Research Article

Thomas Wunderer*

A detailed analysis of the hybrid lattice-reduction and meet-in-the-middle attack

https://doi.org/10.1515/jmc-2016-0044
Received July 28, 2016; revised August 28, 2018; accepted September 3, 2018

Abstract: Over the past decade, the hybrid lattice-reduction and meet-in-the middle attack (called hybrid attack) has been used to evaluate the security of many lattice-based cryptographic schemes such as NTRU, NTRU Prime, BLISS and more. However, unfortunately, none of the previous analyses of the hybrid attack is entirely satisfactory: They are based on simplifying assumptions that may distort the security estimates. Such simplifying assumptions include setting probabilities equal to 1, which, for the parameter sets we analyze in this work, are in fact as small as 2^{-80}. Many of these assumptions lead to underestimating the scheme's security. However, some lead to security overestimates, and without further analysis, it is not clear which is the case. Therefore, the current security estimates against the hybrid attack are not reliable, and the actual security levels of many lattice-based schemes are unclear. In this work, we present an improved runtime analysis of the hybrid attack that is based on more reasonable assumptions. In addition, we reevaluate the security against the hybrid attack for the NTRU, NTRU Prime and R-BinLWEEnc encryption schemes as well as for the BLISS and GLP signature schemes. Our results show that there exist both security over- and underestimates in the literature.

Keywords: Hybrid attack, lattice-based cryptography, cryptanalysis, SVP, LWE, NTRU, NTRU Prime, BLISS

MSC 2010: 94A60, 11T71

Communicated by: Rainer Steinwandt

1 Introduction

In 2007, Howgrave-Graham proposed the hybrid lattice-reduction and meet-in-the-middle attack [20] (referred to as the hybrid attack in the following) against the NTRU encryption scheme [19]. Several works [17, 18, 20, 21, 30] claim that the hybrid attack is by far the best known attack on NTRUEncrypt. In the following years, numerous cryptographers have applied the hybrid attack to their cryptographic schemes in order to estimate their security. These considerations include more variants of the NTRU encryption scheme [17, 18, 30], the recently proposed encryption scheme NTRU Prime [8], a lightweight encryption scheme based on Ring-LWE with binary error [9, 10], and the signature schemes BLISS [14] and GLP [14, 16]. However, the above analyses of the hybrid attack make use of over-simplifying assumptions, yielding unreliable security estimates of the schemes. Many of these assumptions lead to more conservative security estimates (lower than necessary), as they give more power to the attacker. While this is not a problem from a security perspective, in those cases, the schemes might be instantiated more efficiently while preserving the desired security level. On the other hand, there also exist the more dangerous cases in which the security

***Corresponding author: Thomas Wunderer,** Technische Universität Darmstadt, Hochschulstr. 10, 64289 Darmstadt, Germany, e-mail: twunderer@cdc.informatik.tu-darmstadt.de

of a scheme is overestimated, i.e., the scheme is less secure than it is claimed to be, as we show in this work. In [30], Schanck summarizes the current state of the analyses of the hybrid attack as follows:

> "[...] it should be noted that in the author's opinion, no analysis of the hybrid attack presented thus far is entirely satisfactory. [...] it is hoped that future work will answer some of the outstanding questions related to the attack's probability of success as a function of the effort spent in lattice reduction and enumeration."

One of the most common [8, 14, 18, 21, 30] over-simplifications is to assume that collisions in the meet-in-the-middle phase of the attack will always be detected. However, in reality, collisions can only be detected with some (possibly very low) probability. For instance, for the cryptographic schemes we analyze in this work, this probability is sometimes as low as 2^{-80}, highlighting the unrealistic nature of some simplifying assumptions made in previous analyses of the hybrid attack.

Our contribution

In this work, we provide a detailed and more satisfying analysis of the hybrid attack. This is achieved in the following way. We present a generalized version of the hybrid attack applied to shortest vector problems (SVP) and show how it can also be used to solve bounded distance decoding (BDD) problems. This general framework for the hybrid attack can naturally be applied to many lattice-based cryptographic constructions, as we also show in this work. We further provide a detailed and improved analysis of the generalized version of the hybrid attack, which can be used to derive updated security estimates. We offer two types of formulas for our security estimates – one that reflects the current state of the art and a more conservative one that reflects potential advances in cryptanalysis – giving a possible range for the security level. In our analysis of the attack, we reduce the amount of underlying assumptions, eliminate the ones that are over-simplifying and clearly state the remaining ones in order to offer as much transparency as possible. We further provide some experimental results and comparisons to the literature to support the validity of the remaining assumptions on the involved success probabilities.

Our second main contribution is the following: Since previous analyses of the hybrid attack are unreliable, the security estimates of many lattice-based cryptographic schemes against the hybrid attack might be inaccurate, and their actual security level is unclear. We therefore apply our improved analysis to reevaluate the security of various cryptographic schemes against the hybrid attack in order to derive updated security estimates.[1] We first revisit the security against the hybrid attack of the NTRU [18], NTRU Prime [8] and R-BinLWEEnc [9] encryption schemes, and end with the BLISS [14] and GLP [16] signature schemes. Our results show that there exist both security over- and underestimates against the hybrid attack across the literature.

Outline

This work is structured as follows. First, we fix notation and provide the necessary background in Section 2. In Section 3, we describe a generalized version of the hybrid attack on shortest vector problems (SVP) and further explain how it can also be used to solve bounded distance decoding (BDD) problems. Our runtime analysis of the hybrid attack is presented in Section 4. In Section 5, we apply our analysis of the hybrid attack to various cryptographic schemes in order to derive updated security estimates against the hybrid attack. We end this work by giving a conclusion and outlook for possible future work.

1 Our implementation used for the security estimates is available on https://www.informatik.tu-darmstadt.de/cdc/home_cdc/personen_cdc/thomas_wunderer/index.de.jsp.

2 Preliminaries

Notation. In this work, we write vectors in bold lowercase letters, e.g., $\mathbf{a}$, and matrices in bold uppercase letters, e.g., $\mathbf{A}$. Polynomials are written in normal lower case letters, e.g., a. We frequently identify polynomials $a = \sum_{i=0}^{n} a_i x^i$ with their coefficient vectors $\mathbf{a} = (a_0, \dots, a_n)$, indicated by using the corresponding bold letter. Let $n, q \in \mathbb{N}$, $f \in \mathbb{Z}[x]$ be a polynomial of degree n and $R_q = \mathbb{Z}_q[x]/(f)$. We define the rotation matrix of a polynomial $a \in R_q$ as $\mathrm{rot}(a) = (\mathbf{a}, \mathbf{ax}, \mathbf{ax}^2, \dots, \mathbf{ax}^{n-1}) \in \mathbb{Z}_q^{n\times n}$. Then for $a, b \in R_q$, the matrix-vector product $\mathrm{rot}(a) \cdot \mathbf{b} \bmod q$ corresponds to the product of polynomials $ab \in R_q$.

We use the abbreviation $\log(\cdot)$ for $\log_2(\cdot)$. We further write $\|\cdot\|$ instead of $\|\cdot\|_2$ for the Euclidean norm. For $N \in \mathbb{N}_0$ and $m_1, \dots, m_k \in \mathbb{N}_0$ with $m_1 + \dots + m_k = N$, the multinomial coefficient is defined as

$$\binom{N}{m_1, \dots, m_k} = \frac{N!}{m_1! \cdot \ldots \cdot m_k!}.$$

Lattices and bases. In this work, we use the following definition of lattices. A discrete additive subgroup of $\mathbb{R}^m$, for some $m \in \mathbb{N}$, is called a lattice. Let m be a positive integer. For a set of vectors $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\} \subset \mathbb{R}^m$, the lattice spanned by $\mathbf{B}$ is defined as

$$\Lambda(\mathbf{B}) = \left\{ \mathbf{x} \in \mathbb{R}^m \;\middle|\; \mathbf{x} = \sum_{i=1}^{n} \alpha_i \mathbf{b}_i \text{ for } \alpha_i \in \mathbb{Z} \right\}.$$

Let $\Lambda \subset \mathbb{R}^m$ be a lattice. A set of vectors $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_n\} \subset \mathbb{R}^m$ is called a basis of Λ if $\mathbf{B}$ is $\mathbb{R}$-linearly independent and $\Lambda = \Lambda(\mathbf{B})$. Abusing notation, we identify lattice bases with matrices and vice versa by taking the basis vectors as the columns of the matrix. The number of vectors in a basis of a lattice is called the rank (or dimension) of the lattice. If the rank is maximal, i.e., $m = n$, the lattice is called full-rank. In the following, we only consider full-rank lattices. Let q be a positive integer. The length of the shortest non-zero vector of a lattice Λ is denoted by $\lambda_1(\Lambda)$. An integer lattice $\Lambda \subset \mathbb{Z}^m$ that contains $q\mathbb{Z}^m$ is called a q-ary lattice. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, we define the q-ary lattice

$$\Lambda_q(\mathbf{A}) := \{\mathbf{v} \in \mathbb{Z}^m \mid \text{there exists } \mathbf{w} \in \mathbb{Z}^n \text{ such that } \mathbf{Aw} = \mathbf{v} \bmod q\}.$$

For a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_m\} \subset \mathbb{R}^m$, its fundamental parallelepiped is defined as

$$\mathcal{P}(\mathbf{B}) = \left\{ \mathbf{x} = \sum_{i=1}^{m} \alpha_i \mathbf{b}_i \in \mathbb{R}^m \;\middle|\; -\frac{1}{2} \le \alpha_i < \frac{1}{2} \text{ for all } i \in \{1, \dots, m\} \right\}.$$

The determinant $\det(\Lambda)$ of a lattice $\Lambda \subset \mathbb{R}^m$ is defined as the m-dimensional volume of the fundamental parallelepiped of a basis of Λ. Note that the determinant of the lattice is well-defined, i.e., it is independent of the basis. The Hermite delta (or Hermite factor) δ of a lattice basis $\mathbf{B} = \{\mathbf{b}_1, \dots, \mathbf{b}_m\} \subset \mathbb{R}^m$ is defined via the equation $\|\mathbf{b}_1\| = \delta^m \det(\Lambda)^{\frac{1}{m}}$. It provides a measure for the quality of the basis.

Lattice-based cryptography is based on the presumed hardness of computational problems in lattices. Two of the most important lattice problems are the following:

Problem (Shortest vector problem, SVP). *Given a lattice basis* $\mathbf{B}$, *the task is to find a shortest non-zero vector in the lattice* $\Lambda(\mathbf{B})$.

Problem (Bounded distance decoding, BDD). *Given* $\alpha \in \mathbb{R}_{\ge 0}$, *a lattice basis* $\mathbf{B} \subset \mathbb{R}^m$ *and a target vector* $\mathbf{t} \in \mathbb{R}^m$ *with* $\mathrm{dist}(\mathbf{t}, \Lambda(\mathbf{B})) < \alpha\lambda_1(\Lambda(\mathbf{B}))$, *the task is to find a vector* $\mathbf{e} \in \mathbb{R}^m$ *with* $\|\mathbf{e}\| < \alpha\lambda_1(\Lambda(\mathbf{B}))$ *such that* $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$.

Babai's nearest plane. Babai's nearest plane algorithm [5] (denoted by NP in the following) is an important building block of the hybrid attack. For more details on the algorithm, we refer to Babai's original work [5] or Lindner and Peikert's work [24]. We use the nearest plane algorithm in a black box manner. For the reader, it is sufficient to know the following: The input for the nearest plane algorithm is a lattice basis $\mathbf{B} \subset \mathbb{Z}^m$ and a target vector $\mathbf{t} \in \mathbb{R}^m$, and the corresponding output is a vector $\mathbf{e} \in \mathbb{R}^m$ such that $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$. We denote the output by $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}) = \mathbf{e}$. If there is no risk of confusion, we might omit the basis in the notation, writing $\mathrm{NP}(\mathbf{t})$ instead of $\mathrm{NP}_{\mathbf{B}}(\mathbf{t})$. The output of the nearest plane algorithm satisfies the following condition, as shown in [5].

Lemma 2.1. *Let $\mathbf{B} \subset \mathbb{Z}^m$ be a lattice basis, and let $\mathbf{t} \in \mathbb{R}^m$ be a target vector. Then $\mathrm{NP}_{\mathbf{B}}(\mathbf{t})$ is the unique vector $\mathbf{e} \in \mathcal{P}(\overline{\mathbf{B}})$ that satisfies $\mathbf{t} - \mathbf{e} \in \Lambda(\mathbf{B})$, where $\overline{\mathbf{B}}$ is the Gram–Schmidt basis of $\mathbf{B}$.*

The lengths of the Gram–Schmidt vectors of a reduced basis can be estimated by the following heuristic (for more details, we refer to [24]).

Heuristic (Geometric series assumption (GSA)). *Let $\mathbf{B} \subset \mathbb{Z}^m$ be a reduced basis of some full-rank lattice with Hermite delta δ, and let D denote the determinant of $\Lambda(\mathbf{B})$. Further, let $\overline{\mathbf{b}}_1, \dots, \overline{\mathbf{b}}_m$ denote the corresponding Gram–Schmidt vectors of $\mathbf{B}$. Then the length of $\overline{\mathbf{b}}_i$ is approximately*

$$\|\overline{\mathbf{b}}_i\| \approx \delta^{-2(i-1)+m} D^{\frac{1}{m}}.$$

While the GSA is widely relied upon in lattice-based cryptography (see, e.g., [2–4, 10, 13, 20, 27]), we emphasize that it does not offer precise estimates, in particular for the last indices of highly reduced bases, see, e.g., [12].

3 The hybrid attack

In this section, we present a generalized version of the hybrid attack to solve shortest vector problems. Our framework for the hybrid attack is the following: The task is to find the shortest vector $\mathbf{v}$ in a lattice Λ, given a basis of Λ of the form

$$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m},$$

where $0 < r < m$ is the meet-in-the-middle dimension, $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$. In Appendix A.1, we show that for q-ary lattices, where q is prime, one can always construct a basis of this form, provided that the determinant of the lattice is at most q^{m-r}. Additionally, in Section 5, we show that our framework can be applied to many lattice-based cryptographic schemes.

The main idea of the attack is the following: Let $\mathbf{v}$ be a short vector contained in the lattice Λ. We split the short vector $\mathbf{v}$ into two parts $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. The second part $\mathbf{v}_g$ represents the part of $\mathbf{v}$ that is recovered by guessing (meet-in-the-middle) during the attack, while the first part $\mathbf{v}_l$ is recovered with lattice techniques (solving BDD problems). Because of the special form of the basis $\mathbf{B}'$, we have that

$$\mathbf{v} = \begin{pmatrix} \mathbf{v}_l \\ \mathbf{v}_g \end{pmatrix} = \mathbf{B}' \begin{pmatrix} \mathbf{x} \\ \mathbf{v}_g \end{pmatrix} = \begin{pmatrix} \mathbf{B}\mathbf{x} + \mathbf{C}\mathbf{v}_g \\ \mathbf{v}_g \end{pmatrix}$$

for some vector $\mathbf{x} \in \mathbb{Z}^{m-r}$, hence $\mathbf{C}\mathbf{v}_g = -\mathbf{B}\mathbf{x} + \mathbf{v}_l$. This means $\mathbf{C}\mathbf{v}_g$ is close to the lattice $\Lambda(\mathbf{B})$, since it only differs from the lattice by the short vector $\mathbf{v}_l$, and therefore $\mathbf{v}_l$ can be recovered solving a BDD problem if $\mathbf{v}_g$ is know. The idea now is that if we can correctly guess the vector $\mathbf{v}_g$, we can hope to find $\mathbf{v}_l$ using the nearest plane algorithm (see Section 2) via $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$, which is the case if the basis $\mathbf{B}$ is sufficiently reduced. Solving the BDD problem using the nearest plane algorithm is the lattice part of the attack. The lattice $\Lambda(\mathbf{B})$ in which we need to solve BDD has the same determinant as the lattice $\Lambda(\mathbf{B}')$ in which we want to solve SVP, but it has smaller dimension, i.e., $m - r$ instead of m. Therefore, the newly obtained BDD problem is potentially easier to solve than the original SVP instance.

In the following, we explain how one can speed up the guessing part of the attack by Odlyzko's meet-in-the-middle approach. Using this technique, one is able to reduce the number of necessary guesses to the square root of the number of guesses needed in a naive brute-force approach. Odlyzko's meet-in-the-middle attack on NTRU was first described in [22] and applied in the hybrid lattice-reduction and meet-in-the-middle attack against NTRU in [20]. The idea is that instead of guessing $\mathbf{v}_g$ directly in a large set M of possible vectors, we guess sparser vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in a smaller set N of vectors such that $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$. In our attack, the larger set M will be the set of all vectors with a fixed number $2c_i$ of the non-zero entries equal to i for all $i \in \{\pm 1, \dots, \pm k\}$, where $k = \|\mathbf{v}_g\|_\infty$. The smaller set N will be the set of all vectors with only half as many, i.e.,

only c_i, of the non-zero entries equal to i for all $i \in \{\pm 1, \dots, \pm k\}$. Assume that $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. First, we guess vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in the smaller set N. We then compute $\mathbf{v}_l' = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g')$ and $\mathbf{v}_l'' = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g'')$. We hope that if $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$, then also $\mathbf{v}_l' + \mathbf{v}_l'' = \mathbf{v}_l$, i.e., that the nearest plane algorithm is additively homomorphic on those inputs. The probability that this additive property holds is one crucial element in the runtime analysis of the attack. We further need to detect when this property holds during the attack, i.e., we need to be able to recognize matching vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ with $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$ and $\mathbf{v}_l' + \mathbf{v}_l'' = \mathbf{v}_l$, which we call a collision. In order to do so, we store $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in (hash) boxes whose addresses depend on $\mathbf{v}_l'$ and $\mathbf{v}_l''$, respectively, such that they collide in at least one box. To define those addresses properly, note that in case of a collision, we have $\mathbf{v}_l' = -\mathbf{v}_l'' + \mathbf{v}_l$. Thus $\mathbf{v}_l'$ and $-\mathbf{v}_l''$ differ only by a vector of infinity norm $y = \|\mathbf{v}_l\|_\infty$. Therefore, the addresses must be crafted such that for any $\mathbf{x} \in \mathbb{Z}^m$ and $\mathbf{z} \in \mathbb{Z}^m$ with $\|\mathbf{z}\|_\infty \le y$, it holds that the intersection of the addresses of $\mathbf{x}$ and $\mathbf{x} + \mathbf{z}$ is non-empty, i.e., $\mathcal{A}_{\mathbf{x}}^{(m,y)} \cap \mathcal{A}_{\mathbf{x}+\mathbf{z}}^{(m,y)} \neq \emptyset$. Furthermore, the set of addresses should not be unnecessarily large so the hash tables do not grow too big and unwanted collisions are unlikely to happen. The following definition satisfies these properties, as can easily be verified.

Definition 3.1. Let $m, y \in \mathbb{N}$. For a vector $\mathbf{x} \in \mathbb{Z}^m$, the set $\mathcal{A}_{\mathbf{x}}^{(m,y)} \subset \{0,1\}^m$ is defined as

$$\begin{aligned}\mathcal{A}_{\mathbf{x}}^{(m,y)} = \{\mathbf{a} \in \{0,1\}^m \mid (\mathbf{a})_i = 1 \text{ if } (\mathbf{x})_i > \lceil \tfrac{y}{2} - 1 \rceil \text{ for } i \in \{1, \dots, m\},\\ (\mathbf{a})_i = 0 \text{ if } (\mathbf{x})_i < -\lfloor \tfrac{y}{2} \rfloor \text{ for } i \in \{1, \dots, m\}\}.\end{aligned}$$

We illustrate Definition 3.1 with some examples.

Example. Let $m = 5$ be fixed. For varying bounds y and input vectors $\mathbf{x}$, we have

$$\begin{aligned}
\mathcal{A}_{(7,0,-1,1,-5)}^{(5,1)} &= \{(1,0,0,1,0), (1,1,0,1,0)\}\\
\mathcal{A}_{(8,0,-1,1,-2)}^{(5,2)} &= \{(1,0,0,1,0), (1,1,0,1,0), (1,0,1,1,0), (1,1,1,1,0)\}\\
\mathcal{A}_{(2,-1,9,1,-2)}^{(5,3)} &= \{(1,0,1,0,0), (1,0,1,1,0), (1,1,1,0,0), (1,1,1,1,0)\}\\
\mathcal{A}_{(2,-5,0,7,-2)}^{(5,4)} &= \{(1,0,0,1,0), (1,0,0,1,1), (1,0,1,1,0), (1,0,1,1,1)\}
\end{aligned}$$

The hybrid attack on SVP without precomputation is presented in Algorithm 1. A list of the attack parameters and the parameters used in the runtime analysis of the attack and their meaning is given in Table 1. In order to increase the chance of Algorithm 1 being successful, one performs a basis reduction step as precomputation. Therefore, the complete hybrid attack, presented in Algorithm 2, is in fact a combination of a basis reduction step and Algorithm 1.

Input: $m, r \in \mathbb{N}$ with $r < m$, $y, k \in \mathbb{N}$, $c_{-k}, \dots, c_k \in \mathbb{N}_0$ with $r = \sum_{i=-k}^{k} 2c_i$,
$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m \times m}$, where $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$

while *true* **do**
  guess $\mathbf{v}_g' \in \{-k, \dots, k\}^r$ with exactly c_i entries equal to i for all $i \in \{-k, \dots, k\}$;
  calculate $\mathbf{v}_l' = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g') \in \mathbb{Z}^{m-r}$;
  store $\mathbf{v}_g'$ in all the boxes addressed by $\mathcal{A}_{\mathbf{v}_l'}^{(m-r,y)} \cup \mathcal{A}_{-\mathbf{v}_l'}^{(m-r,y)}$ (cf. Definition 3.13.1;
  for *all* $\mathbf{v}_g'' \neq \mathbf{v}_g'$ *in all the boxes addressed by* $\mathcal{A}_{\mathbf{v}_l'}^{(m-r,y)} \cup \mathcal{A}_{-\mathbf{v}_l'}^{(m-r,y)}$ **do**
    set $\mathbf{v}_g = \mathbf{v}_g' + \mathbf{v}_g''$ and calculate $\mathbf{v}_l = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) \in \mathbb{Z}^{m-r}$;
    if $\mathbf{v} = \binom{\mathbf{v}_l}{\mathbf{v}_g} \in \Lambda(\mathbf{B}')$ *and* $\|\mathbf{v}_l\|_\infty \le y$ *and* $\|\mathbf{v}_g\|_\infty \le k$ **then**
      return $\mathbf{v}$;

Algorithm 1: The hybrid attack on SVP without basis reduction.

Input: $m, r \in \mathbb{N}$ with $r < m$, $y, k \in \mathbb{N}$, $c_{-k}, \ldots, c_{-1}, c_1, \ldots, c_k \in \mathbb{N}_0$ with $r = \sum_{i=-k}^{k} 2c_i$,
$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m\times m}$, where $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$ and $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$
reduce $\mathbf{B}$ to some basis $\tilde{\mathbf{B}}$;
run Algorithm 1 on input $m, r, y, k, c_{-k}, \ldots, c_{-1}, c_1, \ldots, c_k, \left(\begin{array}{c|c} \tilde{\mathbf{B}} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right)$;

Algorithm 2: The hybrid attack on SVP including basis reduction.

Parameter	Meaning
m	lattice dimension
r	meet-in-the-middle dimension
$\mathbf{B}'$	lattice basis of the whole lattice
$\mathbf{B}$	partially reduced lattice basis of the sublattice
c_i	number of i-entries guessed during attack
y	infinity norm bound on $\mathbf{v}_l$
k	infinity norm bound on $\mathbf{v}_g$
Y	expected Euclidean norm of $\mathbf{v}_l$
R_i	Gram–Schmidt lengths corresponding to $\mathbf{B}$
r_i	scaled Gram–Schmidt lengths corresponding to $\mathbf{B}$

Table 1: Attack parameters and parameters in the runtime analysis.

The hybrid attack on BDD

The hybrid attack can also be applied to BDD instead of SVP by rewriting a BDD instance into an SVP instance. This can be done in the following way (see for example [1]): Let $\mathbf{B}'$ be a lattice basis of the form

$$\mathbf{B}' = \left(\begin{array}{c|c} \mathbf{B} & \mathbf{C} \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{m\times m}$$

with $\mathbf{B} \in \mathbb{Z}^{(m-r)\times(m-r)}$, $\mathbf{C} \in \mathbb{Z}^{(m-r)\times r}$, and let $\mathbf{t}$ be a target vector for BDD. Suppose $\mathbf{t} - \mathbf{v} \in \Lambda(\mathbf{B}')$, where $\mathbf{v}$ is the short (bounded) vector we are looking for. Then the short vector $(\mathbf{v}, 1)^t$ is contained in the lattice $\Lambda(\mathbf{B}'')$ spanned by

$$\mathbf{B}'' = \left(\begin{array}{c|c} \mathbf{B}' & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right) \in \mathbb{Z}^{(m+1)\times(m+1)},$$

which is of the required form for the hybrid attack on SVP. Therefore, we can apply the hybrid attack on SVP to find $(\mathbf{v}, 1)^t$, solving the BDD problem. The SVP lattice $\Lambda(\mathbf{B}'')$ has the same determinant as the BDD lattice $\Lambda(\mathbf{B}')$ and dimension $m + 1$ instead of m. However, the additional dimension can be ignored, since we know the last entry of $(\mathbf{v}, 1)^t$ and therefore do not have to guess it during the meet-in-the-middle phase. Note that by definition of BDD, it is very likely that $\pm\mathbf{v}$ are the only short vectors in the lattice $\Lambda(\mathbf{B}'')$. By fixing the last coordinate to be plus one, only $\mathbf{v}$, not also $-\mathbf{v}$, can be found by the attack.

4 Analysis

In this section, we analyze the runtime of the hybrid attack. First, in Heuristic 1 in Section 4.1, we estimate the runtime of the attack in case sufficient success conditions are satisfied. In Section 4.2, we then show how to determine the probability that those sufficient conditions are satisfied, i.e., how to determine (a lower bound on) the success probability. We conclude the runtime analysis of the attack by showing how to optimize the attack parameters to minimize its runtime in Section 4.3. We end the section by highlighting our improvements over previous analyses of the hybrid attack, see Section 4.4.

4.1 Runtime analysis

We now present our main result about the runtime of the generalized hybrid attack. It shows that under sufficient conditions, the attack is successful and estimates the expected runtime.

Heuristic 1. *Let all inputs be denoted as in Algorithm 1, $Y \in \mathbb{R}_{\geq 0}$, and let $R_1, \dots, R_{m-r}$ denote the lengths of the Gram–Schmidt basis vectors of the basis $\mathbf{B}$. Further, let $S \subset \Lambda(\mathbf{B}')$ denote the set of all non-zero lattice vectors $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in \Lambda(\mathbf{B}')$, where $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$ with $\|\mathbf{v}_l\|_\infty \leq y$, $\|\mathbf{v}_l\| \approx Y$, $\|\mathbf{v}_g\|_\infty \leq k$, exactly $2c_i$ entries of $\mathbf{v}_g$ are equal to i for all $i \in \{\pm 1, \dots, \pm k\}$, and $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. Assume that the set S is non-empty.*

Then Algorithm 1 is successful, and the expected number of loops can be estimated by

$$L = \binom{r}{c_{-k}, \dots, c_k} \left(p \cdot |S| \cdot \prod_{i \in \{\pm 1, \dots, \pm k\}} \binom{2c_i}{c_i} \right)^{-\frac{1}{2}},$$

where

$$p = \prod_{i=1}^{m-r} \left(1 - \frac{1}{r_i B\left(\frac{(m-r)-1}{2}, \frac{1}{2}\right)} \int_{-r_i-1}^{-r_i} \int_{\max(-1, z-r_i)}^{z+r_i} (1-t^2)^{\frac{(m-r)-3}{2}} \, dt \, dz \right),$$

$B(\cdot, \cdot)$ denotes the Euler beta function [28], *and*

$$r_i = \frac{R_i}{2Y} \quad \text{for all } i \in \{1, \dots, m-r\}.$$

Furthermore, the expected number of operations of Algorithm 1 can be estimated by $T_{\mathrm{NP}} \cdot L$, where T_{NP} denotes the number of operations of one nearest plane call in the lattice $\Lambda(\mathbf{B})$ of dimension $m - r$.

In the following remark, we explain the meaning of the (attack) parameters that appear in Heuristic 1 in more detail.

Remark 4.1. (1) The parameters r, y, k, $c_{-k}, \dots, c_k$ are the attack parameters that can be chosen by the attacker. The meet-in-the-middle dimension and the remaining lattice dimension are determined by the parameter r. The remaining parameters must be chosen in such a way that the requirements of Heuristic 1 are likely to be fulfilled in order to obtain a high success probability of the attack. Choosing those parameters depends heavily on the distribution of the short vectors $\mathbf{v} \in S$. In order to obtain more flexibility, this distribution is not specified in Heuristic 1. However, in Section 5, we show how one can choose the attack parameters and calculate the success probability for several distributions arising in various cryptographic schemes. At this point, we only want to remark that y should be (an upper bound on) $\|\mathbf{v}_l\|_\infty$, k (an upper bound on) $\|\mathbf{v}_g\|_\infty$ and $2c_i$ the (expected) number of entries of $\mathbf{v}_g$ that is equal to i for $i \in \{\pm 1, \dots, \pm k\}$.

(2) The attacker can further influence the lengths $R_1, \dots, R_{m-r}$ of the Gram–Schmidt vectors by providing a different basis than $\mathbf{B}$ with Gram–Schmidt lengths that lead to a more efficient attack. This is typically done by performing a basis reduction on $\mathbf{B}$ or parts of $\mathbf{B}$ as precomputation, see Algorithm 2. The lengths of the Gram–Schmidt vectors achieved by the basis reduction with Hermite delta δ are typically estimated by the GSA (see Section 2). However, for bases of q-ary lattices of a special form, the GSA may be modified. For further details, see Appendix A.2. Notice that spending more time on basis reduction increases the probability p in Heuristic 1, and the probability that the condition $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$ holds, as can be seen later in this section and Section 4.2.

(3) Because of the previous remark, the complete attack – presented in Algorithm 2 – is actually a combination of precomputation (basis reduction) and Algorithm 1. Therefore, the runtime of both steps must be considered, and they have to be balanced in order to estimate the total runtime. In particular, the amount of precomputation must be chosen such that the precomputed basis offers the best trade-off between its quality with respect to the hybrid attack (i.e., amplifying the success probability and decreasing the number of operations) and the cost to compute this basis. We show how to optimize the total runtime in Section 4.3.

In the following, we show how Heuristic 1 can be derived. For the rest of this section, let all notations be as in Heuristic 1. We further assume in the following that the assumption of Heuristic 1, i.e., $S \neq \emptyset$, is satisfied. We first provide the following useful definition already given in [10, 20]. We use the notation of [10].

Definition 4.2. Let $n \in \mathbb{N}$. A vector $\mathbf{x} \in \mathbb{R}^n$ is called $\mathbf{y}$-admissible (with respect to the basis $\mathbf{B}$) for some vector $\mathbf{y} \in \mathbb{R}^n$ if $\mathrm{NP}_{\mathbf{B}}(\mathbf{x}) = \mathrm{NP}_{\mathbf{B}}(\mathbf{x} - \mathbf{y}) + \mathbf{y}$.

This means, that if $\mathbf{x}$ is $\mathbf{y}$-admissible then $\mathrm{NP}_{\mathbf{B}}(\mathbf{x})$ and $\mathrm{NP}_{\mathbf{B}}(\mathbf{x} - \mathbf{y})$ yield the same lattice vector. We recall the following Lemma from [10] about Definition 4.2. It showcases the relevance of the definition by relating it to the equation $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1) + \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_2) = \mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$, which is necessary to hold for our attack to work.

Lemma 4.3 ([10, Lemma 2]). *Let $\mathbf{t}_1 \in \mathbb{R}^n$, $\mathbf{t}_2 \in \mathbb{R}^n$ be two arbitrary target vectors. Then the following are equivalent.*
(i) $\mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1) + \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_2) = \mathrm{NP}_{\mathbf{B}}(\mathbf{t}_1 + \mathbf{t}_2)$.
(ii) $\mathbf{t}_1$ *is* $\mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*
(iii) $\mathbf{t}_2$ *is* $\mathrm{NP}(\mathbf{t}_1 + \mathbf{t}_2)$*-admissible.*

Success of the attack and number of loops

We now estimate the expected number of loops in case Algorithm 1 terminates. In the following, we use the subscript $\mathbf{B}$ for probabilities to indicate that the probability is taken over the randomness of the basis (with Gram–Schmidt length $R_1, \dots, R_{m-r}$). In each loop of the algorithm, we sample a vector $\mathbf{v}_g'$ in the set

$$W = \{\mathbf{w} \in \mathbb{Z}^r \mid \text{exactly } c_i \text{ entries of } \mathbf{w} \text{ are equal to } i \text{ for all } i \in \{-k, \dots, k\}\}.$$

The attack succeeds if $\mathbf{v}_g' \in W$ and $\mathbf{v}_g'' \in W$ such that $\mathbf{v}_g' + \mathbf{v}_g'' = \mathbf{v}_g$ and

$$\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g') + \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g'') = \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g' + \mathbf{C}\mathbf{v}_g'') = \mathbf{v}_l$$

for some vector $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in S$ are sampled in different loops of the algorithm. By Lemma 4.3, the second condition is equivalent to the fact that $\mathbf{C}\mathbf{v}_g'$ is $\mathbf{v}_l$-admissible. We assume that the algorithm only succeeds in this case. We are therefore interested in the following subset of W:

$$V = \{\mathbf{w} \in W \mid \mathbf{v}_g - \mathbf{w} \in W \text{ and } \mathbf{C}\mathbf{w} \text{ is } \mathbf{v}_l\text{-admissible for some } \mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in S\}.$$

For all $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in S$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$, let $p(\mathbf{v})$ denote the probability

$$p(\mathbf{v}) = \Pr_{\mathbf{B}, \mathbf{w} \leftarrow W}[\mathbf{C}\mathbf{w} \text{ is } \mathbf{v}_l\text{-admissible}],$$

and let $p_1(\mathbf{v})$ denote the probability

$$p_1(\mathbf{v}) = \Pr_{\mathbf{w} \leftarrow W}[\mathbf{v}_g - \mathbf{w} \in W].$$

By construction, we have that $p_1(\mathbf{v})$ is constant for all $\mathbf{v} \in S$, so we can simply write p_1 instead of $p_1(\mathbf{v})$. We make the following reasonable assumption on $p(\mathbf{v})$ and p_1.

Assumption 1. *For all $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in S$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$, we assume that the independence condition*

$$p(\mathbf{v}) = \Pr_{\mathbf{B}, \mathbf{w} \leftarrow W}[\mathbf{C}\mathbf{w} \text{ is } \mathbf{v}_l\text{-admissible} \mid \mathbf{v}_g - \mathbf{w} \in W]$$

holds. We further assume that $p(\mathbf{v})$ is equal to some constant probability p (as in Heuristic 1) for all $\mathbf{v} \in S$.

Assuming independence of p and p_1 and disjoint events for the elements of S, we can make the following reasonable assumption (analogously to [20, Lemma 6 and Theorem 3]).

Assumption 2. *We assume that*

$$\frac{|V|}{|W|} = \Pr_{\mathbf{w} \leftarrow W}[\mathbf{w} \in V] = p_1 p |S|.$$

The probability p_1 is calculated by

$$p_1 = \frac{\prod_{i\in\{\pm1,\dots,\pm k\}}\binom{2c_i}{c_i}}{|W|}, \quad \text{where } |W| = \binom{r}{c_{-k},\dots,c_k}.$$

From Assumption 2, it follows that $|V| \approx p_1 p|W||S|$. As long as the product $p_1 p$ is not too small, we can therefore assume that $V \neq \emptyset$.

Assumption 3. *We assume that $V \neq \emptyset$.*

Assumption 3 implies that the attack is successful, since by Lemma 4.3, if $\mathbf{v}_g' \in V$, then also $\mathbf{v}_g'' = \mathbf{v}_g - \mathbf{v}_g' \in V$ for all $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t \in S$. Such two vectors $\mathbf{v}_g'$ and $\mathbf{v}_g''$ in V will eventually be guessed in two separate loops of the algorithm, and they are recognized as a collision, since by the assumption $\|\mathbf{v}_l\|_\infty \leq y$ of Heuristic 1, they share at least one common address. By Assumption 2, we expect that during the algorithm, we sample in V every $\frac{1}{p_1 p|S|}$ loops, and by the birthday paradox, we expect to find a collision $\mathbf{v}_g' \in V$ and $\mathbf{v}_g'' \in V$ with $\mathbf{v}_g'' + \mathbf{v}_g' = \mathbf{v}_g$ after $L \approx \frac{1}{p_1 p|S|}\sqrt{|V|}$ loops. In conclusion, we can estimate the expected number of loops by

$$L \approx \frac{\sqrt{|V|}}{p_1 p|S|} = \frac{\sqrt{|W|}}{\sqrt{p_1 p|S|}} = \binom{r}{c_{-k},\dots,c_k}\left(p|S|\prod_{i\in\{\pm1,\dots,\pm k\}}\binom{2c_i}{c_i}\right)^{-\frac{1}{2}}.$$

It remains to calculate the probability p. This can be done analogously to [10, Heuristic 3] and the calculations following it. For a detailed and convincing justification of the heuristic and the intuition behind it, including a geometric intuition behind the $\mathbf{y}$-admissibility and its mathematical modeling, we refer to [10]. Following the calculations of [10], we obtain the following assumption.

Assumption 4. *We assume that the probability p is approximately*

$$p \approx \prod_{i=1}^{m-r}\left(1 - \frac{1}{r_i B(\frac{(m-r)-1}{2}, \frac{1}{2})}\int_{-r_i-1}^{-r_i}\int_{\max(-1,z-r_i)}^{z+r_i}(1-t^2)^{\frac{(m-r)-3}{2}}\,dt\,dz\right),$$

where $B(\cdot,\cdot)$ and $r_1,\dots,r_{m-r}$ are defined as in Heuristic 1.

The integrals in the above formula for p can, for instance, be calculated using SageMath [31]. In order to calculate p, one needs to estimate the lengths r_i, as discussed in the Remark 4.1. In Appendix A.3, we provide the results of some preliminary experiments supporting the validity of Assumption 4.

Number of operations

We now estimate the expected total number of operations of the hybrid attack under the conditions of Heuristic 1. In order to do so, we need to estimate the runtime of one inner loop and multiply it by the expected number of loops. As in [20] and [10], we make the following assumption, which is plausible as long the sets of addresses are not extremely large.

Assumption 5. *We assume that the number of operations of one inner loop of Algorithm 1 is dominated by the number of operations of one nearest plane call.*

We remark that we see Assumption 5 as one of the more critical ones. Obviously, it does not hold for all parameter choices[2], but it is reasonable to believe that it holds for many relevant parameter sets, as claimed

2 For instance, if the infinity norm y is too big, it is likely to have exponentially many addresses per vector, and storing a vector at all addresses takes more time than a nearest plane call.

in [20] and [10]. However, the claim in [20] is based on the observation that for random vectors in $\mathbb{Z}_q^m$, it is highly unlikely that adding a binary vector will flip the sign of many coordinates (i.e., that a random vector in $\mathbb{Z}_q^m$ has many minus one coordinates). While this is true, the vectors in question are in fact not random vectors in $\mathbb{Z}_q^m$ but outputs of a nearest plane call, and thus potentially shorter than typical vectors in $\mathbb{Z}_q^m$. Therefore, it can be expected that adding a binary vector will flip more signs. Additionally, in general, it is not only a binary vector that is added, but a vector of infinity norm y, which makes flipping signs even more likely. However, we believe that Assumption 5 is still plausible for most relevant parameter sets and small y, and even in the worst case the assumption leads to more conservative security estimates.

In [17], Hirschhorn et al. give an experimentally verified number of bit operations (defined as in [23]) of one nearest plane call and state a conservative assumption on the runtime of the nearest plane algorithm using precomputation. Based on their results, we use the following assumption for our security estimates. We provide two different kinds of security estimates, one which we call "standard" (std) and one which we call "conservative" (cons). The latter accounts for possible cryptanalytic improvements which are plausible but not yet known to be applicable.

Assumption 6. *Let $d \in \mathbb{N}$ be the lattice dimension. For our standard security estimates, we assume that the number of bit operations of one nearest plane call is approximately $d^2/2^{1.06}$. For our conservative security estimates, we assume that the number of bit operations of one nearest plane call is approximately $d/2^{1.06}$.*

4.2 Determining the success probability

In Heuristic 1, it is guaranteed that Algorithm 1 is successful if the lattice Λ contains a non-empty set S of short vectors of the form $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t$, where $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$ with $\|\mathbf{v}_l\| \approx Y$, $\|\mathbf{v}_l\|_\infty \le y$, $\|\mathbf{v}_g\|_\infty \le k$, exactly $2c_i$ entries of $\mathbf{v}_g$ are equal to i for all $i \in \{\pm 1, \dots, \pm k\}$, and $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. In order to determine a lower bound on the success probability, one must calculate the probability that the set S of such vectors is non-empty, since

$$p_{\mathrm{succ}} \ge \Pr[S \neq \emptyset].$$

However, this probability depends heavily on the distribution of the short vectors contained in Λ and is therefore not done in Heuristic 1, allowing for more flexibility. In consequence, this analysis must be performed for the specific distribution at hand, originating from the cryptographic scheme that is to be analyzed. The most involved part in calculating the success probability is typically calculating the probability p_{NP} that $\mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$. As shown in [10], the probability p_{NP} is approximately

$$p_{\mathrm{NP}} \approx \prod_{i=1}^{m-r} \left(1 - \frac{2}{B\left(\frac{(m-r)-1}{2}, \frac{1}{2}\right)} \int_{-1}^{\max(-r_i,-1)} (1-t^2)^{\frac{(m-r)-3}{2}} \, dt \right), \tag{4.1}$$

where r_i are defined as in Heuristic 1.

In [24], Lindner and Peikert calculated the success probability of the nearest plane(s) algorithm for the case that the difference vector is drawn from a discrete Gaussian distribution with standard deviation σ. In our case, this would result in the formula

$$p_{\mathrm{NP}} = \Pr[\mathrm{NP}_{\mathbf{B}}\, \mathbf{C}\mathbf{v}_g \approx \mathbf{v}_l] = \prod_{i=1}^{m-r} \mathrm{erf}\left(\frac{R_i \sqrt{2}}{\sigma} \right). \tag{4.2}$$

In the following, we compare the formulas (4.1) and (4.2) in the case of discrete Gaussian distributions with standard deviation σ. To this end, we evaluated both formulas for a lattice of dimension $d = m - r = 200$ of determinant 128^{100} for different standard deviations. For formula (4.1), we assumed that the norm of $\mathbf{v}_l$ is $\sigma\sqrt{200}$ as expected, and that the basis follows the GSA with Hermite delta 1.008. The results, presented in Table 2, show that both formulas virtually give the same results for the analyzed instances. This indicates that formula (4.1) is a good generalization of the one provided in [24].

Gaussian parameter	$s = 1$	$s = 2$	$s = 4$	$s = 8$	$s = 16$
p_{NP} according to (4.1)	$2^{-0.033}$	$2^{-3.658}$	$2^{-27.775}$	$2^{-87.506}$	$2^{-188.445}$
p_{NP} according to (4.2)	$2^{-0.036}$	$2^{-3.669}$	$2^{-27.680}$	$2^{-87.217}$	$2^{-187.932}$

Table 2: Comparison of (4.1) and (4.2) for standard deviation $\sigma = s/\sqrt{2\pi}$ and varying Gaussian parameter s.

4.3 Optimizing the runtime

The final step in our analysis is to determine the runtime of the complete hybrid attack (Algorithm 2) including precomputation, which involves the runtime of the basis reduction T_{red}, the runtime of the actual attack T_{hyb} and the success probability p_{succ}. All these quantities depend on the attack parameter r and the quality of the basis $\mathbf{B}$ given by the lengths of the Gram–Schmidt vectors achieved by the basis reduction performed in the precomputation step of the attack. The quality of the basis can be measured by its Hermite delta δ (see Section 2). In order to unfold the full potential of the attack, one must minimize the runtime over all possible attack parameters r and δ. For our standard security estimates, we assume that the total runtime (which is to be minimized) is given by

$$T_{\mathrm{total,std}}(\delta, r) = \frac{T_{\mathrm{red,std}}(\delta, r) + T_{\mathrm{hyb,std}}(\delta, r)}{p_{\mathrm{succ}}(\delta, r)}.$$

For our conservative security estimates, we assume that given a reduced basis with quality δ, it is significantly easier to find another reduced basis with same quality δ than it is to find one given an arbitrary non-reduced basis. We therefore assume that even if the attack is not successful and needs to be repeated, the large precomputation cost for the basis reduction only needs to be paid once, and hence

$$T_{\mathrm{total,cons}}(\delta, r) = T_{\mathrm{red,cons}}(\delta, r) + \frac{T_{\mathrm{hyb,cons}}(\delta, r)}{p_{\mathrm{succ}}(\delta, r)}.$$

In order to calculate $T_{\mathrm{total,cons}}(\delta, r)$ and $T_{\mathrm{total,std}}(\delta, r)$, one must calculate $T_{\mathrm{hyb,cons}}(\delta, r)$, $T_{\mathrm{hyb,std}}(\delta, r)$, $T_{\mathrm{red,cons}}(\delta, r)$, $T_{\mathrm{red,std}}(\delta, r)$ and $p_{\mathrm{succ}}(\delta, r)$. How to calculate $T_{\mathrm{total,cons}}(\delta, r)$ and $T_{\mathrm{total,std}}(\delta, r)$ is shown in Heuristic 1. The success probability $p_{\mathrm{succ}}(\delta, r)$ is calculated in Section 4.2.

Basis reduction. Estimating the necessary runtime $T_{\mathrm{red}}(\delta, r)$ for a basis reduction of quality δ is highly non-trivial and still an active research area, and precise estimates are hard to derive. For this reason, our framework is designed such that the cost model for basis reduction can be replaced by a different one while the rest of the analysis remains intact. Thus, if future research shows significant improvements in estimating the cost of basis reduction, these cost models can be applied in our framework. To illustrate our method, we fix two common approaches to estimate the cost of basis reduction. For our standard security estimates, we apply the following approach. We first determine the (minimal) block size β necessary to achieve the targeted Hermite delta δ via

$$\delta \approx \left(\frac{\beta \cdot (\pi\beta)^{\frac{1}{\beta}}}{2\pi e} \right)^{\frac{1}{2(\beta-1)}}$$

according to Chen's thesis [12] (see also, e.g., [27]). We then use the BKZ 2.0 simulator[3] of the full version of [13] to determine the corresponding necessary number of rounds k. Finally, we use the estimate

$$\mathrm{Estimate}_{\mathrm{std}}(\beta, n, k) = 0.187281\beta \log(\beta) - 1.0192\beta + \log(n \cdot k) + 16.1$$

provided in [2] to determine the (base-two) logarithm of the runtime, where n is the lattice dimension.

For the conservative security estimates, we assume that only one round of BKZ 2.0 with the determined block size β is needed. The reason for this assumption is that one can use progressive BKZ strategies to reduce the number of rounds needed with blocksize β by running BKZ with block sizes smaller than β in advance,

3 For our implementations, we used the publicly available code from https://github.com/NTRUOpenSourceProject/ntru-params.

see [4, 12]. Since BKZ with smaller block sizes is considerably cheaper, we do not consider the BKZ costs with smaller block sizes in our conservative security estimates. Furthermore, for our conservative security estimates, we assume that the number of rounds can be decreased to one, giving

$$\text{Estimate}_{\text{cons}}(\beta, n) = 0.187281\beta \log(\beta) - 1.0192\beta + \log(n + 1 - \beta) + 16.1.$$

Runtime optimization. The optimization of the total runtime $T_{\text{total}}(\delta, r)$ is performed in the following way. For each possible r, we find the optimal δ_r that minimizes the runtime $T_{\text{total}}(\delta, r)$. Consequently, the optimal runtime is given by $\min\{T_{\text{total}}(\delta_r, r)\}$, the smallest of those minimized runtimes. Note that for fixed r, the optimal δ_r for our conservative security estimates can easily be found in the following way. For fixed r, the function $T_{\text{red,cons}}(\delta, r)$ is monotonically decreasing in δ, and the function $\frac{T_{\text{hyb,cons}}(\delta,r)}{p_{\text{succ}}(\delta,r)}$ is monotonically increasing in δ. Therefore, $T_{\text{total,cons}}(\delta, r)$ is (close to) optimal when both those functions are balanced, i.e., take the same value. Thus the optimal δ_r can, for example, be found by a simple binary search.

For our standard security estimates, we assume the function $\frac{T_{\text{red,std}}(\delta,r)}{p_{\text{succ}}(\delta,r)}$ is monotonically decreasing in δ in the relevant range, hence the optimal $T_{\text{total,std}}(\delta, r)$ can be found by balancing the functions $\frac{T_{\text{red,std}}(\delta,r)}{p_{\text{succ}}(\delta,r)}$ and $\frac{T_{\text{hyb,std}}(\delta,r)}{p_{\text{succ}}(\delta,r)}$ as above. Note that this assumption might note be true, but it surely leads to upper bounds on the optimal runtime of the attack.

4.4 Typical flaws in previous analyses of the hybrid attack

We end this section by listing some typical over-simplifications which can be found in previous analyses of the hybrid attack. We remark that some simplifying assumptions lead to overestimating the security of the schemes and others to underestimating it. In some analyses, both types occurred at the same time and somewhat magically almost canceled out each others effect on the security estimates for some parameter sets.

Ignoring the probability *p*

One of the most frequently encountered simplifications that appeared in several works is the lack of a (correct) calculation of the probability p defined in Assumption 1. As can be seen in Heuristic 1, this probability plays a crucial role in the runtime analysis of the attack. Nevertheless, in several works [8, 14, 18, 21, 30], the authors ignore the presence of this probability by setting $p = 1$ for the sake of simplicity. However, even though we took the probability into account when optimizing the attack parameters[4], for the parameter sets we analyze in Section 5, the probability p was sometimes as low as 2^{-80}, see Table 4. Note that the wrong assumption $p = 1$ gives more power to the attacker, since it assumes that collisions can always be detected by the attacker, although this is not the case, resulting in security underestimates. We also remark that in some works, the probability p is not completely ignored but determined in a purely experimental way [20] or calculated using additional assumptions [17].

Unrealistic demands on the basis reduction

In most works [8, 14, 17, 18, 20, 21, 30], the authors demand a sufficiently good basis reduction such that the nearest plane algorithm must unveil the searched short vector (or at least with very high probability). To be more precise, [20, Lemma 1] is used to determine what sufficiently good exactly means. In our opinion,

4 If the probability p is ignored in the optimization process, it can even be lower for the "optimized" attack parameters.

this demand is unrealistic, and instead we account for the probability of this event in the success probability, which reflects the attacker's power in a more accurate way. In addition, we note that in most cases, [20, Lemma 1] is not applicable the way it is claimed in several works. We briefly sketch why this is the case. Often, [20, Lemma 1] is applied to determine the necessary quality of a reduced basis such that the nearest plane (on correct input) unveils a vector $\mathbf{v}$ of infinity norm at most y. However, this lemma is only applicable if the basis matrix is in triangular form, which is not the case is general. Therefore, one needs to transform the basis with an orthonormal matrix $\mathbf{Y}$ in order to obtain a triangular basis. This basis however does not span the same lattice but an isomorphic one, which contains the transformed vector $\mathbf{vY}$, but (in general) not the vector $\mathbf{v}$. While the transformation $\mathbf{Y}$ preserves the Euclidean norm of the vector $\mathbf{v}$, it does not preserve its infinity norm. Therefore, the lemma cannot be applied with the same infinity norm bound y, which is done in most works. In fact, in the worst case, the new infinity norm bound can be up to $\sqrt{m}y$, where m is the lattice dimension. In consequence, one would have to apply [20, Lemma 1] with infinity norm bound $\sqrt{m}y$ instead of y in order to get a rigorous statement, which demands a much better basis reduction. This problem is already mentioned – but not solved – in [30]. Note that the worst case, where (i) the vector $\mathbf{v}$ has Euclidean norm $\sqrt{m}y$, and (ii) all the weight of the transformed vector is on one coordinate such that $\sqrt{m}y$ is a tight bound on the infinity norm after transformation, is highly unlikely. In the following, we give an example to illustrate the different success conditions for the nearest plane algorithm.

Example. Let $d = 512$ and $q = 1024$. We consider the nearest plane algorithm on a BDD instance $\mathbf{t} \in \Lambda + \mathbf{e}$ in a d-dimensional lattice Λ of determinant $q^{d/2}$, where $\mathbf{e}$ is a random binary vector. Naively applying [20, Lemma 1] with infinity norm bound 1 would suggest that a basis reduction of quality $\delta_1 \approx 1.0068$ is sufficient to recover $\mathbf{e}$. Applying the cost model used for our conservative security estimates described in Section 4.3, this would take roughly $T_1 \approx 2^{91}$ operations. However, as described above, the lemma cannot be applied with that naive bound. Instead, using the worst case bound $\sqrt{m}y$ on the infinity norm and applying [20, Lemma 1] would lead to a basis reduction of quality $\delta_2 \approx 1.0007$, taking roughly $T_2 \approx 2^{357}$ operations to guarantee the success of the nearest plane algorithm. This shows the impracticality of this approach. Instead, taking the success probability of the nearest plane algorithm into account, as done in this work, one can achieve the following results. Assuming that the Euclidean norm of a random binary vector is roughly $\|\mathbf{e}\| \approx \sqrt{\frac{m}{2}}$, one can balance the quality of the basis reduction and the success probability of the nearest plane algorithm to obtain the optimal trade-off $\delta_3 \approx 1.0067$, taking roughly $T_3 \approx 2^{94}$ operations, with a success probability of roughly 2^{-31}.

Missing or incorrect optimization

In some works such as [14, 20], the optimization of the attack parameters is either completely missing, ignoring the fact that there is a trade-off between the time spent on basis reduction and the actual attack, or incorrect. As a result, one only obtains upper bounds on the estimated security level but not precise estimates.

Other inaccuracies

Further inaccuracies we encountered include the following:

(1) Implicitly assuming that the meet-in-the-middle part $\mathbf{v}_g$ of the short vector has the right number of i-entries for each i [8, 14, 18, 21, 30]. This is not the case in general and therefore needs to be accounted for in the success probability.
(2) Simplifying the structure of the secret key when convenient in order to ease the analysis [18, 30]. This can drastically change the norm of the secret vector and in consequence manipulate the runtime estimates.
(3) Assuming that an attacker could maybe utilize some algebraic structure without any evidence that this is the case [17, 18, 30]. This assumption results in security underestimates if the assumption is in fact wrong.

5 Updating security estimates against the hybrid attack

In this section, we apply our improved analysis of the hybrid attack to various cryptographic schemes in order to reevaluate their security and derive updated security estimates. The section is structured as follows: Each scheme is analyzed in a separate subsection. We begin with subsections on the encryption schemes NTRU, NTRU Prime and R-BinLWEEnc, and end with subsections on the signature schemes BLISS and GLP. In each subsection, we first give a brief introduction to the scheme. We then apply the hybrid attack to the scheme and analyze its complexity according to Section 4. This analysis is performed with the following four steps:

(1) *Constructing the lattice.* We first construct a lattice of the required form which contains the secret key as a short vector.
(2) *Determining the attack parameters.* We find suitable attack parameters c_i (depending on the meet-in-the-middle dimension r), infinity norm bounds y and k, and estimate the Euclidean Y.
(3) *Determining the success probability.* We determine the success probability of the attack according to Section 4.2.
(4) *Optimizing the runtime.* We optimize the runtime of the attack for our standard and conservative security estimates according to Section 4.3.

We end each subsection by providing a table of updated security estimates against the hybrid attack obtained by our analysis. In the tables, we also provide the optimal attack parameters (δ_r, r) derived by our optimization process and the corresponding probability p with which collisions can be detected. For comparison, we further provide the security estimates of the previous works. In our runtime optimization of the attack, we optimized with a precision of up to one bit. As a result, there may not be one unique optimal attack parameter pair (δ_r, r), and for the table, we simply pick one that minimizes the runtime (up to one bit precision).

5.1 NTRU

The NTRU encryption system was officially introduced in [19] and is one of the most important lattice-based encryption schemes today due to its high efficiency. The hybrid attack was first developed to attack NTRU [20] and has been applied to various proposed parameter sets since [17, 18, 20, 21, 30]. In this work, we restrict our studies to the NTRU EESS # 1 parameter sets given in [18, Table 3].

Constructing the lattice

The NTRU cryptosystem is defined over the ring $R_q = \mathbb{Z}_q[X]/(X^N - 1)$, where $N, q \in \mathbb{N}$, and N is prime. The parameters N and q are public. Furthermore, there exist public parameters $d_1, d_2, d_3, d_g \in \mathbb{Z}$. For the parameter sets considered in [18], the private key is a pair of polynomials $(f, g) \in R_q^2$, where g is a trinary polynomial with exactly $d_g + 1$ ones and d_g minus ones and $f = 1 + 3F$ invertible in R_q with $F = A_1A_2 + A_3$ for some trinary polynomials A_i with exactly d_i one and d_i minus one entries. The corresponding public key is $(1, h)$, where $h = f^{-1}g$. In the following, we assume that h and 3 are invertible in R_q. We further identify polynomials with their coefficient vectors. We can recover the private key by finding the secret vector $\mathbf{v} = (\mathbf{F}, \mathbf{g})^t$.[5] Since $h = (1 + 3F)^{-1}g$, we have $3^{-1}h^{-1}g = F + 3^{-1}$, and therefore it holds that

$$\mathbf{v} + \begin{pmatrix} \mathbf{3}^{-1} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 3^{-1}\mathbf{h}^{-1}\mathbf{g} + q\mathbf{w} \\ \mathbf{g} \end{pmatrix} = \left(\begin{array}{c|c} q\mathbf{I}_n & 3^{-1}\overline{\mathbf{H}} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right) \begin{pmatrix} \mathbf{w} \\ \mathbf{g} \end{pmatrix}$$

for some $\mathbf{w} \in \mathbb{Z}^n$, where $\overline{\mathbf{H}}$ is the rotation matrix of $\mathbf{h}^{-1}$. Hence, $\mathbf{v}$ can be recovered by solving the BDD on

5 Note that we put $\mathbf{g}$ in the half of the vector $\mathbf{v}$ that is guessed in the meet-in-the-middle part of the attack. The reason for this choice is that we exactly know the structure of $\mathbf{g}$ but not the structure of the product form polynomial $\mathbf{F}$.

input $(-\mathbf{3}^{-1}, \mathbf{0})^t$ in the q-ary lattice

$$\Lambda = \Lambda\left(\left(\begin{array}{c|c} q\mathbf{I}_n & 3^{-1}\overline{\mathbf{H}} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\right),$$

since $(-\mathbf{3}^{-1}, \mathbf{0})^t - \mathbf{v} \in \Lambda$.[6] A similar way to recover the private key was already mentioned in [30]. The lattice Λ has dimension $2n$ and determinant q^n. Since we take the BDD approach for the hybrid attack, we assume that only $\mathbf{v}$, not its rotations or additive inverse, can be found by the attack, see Section 3. Hence, we assume that the set S, as defined in Heuristic 1, consists of at most one element.

Determining the attack parameters

Let $\mathbf{v} = (\mathbf{F}, \mathbf{g})^t = (\mathbf{v}_l, \mathbf{v}_g)^t$ with $\mathbf{v}_l \in \mathbb{Z}^{2n-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Since $\mathbf{g}$ is a trinary vector, we can set the infinity norm bound k on $\mathbf{v}_g$ equal to one. In contrast, determining an infinity norm bound on the vector $\mathbf{v}_l$ is not that trivial, since $\mathbf{F}$ is not trinary but of product form. For a specific parameter set, this can either be done theoretically or experimentally. The same holds for estimating the Euclidean norm of $\mathbf{v}_l$. For our runtime estimates, we determined the expected Euclidean norm of $\mathbf{F}$ experimentally and set the expected Euclidean norm of $\mathbf{v}_l$ to

$$\|\mathbf{v}_l\| \approx \sqrt{\|\mathbf{F}\|^2 + \frac{n-r}{n} \cdot (2d_g + 1)}.$$

We set $2c_{-1} = \frac{r}{n} \cdot (d_g + 1)$ and $2c_1 = \frac{r}{n} \cdot d_g$ to be equal to the expected number of minus one entries and one entries, respectively, in $\mathbf{g}$.[7] For simplicity, we assume that c_{-1} and c_1 are integers in the following in order to avoid writing down the rounding operates.

Determining the success probability

The next step is to determine the success probability p_{succ}, i.e., the probability that $\mathbf{v}$ has exactly $2c_{-1}$ entries equal to minus one, $2c_1$ entries equal to one, and $\text{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$ holds, where $\mathbf{B}$ is as given in Heuristic 1. Assuming independence, the success probability is approximately

$$p_{\text{succ}} \approx p_c \cdot p_{\text{NP}},$$

where p_c is the probability that $\mathbf{v}$ has exactly $2c_{-1}$ entries equal to minus one and $2c_1$ entries equal to one, and p_{NP} is defined and calculated as in Section 4.2. Obviously, p_c is given by

$$p_c = \frac{\binom{r}{2\tilde{c}_0, 2c_{-1}, 2c_1}\binom{n-r}{d_0 - 2\tilde{c}_0, d_g - 2c_{-1}, d_g + 1 - 2c_1}}{\binom{n}{d_0, d_g, d_g + 1}},$$

where $2\tilde{c}_0 = r - 2c_{-1} - 2c_1$ and $d_0 = n - (d_g + 1) - d_g$. As explained earlier, since we use the BDD approach of the hybrid attack, we assume that $|S| = 1$ in case the attack is successful.

Optimizing the runtime

We determined the optimal attack parameters to estimate the minimal runtime of the hybrid attack for the NTRU EESS # 1 parameter sets given in [18, Table 3]. The results, including the optimal r, corresponding δ_r and resulting probability p that collisions can be found, are presented in Table 3. Our analysis shows that

6 It is also possible to construct a lattice that contains $(\mathbf{f}, \mathbf{g})$ as a short vector instead. However, since $f = 1 + 3F$ has norm larger than F, this leads to a less efficient attack.

7 Note that this must not necessarily be the optimal choice for the c_i. However, we expect that this choice comes very close to the optimal one and therefore restrict our studies to this case.

Parameter set	n = 401	n = 439	n = 593	n = 743
Optimal $r_{\text{cons}}/r_{\text{std}}$	104/122	122/140	206/219	290/308
Optimal $\delta_{\text{r,cons}}$	1.00544	1.00509	1.00412	1.00352
Optimal $\delta_{\text{r,std}}$	1.00552	1.00518	1.00420	1.00357
Corresp. p_{succ} cons/std	$2^{-70}/2^{-43}$	$2^{-56}/2^{-47}$	$2^{-67}/2^{-62}$	$2^{-78}/2^{-69}$
Security cons/std in bits	**145/162**	**165/182**	**249/267**	**335/354**
In [18] cons/std	116/127	133/145	204/236	280/330
$r_{\text{cons}}/r_{\text{std}}$ used in [18]	154/166	175/192	264/303	360/423

Table 3: Optimal attack parameters and security levels against the hybrid attack for NTRU.

the security levels against the hybrid attack claimed in [18] are lower than the actual security levels for all parameter sets. In addition, our results show that for all of the analyzed parameter sets, the hybrid attack does not perform better than a purely combinatorial meet-in-the-middle search, see [18, Table 3]. Our results therefore disprove the common claim that the hybrid attack is necessarily the best attack on NTRU.

5.2 NTRU Prime

The NTRU Prime encryption scheme was recently introduced [8] in order to eliminate worrisome algebraic structures that exist within NTRU [19] or Ring-LWE based encryption schemes such as [3, 25].

Constructing the lattice

The Streamlined NTRU Prime family of cryptosystems is parameterized by three integers $(n, q, t) \in \mathbb{N}^3$, where n and q are odd primes. The base ring for Streamlined NTRU Prime is $R_q = \mathbb{Z}_q[X]/(X^n - X - 1)$. The private key is (essentially) a pair of polynomials $(g, f) \in R_q^2$, where g is drawn uniformly at random from the set of all trinary polynomials, and f is drawn uniformly at random from the set of all trinary polynomials with exactly $2t$ non-zero coefficients. The corresponding public key is $h = g(3f)^{-1} \in R_q$. In the following, we identify polynomials with their coefficient vectors. As described in [8], the secret vector $\mathbf{v} = (\mathbf{g}, \mathbf{f})$ is contained in the q-ary lattice

$$\Lambda = \Lambda\left(\left(\begin{array}{c|c} q\mathbf{I}_n & 3\mathbf{H} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\right),$$

where $\mathbf{H}$ is the rotation matrix of h, since

$$\left(\begin{array}{c|c} q\mathbf{I}_n & 3\mathbf{H} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\begin{pmatrix} \mathbf{w} \\ \mathbf{f} \end{pmatrix} = \begin{pmatrix} q\mathbf{w} + 3\mathbf{hf} \\ \mathbf{f} \end{pmatrix} = \begin{pmatrix} \mathbf{g} \\ \mathbf{f} \end{pmatrix} = \mathbf{v}$$

for some $\mathbf{w} \in \mathbb{Z}^n$. The determinant of the lattice Λ is given by q^n, and its dimension is equal to $2n$. Note that in the case of Streamlined NTRU Prime, the rotations of a trinary polynomial are not necessarily trinary, but it is likely that some are. The authors of [8] conservatively assume that the maximum number of good rotations of $\mathbf{v}$ that can be utilized by the attack is $n - t$, which we also assume in the following. Counting their additive inverses leaves us with $2(n - t)$ short vectors that can be found by the attack.

Determining the attack parameters

Let $\mathbf{v} = (\mathbf{f}, \mathbf{g})^t = (\mathbf{v}_l, \mathbf{v}_g)^t$ with $\mathbf{v}_l \in \mathbb{Z}^{2n-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Since $\mathbf{v}$ is trinary, we can set the infinity norm bounds y and k equal to one. The expected Euclidean norm of $\mathbf{v}_l$ is given by

$$\|\mathbf{v}_l\| \approx \sqrt{\frac{2}{3}n + \frac{n-r}{n}2t}.$$

We set $2c_1 = 2c_{-1} = \frac{r}{n} \cdot \frac{t}{2}$ equal to the expected number of one entries (or minus one entries, respectively) in $\mathbf{f}$. For simplicity, we assume that c_1 is an integer in the following.

Determining the success probability

Next, we determine the success probability $p_{\text{succ}} = \Pr[S \neq \emptyset]$, where S denotes the following subset of the lattice Λ:

$$\begin{aligned} S = \{\mathbf{w} \in \Lambda \mid \mathbf{w} = (\mathbf{w}_l, \mathbf{w}_g)^t \text{ with } \mathbf{w}_l \in \{0, \pm 1\}^{2n-r},\ \mathbf{w}_g \in \{0, \pm 1\}^r, \\ \text{exactly } 2c_i \text{ entries of } \mathbf{w}_g \text{ equal to } i \text{ for all } i \in \{-1, 1\}, \\ \mathrm{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{w}_g) = \mathbf{w}_l\}, \end{aligned}$$

where $\mathbf{B}$ is as defined in Heuristic 1. We assume that S is a subset of all the rotations of $\mathbf{v}$ that can be utilized by the attack and their additive inverses. In particular, we assume that S has at most $2(n-t)$ elements. Note that if some vector $\mathbf{w}$ is contained in S, then we also have $-\mathbf{w} \in S$. Assuming independence, the probability p_S that $\mathbf{v} \in S$ is approximately given by

$$p_S \approx \frac{\binom{r}{2\tilde{c}_0, 2c_{-1}, 2c_1}\binom{n-r}{2t-4c_1}2^{2t-4c_1}}{\binom{n}{2t}2^{2t}} \cdot p_{\text{NP}},$$

where $d_0 = n - 2t$ and $2\tilde{c}_0 = r - 4c_1$, and p_{NP} is defined and calculated as in Section 4.2. Assuming independence, all of the $n-t$ good rotations of $\mathbf{v}$ are contained in S with probability p_S as well. Therefore, the probability p_{succ} that we have at least one good rotation is approximately

$$p_{\text{succ}} = \Pr[S \neq \emptyset] \approx 1 - (1 - p_S)^{n-t}.$$

Next, we estimate the size of the set S in the case $S \neq \emptyset$, i.e., Algorithm 1 is successful. In that case, at least one rotation is contained in S. Then also its additive inverse is contained in S, hence $|S| \geq 2$. We can estimate the size of S in case of success to be

$$|S| \approx 2 + 2(n - t - 1)p_S,$$

where p_S is defined as above.

Optimizing the runtime

We applied our new techniques to estimate the minimal runtimes for several NTRU Prime parameter sets proposed in [8, Appendix D]. Besides the "case study parameter set", for our analysis, we picked one parameter set that offers the lowest bit security and one that offers the highest according to the analysis of [8]. Our resulting security estimates are presented in Table 4. Our analysis shows that the authors of [8] underestimate the security of their scheme for all parameter sets we evaluated.

5.3 R-BinLWEEnc

In [9], Buchmann et al. presented R-BinLWEEnc, a lightweight public key encryption scheme based on binary Ring-LWE[8]. To determine the security of their scheme, the authors evaluate the hardness of binary LWE against the hybrid attack using the methodology of [10].

8 For more details on (the hardness of) LWE, Ring-LWE and binary LWE, we refer to [2, 6, 10, 25, 26, 29].

Parameter set	n = 607 q = 18749	n = 739 q = 9829	n = 929 q = 12953
Optimal $r_{\text{cons}}/r_{\text{std}}$	148/162	235/257	328/353
Optimal $\delta_{\text{r,cons}}$	1.00466	1.00405	1.00346
Optimal $\delta_{\text{r,std}}$	1.00466	1.00407	1.00346
Corresponding p_{succ} cons/std	$2^{-63}/2^{-54}$	$2^{-73}/2^{-60}$	$2^{-80}/2^{-65}$
Security cons/std in bits	**197/211**	**258/273**	**346/363**
In [8]	128	228	310

Table 4: Optimal attack parameters and security levels against the hybrid attack for NTRU Prime.

Constructing the lattice

Let $m, n, q \in \mathbb{Z}$ with $m > n$ and $(\mathbf{A}, \mathbf{b}' = \mathbf{A}\mathbf{s} + \mathbf{e}' \bmod q)$ be a binary LWE instance with $\mathbf{A} \in \mathbb{Z}_q^{m\times n}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and binary error $\mathbf{e}' \in \{0, 1\}$.[9] To obtain a more efficient attack, we first subtract the vector $(0.5, \dots, 0.5, 0, \dots, 0)$ with $m - r$ non-zero and r zero entries from both sides of the equation $\mathbf{b}' = \mathbf{A}\mathbf{s} + \mathbf{e}' \bmod q$ to obtain a new LWE instance $(\mathbf{A}, \mathbf{b} = \mathbf{A}\mathbf{s} + \mathbf{e} \bmod q)$, where $\mathbf{e} \in \{\pm 0.5\}^{m-r} \times \{0, 1\}^r$. This way, the expected norm of the first $m - r$ entries is reduced while the last r entries, which are guessed during the attack, remain unchanged. In the following, we only consider this transformed LWE instance with smaller error. Obviously, the vector $(\mathbf{e}, 1)$ is contained in the q-ary lattice

$$\Lambda = \Lambda_q(\mathbf{A}') = \{\mathbf{v} \in \mathbb{Z}^{m+1} \mid \text{there exists } \mathbf{w} \in \mathbb{Z}^{n+1} \text{ such that } \mathbf{v} = \mathbf{A}'\mathbf{w} \bmod q\},$$

where

$$\mathbf{A}' = \begin{pmatrix} \mathbf{A} & \mathbf{b} \\ \mathbf{0} & 1 \end{pmatrix} \in \mathbb{Z}_q^{(m+1)\times(n+1)}.$$

Note that constructing the lattice this way, we only need the error vector $\mathbf{e}'$ to be binary and not also the secret $\mathbf{s}$ as in [9, 10]. The dimension of the lattice Λ is equal to $m + 1$, and with high probability, its determinant is $q^{m-(n+1)}$, see, for example, [6]. However, as we know the last component of $(\mathbf{e}, 1)$, it does not need to be guessed, and we may hence ignore it for the hybrid attack and consider the lattice to be of dimension m.

Determining the attack parameters

Let $\mathbf{v} = \mathbf{e} = (\mathbf{v}_l, \mathbf{v}_g)^t$ with $\mathbf{v}_l \in \{\pm\frac{1}{2}\}^{m-r}$ and $\mathbf{v}_g \in \{\pm\frac{1}{2}\}^r$. Then obviously, we have $\|\mathbf{v}\|_\infty \le \frac{1}{2}$, so we set the infinity norm bounds $y = k = \frac{1}{2}$. Since $\mathbf{v}_l$ is a uniformly random vector in $\{\pm\frac{1}{2}\}^{m-r}$, the expected Euclidean norm of $\mathbf{v}_l$ is

$$\|\mathbf{v}_l\| \approx \sqrt{\frac{m-r}{4}}.$$

We set $2c_{-1/2} = 2c_{1/2} = \frac{r}{2}$ to be the expected number of $-\frac{1}{2}$ and $\frac{1}{2}$ entries of $\mathbf{v}_g$. In the following, we assume that $c_{-1/2} = c_{1/2}$ is an integer in order to not have to deal with rounding operators.

Determining the success probability

We can approximate the success probability p_{succ} by $p_{\text{succ}} \approx p_c \cdot p_{\text{NP}}$, where p_c is the probability that $\mathbf{v}_g$ has exactly $2c_{-1/2}$ entries equal to $-\frac{1}{2}$ and $2c_{1/2}$ entries equal to $\frac{1}{2}$, and p_{NP} is defined as in Section 4.2. Using the fact that $2c_{-1/2} + 2c_{1/2} = r$, we therefore obtain

$$p_{\text{succ}} \approx p_c \cdot p_{\text{NP}} = 2^{-r}\binom{r}{2c_{1/2}} p_{\text{NP}}.$$

9 Note that with our approach, we only need that the error vector $\mathbf{e}'$ is binary, and not also that the secret vector $\mathbf{s}$ is binary, as demanded in [9, 10].

Parameter set	Set-I	Set-II	Set-III
Optimal $r_{\text{cons}}/r_{\text{std}}$	104/116	88/96	276/268
Optimal $\delta_{\text{r,cons}}$	1.00691	1.00731	1.00478
Optimal $\delta_{\text{r,std}}$	1.00698	1.00741	1.00487
Corresponding p_{succ} cons/std	$2^{-33}/2^{-27}$	$2^{-31}/2^{-28}$	$2^{-38}/2^{-43}$
Security cons/std in bits	**88/99**	**79/90**	**187/197**
In [9]	94	84	190

Table 5: Optimal attack parameters and security levels against the hybrid attack for R-BinLWEEnc.

We assume that if the attack is successful, then $|S| = 2$, where S is defined as in Heuristic 1, since $\mathbf{e}$ and $-\mathbf{e}$ are assumed to be the only vectors that can be found by the attack.

Optimizing the runtime

We reevaluated the security of the R-BinLWEEnc parameter sets proposed in [9]. Our security estimates, the optimal attack parameters r and δ_r, and the corresponding probability p are presented in Table 5. The original security estimates given in [9] are within the security range we determined.

5.4 BLISS

The signature scheme BLISS, introduced in [14], is one of the most important lattice-based signature schemes. In the original paper, the authors considered the hybrid attack on their signature scheme for their security estimates; however, their analysis is rather vague.

Constructing the lattice

In the BLISS signature scheme, the setup is the following: Let n be a power of two, $d_1, d_2 \in \mathbb{N}$ such that $d_1 + d_2 \leq n$ holds, q a prime modulus with $q \equiv 1 \bmod 2n$, and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. The signing key is of the form $(s_1, s_2) = (f, 2g + 1)$, where $f \in \mathcal{R}_q^\times$, $g \in \mathcal{R}_q$, each with d_1 coefficients in $\{\pm 1\}$ and d_2 coefficients in $\{\pm 2\}$, and the remaining coefficients equal to 0. The public key is essentially $a = \frac{s_2}{s_1} \in \mathcal{R}_q$. We assume that a is invertible in $\mathcal{R}_q$, which is the case with very high probability. Hence, we obtain the equation $s_1 = s_2 a^{-1} \in \mathcal{R}_q$, or equivalently $f = 2ga^{-1} + a^{-1} \bmod q$. In the following, we identify polynomials with their coefficient vectors.

In order to recover the signing key, it is sufficient to find the vector $\mathbf{v} = (\mathbf{f}, \mathbf{g})^t$. Similar to our previous analysis of NTRU in Section 5.1, we have that

$$\mathbf{v} + \begin{pmatrix} -\mathbf{a}^{-1} \\ \mathbf{0} \end{pmatrix} = \begin{pmatrix} 2\mathbf{g}\mathbf{a}^{-1} + q\mathbf{w} \\ \mathbf{g} \end{pmatrix} = \left(\begin{array}{c|c} q\mathbf{I}_n & 2\mathbf{A} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right) \begin{pmatrix} \mathbf{w} \\ \mathbf{g} \end{pmatrix}$$

for some $\mathbf{w} \in \mathbb{Z}^n$, where $\mathbf{A}$ is the rotation matrix of $\mathbf{a}^{-1}$. Hence, $\mathbf{v}$ can be recovered by solving the BDD on input $(\mathbf{a}^{-1}, \mathbf{0})^t$ in the q-ary lattice

$$\Lambda = \Lambda\left(\left(\begin{array}{c|c} q\mathbf{I}_n & 2\mathbf{A} \\ \hline \mathbf{0} & \mathbf{I}_n \end{array}\right)\right),$$

since $(\mathbf{a}^{-1}, \mathbf{0})^t - \mathbf{v} \in \Lambda$. The determinant of the lattice Λ is q^n, and its dimension is equal to $2n$.

Determining the attack parameters

In the following, let $\mathbf{v} = (\mathbf{f}, \mathbf{g})^t = (\mathbf{v}_l, \mathbf{v}_g)^t$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Since we are using the hybrid attack to solve a BDD problem, the rotations of $\mathbf{v}$ cannot be utilized in the attack (or at least it is not known how),

Parameter set	BLISS-I	BLISS-II	BLISS-III	BLISS-IV
Optimal $r_{\text{cons}}/r_{\text{std}}$	152/152	152/152	109/144	99/137
Optimal $\delta_{\text{r,cons}}$	1.00588	1.00588	1.00532	1.00518
Optimal $\delta_{\text{r,std}}$	1.00600	1.00600	1.00541	1.00524
Corresp. p_{succ} cons/std	$2^{-35}/2^{-38}$	$2^{-35}/2^{-38}$	$2^{-58}/2^{-40}$	$2^{-67}/2^{-44}$
Security cons/std in bits	**124/139**	**124/139**	**152/170**	**160/182**
In [14]	128	128	160	192
r used in [14]	194	194	183	201

Table 6: Optimal attack parameters and security levels against the hybrid attack for BLISS.

see Section 3. We therefore assume that $\mathbf{v}$ is the only rotation useful in the attack, i.e., that the set of good rotations S contains at most $\mathbf{v}$. The first step is to determine proper bounds y on $\|\mathbf{v}_l\|_\infty$ and k on $\|\mathbf{v}_g\|_\infty$ and find suitable guessing parameters c_i. By construction, we obviously have $\|\mathbf{v}\|_\infty \leq 2$, thus we can set the infinity norm bounds $y = k = 2$. The expected Euclidean norm of $\mathbf{v}_l$ is given by

$$\|\mathbf{v}_l\| \approx \sqrt{d_1 + 4d_2 + \frac{n-r}{n}(1d_1 + 4d_2)}.$$

We set $2c_i$ equal to the expected number of i-entries in $\mathbf{v}_g$, i.e., $c_{-2} = c_2 = \frac{r}{n} \cdot \frac{1}{4} d_2$ and $c_{-1} = c_1 = \frac{r}{n} \cdot \frac{1}{4} d_1$. For simplicity, we assume that c_1 and c_2 are integers in the following.

Determining the success probability

Next, we determine the success probability p_{succ}, which is the probability that $\text{NP}_{\mathbf{B}}(\mathbf{C}\mathbf{v}_g) = \mathbf{v}_l$ and exactly $2c_i$ entries of $\mathbf{v}_g$ are equal to i for $i \in \{\pm 1, \ldots, \pm k\}$. The probability p_c that exactly $2c_i$ entries of the vector $\mathbf{v}_g$ are equal to i for all $i \in \{\pm 1, \ldots, \pm k\}$ is given by

$$\frac{\binom{r}{2\tilde{c}_0, 2c_{-2}, 2c_2, 2c_{-4}, 2c_4}\binom{n-r}{d_0 - 2\tilde{c}_0, d_1 - 4c_2, d_2 - 4c_4} 2^{d_1+d_2-4(c_2+c_4)}}{\binom{n}{d_0, d_1, d_2} 2^{d_1+d_2}},$$

where $d_0 = n - d_1 - d_2$ and $2\tilde{c}_0 = r - 2(c_{-2} + c_2 + c_{-4} + c_4)$. Assuming independence, the success probability is approximately given by

$$p_{\text{succ}} \approx p_c \cdot p_{\text{NP}},$$

where p_{NP} is defined as in Section 4.2. As explained earlier, we assume that $S \subset \{\mathbf{v}\}$; so if Algorithm 1 is successful, we have $|S| = 1$.

Optimizing the runtime

We performed the optimization process for the BLISS parameter sets proposed in [14]. The results are presented in Table 6. Besides the security levels against the hybrid attack, we provide the optimal attack parameters r and δ_r leading to a minimal runtime of the attack, as well as the probability p. Our results show that the security estimates for the BLISS-I, BLISS-II, and BLISS-III parameter sets given in [14] are within the range of security we determined, whereas the BLISS-IV parameter set is less secure than originally claimed. In addition, the authors of [14] claim that there are at least 17 bits of security margins built into their security estimates, which is incorrect for all parameter sets according to our analysis.

5.5 GLP

The GLP signature scheme was introduced in [16]. In the original work, the authors did not consider the hybrid attack when deriving their security estimates. Later, in [14], the hybrid attack was also applied to the GLP-I parameter set. The GLP-II parameter set has not been analyzed regarding the hybrid attack so far.

Constructing the lattice

For the GLP signature scheme the setup is the following: Let n be a power of two, q a prime modulus with $q \equiv 1 \bmod 2n$, and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$. The signing key is of the form (s_1, s_2), where s_1 and s_1 are sampled uniformly at random among all polynomials of $\mathcal{R}_q$ with coefficients in $\{-1, 0, 1\}$. The corresponding public key is then of the form $(a, b = as_1 + s_2) \in \mathcal{R}_q^2$, where a is drawn uniformly at random in $\mathcal{R}_q$. So we know that $0 = -b + as_1 + s_2$. Identifying polynomials with their coefficient vectors, we therefore have that

$$\mathbf{v} := \begin{pmatrix} -1 \\ \mathbf{s}_1 \\ \mathbf{s}_2 \end{pmatrix} \in \Lambda := \Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{w} \in \mathbb{Z}^{2n+1} \mid \mathbf{A}\mathbf{w} \equiv \mathbf{0} \bmod q\} \subset \mathbb{Z}^{2n+1},$$

where $\mathbf{A} = (\mathbf{b}|\mathrm{rot}(\mathbf{a})|\mathbf{I}_n)$, and $\mathrm{rot}(\mathbf{a})$ is the rotation matrix of $\mathbf{a}$. Because of how the lattice is constructed, we do not assume that rotations of $\mathbf{v}$ can by utilized by the attack.[10] Therefore, with very high probability, $\mathbf{v}$ and $-\mathbf{v}$ are the only non-zero trinary vectors contained in Λ, which we assume in the following. Since q is prime and $\mathbf{A}$ has full rank, we have that $\det \Lambda = q^n$, see for example [7]. In Appendix A.1, we show how to construct a basis of the form

$$\mathbf{B}' = \left(\begin{array}{c|c} q\mathbf{I}_n & \star \\ \hline \mathbf{0} & \mathbf{I}_{n+1} \end{array}\right) \in \mathbb{Z}^{(2n+1)\times(2n+1)}$$

for the q-ary lattice Λ.

Determining the attack parameters

Ignoring the first -1 coordinate, the short vector $\mathbf{v}$ is drawn uniformly from $\{-1, 0, 1\}^{2n+1}$. Let $\mathbf{v} = (\mathbf{v}_l, \mathbf{v}_g)^t$ with $\mathbf{v}_l \in \mathbb{Z}^{m-r}$ and $\mathbf{v}_g \in \mathbb{Z}^r$. Then obviously, $\|\mathbf{v}_l\|_\infty \le 1$ and $\|\mathbf{v}_g\|_\infty \le 1$ hold; so we can set the infinity norm bounds y and k equal to one. The expected Euclidean norm of $\mathbf{v}_l$ is approximately

$$\|\mathbf{v}_l\| \approx \sqrt{2(m-r)/3}.$$

We set $2c_{-1} = 2c_1 = \frac{r}{3}$ to be the expected number of ones and minus ones. For simplicity, we assume that $c_{-1} = c_1$ is an integer in the following.

Determining the success probability

The success probability p_{succ} of the attack is approximately $p_{\mathrm{succ}} \approx p_c \cdot p_{\mathrm{NP}}$, where p_c is the probability that $\mathbf{v}_g$ hat exactly $2c_{-1}$ minus one entries and $2c_1$ one entries, and p_{NP} is defined as in Section 4.2. Calculating p_c yields

$$p_{\mathrm{succ}} \approx p_c \cdot p_{\mathrm{NP}} = 3^{-r}\binom{r}{\frac{r}{3}, \frac{r}{3}, \frac{r}{3}} p_{\mathrm{NP}}.$$

As previously mentioned, we assume that if the attack is successful, then $|S| = 2$.

10 It is possible to construct a lattice which contains all the rotations as short vectors, but this unnecessarily blows up the lattice dimension.

Parameter set	GLP-I	GLP-II
Optimal $r_{\text{cons}}/r_{\text{std}}$	30/54	168/192
Optimal $\delta_{\text{r,cons}}$	1.00776	1.00450
Optimal $\delta_{\text{r,std}}$	1.00769	1.00451
Corresponding p_{succ} cons/std	$2^{-41}/2^{-25}$	$2^{-61}/2^{-49}$
Security cons/std in bits	**71/88**	**212/233**
In [14, 16]	75 to 80	≥ 256
r used in [14]	85	—

Table 7: Optimal attack parameters and security levels against the hybrid attack for GLP.

Optimizing the runtime

We performed the optimization for the GLP parameter sets proposed in [16]. The results, including the optimal attack parameters r and δ_r and the probability p, are shown in Table 7. The security level of the GLP-I parameter set claimed in [14] is within the range of security we determined. In [14], the authors did not analyze the hybrid attack for the GLP-II parameter set. Güneysu et al. [16] claimed a security level of at least 256 bits (not considering the hybrid attack) for the GLP-II parameter set, whereas we show that it offers at most 233 bits of security against the hybrid attack.

6 Conclusion and future work

In this work, we described a general version of the hybrid attack and presented improved techniques to analyze its runtime. We further reevaluated various cryptographic schemes regarding their security against the hybrid attack. Our analysis shows that several of the old security estimates of previous works were in fact unreliable. By updating these unreliable estimates, we contributed to the trustworthiness of security estimates of lattice-based cryptography.

For future work, we hope that more provable statements about the practicality of the hybrid attack can be derived. For instance, our results show that the hybrid attack is not the best known attack on all NTRU instances as previously thought. It would be interesting to prove that under certain conditions on the key structure, the hybrid attack is always outperformed by some other attack. Another possible line of future work is applying the hybrid attack to a broader range of cryptographic schemes than already done in this work. Furthermore, analyzing the memory requirements of the hybrid attack was out of the scope of this work. In future research, the memory requirements can be analyzed and reduced (as for example done in [32] for binary NTRU keys). If required, the memory consumption can then be taken into account when optimizing the attack parameters for the hybrid attack.

A Appendix: On q-ary lattices

A.1 Constructing a basis of the required form

In the following lemma, we show that for q-ary lattices, where q is prime, there always exists a basis of the form required for the attack. The size of the identity in the bottom right corner of the basis depends on the determinant of the lattice. In the proof, we also show how to construct such a basis.

Lemma A.1. *Let q be prime, $m \in \mathbb{N}$, and let $\Lambda \subset \mathbb{Z}^m$ be a q-ary lattice.*
(i) *There exists some $k \in \mathbb{Z}$, $0 \le n \le m$ such that $\det(\Lambda) = q^k$.*
(ii) *Let $\det(\Lambda) = q^k$. Then there is a matrix $\mathbf{A} \in \mathbb{Z}_q^{m\times(m-k)}$ of rank $m-k$ (over $\mathbb{Z}_q$) such that $\Lambda = \Lambda_q(\mathbf{A})$.*

(iii) *Let* $\det(\Lambda) = q^k$ *and* $\mathbf{A} = \binom{\mathbf{A}_1}{\mathbf{A}_2}$ *with* $\mathbf{A}_1 \in \mathbb{Z}_q^{k\times(m-k)}$, $\mathbf{A}_2 \in \mathbb{Z}_q^{(m-k)\times(m-k)}$ *be a matrix of rank* $m-k$ *(over* $\mathbb{Z}_q$*) such that* $\Lambda = \Lambda_q(\mathbf{A})$. *If* $\mathbf{A}_2$ *is invertible over* $\mathbb{Z}_q$, *then the columns of the matrix*

$$\mathbf{B}' = \left(\begin{array}{c|c} q\mathbf{I}_k & \mathbf{A}_1\mathbf{A}_2^{-1} \\ \hline \mathbf{0} & \mathbf{I}_{m-k} \end{array}\right) \in \mathbb{Z}^{m\times m}$$

form a basis of the lattice Λ.

Proof. (i) Obviously, $\det(\Lambda) \mid \det(q\mathbb{Z}^m) = q^m$, since $q\mathbb{Z}^m \subset \Lambda$, and therefore $\det(\Lambda)$ is some non-negative power of q, because q is prime.

(ii) We have $(\mathbb{Z}^m : q\mathbb{Z}^m) = (\mathbb{Z}^m : \Lambda) \cdot (\Lambda : q\mathbb{Z}^m)$, and therefore

$$(\Lambda : q\mathbb{Z}^m) = \frac{(\mathbb{Z}^m : q\mathbb{Z}^m)}{(\mathbb{Z}^m : \Lambda)} = \frac{\det(q\mathbb{Z}^m)}{\det(\Lambda)} = q^{m-k}.$$

Let $\mathbf{A}' \in \mathbb{Z}_q^{m\times m}$ be some lattice basis of Λ. Since $\Lambda/q\mathbb{Z}^m$ is in one-to-one correspondence to the $\mathbb{Z}_q$-vector space spanned by $\mathbf{A}'$, this vector space has to be of dimension $m-k$, and therefore $\mathbf{A}'$ has rank $m-k$ over $\mathbb{Z}_q$. This implies that there is some matrix $\mathbf{A}$ consisting of $m-k$ columns of $\mathbf{A}'$ such that $\Lambda = \Lambda(q\mathbf{I}_m \mid \mathbf{A}) = \Lambda_q(\mathbf{A})$.

(iii) By assumption, $\mathbf{A}_2$ is invertible, and thus we have

$$\begin{aligned}
\Lambda &= \{\mathbf{v} \in \mathbb{Z}^m \mid \text{there exists a } \mathbf{w} \in \mathbb{Z}^{(m-k)} \text{ such that } \mathbf{v} = \mathbf{A}\mathbf{w} \bmod q\} \\
&= \left\{\mathbf{v} \in \mathbb{Z}^m \;\middle|\; \text{there exists a } \mathbf{w} \in \mathbb{Z}^{(m-k)} \text{ such that } \mathbf{v} = \begin{pmatrix}\mathbf{A}_1 \\ \mathbf{A}_2\end{pmatrix}\mathbf{A}_2^{-1}\mathbf{w} \bmod q\right\} \\
&= \left\{\begin{pmatrix}\mathbf{A}_1\mathbf{A}_2^{-1} \\ \mathbf{I}_{m-k}\end{pmatrix}\mathbf{w} \;\middle|\; \mathbf{w} \in \mathbb{Z}^{(m-k)}\right\} + q\mathbb{Z}^m.
\end{aligned}$$

Therefore, the columns of the matrix

$$\left(q\mathbf{I}_m \middle| \begin{matrix}\mathbf{A}_1\mathbf{A}_2^{-1} \\ \mathbf{I}_{m-k}\end{matrix}\right) \in \mathbb{Z}^{m\times(m+(m-k))}$$

form a generating set of the lattice Λ, which can be reduced to the basis $\mathbf{B}'$. □

A.2 Modifying the GSA for q-ary lattices

Typically, the Gram–Schmidt lengths obtained after performing a basis reduction with quality δ can be approximated by the geometric series assumption (GSA), see Section 2. However, for q-ary lattices of the above form, this assumption may be modified. This has already been considered and confirmed with experimental results in previous works, see for example [17, 18, 20, 30]. However, in this work, we derive simple formulas predicting the quality of the reduction, and therefore explain how to obtain these formulas in more detail. We begin by sketching the reason for modifying the GSA for q-ary lattices, given a lattice basis $\mathbf{B}$ of the form

$$\mathbf{B} = \left(\begin{array}{c|c} q\mathbf{I}_a & \star \\ \hline \mathbf{0} & \mathbf{I}_b \end{array}\right) \in \mathbb{Z}^{d\times d},$$

where $d = a + b$. If the basis reduction is not strong enough, i.e., the Hermite delta is too large, the GSA predicts that the first Gram–Schmidt vectors of the reduced basis have norm bigger than q. However, in practice, this will not happen, since in this case, the first vectors will simply not be reduced. This means that instead of reducing the whole basis $\mathbf{B}$, one can just reduce the last vectors that will actually be reduced. Let k denote the (so far unknown) number of the last vectors that are actually reduced (i.e., their corresponding Gram–Schmidt vectors according to the GSA have norm smaller than q). We assume that the basis reduction is sufficiently weak such that $k < d$ and sufficiently strong such that $k > b$. We write $\mathbf{B}$ in the form

$$\mathbf{B} = \left(\begin{array}{c|c} q\mathbf{I}_{d-k} & \mathbf{D} \\ \hline \mathbf{0} & \mathbf{B}_1 \end{array}\right)$$

for some $\mathbf{B}_1 \in \mathbb{Z}^{k\times k}$ and $\mathbf{D} \in \mathbb{Z}^{(d-k)\times k}$. Now, instead of $\mathbf{B}$, we only reduce $\mathbf{B}_1$ to $\mathbf{B}_1' = \mathbf{B}_1\mathbf{U}$ for some unimodular $\mathbf{U} \in \mathbb{Z}^{k\times k}$. This yields a reduced basis

$$\mathbf{B}' = \left(\begin{array}{c|c} q\mathbf{I}_{d-k} & \mathbf{DU} \\ \hline \mathbf{0} & \mathbf{B}_1' \end{array}\right)$$

of $\mathbf{B}$. The Gram–Schmidt basis of this new basis $\mathbf{B}'$ is given by

$$\overline{\mathbf{B}}' = \left(\begin{array}{c|c} q\mathbf{I}_{d-k} & \mathbf{0} \\ \hline \mathbf{0} & \overline{\mathbf{B}}_1' \end{array}\right).$$

Therefore, the lengths of the Gram–Schmidt basis vectors $\overline{\mathbf{B}}'$ are q for the first $d-k$ vectors and then equal to the lengths of the Gram–Schmidt basis vectors $\overline{\mathbf{B}}_1'$, which are smaller than q. In order to predict the lengths of $\overline{\mathbf{B}}'$, we can apply the GSA to the lengths of the Gram–Schmidt basis vectors $\overline{\mathbf{B}}_1'$, since they are actually reduced. What remains is to determine k. Assume we apply a basis reduction on $\mathbf{B}_1$ that results in a reduced basis $\mathbf{B}_1'$ of Hermite delta δ. By our construction, we can assume that the first Gram–Schmidt basis vector of $\overline{\mathbf{B}}_1'$ has norm roughly equal to q, so the GSA implies

$$\delta^k \det(\Lambda(\mathbf{B}_1))^{\frac{1}{k}} = q.$$

Using the fact that $\det(\Lambda(\mathbf{B}_1)) = q^{k-b}$ and $k < d$, we can solve for k and obtain

$$k = \min\left(\left\lfloor \sqrt{\frac{b}{\log_q(\delta)}} \right\rfloor, d\right). \tag{A.1}$$

Summarizing, we expect that after the basis reduction, our Gram–Schmidt basis $\overline{\mathbf{B}}_1'$ has lengths $R_1, \dots, R_d$, where

$$R_i = \begin{cases} q & \text{if } i \le d-k, \\ \delta^{-2(i-(d-k)-1)+k} q^{\frac{k-b}{k}} & \text{else,} \end{cases}$$

and k is given as in equation (A.1).

Note that it might also happen that the last Gram–Schmidt lengths are predicted to be smaller than 1. In this case, these last vectors will also not be reduced in reality, since the basis matrix has the identity in the bottom right corner. Therefore, in this case, the GSA may be further modified. However, for realistic attack parameters, this phenomenon never occurred during our runtime optimizations, and therefore we do not include it in our formulas and leave it to the reader to do the easy calculations if needed.

A.3 Experimental results for the probability *p*

In the following, we provide the results of some preliminary experiments supporting the validity of the formula for the collision-finding probability p provided in Assumption 4. To this end, for $n = 40$, $n = 60$ and $n = 80$, we created q-ary lattices Λ_n that contain a short binary vector by embedding binary LWE instances into an SVP problem according to Section 5.3 (however, we did not shift the first components of the error vector by 0.5). For each n, we chose a random binary LWE instance with LWE parameters n, $m = 2n$, $q = 521$ and created a basis $\mathbf{B}_\mathbf{n}'$ of the corresponding SVP lattice of the form

$$\mathbf{B}_\mathbf{n}' = \left(\begin{array}{c|c} \mathbf{B}_n & \mathbf{C}_n \\ \hline \mathbf{0} & \mathbf{I}_r \end{array}\right) \in \mathbb{Z}^{(m+1)\times(m+1)}$$

with $r = 28$. We then BKZ-reduced the upper-left part $\mathbf{B}_n$ of the basis with block size 20. Let $(\mathbf{v}_l^{(n)}, \mathbf{v}_g^{(n)}) \in \Lambda_n$ with $\mathbf{v}_g^{(n)} \in \{0,1\}^r$ be the short binary vector in Λ_n. For each n, we repeated the following experiment 10,000 times and recorded the number of success cases: Guess a vector $\mathbf{w} \in \{0,1\}^r$ with $\frac{r}{4} = 7$ non-zero entries as in the hybrid attack and check if the vector $\mathbf{Cw}$ is $\mathbf{v}_l^{(n)}$-admissible with respect to the basis $\mathbf{B}_n$ by checking if $\mathrm{NP}(\mathbf{C}_n\mathbf{w}) = \mathrm{NP}(\mathbf{C}_n\mathbf{w} - \mathbf{v}_l^{(n)}) + \mathbf{v}_l^{(n)}$. The experiments were performed using SageMath [31]. For comparison, we calculated the probability p according to the formula provided in Assumption 4 using the actual Gram–Schmidt norms of the basis $\mathbf{B}_n$ and the norm of $\mathbf{v}_l^{(n)}$ to calculate the r_i. The results are presented in Table 8. They suggest that for the analyzed instances, Assumption 4 is a good approximation of the actual probability.

	$n = 40$	$n = 60$	$n = 80$
p according to Assumption 4	$2^{-0.399}$	$2^{-1.709}$	$2^{-3.857}$
p according to experiments	$2^{-0.400}$	$2^{-1.563}$	$2^{-4.088}$

Table 8: Comparison of Assumption 4 and our experimental results for binary LWE instances with LWE parameters $n, m = 2n, q = 521$ for $n = 40$, $n = 60$, and $n = 80$.

Acknowledgment: We thank Florian Göpfert and John Schanck for helpful discussions and comments.

Funding: This work has been co-funded by the DFG as part of project P1 within the CRC 1119 CROSSING.

References

[1] M. R. Albrecht, R. Fitzpatrick and F. Göpfert, On the efficacy of solving LWE by reduction to unique-SVP, in: *Information Security and Cryptology—ICISC 2013*, Lecture Notes in Comput. Sci. 8565, Springer, Cham (2014), 293–310.

[2] M. R. Albrecht, R. Player and S. Scott, On the concrete hardness of learning with errors, *J. Math. Cryptol.* **9** (2015), no. 3, 169–203.

[3] E. Alkim, L. Ducas, T. Pöppelmann and P. Schwabe, Post-quantum key exchange - A new hope, in: *Proceedings of the 25th USENIX Security Symposium*, USENIX, Berkeley (2016), 327–343.

[4] Y. Aono, Y. Wang, T. Hayashi and T. Takagi, Improved progressive BKZ algorithms and their precise cost estimation by sharp simulator, in: *Advances in Cryptology—EUROCRYPT 2016. Part I*, Lecture Notes in Comput. Sci. 9665, Springer, Berlin (2016), 789–819.

[5] L. Babai, On Lovász' lattice reduction and the nearest lattice point problem, in: *Annual Symposium on Theoretical Aspects of Computer Science—STACS 85*, Lecture Notes in Comput. Sci. 182, Springer, Berlin, (1985) 13–20.

[6] S. Bai and S. D. Galbraith, Lattice decoding attacks on binary LWE, in: *Information Security and Privacy—ACISP 2014*, Lecture Notes in Comput. Sci. 8544, Springer, Berlin (2014), 322–337.

[7] D. J. Bernstein, J. Buchmann and E. Dahmen, *Post-Quantum Cryptography*, Springer, Berlin, 2009.

[8] D. J. Bernstein, C. Chuengsatiansup, T. Lange and C. van Vredendaal, NTRU prime: Reducing attack surface at low cost, in: *Selected Areas in Cryptography—SAC 2017*, Lecture Notes in Comput. Sci. 10719, Springer, Cham (2018), 235–260.

[9] J. Buchmann, F. Göpfert, T. Güneysu, T. Oder and T. Pöppelmann, High-performance and lightweight lattice-based public-key encryption, in: *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, ACM, New York (2016), 2–9.

[10] J. Buchmann, F. Göpfert, R. Player and T. Wunderer, On the hardness of LWE with binary error: Revisiting the hybrid lattice-reduction and meet-in-the-middle attack, in: *Progress in Cryptology—AFRICACRYPT 2016*, Lecture Notes in Comput. Sci. 9646, Springer, Cham (2016), 24–43.

[11] R. Canetti and J. A. Garay, *Advances in Cryptology—CRYPTO 2013. Part I*, Lecture Notes in Comput. Sci. 8042, Springer, Heidelberg, 2013.

[12] Y. Chen, *Réduction de réseau et sécurité concrete du chiffrement completement homomorphe*, PhD thesis, Paris 7, 2013.

[13] Y. Chen and P. Q. Nguyen, BKZ 2.0: Better lattice security estimates, in: *Advances in Cryptology—ASIACRYPT 2011*, Lecture Notes in Comput. Sci. 7073, Springer, Heidelberg (2011), 1–20.

[14] L. Ducas, A. Durmus, T. Lepoint and V. Lyubashevsky, Lattice signatures and bimodal Gaussians, in: *Advances in Cryptology—CRYPTO 2013. Part I*, Lecture Notes in Comput. Sci. 8042, Springer, Heidelberg (2013), 40–56.

[15] M. Fischlin and J.-S. Coron, *Advances in Cryptology—EUROCRYPT 2016. Part I*, Lecture Notes in Comput. Sci. 9665, Springer, Berlin, 2016.

[16] T. Güneysu, V. Lyubashevsky and T. Pöppelmann, Practical lattice-based cryptography: A signature scheme for embedded systems, in: *Cryptographic Hardware and Embedded Systems—CHES 2012*, Lecture Notes in Comput. Sci. 7428, Springer, Berlin (2012), 530–547.

[17] P. S. Hirschhorn, J. Hoffstein, N. Howgrave-Graham and W. Whyte, Choosing NTRUencrypt parameters in light of combined lattice reduction and MITM approaches, in: *Applied Cryptography and Network Security—ACNS 2009*, Lecture Notes in Comput. Sci. 5536, Springer, Berlin (2009), 437–455.

[18] J. Hoffstein, J. Pipher, J. M. Schanck, J. H. Silverman, W. Whyte and Z. Zhang, Choosing parameters for `NTRUEncrypt`, in: *Topics in Cryptology—CT-RSA 2017*, Lecture Notes in Comput. Sci. 10159, Springer, Cham (2017), 3–18.

[19] J. Hoffstein, J. Pipher and J. H. Silverman, NTRU: A ring-based public key cryptosystem, in: *Algorithmic Number Theory* (Portland 1998), Lecture Notes in Comput. Sci. 1423, Springer, Berlin (1998), 267–288.

[20] N. Howgrave-Graham, A hybrid lattice-reduction and meet-in-the-middle attack against NTRU, in: *Advances in Cryptology—CRYPTO 2007*, Lecture Notes in Comput. Sci. 4622, Springer, Berlin (2007), 150–169.
[21] N. Howgrave-Graham, A hybrid lattice-reduction and meet-in-the-middle attack against NTRU, in: *Advances in Cryptology—CRYPTO 2007*, Lecture Notes in Comput. Sci. 4622, Springer, Berlin (2007), 150–169.
[22] N. Howgrave-Graham, J. H. Silverman and W. Whyte, A meet-in-the-middle attack on an NTRU private key, https://www.securityinnovation.com/uploads/Crypto/NTRUTech004v2.pdf.
[23] A. K. Lenstra and E. R. Verheul, Selecting cryptographic key sizes, *J. Cryptology* **14** (2001), no. 4, 255–293.
[24] R. Lindner and C. Peikert, Better key sizes (and attacks) for LWE-based encryption, in: *Topics in Cryptology—CT-RSA 2011*, Lecture Notes in Comput. Sci. 6558, Springer, Heidelberg (2011), 319–339.
[25] V. Lyubashevsky, C. Peikert and O. Regev, On ideal lattices and learning with errors over rings, *J. ACM* **60** (2013), no. 6, Article ID 43.
[26] D. Micciancio and C. Peikert, Hardness of SIS and LWE with small parameters, in: *Advances in Cryptology—CRYPTO 2013. Part I*, Lecture Notes in Comput. Sci. 8042, Springer, Heidelberg (2013), 21–39.
[27] D. Micciancio and M. Walter, Practical, predictable lattice basis reduction, in: *Advances in Cryptology—EUROCRYPT 2016. Part I*, Lecture Notes in Comput. Sci. 9665, Springer, Berlin (2016), 820–849.
[28] F. W. J. Olver, *NIST Handbook Of Mathematical Functions*, Cambridge University Press, Cambridge, 2010.
[29] O. Regev, On lattices, learning with errors, random linear codes, and cryptography, in: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing—STOC'05*, ACM, New York (2005), 84–93.
[30] J. Schanck, *Practical lattice cryptosystems: NTRUEncrypt and NTRUMLS*, Master's thesis, University of Waterloo, 2015.
[31] W. Stein, Sage Mathematics Software Version 7.5.1, The Sage Development Team 2017, http://www.sagemath.org.
[32] C. van Vredendaal, Reduced memory meet-in-the-middle attack against the NTRU private key, *LMS J. Comput. Math.* **19** (2016), no. Suppl. A, 43–57.