Secure Parameters for SWIFFT
- Extended Absract -

Johannes Buchmann and Richard Lindner

Technische Universität Darmstadt, Department of Computer Science
Hochschulstraße 10, 64289 Darmstadt, Germany
buchmann,rlindner@cdc.informatik.tu-darmstadt.de

Abstract. The SWIFFT compression functions, proposed by Lyubashevsky *et al.* at FSE 2008, are very efficient instantiations of generalized compact knapsacks for a specific set of parameters. They have the property that, *asymptotically*, finding collisions for a randomly chosen compression function implies being able to solve computationally hard ideal lattice problems in the *worst-case*.
We present three results. First, we propose a parameter generation algorithm for SWIFFT where the main parameter n can be any integer in the image of Euler's totient function, and not necessarily a power of 2 as before. Second, we give some efficiency improvements that apply to SWIFFT in general. Third, we give experimental evidence that finding *pseudo-collisions* for SWIFFT is as hard as breaking a 68-bit symmetric cipher according to the well-known heuristic by Lenstra and Verheul. We also recommend conservative parameters corresponding to a 127-bit symmetric cipher.

Keywords: post-quantum cryptography, hash functions, lattices.

1 Introduction

In 2007, the National Institute for Standards and Technology (NIST) announced a comptetion to develop a new cryptographic hash algorithm. The winner of this competition will be standardized as Secure Hash Algorithm 3 (SHA-3) and will replace the current standards SHA-1 and SHA-2. One candidate that has been submitted to this competition is SWIFFTX [2] which is constructed using the SWIFFT compression function family [15]. This hash function is most interesting because its collision resistance, the most important security property, has been proven to rely on the worst case hardness of finding short vectors in a large class of lattices. This kind of security reduction is unique among all the SHA-3 candidates. In order for the submission to be successful, the inventors have presented parameters for SWIFFT that guarantee 100-bit security and they have argued that SWIFFTX ist sufficiently efficient.

In this paper we show that the proposed parameters for SWIFFT are inadequate for 100-bit security. We do this by presenting evidence based on the method of Lenstra and Verheul [12] and our extensive experiments that lattice attacks require significantly less than 2^{100} operations. To be more precise, we argue that the effort to find so-called pseudo-collisions[1] is comparable to breaking a 68-bit symmetric cipher. In lattice attacks against the SWIFFT collision resistance, we consider finding such pseudo-collisions the most time-consuming step.

We also present a new parameter set that we claim yields 100-bit security. However, with these new parameters, SWIFFTX does not satisfy the SHA-3 efficiency requirements any more. We partially solve this problem. We propose an efficient parameter generation algorithm that can be used for a much wider class of SWIFFT compression functions, compared to in the method the original SWIFFT proposal. There, only the case where the security parameter n is a power of 2 was considered. We generalize this to all $n = \varphi(k)$ where φ is the Euler totient function. We are convinced that using those compression functions a new version of SWIFFTX can be designed that satisfies the SHA-3 efficiency requirements.

The paper is organized as follows. Section 2 deals with basics about lattices. Section 3 explains the SWIFFT compression function. Section 4 presents the parameter generation algorithm and Section 5 discusses the SWIFFT security.

2 Preliminaries

A lattice Λ is a discrete, additive subgroup of $\mathbb{R}^n$. It can always be described as $\Lambda = \{\sum_{i=1}^{d} x_i\,\mathbf{b}_i \mid x_i \in \mathbb{Z}\}$, where $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \mathbb{R}^n$ are linearly independent. The matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_d]$ is a *basis* of Λ and we write $\Lambda = \Lambda(\mathbf{B})$. The number of vectors in the basis is the dimension of the lattice.

For each basis $\mathbf{B}$ there is a decomposition $\mathbf{B} = \mathbf{B}^*\,\mu$, where $\mathbf{B}^*$ is orthogonal and μ is upper triangular. The decomposition is uniquely defined by these rules

$$\mu_{j,i} = \langle \mathbf{b}_i, \mathbf{b}_j^* \rangle / \|\mathbf{b}_j^*\|^2, \quad \mathbf{b}_i = \mu_{1,i}\mathbf{b}_1^* + \cdots + \mu_{i-1,i}\mathbf{b}_{i-1}^* + \mathbf{b}_i^*, \quad 1 \le j \le i \le n.$$

It can be computed efficiently with the Gram-Schmidt process and $\mathbf{B}^*$ is the *Gram-Schmidt Orthogonalization* (GSO) of $\mathbf{B}$.

Conforming with notations in previous works, we will write vectors and matrices in boldface. Special tuples of vectors will be denoted with a hat (see for example Section 3). The residue class ring $\mathbb{Z}/\langle p \rangle$ is $\mathbb{Z}_p$.

3 SWIFFT compression functions

The SWIFFT compression function family was proposed by Lyubashevsky *et al.* at FSE 2008 [15]. They showed that for one set of parameters, its efficiency is comparable to SHA-2, while its collision resistance is asymptotically based on *worst-case* computational problems in ideal lattices.

[1] For the definition see Section 3.2.

Specifically, for a set of integer parameters (n, m, p), in their case $(64, 16, 257)$, they use the polynomial $f(x) = x^n + 1$, the ring $R_{p,n} = \mathbb{Z}_p[x]/\langle f\rangle$, and the subset $D_n = \{0,1\}[x]/\langle f\rangle$ to define the family

$$\mathcal{H}_{n,m,p} = \left\{ h_{\widehat{\mathbf{a}}} \colon D_n^m \ni \widehat{\mathbf{x}} \longmapsto \sum_{i=1}^{m} \mathbf{a}_i \mathbf{x}_i \pmod p \;\middle|\; (\mathbf{a}_1, \ldots, \mathbf{a}_m) = \widehat{\mathbf{a}} \in R_{p,n}^m \right\}.$$

These functions can be computed efficiently. Let $\omega_0, \ldots, \omega_{n-1}$ be the roots of f in $\mathbb{Z}_p$ in any order, and $\mathbf{V}$ be the Vandermonde matrix generated by them

$$\mathbf{V} = \begin{pmatrix} 1 & \omega_0 & \ldots \omega_0^{n-1} \\ \vdots & \vdots & \vdots \\ 1 & \omega_{n-1} & \ldots \omega_{n-1}^{n-1} \end{pmatrix}.$$

Applying the *Fast Fourier Transform* over $\mathbb{Z}_p$ to SWIFFT we get

$$\mathbf{z} \equiv \sum_{i=0}^{m-1} \mathbf{a}_i \mathbf{x}_i \bmod f \equiv \mathbf{V}^{-1} \left(\sum_{i=0}^{m-1} \mathbf{V}\mathbf{a}_i \odot \mathbf{V}\mathbf{x}_i \right) \pmod p, \tag{1}$$

where $\odot$ is the pointwise multiplication in $\mathbb{Z}_p^n$. Since $\mathbf{V}$ is invertible, we may use $\mathbf{z}' = \mathbf{V}\mathbf{z}$ as hash, instead of $\mathbf{z}$. Since the compression function key $\widehat{\mathbf{a}}$ is fix, we may precompute $\mathbf{a}_i' = \mathbf{V}\mathbf{a}_i$ for all i. So evaluating the compression function amounts to computing all n components of $\mathbf{z}'$ with

$$z_j' = \sum_{i=0}^{m-1} a_{i,j}' \, x_{i,j}' \bmod p, \qquad\qquad x_{i,j}' = \sum_{l=0}^{n-1} \omega_j^l \, x_{i,l} \bmod p.$$

Due to the form of f we can set $\omega_j \leftarrow \omega^{2j+1}$ for any element ω of order $2n$ in $\mathbb{Z}_p$. We insert the parameters and split up the indices $j = j_0 + 8j_1$ and $l = l_0 + 8l_1$.

$$\begin{aligned} x_{i,j_0+8j_1}' &= \sum_{l_0=0}^{7} \sum_{l_1=0}^{7} \omega^{(l_0+8l_1)(2(j_0+8j_1)+1)} \, x_{i,l_0+8l_1} \bmod p \\ &= \sum_{l_0=0}^{7} \omega^{16 l_0 j_1} \cdot \underbrace{\omega^{l_0(2j_0+1)}}_{m_{l_0,j_0}} \cdot \underbrace{\sum_{l_1=0}^{7} \omega^{8l_1(2j_0+1)} \, x_{i,l_0+8l_1}}_{t_{l_0,j_0}} \bmod p \end{aligned} \tag{2}$$

The quantities t_{l_0,j_0} for all 2^8 possible x_{i,l_0+8l_1} and m_{l_0,j_0} can be precomputed. The SWIFFT authors recommend using $\omega = 42$, because then $\omega^{16} \equiv 4 \bmod p$, so some multiplications in the last expression can be realized with bit-shifts. A single $\mathbf{x}_i'$, i.e. the last expression for all j, can then be evaluated with a total of 64 multiplications, $8 \cdot 24$ additions/substractions using an FFT network. The total number of operations (ignoring index-calculations and modular reduction) for the standard SWIFFT parameters is

$$\underbrace{16 \cdot 64}_{\text{computing } x'_{i,j}} + \underbrace{16 \cdot 64}_{\text{all } a'_{i,j} \cdot x'_{i,j}} = 2048 \text{ multiplications}$$

$$\underbrace{16 \cdot 8 \cdot 24}_{\text{computing } x'_{i,j}} + \underbrace{16 \cdot 64 - 1}_{\text{summing } a'_{i,j} \cdot x'_{i,j}} = 4095 \text{ additions/substractions}$$

Lyubashevsky and Micciancio showed in [13] that *asymptotically* these compression functions are collision resistant, as long as standard lattice problems in lattices corresponding to ideals of $\mathbb{Z}[x]/\langle f \rangle$ are hard in the *worst-case*. The arguments given later by Peikert and Rosen in [17] can also be adapted to prove collision resistance of SWIFFT with a *tighter* connection to the same worst-case problem.

3.1 More parameters

Let $k > 0$ be some integer, p be prime and $n = \varphi(k)$, where φ is Euler's totient function. Furthermore, let f be the kth cyclotomic polynomial, which is monic, irreducible over the integers, and has degree equal to n. Using the same structures as above, i.e. the ring $R_{p,n} = \mathbb{Z}_p[x]/\langle f \rangle$, and subset $D_n = \{0,1\}[x]/\langle f \rangle$ with this new f, we can construct the same compression function family as above and the asymptotic security argument given in [17,13] still holds. In order to apply FFT as before, we need to ensure that elements of order k exist in $\mathbb{Z}_p$. This is guaranteed whenever $k \mid (p-1)$.

Optimizations similar to the ones available for SWIFFT in this more general setting are still an area of investigation. We give some preliminary general results here and some specific results for the parameters we recommend in Section 4.1.

We found that using additions in a logarithmic table instead of multiplications in $\mathbb{Z}_p$ is comparable in speed to the normal multiplication and special bit shifting reduction modulo 257 used in SWIFFT.

Another optimization works for all SWIFFT parameters, including the ones in use now. Consider the following Lemma.

Lemma 1. *Let $h_{\widehat{\mathbf{a}}} \in \mathcal{H}_{n,m,p}$, assuming there exists an i, s.t. $\mathbf{a}_i$ is invertible in R then for $\widehat{\mathbf{a}}' = \mathbf{a}_i^{-1}\widehat{\mathbf{a}}$, any collision of $h_{\widehat{\mathbf{a}}}$ is one for $h_{\widehat{\mathbf{a}}'}$ and vice versa.*

Proof. Let $\widehat{\mathbf{x}}, \widehat{\mathbf{y}}$ be an $h_{\widehat{\mathbf{a}}}$ collision, then $h_{\widehat{\mathbf{a}}'}(\widehat{\mathbf{x}}) = \mathbf{a}_i^{-1} h_{\widehat{\mathbf{a}}}(\widehat{\mathbf{x}}) = \mathbf{a}_i^{-1} h_{\widehat{\mathbf{a}}}(\widehat{\mathbf{y}}) = h_{\widehat{\mathbf{a}}'}(\widehat{\mathbf{y}})$. □

The probability of the event we assume here, is $1-(1-|R^*|/|R|)^m$ given the way we choose $\widehat{\mathbf{a}}$, where R^* is the group of invertible elements in R. This probability well exceeds 99% for the SWIFFT parameters in use. All entries in $\widehat{\mathbf{a}}'$ can be precomputed in practice and having one of them equal $\mathbf{1}$ saves some multiplications during evaluation depending on the implementation. Recall Equation (1), if we would computed $\mathbf{z}$ instead of $\mathbf{z}'$ the speed-up is $1/m$. For $m = 16$ this is $\approx 6\%$. This may be further increased with the sliding window method used for

NTRU [3]. However, at the moment it is more efficient to compute $\mathbf{z}'$. In this case we save n multiplications, which is about 1% of all operations for standard parameters.

We believe that optimizations are easiest to find in the cases where k is prime or a power of two. Focusing on these two special cases, we can already see much more variety in the choice of parameters. See Table 1 for comparison of parameters where n is between 64 and 128.

3.2 SWIFFT lattice

Let $\widehat{\mathbf{a}} \in R_{p,n}$. Consider the function $h_{\widehat{\mathbf{a}}} \in \mathcal{H}_{n,m,p}$ and extended the domain to $R_n = \mathbb{Z}[x]/\langle f \rangle$. The coefficient vectors of the *periods* of this function form the set

$$\Lambda_p^{\perp}(\widehat{\mathbf{a}}) = \left\{ (x_1, \ldots, x_{nm}) \in \mathbb{Z}^{nm} \;\middle|\; h_{\widehat{\mathbf{a}}} \left(\sum_{i=0}^{n-1} x_{i+1} x^i, \ldots, \sum_{i=0}^{n-1} x_{m(i+1)} x^i \right) = \mathbf{0} \right\}.$$

This is a lattice of dimension nm, since the extended $h_{\widehat{\mathbf{a}}}$ is R_n-linear. A basis for this lattice can be found efficiently using a method described by Buchmann *et al.* [4]. Collisions in the original (unextended) function $h_{\widehat{\mathbf{a}}}$ correspond exactly to vectors in this lattice with ℓ_∞-norm bounded by 1. Therefore we refer to these lattices as SWIFFT lattices.

A *pseudo-collision* is a vector in this lattice with Euclidean norm less than $\sqrt{nm}$, i.e. all vectors in the smallest ball containing all collisions. So every collision is a pseudo-collision, but not vice versa.

4 Parameter generation

We now describe an algorithm for generating parameter sets (n, m, p) for the SWIFFT compression function families in Section 3. For the polynomial f we will use the kth cyclotomic polynomial, such that $n = \varphi(k)$. If multiple polynomials are possible, we choose the one, where the resulting bitlength of the output is shorter, i.e. the one with smaller p. For example, if the first parameter n is a predecessor of a prime, we will use the polynomial $f(x) = x^n + x^{n-1} + \cdots + 1$, and if n is a power of two, we will use the polynomial $f(x) = x^n + 1$.

```
Input: Integer $n$, s.t. $n = \varphi(k), k > 0$
Output: Parameters $(n, m, p)$

$l \leftarrow 1$
$p \leftarrow k + 1$
while not isPrime(p) do
    $l \leftarrow l + 1$
    $p \leftarrow l \cdot k + 1$
end
$m \leftarrow \lceil 1.99 \cdot \log_2(p) \rceil$
```

Algorithm 1: Parameter generation for $n = \varphi(k), k > 0$.

For each set of parameters, we may additionally compute the output bitlength $out = n(\lfloor \log_2(p) \rfloor + 1)$, the compression rate $cr = m/\log_2(p)$, the Hermite factor δ required for finding pseudo-collisions, and the minimal dimension d where we can expect to find pseudo-collisions. These values are listed in Table 1.

k	n	m	p	out	cr	δ	d
128	64	16	257	513	1.999	1.0084	206
67	66	17	269	529	2.106	1.0084	211
71	70	19	569	631	2.076	1.0073	248
73	72	17	293	577	2.074	1.0077	231
79	78	17	317	625	2.046	1.0072	251
83	82	15	167	575	2.032	1.0075	237
89	88	15	179	617	2.004	1.0071	255
97	96	18	389	769	2.092	1.0061	308
101	100	19	607	901	2.055	1.0056	340
103	102	19	619	919	2.049	1.0055	348
107	106	19	643	955	2.037	1.0053	361
109	108	21	1091	1081	2.081	1.0049	392
113	112	16	227	785	2.044	1.0058	325
127	126	18	509	1009	2.002	1.0047	408
256	128	16	257	1025	1.999	1.0051	373

Table 1. Parameters for $64 \leq n \leq 128$, k prime or a power of two.

The two latter values δ and d are computed in the following fashion. Consider the function $len(d) = p^{n/d}\delta^d$. According to an analysis by Gama and Nguyen [7][2] this is the Euclidean size of the smallest vector we are likely to find when reducing a sublattice with dimension d of any SWIFFT lattice $\Lambda_p^\perp(\widehat{\mathbf{a}})$. Micciancio and Regev observed in [16] that this function takes its minimal value

$$len(d_{min}) = \delta^{2\sqrt{n\log(p)/\log(\delta)}} \qquad \text{for} \qquad d_{min} = \sqrt{n\log(p)/\log(\delta)}.$$

A pseudo-collision is a vector in $\Lambda_p^\perp(\widehat{\mathbf{a}})$ with Euclidean norm $\sqrt{nm}$. In order to find such a vector, we need a δ, s.t. $len(d_{min}) = \sqrt{nm}$. We say this is the Hermite factor required for finding pseudo-collisions, and the corresponding d_{min} is the minimal dimension, where we can expect to find a pseudo-collision. Note that these minimal dimensions, which we will work in are about 5 *times* smaller than the corresponding dimensions of the SWIFFT lattices. To give an intuition, Gama and Nguyen state that the best lattice reduction algorithms known today can achieve a Hermite factor of roughly $\delta = 1.01$ in high dimension within acceptable time.

[2] Their experiments were performed on random lattices following a different distribution, but experimentally their results apply here as well.

4.1 Recommended parameters

We will give arguments in Section 5.2 that parameters with $d \geq 260$ correspond to SWIFFT instances, where finding pseudo-collisions is at least as hard as breaking a 100-bit symmetric cipher. The smallest such parameters in Table 1 are $(n, m, p) = (96, 18, 389)$. Finding pseudo-collisions for these parameters is as hard as breaking a 127-bit symmetric cipher. Concerning all other known attacks, these parameters are more secure than $(64, 16, 257)$.

Note that some of the efficiency improvements we outlined in Section 4 for the original SWIFFT function work here as well. Recall Equation 2, since $k = 97$ is prime we can set $\omega_j \leftarrow \omega^{j+1}$ for any element ω of order k in $\mathbb{Z}_p$. We recommend to split up the indices $l = l_0 + 8l_1$, where $0 \leq l_0 \leq 7, 0 \leq l_1 \leq 11$, j similar and use $\omega = 275$, and since multiplying with $\omega^8 = 16$ can then be realized with bit-shifts. Corresponding to Equation 2 we get

$$x'_{i,j_0+8j_1} = \sum_{l_0=0}^{7} \omega^{8l_0j_1} \cdot \underbrace{\omega^{l_0(j_0+1)}}_{m_{l_0,j_0}} \cdot \underbrace{\sum_{l_1=0}^{11} \omega^{l_1(8j_0+64j_1+8)}\, x_{i,l_0+8l_1}}_{t_{l_0,j_0,j_1}} \operatorname{mod} p.$$

Note that the precomputed t part depends on j_1 now, and needs to be available for 2^{12} possible $x_{i,l}$. So this part will need $12 \cdot 2^4 = 192$ times the space it did before. Although we have no implementation yet, we are confident that efficiency of evaluation here is comparative to the original SWIFFT function.

5 Security Analysis

The collision resistance of SWIFFT has the desirable property of being reducible to a worst-case computational problem. In particular, this means an algorithm which breaks random instances of SWIFFT compression functions with main parameter n can also be used to find short nonzero vectors in *all* ideals of the ring $\mathbb{Z}[x]/\langle x^n + 1\rangle$. Finding such vectors is assumed to be infeasible for large n. However, for the current parameter, $n = 64$, exhaustive search algorithms find these short vectors in less than one hour. In the lattice challenge [4] open for all enthusiasts similar problems have been solved[3] up to $n = 108$. Gama and Nguyen even state that finding the *shortest* vector in n-dimensional lattices for $n \leq 70$ should be considered easy [7]. So the resulting lower bound on the attacker's runtime is insignificant.

We will analyze the *practical* security of SWIFFT. With all known attacks, breaking SWIFFT directly is harder than breaking the underlying worst-case problem. As we have seen in Section 3.2, collisions in the SWIFFT compression functions naturally correspond to vectors with ℓ_∞-norm bounded by 1 in certain lattices. These may be recovered with lattice basis reduction algorithms. Since these algorithms are highly optimized to find small vectors in the Euclidean norm, it is reasonable to analyze the computational problem of finding

[3] See `http://www.latticechallenge.org`

pseudo-collisions instead of collisions. These are vectors in the *smallest ball* which contains all vectors corresponding to collisions, so an algorithm which minimizes the Euclidean norm cannot distinguish between the two. In this section, we give experimental evidence that according to a well-known heuristic by Lenstra and Verheul [12], finding pseudo-collisions is comparable to breaking a 68-bit symmetric cipher. In comparison, all other attacks analyzed by the SWIFFT authors take 2^{106} operations and almost as much space.

In their original proposal of SWIFFT, Lyubashevsky *et al.* provide a first analysis of all standard attacks. When it comes to attacks using lattice reduction they state however, they state that the dimension 1024 of SWIFFT lattices is too big for current algorithms. We will start by showing in Section 5.1 that reducing sublattices of dimension 251, which corresponds roughly to $m = 4$, is sufficient to find pseudo-collisions and dimension 513 ($m = 8$) is sufficient for collisions and beyond this point as Micciancio and Regev observe in [16] “the problem [SVP] cannot become harder by increasing m”. This means if we find a pseudo-collision in dimension 251, we can pad it with zeroes to obtain a pseudo-collision for SWIFFT. In practice, even dimension $d = 205$ is sufficient to find pseudo-collisions (cf. Table 1). In particular this means SWIFFTX, where internally SWIFFT is used with $m = 32$ is not more secure.

5.1 Existence of (pseudo-)collisions in *d*-dimensional sublattices

The method we have given in Section 4 for choosing the dimension of the sublattice we attack with lattice-basis reduction algorithms is a heuristic, because it is based on extensive experiments by Gama and Nguyen. We will now give a related result independent of experiments but dependent on the construction of SWIFFT lattices and other lattices of the form $\{\mathbf{v} \in \mathbb{Z}^d : \mathbf{A}\mathbf{v} \equiv \mathbf{0} \pmod{p}\}$, where $\mathbf{A}$ is some integral matrix. These lattices are widely used in practice for constructing provably secure cryptosystems (see e.g. [8,14,18]) and they originate from Ajtai’s work [1].

Let $h_{\widehat{\mathbf{a}}}$ be a random SWIFFT compression function with parameters (n, m, p). The range of this function has size $|R| = p^n$. We change the domain of h to all vectors in a d-dimensional subspace of $\mathbb{Z}^{nm}$ that have Euclidean norm less than $r = \sqrt{nm}/2$. The size of this space can be very well approximated by the volume of a d-dimensional ball with radius r, i.e. $|D| = r^d \pi^{d/2} / \Gamma(d/2 + 1)$.

Now any collision in the modified $h_{\widehat{\mathbf{a}}}$ corresponds to a pseudo-collision of the corresponding SWIFFT function by the triangle inequality. These collisions exist for certain by the pigeonhole principle for all $d \geq 251$. So the dimension $d = 205$ suggested by the heuristic looks too optimistic, but remember that this argument only gives an upper bound on the required d and doesn’t take into account the randomness in the choice of $\widehat{\mathbf{a}}$.

The situation for proper collisions is similar. Here, we shrink the input to all vectors in a d-dimensional subspace that have coefficients in $\{0, 1\}$. The size of this input space is $|D| = 2^d$. Again, collisions exist by the pigeonhole principle for all $d \geq 513$. This suggests that proper collisions are harder to find.

5.2 Experiments

For our experiments we chose the sublattice dimension where lattice basis reduction algorithms like LLL/BKZ behave optimal in practice (see Section 4). We then proceeded to compare the following lattice basis reduction algorithms to see which performs best in practice on the lattices in our experiment. BKZ as implemented in version 5.4.2 of the "Number Theory Library"(NTL) by Shoup [20], Primal-Dual (PD) as implemented by Filipović and Koy, and finally RSR as implemented by Ludwig. Both latter algorithms are available on request from the authors[4]. It became apparent that Primal-Dual runs much slower than both competitors, so for the main experiment we omitted it.

For our experiments, we fixed $n = 64$, $m = 16$ to their standard values and chose the third parameter p variable. This results in a steady decrease in the Hermite factor and increase in the dimension required to find pseudo-collisions (see Table 2). We found that for smaller values of p, corresponding to smaller

n	m	p	δ	d
64	16	29	1.0140	125
64	16	33	1.0135	130
64	16	37	1.0131	134
64	16	41	1.0127	138
64	16	45	1.0124	141
64	16	49	1.0121	144
64	16	53	1.0119	147
64	16	57	1.0117	150
64	16	61	1.0115	152

Table 2. Parameters used for our experiments.

values of d, pseudo-collisions were found too fast to make sensible measurements.

For each of these 9 parameter sets, we created 10 random SWIFFT lattices using the PRNG which is part of NTL. We then proceeded to break all instances with the NTL floating-point variant of BKZ (bkzfp), by increasing the BKZ parameter β until a pseudo-collision was found and recording the total time taken in each case. We also broke all instances with a floating-point variant of Schnorr's random sampling reduction (RSR) algorithm [19] (rsrfp) implemented by Ludwig [5] using the parameters $\delta = 0.9, u = 22$ and again increasing β until a pseudo-collision was found.

In all cases, we computed the average runtime of both algorithms and plotted the base two log of this value relative to the dimension d. We also plotted a conservative extrapolation (assuming linear growth in logscale) for the average

[4] PD, Bartol Filipović, `bartol.filipovic@sit.fraunhofer.de`
PSR, Christoph Ludwig, `cludwig@cdc.informatik.tu-darmstadt.de`

runtime of both algorithms (see Figure 1). The same growth assumption has often been made when analyzing NTRU lattices [9]. These are similar to SWIFFT lattices as we will see in Section 5.3.

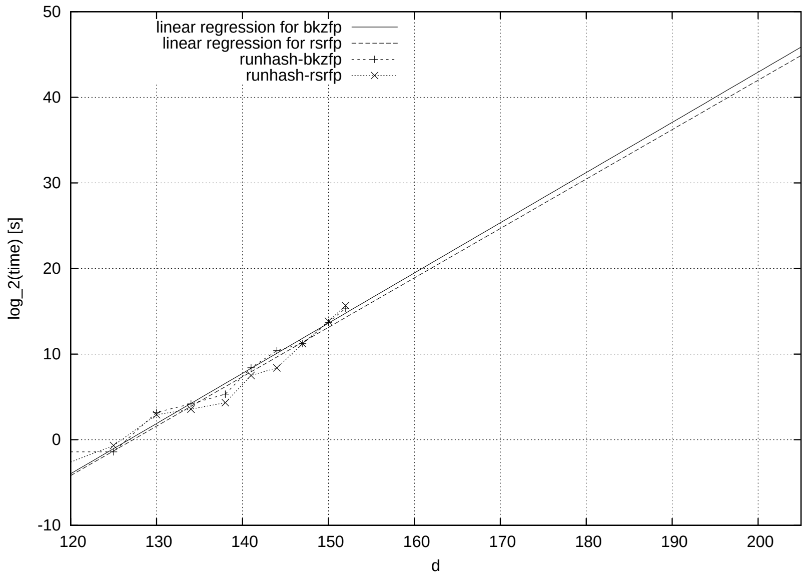

Fig. 1. Average runtimes of our experiments.

All our experiments where run on a single 2.3 GHz AMD Opteron processor. According to the predictions of Lenstra and Verheul [12] the computational hardness of a problem solved after t seconds on such a machine is comparable to breaking a k-bit symmetric cipher, where

$$k = \log_2(t) + \log_2(2300) - \log_2(60 \cdot 60 \cdot 24 \cdot 365.25) - \log_2(5 \cdot 10^5) + 56.$$

Using the data in Figure 1, we can compute the security level k corresponding to the average runtime of each algorithm relative to the dimension d for each parameter set.

The rightmost side of Figure 1 corresponds to $p = 257$, i.e. a *real* SWIFFT lattice. The extrapolated symmetric bit security for finding pseudo-collisions on these lattices is $k = 68.202$. Any parameter set, where $d \geq 260$ would correspond to a cipher with symmetric bit-security at least 100 according to our extrapolation. Parameters realizing this paradigm are given in Section 4.1.

5.3 Hybrid lattice reduction

There is a *strong* similarity between NTRU lattices and SWIFFT lattices which we will make explicit. According to the most recent NTRU flavor [9], an NTRU trapdoor one-way function family is described by the parameters

$$(q^{NTRU}, p^{NTRU}, N^{NTRU}, d_f^{NTRU}, d_g^{NTRU}, d_r^{NTRU}).$$

These relate to SWIFFT families in the following way. Choose $n = N^{NTRU}, m = 2, p = q^{NTRU}$. Use the polynomial $f(x) = x^n - 1$ for the ring $R_{p,n}$. Let $\mathcal{T}_d$ be the set of trinary polynomials of degree $n-1$ with $d+1$ entries equal to 1 and d entries equal to -1. In the NTRU setting, we choose our hash-keys (a_1, a_2) not uniformly from $R_{p,n}^2$ but rather from $(1 + p^{NTRU}\mathcal{T}_{d_f^{NTRU}}) \times \mathcal{T}_{d_g^{NTRU}}$ which are the NTRU secret key spaces.

The strong limitation on the choice of keys allows the trapdoor to work. The use of a reducible polynomial does not guarantee collision resistance anymore [13], but one-wayness is sufficient for NTRUs security. In summation, the step from NTRU to SWIFFT is exchanging a huge $N^{NTRU} = 401, q^{NTRU} = 2048$ with $n = 64, p = 257$ but in turn increase m from 2 to 16. This seems risky because as we mentioned at the beginning of this section, the problem cannot become harder by increasing m beyond some unknown threshold which is at most 8. This upper bound for the threshold given by the dimension d of a sublattice in which short enough lattice vectors must exist (see Section 5.1).

The strongest attack on NTRU lattices is a hybrid method presented at CRYPTO 2007 by Howgrave-Graham [10]. It combines both Meet-in-the-middle (MITM) attacks by Odlyzko [11] and lattice reduction attacks by Coppersmith and Shamir [6]. In our brief summary of the attack we describe three distinct phases.

1. Reduce the public NTRU lattice and save the result in $\mathbf{B}$.
2. Reduce the maximal *sublattice* of $\mathbf{B}$, which satisfies the geometric series assumption (GSA), i.e. for which the $\|\mathbf{b}_i^*\|$ descend linearly in logscale.
3. Let k be the last index of a length contributing vector in $\mathbf{B}^*$, meaning $\|\mathbf{b}_i^*\| \approx 0$ for all $i > k$. Howgrave-Graham introduced a modification of Babai's Nearest Plane algorithm that allows us to perform a MITM attack on the final $\dim(\mathbf{B}) - k$ entries of the secret keys.

Phases 1–2 ensure that $\|\mathbf{b}_k^*\|$ is as big as possible. This allows Babai's original algorithm, and the modification to better approximate CVP in the lattice spanned by the first k basis vectors.

Stated in this form the same algorithm can be used to search for collisions (not pseudo-collisions) in SWIFFT lattices. However, preliminary experiments show that this methodology is not helpful. At the end of phase 2 we find that $k \approx 128$. Obviously, even if the CVP oracle works perfectly we would still have to do a MITM attack on the last $\dim(\mathbf{B}) - k \approx 896$ entries. This is too much to be practical.

We are currently working on a generalization of the attack, where step 2 is iterated for $m-1$ different overlapping parts of the basis, namely

$$[\mathbf{b}_1, \ldots, \mathbf{b}_{2n}], [\mathbf{b}_{n+1}, \ldots, \mathbf{b}_{3n}], \ldots, [\mathbf{b}_{(m-2)n+1}, \ldots, \mathbf{b}_{mn}].$$

This modification is only sensible for SWIFFT and not NTRU. It should bring k closer to $\dim(\mathbf{B})$ possibly at the expense of CVP approximation quality. It remains to be seen if this is a good strategy.

5.4 Acknowledgments

We would like to thank Chris Peikert and Alon Rosen for helpful advice and encouragement. We also want to thank Bartol Filipović, Henrik Koy and Christoph Ludwig for letting us use their lattice reduction code. Finally, we thank Markus Rückert and Michael Schneider for their patience and unbounded cooperation.

References

1. Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In *Proceedings of the Annual Symposium on the Theory of Computing (STOC) 1996*, pages 99–108. ACM Press, 1996.
2. Yuriy Arbitman, Gil Dogon, Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFTX: A proposal for the SHA-3 standard. `http://www.eecs.harvard.edu/~alon/PAPERS/lattices/swifftx.pdf`, 2008.
3. Johannes Buchmann, Martin Döring, and Richard Lindner. Efficiency improvement for NTRU. In Ammar Alkassar and Jörg Siekmann, editors, *SICHERHEIT 2008*, volume 128 of *Lecture Notes in Informatics*, pages 79–94. Bonner Köllen Verlag, 2008.
4. Johannes Buchmann, Richard Lindner, and Markus Rückert. Explicit hard instances of the shortest vector problem. In Johannes Buchmann and Jintai Ding, editors, *PQCrypto*, volume 5299 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2008.
5. Johannes Buchmann and Christoph Ludwig. Practical lattice basis sampling reduction. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS*, volume 4076 of *Lecture Notes in Computer Science*, pages 222–237. Springer, 2006.
6. Don Coppersmith and Adi Shamir. Lattice attacks on NTRU. In *EUROCRYPT*, pages 52–61, 1997.
7. Nicolas Gama and Phong Q. Nguyen. Predicting lattice reduction. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 31–51. Springer, 2008.
8. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the Annual Symposium on the Theory of Computing (STOC) 2008*, pages 197–206. ACM Press, 2008.
9. Phil Hirschhorn, Jill Hoffstein, Nick Howgrave-Graham, and William Whyte. Choosing NTRU parameters in light of combined lattice reduction and MITM approaches. `http://www.ntru.com/cryptolab/pdf/params.pdf`, will be published at ACNS 2009.
10. Nick Howgrave-Graham. A hybrid lattice-reduction and meet-in-the-middle attack against ntru. In Alfred Menezes, editor, *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 150–169. Springer, 2007.
11. Nick Howgrave-Graham, Jospeh H. Silverman, and William Whyte. A meet-in-the-middle attack on an NTRU private key. `http://www.ntru.com/cryptolab/tech_notes.htm#004`.
12. Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *J. Cryptology*, 14(4):255–293, 2001.
13. Vadim Lyubashevsky and Daniele Micciancio. Generalized compact knapsacks are collision resistant. In *International Colloquium on Automata, Languages and*

Programming (ICALP) 2006, Lecture Notes in Computer Science, pages 144–155. Springer-Verlag, 2006.

14. Vadim Lyubashevsky and Daniele Micciancio. Asymptotically efficient lattice-based digital signatures. In *Theory of Cryptography Conference (TCC) 2008*, Lecture Notes in Computer Science, pages 37–54. Springer-Verlag, 2008.
15. Vadim Lyubashevsky, Daniele Micciancio, Chris Peikert, and Alon Rosen. SWIFFT: A modest proposal for FFT hashing. In *Fast Software Encryption (FSE) 2008*, Lecture Notes in Computer Science, pages 54–72. Springer-Verlag, 2008.
16. Daniele Micciancio and Oded Regev. *Post Quantum Cryptography*, chapter Lattice-based Cryptography. Springer-Verlag, 2009.
17. Chris Peikert and Alon Rosen. Efficient collision-resistant hashing from worst-case assumptions on cyclic lattices. In *Theory of Cryptography Conference (TCC) 2006*, Lecture Notes in Computer Science, pages 145–166. Springer-Verlag, 2006.
18. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
19. Claus-Peter Schnorr. Lattice reduction by random sampling and birthday methods. In Helmut Alt and Michel Habib, editors, *STACS*, volume 2607 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2003.
20. Victor Shoup. Number theory library (NTL) for C++. `http://www.shoup.net/ntl/`.