Divergence bounds for random fixed-weight vectors obtained by sorting

Daniel J. Bernstein

Department of Computer Science
University of Illinois at Chicago, Chicago, IL 60607–7045, USA
djb@cr.yp.to

Abstract. This note analyzes the distribution of fixed-weight vectors obtained by the following procedure: generate a sequence of uniform random integers with a moderate number of bits; force the bottom bits of the sequence to be a standard fixed-weight vector; sort the sequence; and extract the bottom bits. This note shows, for example, that this procedure with 32-bit integers produces any particular weight-119 6960-bit vector with probability $<1.02/\binom{6960}{119}$.

Keywords: McEliece, NTRU, sorting

1 Introduction

Given a large file of N words, how would you "shuffle" it into a random rearrangement? ... One way is to attach random distinct key values, sort on these keys, then discard the keys.

—Knuth [**5**, Section 5, Exercise 11 and answer], 1973;
slightly rephrased: [**6**, Section 5, Exercise 13 and answer], 1998

One can generate a random n-bit vector of Hamming weight w, i.e., having exactly w nonzero bits, by randomly rearranging the following standard weight-w n-bit vector: $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$ with w copies of 1 and $n - w$ copies of 0. One way to carry out this rearrangement is to generate a random sequence $(r_1, r_2, \ldots, r_w, r_{w+1}, r_{w+2}, \ldots, r_n)$ of b-bit integers; sort the sequence; and apply a corresponding permutation to the standard vector $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$.

If the sequence is a uniform random sequence of n *distinct* b-bit integers—assume that $n \leq 2^b$ so such sequences exist—then the permutation is a uniform random permutation, and the output is a uniform random weight-w n-bit vector. One can produce such a sequence by rejection sampling: first generate a uniform

This work was supported by the European Commission under Contract ICT-645622 PQCRYPTO; by the Netherlands Organisation for Scientific Research (NWO) under grant 639.073.005; and by the U.S. National Science Foundation under grant 1314919. "Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation" (or other funding agencies). Permanent ID of this document: `a04dbdd157ddfbd056db4672629d74d27dfbfacf`. Date: 2017.12.12.

random sequence of b-bit integers, and reject—i.e., try again—if the integers collide (which is easy to see after sorting). This is intolerably slow if b is not much larger than $\lg n$, but the rejection probability becomes tolerable as b grows past $2\lg n$, and becomes much smaller as b continues to grow.

A faster, easier-to-implement approach is to skip the rejection: to generate a uniform random sequence $(r_1, \ldots, r_n)$ of b-bit integers without checking that the integers are distinct. If b is large, say $b = 256$, then rejection is extremely unlikely to be observed in the foreseeable future, so skipping it makes no difference in the outputs. But what if b is smaller: fitting integers into 64-bit words, for example, or 32-bit words? This is even faster, but now collisions have a noticeable chance of occurring. What is the impact of allowing them rather than rejecting them?

For each weight-w n-bit vector s, write p_s for the probability that this procedure outputs s. Then p_s is a multiple of $1/2^{bn}$: there are exactly 2^{bn} sequences $(r_1, \ldots, r_n)$, of which some number produce s. For comparison, if weight-w n-bit vectors were generated uniformly at random then s would appear with probability $q_s = 1/\binom{n}{w}$. If $\binom{n}{w}$ does not divide 2^{bn} then p_s cannot match q_s. Does this non-uniformity create security problems?

Assume, for example, that the resulting weight-w vector s is kept secret, but that $F(s)$ is revealed for some public function F. Assume that the attacker runs some randomized algorithm A to try to find s given $F(s)$. Write α_s for the conditional probability that A outputs s, given that s is in fact the secret vector. Assume that this attack fails against the uniform distribution: that $\sum_s \alpha_s q_s$ is tiny. Could $\sum_s \alpha_s p_s$ be much larger? In other words, could the attack succeed against the p distribution?

One way to put a bound on $\sum_s \alpha_s p_s$, given a bound on $\sum_s \alpha_s q_s$, is to show that the "divergence" of the p distribution from the q distribution is, say, ≤ 2. This means that $p_s \le 2q_s$ for each s. This immediately implies that $\sum_s \alpha_s p_s \le 2\sum_s \alpha_s q_s$: i.e., switching from the q distribution to the p distribution gives the attacker an extra factor ≤ 2 in success probability. For example, if $\sum_s \alpha_s q_s \le 2^{-128}$, then $\sum_s \alpha_s p_s \le 2^{-127}$.

The point of this note is an easy-to-compute, and fairly tight, upper bound on the divergence of p from q. This upper bound shows, for example, that the divergence is <1.02 when $n = 6960$, $w = 119$, and $b = 31$. The attacker thus gains a factor <1.02 in success probability, compared to the uniform distribution; and this distribution is easier to compute than the uniform distribution.

The results here are easy exercises, but it is important to write down the details for verification. There is a long history of security problems lurking inside allegedly easy exercises that turned out to be incorrect.

1.1. Divergence vs. distance. A different way to put a bound on $\sum_s \alpha_s p_s$, given a bound on $\sum_s \alpha_s q_s$, is to show that the "distance" of the p distribution from the q distribution is, say, $\le 2^{-128}$. This means that $\sum_s |p_s - q_s|/2 \le 2^{-128}$; equivalently, the sum of all positive $p_s - q_s$ is $\le 2^{-128}$. This immediately implies that $\sum_s \alpha_s p_s - \sum_s \alpha_s q_s = \sum_s \alpha_s(p_s - q_s) \le 2^{-128}$, since each α_s is between 0 and 1: i.e., switching from the q distribution to the p distribution adds $\le 2^{-128}$ to the attacker's success probability.

Like the divergence bound, this distance bound would show that if $\sum_s \alpha_s q_s \le 2^{-128}$ then $\sum_s \alpha_s p_s \le 2^{-127}$. Unlike the divergence bound, the distance bound would show that if $\sum_s \alpha_s q_s \le 0.5 + 2^{-128}$ then $\sum_s \alpha_s p_s \le 0.5 + 2^{-127}$. This extra feature of the distance bound is useful for "indistinguishability" security definitions that challenge the attacker to guess a secret bit, and that compare the resulting success probability to 0.5.

On the other hand, this extra feature is not necessary in the context of "unfindability" security definitions that challenge the attacker to guess much larger secrets: for example, to

- forge an authenticator,
- forge a signature, or
- find a random plaintext given a public key and the corresponding ciphertext.

Furthermore, there are various ways to build systems believed to meet various indistinguishability notions out of systems believed to meet various unfindability notions. The big advantage of the divergence bound is that it applies to much smaller values of b than the distance bound.

Distance statements appear frequently in the cryptographic literature, while divergence statements are relatively rare. The facts that divergence bounds are

- adequate in the context of unfindability and
- often stronger than distance bounds

have appeared in some papers on lattice-based cryptography in the last few years (see, e.g., [**1**]), but were already used elsewhere in cryptography at least a decade earlier; consider, for example, the statement "$\Pr[A(p) = 1] \le \delta \Pr[A(f) = 1]$" in [**2**, Theorem 2.1].

1.2. Terminology. The word "distance" has many meanings. The object called "distance" in this paper is often called the "statistical distance" (although this phrase also has other meanings) or the "total variation distance".

The word "divergence" also has many meanings. The natural logarithm of the object called "divergence" in this paper, the maximum of p_s/q_s, is the same as the "Rényi divergence of order ∞", although it seems likely that this simple concept predates Rényi. Some recent security bounds have used Rényi divergence of other orders.

A "b-bit integer" in this paper means an element of $\{0, 1, \ldots, 2^b - 1\}$. There is no requirement for the bth bit to be set. Negative integers are not included.

2 Warmup: non-uniform coefficients

This section gives a simple example of a divergence calculation. A uniform random sequence of n b-bit integers is being used to produce a vector of n elements of $\{0, 1, \ldots, q - 1\}$; the question is how close the output vector is to uniform.

Theorem 2.1. *Fix integers $b \geq 0$, $q \geq 1$, $P \geq 1$, and $n \geq 0$. Let $(r_1, \ldots, r_n)$ be a uniform random element of $\{0, 1, \ldots, 2^b - 1\}^n$. Let F be a function from $\{0, 1, \ldots, 2^b - 1\}$ to $\{0, 1, \ldots, q-1\}$. Assume that each $j \in \{0, 1, \ldots, q-1\}$ has $\leq P$ preimages under F. Let $(s_1, \ldots, s_n)$ be an element of $\{0, 1, \ldots, q-1\}^n$. Then $\Pr[(F(r_1), \ldots, F(r_n)) = (s_1, \ldots, s_n)] \leq \delta/q^n$ where $\delta = (Pq/2^b)^n$.*

In other words, the divergence of $(F(r_1), \ldots, F(r_n))$ from uniform is $\leq \delta$. This is most interesting when δ is not much larger than 1, i.e., when P is not much larger than $2^b/q$. Note that P cannot be smaller than $2^b/q$. Theorems 2.2 and 2.3 give two examples of convenient functions F that work with $P = \lceil 2^b/q \rceil$.

Proof. The condition $(F(r_1), \ldots, F(r_n)) = (s_1, \ldots, s_n)$ is satisfied by $\leq P$ choices of r_1, $\leq P$ choices of r_2, and so on through $\leq P$ choices of r_n, for a total of $\leq P^n$ choices of the sequence $(r_1, r_2, \ldots, r_n)$. These choices occur with probability $\leq P^n/2^{bn} = \delta/q^n$. □

Theorem 2.2. *Fix integers $b \geq 0$ and $q \geq 1$. Define $F : \{0, 1, \ldots, 2^b - 1\} \to \{0, 1, \ldots, q-1\}$ by $F(r) = r \bmod q$. Fix $j \in \{0, 1, \ldots, q-1\}$. Then j has $\leq \lceil 2^b/q \rceil$ preimages under F.*

Proof. If $F(r) = j$ then r is in the set $\{j, j+q, j+2q, \ldots, j+(k-1)q\}$, where k is the smallest integer such that $j + kq \geq 2^b$. This is a set of size $k = \lceil (2^b - j)/q \rceil \leq \lceil 2^b/q \rceil$. □

Theorem 2.3. *Fix integers $b \geq 0$ and $q \geq 1$. Define $F : \{0, 1, \ldots, 2^b - 1\} \to \{0, 1, \ldots, q-1\}$ by $F(r) = \lfloor qr/2^b \rfloor$. Fix $j \in \{0, 1, \ldots, q-1\}$. Then j has $\leq \lceil 2^b/q \rceil$ preimages under F.*

Proof. If $F(r) = j$ then $j \leq qr/2^b < j+1$ so $2^b j/q \leq r < 2^b(j+1)/q$; i.e., r is in the set $\{\lceil 2^b j/q \rceil, \ldots, \lceil 2^b(j+1)/q \rceil - 1\}$. This set has size $\lceil 2^b(j+1)/q \rceil - \lceil 2^b j/q \rceil \leq \lceil 2^b/q \rceil$. □

2.4. Application to Streamlined NTRU Prime 4591^{761}. The first step in key generation for the Streamlined NTRU Prime cryptosystem is to generate a uniform random n-coefficient vector with each coefficient in $\{-1, 0, 1\}$. Proposed parameters take $n = 761$. The software actually generates each coefficient as follows: generate a uniform random 30-bit integer, multiply by 3, divide by 2^{30} (rounding down), and subtract 1.

Take $b = 30$ and $q = 3$. Define F as in Theorem 2.3; then each $j \in \{0, 1, 2\}$ has $\leq P$ preimages under F where $P = \lceil 2^b/q \rceil$. The software starts with a uniform random element $(r_1, \ldots, r_n)$ of $\{0, 1, \ldots, 2^b - 1\}^n$. By Theorem 2.1, the divergence of $(F(r_1), \ldots, F(r_n))$ from a uniform random element of $\{0, 1, 2\}^n$ is $\leq \delta$ where $\delta = (Pq/2^b)^n = (\lceil 2^{30}/3 \rceil 3/2^{30})^{761} = (1 + 1/2^{29})^{761} \approx 1.000001417$. Finally, the software outputs $(F(r_1) - 1, \ldots, F(r_n) - 1)$, which has the same divergence from a uniform random element of $\{-1, 0, 1\}^n$.

To summarize, the non-uniformity of the output vector increases findability by a factor <1.000002.

Streamlined NTRU Prime puts an extra requirement on these vectors: key generation starts over if the vector does not satisfy an algebraic invertibility condition. This restriction cannot increase the divergence: the maximum p_s/q_s within a limited set of s is bounded by the maximum p_s/q_s on the full set.

2.5. Application to NTRU LPRime 4591$^{\mathbf{761}}$. Let $(r_1, \ldots, r_n)$ be a uniform random sequence of n b-bit integers, where again $n = 761$ but now $b = 32$. Define F as in Theorem 2.2, with $q = 4591$. Then each $j \in \{0, 1, \ldots, q-1\}$ has $\le \lceil 2^b/q \rceil$ preimages under F. By Theorem 2.1, the divergence of $(F(r_1), \ldots, F(r_n))$ from a uniform random element of $\{0, 1, \ldots, q-1\}^n$ is $\le\delta$ where $\delta = (Pq/2^b)^n = (\lceil 2^{32}/4591 \rceil 4591/2^{32})^{761} \approx 1.00007672$.

In the NTRU LPRime 4591^{761} cryptosystem, a public 32-byte seed is mapped to an element of $\{0, 1, \ldots, q-1\}^n$ as follows: the seed is used as an AES-256-CTR key to produce n 32-bit integers $(r_1, \ldots, r_n)$, which are then reduced modulo q. The non-uniformity of reduction modulo q increases findability by a factor <1.000077. Of course, AES-256-CTR output is not uniform random, and it is possible that this AES-256-CTR structure allows attacks, but this structure is outside the scope of this note.

3 Random fixed-weight binary vectors

This section returns to the sorting procedure stated in Section 1. A uniform random sequence $(r_1, r_2, \ldots, r_n)$ of n b-bit integers is sorted, and a standard weight-w n-bit vector $(1, \ldots, 1, 0, \ldots, 0)$ is permuted accordingly, producing a random weight-w n-bit vector.

To clearly define the output distribution, one must pinpoint exactly which permutation is being applied to the standard vector, or at least pinpoint the result of applying this permutation to the standard weight-w vector. If $r_1, r_2, \ldots, r_n$ are distinct then there is no ambiguity: there is exactly one permutation that puts them into order. However, if there are collisions then the permutation is not uniquely defined, and if there are collisions of the form $r_i = r_j$ where $i \le w$ and $j > w$ then the output is not uniquely defined.

An easy-to-implement definition of a specific output is as follows: sort the $(b+1)$-bit integers $2r_1+1, 2r_2+1, \ldots, 2r_w+1, 2r_{w+1}, 2r_{w+2}, \ldots, 2r_n$, and then reduce modulo 2. This is equivalent to lexicographically sorting the pairs

$$(r_1, 1), (r_2, 1), \ldots, (r_w, 1), (r_{w+1}, 0), (r_{w+2}, 0), \ldots, (r_n, 0),$$

and then extracting the second component of each pair. This is also equivalent to applying the unique permutation defined by "anti-stable" sorting, since the standard string $(1, 1, \ldots, 1, 0, 0, \ldots, 0)$ is in anti-sorted order.

This definition has a slight preference for putting 0 before 1: for example, if $r_w = r_{w+1}$ then $2r_w + 1$ will be sorted after $2r_{w+1}$. Theorem 3.1 quantifies the overall non-uniformity of the output distribution.

Theorem 3.1. *Fix integers $b \geq 0$, $w \geq 0$, and $n \geq w$. Let $(r_1, \dots, r_n)$ be a uniform random element of $\{0, 1, \dots, 2^b - 1\}^n$. Define*

$$(t_1, \dots, t_n) = \mathrm{sort}(2r_1 + 1, \dots, 2r_w + 1, 2r_{w+1}, \dots, 2r_n).$$

Fix a weight-w element $(s_1, \dots, s_n) \in \{0, 1\}^n$. Then $(t_1 \bmod 2, \dots, t_n \bmod 2) = (s_1, \dots, s_n)$ with probability $\leq \delta / \binom{n}{w}$, where

$$\delta = \left(1 + \frac{1}{2^b}\right)\left(1 + \frac{2}{2^b}\right)\left(1 + \frac{3}{2^b}\right) \cdots \left(1 + \frac{n-1}{2^b}\right).$$

Proof. There is a permutation π of $\{1, \dots, n\}$ (not necessarily unique) such that $(t_{\pi(1)}, \dots, t_{\pi(n)}) = (2r_1 + 1, \dots, 2r_w + 1, 2r_{w+1}, \dots, 2r_n)$. Write $u_i = \lfloor t_i/2 \rfloor$; then $(u_{\pi(1)}, \dots, u_{\pi(n)}) = (r_1, \dots, r_n)$. The number of possibilities for $(r_1, \dots, r_n)$ is at most the product of the number of possibilities for $(u_1, \dots, u_n)$ and the number of possibilities for π.

By hypothesis $t_1 \leq \cdots \leq t_n$, so $u_1 \leq \cdots \leq u_n$. This limits $(u_1, \dots, u_n)$ to a set of $\binom{2^b - 1 + n}{n}$ possibilities, namely the set of sorted sequences of n b-bit integers.

Notice that if $(t_1 \bmod 2, \dots, t_n \bmod 2) = (s_1, \dots, s_n)$ then $(s_{\pi(1)}, \dots, s_{\pi(n)}) = (1, \dots, 1, 0, \dots, 0)$. This limits π to a set of $w!(n-w)!$ permutations, namely those that take the w nonzero positions in s to positions $1, \dots, w$ in some order.

The probability that $(t_1 \bmod 2, \dots, t_n \bmod 2) = (s_1, \dots, s_n)$ is therefore at most $w!(n-w)!\binom{2^b-1+n}{n}/2^{bn} = (2^b + n - 1) \cdots (2^b + 1)2^b / 2^{bn}\binom{n}{w} = \delta / \binom{n}{w}$. □

If $(s_1, \dots, s_n)$ is not sorted then there are actually fewer possibilities for $(u_1, \dots, u_n)$ in the proof: for example, if $s_1 > s_2$ then $u_1 < u_2$. Even if $(s_1, \dots, s_n)$ is sorted, the δ bound is not tight in general: collisions sometimes mean that many choices of π are consistent with the same $(r_1, \dots, r_n)$.

3.2. Application to McEliece. The original McEliece code-based cryptosystem [7] asks for a uniform random weight-w n-bit vector. The same is true for the Niederreiter "dual" code-based cryptosystem [8] and many newer code-based cryptosystems.

The case $(n, w) = (6960, 119)$ mentioned in Section 1 is a typical example aiming for a high security level. Theorem 3.1 says that the divergence is bounded by $\delta = (1 + 1/2^b) \cdots (1 + 6959/2^b)$. For example, $\delta \approx 1.011341$ for $b = 31$; this choice of b means sorting 6960 32-bit integers. As a larger example, increasing n from 6960 to 8192, again with $b = 31$, increases δ to approximately 1.015746.

4 Random fixed-weight ternary vectors

This section switches from weight-w elements of $\{0, 1\}^n$ to weight-w elements of $\{-1, 0, 1\}^n$. This expands the number of possible outputs from $\binom{n}{w}$ to $2^w\binom{n}{w}$.

One can generate a weight-w element of $\{-1, 0, 1\}^n$ by first generating a weight-w element of $\{0, 1\}^n$ and then multiplying each entry by ± 1. However, it is more efficient to apply the ± 1 to the standard weight-w vector before sorting:

i.e., to randomly rearrange the vector $(-1+2c_1,\dots,-1+2c_w,0,\dots,0)$ where $(c_1,\dots,c_w) \in \{0,1\}^w$. This in turn requires multiplying each r_i by something larger than 2; multiplying by 4 is the obvious choice.

Theorem 4.1 quantifies the non-uniformity of the output distribution. The divergence bound δ is the same as in Theorem 3.1, and the proof proceeds along the same lines. The detailed are spelled out here for verification.

Theorem 4.1. *Fix integers $b \geq 0$, $w \geq 0$, and $n \geq w$. Let $(r_1,\dots,r_n)$ be a uniform random element of $\{0,1,\dots,2^b-1\}^n$. Let $(c_1,\dots,c_w)$ be a uniform random element of $\{0,1\}^w$. Define*

$$(t_1,\dots,t_n) = \mathrm{sort}(4r_1+2c_1,\dots,4r_w+2c_w,4r_{w+1}+1,\dots,4r_n+1).$$

Fix a weight-w element $(s_1,\dots,s_n) \in \{-1,0,1\}^n$. Then

$$(t_1 \bmod 4,\dots,t_n \bmod 4) = (s_1+1,\dots,s_n+1)$$

with probability $\leq \delta/2^w\binom{n}{w}$, where

$$\delta = \left(1+\frac{1}{2^b}\right)\left(1+\frac{2}{2^b}\right)\left(1+\frac{3}{2^b}\right)\cdots\left(1+\frac{n-1}{2^b}\right).$$

Proof. There is a permutation π of $\{1,\dots,n\}$ (not necessarily unique) such that $(t_{\pi(1)},\dots,t_{\pi(n)}) = (4r_1+2c_1,\dots,4r_w+2c_w,4r_{w+1}+1,\dots,4r_n+1)$. Write $u_i = \lfloor t_i/4 \rfloor$. Then $(u_{\pi(1)},\dots,u_{\pi(n)}) = (r_1,\dots,r_n)$. Also, if $(t_1 \bmod 4,\dots,t_n \bmod 4) = (s_1+1,\dots,s_n+1)$ then $(2c_1,\dots,2c_w) = (s_{\pi(1)}+1,\dots,s_{\pi(w)}+1)$.

Consequently $(r_1,\dots,r_n,c_1,\dots,c_w)$ is determined by $(u_1,\dots,u_n)$ and π. The number of possibilities for $(r_1,\dots,r_n,c_1,\dots,c_w)$ is at most the product of the number of possibilities for $(u_1,\dots,u_n)$ and the number of possibilities for π.

By hypothesis $t_1 \leq \cdots \leq t_n$, so $u_1 \leq \cdots \leq u_n$. This limits $(u_1,\dots,u_n)$ to a set of $\binom{2^b-1+n}{n}$ possibilities, namely the set of sorted sequences of n b-bit integers.

As for π: Notice that if $(t_1 \bmod 4,\dots,t_n \bmod 4) = (s_1+1,\dots,s_n+1)$ then $((s_{\pi(1)}+1) \bmod 2,\dots,(s_{\pi(n)}+1) \bmod 2) = (0,\dots,0,1,\dots,1)$. This limits π to a set of $w!(n-w)!$ permutations, namely those that take the w nonzero positions in s to positions $1,\dots,w$ in some order.

The probability that $(t_1 \bmod 4,\dots,t_n \bmod 4) = (s_1+1,\dots,s_n+1)$ is at most $w!(n-w)!\binom{2^b-1+n}{n}/2^{bn+w} = (2^b+n-1)\cdots(2^b+1)2^b/2^{bn+w}\binom{n}{w} = \delta/2^w\binom{n}{w}$. $\square$

4.2. Application to Streamlined NTRU Prime 4591^{761}. The Streamlined NTRU Prime cryptosystem generates a uniform random weight-w element of $\{-1,0,1\}^n$ during key generation, and another during encapsulation. Proposed parameters take $n = 761$ and $w = 286$.

What the software actually does is sort n $(b+2)$-bit integers as described above, where $b = 30$. By Theorem 4.1, the divergence of the output from uniform is $\leq\delta$ where $\delta = (1+1/2^{30})\cdots(1+760/2^{30}) \approx 1.000269$. This quantifies and justifies the statement "the information leak is negligible" in [**3**, Appendix T].

The central "OW-CPA" security question is whether an attacker can find a random plaintext, given a random public key and the corresponding ciphertext. Overall this problem involves three independent random vectors in Streamlined NTRU Prime 4591^{761}:

- the plaintext has a random weight-w vector,
- the secret key has another random weight-w vector, and
- the secret key has another random vector generated as in Section 2.4.

These random vectors have divergence <1.00027, <1.00027, and <1.000002 from uniform respectively, overall increasing the attacker's OW-CPA success probability by a factor <1.001.

4.3. Application to NTRU LPRime 4591^{761}. Similar comments apply to the NTRU LPRime cryptosystem. The sorting procedure is built into the specification of NTRU LPRime 4591^{761}, rather than merely being a choice made in the software; the structure of NTRU LPRime requires the sender and receiver to agree on the details of how weight-w vectors are generated. The choice of w for NTRU LPRime 4591^{761} is different, $w = 250$, but this does not affect δ in Theorem 4.1.

References

[1] Shi Bai, Adeline Langlois, Tancrède Lepoint, Damien Stehlé, Ron Steinfeld, *Improved security proofs in lattice-based cryptography: Using the Rényi divergence rather than the statistical distance*, in Asiacrypt 2015 [4] (2015), 3–24. URL: https://eprint.iacr.org/2015/483. Citations in this document: §1.1.

[2] Daniel J. Bernstein, *Stronger security bounds for permutations* (2005). URL: https://cr.yp.to/papers.html#permutations. Citations in this document: §1.1.

[3] Daniel J. Bernstein, Chitchanok Chuengsatiansup, Tanja Lange, Christine van Vredendaal, *NTRU Prime: reducing attack surface at low cost*, SAC 2017, to appear (2017). URL: https://ntruprime.cr.yp.to/papers.html. Citations in this document: §4.2.

[4] Tetsu Iwata, Jung Hee Cheon (editors), *Advances in cryptology—ASIACRYPT 2015—21st international conference on the theory and application of cryptology and information security, Auckland, New Zealand, November 29–December 3, 2015, proceedings, part I*, Lecture Notes in Computer Science, 9452, Springer, 2015. ISBN 978-3-662-48796-9. See [1].

[5] Donald E. Knuth, *The art of computer programming, volume 3: sorting and searching*, 1st edition, Addison-Wesley, 1973; see also newer version [6]. ISBN 0-201-03803-X. MR 56:4281. Citations in this document: §1.

[6] Donald E. Knuth, *The art of computer programming, volume 3: sorting and searching*, 2nd edition, Addison-Wesley, 1998; see also older version [5]. ISBN 0-201-89685-0. Citations in this document: §1.

[7] Robert J. McEliece, *A public-key cryptosystem based on algebraic coding theory*, JPL DSN Progress Report (1978), 114–116. URL: https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF. Citations in this document: §3.2.

[8] Harald Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory **15** (1986), 159–166. Citations in this document: §3.2.