 shows that the independence assumption stated in Subsection 3.3 can be considered as valid for NewHope Simple with high and moderate failure rates. It illustrates the failure probability with different mapping options within the additive threshold encoding and with varying values of k. Each test value matches with only minor differences the calculated value. For lower values of k the failure probability is too small in order to find the correct value by testing. Test results have shown that the calculated values for NewHope Simple without compression and NewHope Simple with further compression on polynomial u match the test values as well.

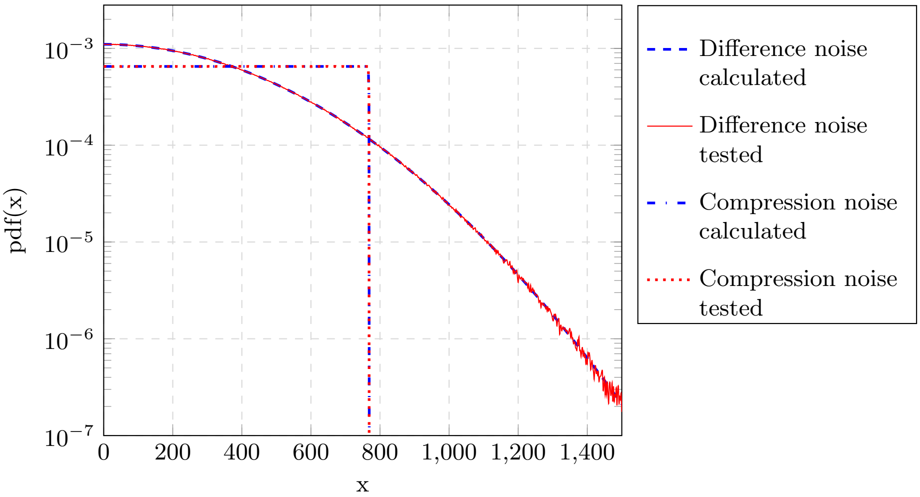

Fig. 12. Comparison of tested and calculated noise distributions for $k = 16$ and compression of c, where $0 \leq x \leq q - 1$

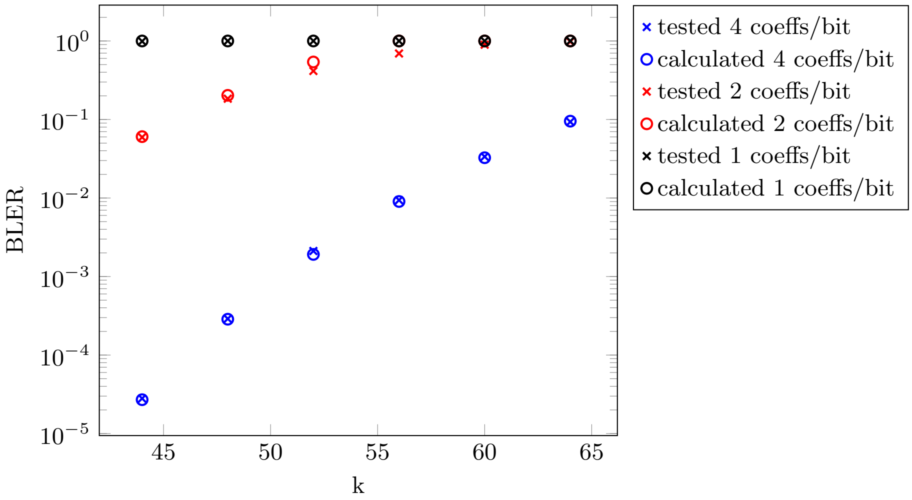

Fig. 13. Comparison of tested and calculated error probability with compression of c