

University of Technology Darmstadt  
Department of Computer Science  
Cryptography and Computer Algebra

Bachelor Thesis

January 2008

# Practical Security of Micciancio's Hash Function



Lyudmil Tserovski

University of Technology Darmstadt  
Department of Mathematics

Supervised by Prof. Dr. Johannes Buchmann,  
Richard Lindner



## Acknowledgments

I would like to thank my direct supervisor Richard Lindner for all his tips and his help throughout this bachelor thesis. I would also like to thank Prof. Buchmann for accepting me to write this thesis and showing me some of the interesting aspects of number theory and cryptography. Not to forget my family and my friends for helping and supporting me all the way through my studies.

## Warranty

I hereby warrant that the content of this thesis is the direct result of my own work and that any use made in it of published or unpublished material is fully and correctly referenced.

Date: ..... Signature: .....



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>The Hash Function Family</b>	<b>1</b>
<b>3</b>	<b>An Instance of a Hash Function</b>	<b>2</b>
3.1	A Note on the needed Parameters . . . . .	2
3.2	An Instance of a Hash Function . . . . .	4
<b>4</b>	<b>Algorithm for Finding Collisions</b>	<b>6</b>
4.1	From the Search for Collisions to Zero-Sums . . . . .	6
4.2	In the Search for Zero-Sums . . . . .	6
4.3	Algorithm for finding collisions . . . . .	10
<b>5</b>	<b>Conclusion</b>	<b>10</b>
<b>A</b>	<b>Appendix - Complete List of basic hashes used in 3.2</b>	<b>12</b>



# 1 Introduction

In this Bachelor thesis, we are trying to investigate the practical security of Micciancio's hash function family, constructed in [2]. We first show how this hash function family is constructed. Then we give an exact example of a hash function in a low dimension and see that for it, it is easy to find collisions. Finally, we construct a method for finding collisions in higher dimensions.

## 2 The Hash Function Family

In this section we present the definitions we use and show how Micciancio's hash function family is constructed.

**Definition 2.1.** [2] The *generalized knapsack problem* is defined as follows: given  $m$  random elements  $a_1, \dots, a_m$  in a ring  $R$  and a target  $t \in R$ , find  $z_1, \dots, z_m \in D$  such that  $\sum a_i z_i = t$ , where  $D$  is some fixed subset of  $R$ .

**Definition 2.2.** [2] The *expansion factor* of a function  $f$  is defined as

$$EF(f, k) = \max_{g \in \mathbb{Z}, \deg(g) \leq k(\deg(f)-1)} \|g\|_f / \|g\|_\infty.$$

Now we can define the hash function family, which is an instance of the generalized knapsack problem.

**Definition 2.3.** [2] We call a function  $\mathfrak{h}$  *hash function*, if it maps elements from a set  $A$  to a set  $B$ , where  $|A| > |B|$ .

Specifically in [2] a hash function family is defined as follows.

**Definition 2.4.** Let  $f \in \mathbb{Z}[x]$  be an irreducible monic polynomial of degree  $n$  with expansion factor  $EF(f, 3) \leq \mathcal{E}$ , where  $\mathcal{E}$  is an upper bound for  $EF(f, 3)$ , and let  $p$  be a positive integer. Given a ring of polynomials  $R = \mathbb{Z}_p[x]/\langle f \rangle$  of degree  $< n$ , subset  $D = \{g \in R : \|g\|_f \leq d\}$  for some positive integer  $d$  and a positive constant  $m$ , the *hash function family*  $\mathcal{H}(R, D, m)$  defined in [2] is the collection of all functions

$$\mathfrak{h}_{\mathbf{a}} : D^m \rightarrow R$$

indexed by  $\mathbf{a} = (a_1, \dots, a_m) \in R^m$  mapping  $\mathbf{b} = (b_1, \dots, b_m) \in D^m$  to

$$\mathfrak{h}_{\mathbf{a}}(\mathbf{b}) = \sum_{i=1}^m a_i \cdot b_i \in R.$$

More precisely, by the norm  $\|g\|_f$ , we mean the infinite norm. Moreover, as we identify polynomials of degree  $< n$  with the corresponding  $n$ -dimensional vectors having the coefficients of the polynomial as coordinates,

we have that  $\|g\|_f = \|g + \langle f \rangle\|_f = \|g \bmod f\|_\infty = \max_i |a_i|$  for  $a_i$  being the coefficients of the polynomial  $g$ .

The functions of the family  $\mathcal{H}$  are mapping elements from  $D^m$  to  $R$  where  $|D^m| = (2d+1)^{nm}$  and  $|R| = p^n$ . So, if  $m > \log(p)/\log(2d)$ , then  $\mathcal{H}$  will be a family of functions that has collisions.

Now we need to define what collision resistance of a hash function is.

**Definition 2.5.** [2] For any function family  $\mathcal{H}$ , the problem  $\text{Collision}_{\mathcal{H}}$  is defined as follows: given a function  $\mathfrak{h} \in \mathcal{H}$ , find a collision, i.e. a pair of inputs  $\mathbf{b}, \mathbf{c} \in D^m$  such that  $\mathbf{b} \neq \mathbf{c}$  and  $\mathfrak{h}(\mathbf{b}) = \mathfrak{h}(\mathbf{c})$ .

If there is no polynomial time algorithm that can solve  $\text{Collision}_{\mathcal{H}}$  with non-negligible probability when given an  $\mathfrak{h}$  which is distributed uniformly at random in  $\mathcal{H}$ , then we say that  $\mathcal{H}$  is a *collision resistant* family of hash functions.

A hash function  $\mathfrak{h}$  is said to be *secure* if it is collision resistant.

In order to prove that his hash function family is collision resistant, Micciancio shows in [2] that if there exists a polynomial-time algorithm that succeeds with non-negligible probability in finding  $\mathbf{b} \neq \mathbf{c} \in D^m$  such that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{b}) = \mathfrak{h}_{\mathbf{a}}(\mathbf{c})$ , for a randomly chosen hash function  $\mathfrak{h}_{\mathbf{a}} \in R^m$ , then a certain problem called the “approximate shortest polynomial problem” is solvable in polynomial time for *every* ideal of the ring  $\mathbb{Z}_p[x]$ . More precisely:

**Definition 2.6.** [2] In the *Approximate Shortest Polynomial Problem* ( $SPP_\gamma(I)$ ), we are given an ideal  $I \subseteq \mathbb{Z}_p[x]/\langle f \rangle$ , where  $f$  is a monic polynomial of degree  $n$ , and we are asked to find  $g \in I$  such that  $g \neq 0$  and  $\|g\|_f \leq \gamma \lambda_1^\infty(I)$ , where  $\gamma \lambda_1^\infty(I)$  is the  $\gamma$ -approximate minimum distance in the lattice  $L(I)$ .

The main theorem in [2] is:

**Theorem 2.7.** [2] *Let  $\mathcal{H}$  be a function family as above with  $m > \log(p)/\log(2d)$  and  $p > 2\mathcal{E}dmn^{1.5} \log(n)$ . Then, for  $\gamma = 8\mathcal{E}^2dmn \log^2(n)$ , there is a polynomial time reduction from  $f - SPP_\gamma(I)$  to  $\text{Collision}_{\mathcal{H}}(\mathfrak{h})$ , for any  $I$  where  $\mathfrak{h}$  is chosen uniformly at random from  $\mathcal{H}$ .*

### 3 An Instance of a Hash Function

In this section we show that for a given hash function in a lower dimension it is easy to find collisions. We also try to find an effective way in finding them.

#### 3.1 A Note on the needed Parameters

The parameters that we need to define a hash function family are:

- $n$  – the degree of the irreducible polynomial

- $m$  – the length of the key  $\mathbf{a}$
- $d$  – the norm of the functions  $g$ , used to define the set  $D$
- $p$  – the integer, used to define the set  $R$
- $f$  – the irreducible polynomial, also used to define  $R$
- $\mathcal{E}$  – the expansion factor for  $f$

In Section 2 we stated that if  $m > \log(p)/\log(2d)$ , then  $\mathcal{H}(R, D, m)$  will be a family of functions that will have collisions, and that is what we are interested in. In order that  $|D^m|/|R| \geq 2/1$ , or otherwise stated  $((2d+1)^m/p)^n \geq 2/1$ , we have to compute  $m$  and  $p$  depending on the choice of  $\mathcal{E}$ ,  $d$  and  $n$ . We also want  $p$  to be minimal. So, instead of using the condition stated in Theorem 2.7:  $p > 2\mathcal{E}dmn^{1.5} \log(n)$ , we use:  $p = \lceil 2\mathcal{E}dmn^{1.5} \log(n) \rceil$ . We introduce the following algorithm for calculating  $m$  and  $p$  according to the values of  $\mathcal{E}, d, n$ :

**Input:**  $\mathcal{E}, d, n$   
**Output:**  $m$  and  $p$ , such that  $\left(\frac{(2d+1)^m}{p}\right)^n \geq 2$  and  
 $p = \lceil 2\mathcal{E}dmn^{1.5} \log(n) \rceil$

```

1  $q = 1$ 
2  $m = 0$ 
3 while  $q < 2$  do
4    $m = m + 1$ 
5    $p = \lceil 2\mathcal{E}dmn^{1.5} \log(n) \rceil$ 
6    $q = \left(\frac{(2d+1)^m}{p}\right)^n$ 
7 end
8 return  $m, p$ 

```

**Algorithm 1:** Compute  $m$  and  $p$

In Table 1 we present some sample outputs produced by Algorithm 1.

$\mathcal{E} = 3$ $d = 1$	$n = 2$	$\rightarrow$	$m = 5$	$p = 85$	$q = 8.172$
	$n = 10$	$\rightarrow$	$m = 8$	$p = 5043$	$q = 13.893$
	$n = 20$	$\rightarrow$	$m = 10$	$p = 23194$	$q = 130839202.226$
	$n = 50$	$\rightarrow$	$m = 11$	$p = 131697$	$q = 2740917.691$
$\mathcal{E} = 3$ $d = 3$	$n = 2$	$\rightarrow$	$m = 3$	$p = 153$	$q = 5.025$
	$n = 10$	$\rightarrow$	$m = 5$	$p = 9455$	$q = 314.982$
	$n = 20$	$\rightarrow$	$m = 6$	$p = 41750$	$q = 996862376.772$
	$n = 50$	$\rightarrow$	$m = 7$	$p = 251421$	$q = 5.810e + 25$
$\mathcal{E} = 3$ $d = 5$	$n = 2$	$\rightarrow$	$m = 3$	$p = 255$	$q = 27.244$
	$n = 10$	$\rightarrow$	$m = 4$	$p = 12606$	$q = 4.466$
	$n = 20$	$\rightarrow$	$m = 5$	$p = 57985$	$q = 746371909.409$
	$n = 50$	$\rightarrow$	$m = 6$	$p = 359179$	$q = 4.493e + 34$

$\mathcal{E} = 6$ $d = 1$	$n = 2$	$\rightarrow$	$m = 5$	$p = 170$	$q = 2.043$
	$n = 10$	$\rightarrow$	$m = 9$	$p = 11346$	$q = 246.879$
	$n = 20$	$\rightarrow$	$m = 10$	$p = 46388$	$q = 124.777$
	$n = 50$	$\rightarrow$	$m = 12$	$p = 287339$	$q = 2.254e + 13$
$\mathcal{E} = 6$ $d = 3$	$n = 2$	$\rightarrow$	$m = 4$	$p = 408$	$q = 34.630$
	$n = 10$	$\rightarrow$	$m = 6$	$p = 22691$	$q = 14039319.019$
	$n = 20$	$\rightarrow$	$m = 6$	$p = 83499$	$q = 950.909$
	$n = 50$	$\rightarrow$	$m = 7$	$p = 502842$	$q = 51610343661.5$
$\mathcal{E} = 6$ $d = 5$	$n = 2$	$\rightarrow$	$m = 3$	$p = 510$	$q = 6.811$
	$n = 10$	$\rightarrow$	$m = 5$	$p = 31515$	$q = 12146779.373$
	$n = 20$	$\rightarrow$	$m = 5$	$p = 115970$	$q = 711.795$
	$n = 50$	$\rightarrow$	$m = 6$	$p = 718346$	$q = 3.991e + 19$

Table 1: Sample outputs of Algorithm 1

We see that in order to keep  $p$  as small as possible, first we need to set small  $\mathcal{E}$ , and we know from [2] that if we choose  $f = x^n + 1$ , then  $\mathcal{E}$  will be 3. Also from the table above we see that setting  $d$  small also helps keeping  $p$  small. We will use this information in the construction of the actual instance.

### 3.2 An Instance of a Hash Function

Now that we know what exactly we need for a hash function to exist, we can actually construct one. We start by choosing the irreducible polynomial. Let  $f = x^n + 1$ . We choose that polynomial because for an even  $n$ , we know that  $f$  is irreducible and its upper bound for the Expansion Factor is  $\mathcal{E} = 3$ , which as we already saw keeps  $p$  small. We set  $d = 1$ , because that way  $p$  will be minimal. Now the set  $D = \{g \in R : \|g\|_f \leq 1\}$ . Let  $n = 2$ . With the help of Algorithm 1 we compute then that  $m = 5$  and  $p = 85$ , and we will have  $R = \mathbb{Z}_{85}[x]/\langle x^2 + 1 \rangle$ .

The length of the key we need is 5, so we can randomly choose 5 elements from  $R$ :

$$\mathbf{a} = (10 + 2x, 3 + 4x, x, 80, 5 + 5x).$$

We can also show how the elements in  $D$  look like. Due to our construction  $D = \{g \in R : \|g\|_f \leq 1\} \subset R$ . So  $g = \{-1, 0, 1\} + \{-1, 0, 1\} \cdot x \in R$ .

The hash function looks like as follows:

$$\mathfrak{h}_{\mathbf{a}}(\mathbf{b}) = \langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=1}^5 a_i \cdot b_i, \text{ for } \mathbf{b} \in D^5$$

Now our main task is to find collisions, i.e. elements  $\mathbf{b}, \mathbf{c} \in D^5$ ,  $\mathbf{b} \neq \mathbf{c}$ , such that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{b}) = \mathfrak{h}_{\mathbf{a}}(\mathbf{c})$ . We could search for all possible combinations, and for  $n = 2$  we could find such collisions. But if  $n$  is increased, then  $m$  will

also be increased and the set  $D^m$  becomes extremely large and it would take too much time to try all possible combinations. So we try to find a better way to search for collisions.

We have randomly chosen the key for our hash function:

$$\mathbf{a} = (10 + 2x, 3 + 4x, x, 80, 5 + 5x).$$

We apply the hash function  $\mathfrak{h}_{\mathbf{a}}$  to the following basic elements  $\mathbf{b}_i \in D^5$ . Then, using the linearity of the hash function, we will try to find combinations between the resulting polynomials that lead to collisions.

$$\begin{aligned} \mathbf{b}_{1,1} &= (1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{1,1}) = 10 + 2x \\ \mathbf{b}_{1,2} &= (0, 1, 0, 0, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{1,2}) = 3 + 4x \\ \mathbf{b}_{1,3} &= (0, 0, 1, 0, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{1,3}) = x \\ \mathbf{b}_{1,4} &= (0, 0, 0, 1, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{1,4}) = 80 \\ \mathbf{b}_{1,5} &= (0, 0, 0, 0, 1) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{1,5}) = 5 + 5x \\ &\vdots \\ \mathbf{b}_{8,1} &= (-x - 1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{8,1}) = 77 + 73x \\ \mathbf{b}_{8,2} &= (0, -x - 1, 0, 0, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{8,2}) = 1 + 78x \\ \mathbf{b}_{8,3} &= (0, 0, -x - 1, 0, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{8,3}) = 1 + 84x \\ \mathbf{b}_{8,4} &= (0, 0, 0, -x - 1, 0) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{8,4}) = 5 + 5x \\ \mathbf{b}_{8,5} &= (0, 0, 0, 0, -x - 1) \rightarrow \mathfrak{h}_{\mathbf{a}}(\mathbf{b}_{8,5}) = 75x \end{aligned}$$

The complete list is presented in Appendix A.

Now that we computed these basic functions, we can start looking for combinations of  $\mathbf{b}_{i,j}$  that lead to collisions. Such are, for example:

$$\begin{aligned} \mathbf{b}_{5,4} &= (0, 0, 0, x + 1, 0) \neq (0, 0, 0, 0, -1) = \mathbf{b}_{4,5} \\ \text{whereas: } \mathfrak{h}(\mathbf{b}_{5,4}) &= \mathfrak{h}(\mathbf{b}_{4,5}) = 80 + 80x. \end{aligned}$$

Also

$$\begin{aligned} \mathbf{b}_{8,4} &= (0, 0, 0, -x - 1, 0) \neq (0, 0, 0, 0, 1) = \mathbf{b}_{1,5} \\ \text{but: } \mathfrak{h}(\mathbf{b}_{8,4}) &= \mathfrak{h}(\mathbf{b}_{1,5}) = 5 + 5x. \end{aligned}$$

And

$$\begin{aligned} \mathbf{f} &= (0, 0, 0, -x - 1, -x - 1) \neq (0, 0, 0, 0, -x) = \mathbf{b}_{3,5} \\ \text{but: } \mathfrak{h}(\mathbf{f}) &= \mathfrak{h}(\mathbf{b}_{3,5}) = 5 + 80x \end{aligned}$$

Important to notice is that if we add

$$\mathbf{b}_{5,4} = (0, 0, 0, x + 1, 0) \text{ with } \mathbf{b}_{1,5} = (0, 0, 0, 0, 1),$$

we get

$$\mathbf{c} = \mathbf{b}_{5,4} + \mathbf{b}_{1,5} = (0, 0, 0, x + 1, 1) \text{ with } \mathfrak{h}(\mathbf{c}) = 0.$$

## 4 Algorithm for Finding Collisions

### 4.1 From the Search for Collisions to Zero-Sums

Using the observations from the previous section, we come to the following important result:

**Lemma 4.1.** *If  $\mathbf{d} \in R^m$ ,  $\mathbf{d} \neq \mathbf{0}$  is such that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{d}) = 0$ , then for all  $\mathbf{b} \in D^m$ , such that  $\mathbf{b} + \mathbf{d} \in D^m$  we have that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{b} + \mathbf{d}) = \mathfrak{h}_{\mathbf{a}}(\mathbf{b})$ .*

*Proof.* This lemma can be proved using the properties of the scalar product. We have that:  $\mathfrak{h}_{\mathbf{a}}(\mathbf{d}) = \langle \mathbf{a}, \mathbf{d} \rangle = 0$  then

$$\mathfrak{h}_{\mathbf{a}}(\mathbf{b} + \mathbf{d}) = \langle \mathbf{a}, \mathbf{b} + \mathbf{d} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle + \langle \mathbf{a}, \mathbf{d} \rangle = \langle \mathbf{a}, \mathbf{b} \rangle + 0 = \mathfrak{h}_{\mathbf{a}}(\mathbf{b}).$$

□

Lemma 4.1 allows us to transform the problem of finding collisions, to a problem of finding vectors  $\mathbf{d} \in R^m$  such that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{d}) = 0$  or otherwise stated, finding vectors in  $R^m$  that lead to a zero-sum.

### 4.2 In the Search for Zero-Sums

Now that we know that in order to find collisions i.e. elements  $\mathbf{b}, \mathbf{c} \in D^m$ ,  $\mathbf{b} \neq \mathbf{c}$  such that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{b}) = \mathfrak{h}_{\mathbf{a}}(\mathbf{c})$ , it is enough to find element  $\mathbf{d} \in R^m$ ,  $\mathbf{d} \neq \mathbf{0}$  such that  $\mathfrak{h}_{\mathbf{a}}(\mathbf{d}) = 0$ .

Buchmann, Lindner and Rückert have recently developed in [1] an algorithm that constructs bases for zero-sum solutions.

**Input:** A matrix  $X = [\mathbf{x}_1, \dots, \mathbf{x}_m] \in \mathbb{Z}_p^{n \times m}$   
**Output:** A full-rank lattice basis  $Y = [\mathbf{y}_1, \dots, \mathbf{y}_m] \in \mathbb{Z}^{m \times m}$ , such that  $\mathbf{v} \in L(Y) \Leftrightarrow X\mathbf{v} \equiv \mathbf{0} \pmod{p}$ , where  $L(Y)$  is the lattice spanned by  $Y$ .

- 1 For each residue in  $X$  choose the representative in  $\{0, \dots, p-1\}$
- 2  $Y_1 \leftarrow (X^T \quad pI_m)$
- 3  $(Y_2, T_1) \leftarrow \text{HNF}(\mathcal{L}(Y_1))$ , where  $Y_2 \in \mathbb{Z}^{m \times m}$  is the Hermite Normal Form,  $Y_2 T_1 = Y_1$ , and  $T_1$  is integral
- 4  $T_2 \leftarrow T_1$  without the  $n$  leading columns, such that  $Y_2 T_2 = pI_m$
- 5 **return**  $Y_3 \leftarrow p(Y_2^{-1})^T = T_2^T$

**Algorithm 2:** Constructing bases of zero-sum solutions

In Section 2 we said that we identify polynomials (of degree  $< n$ ) with the corresponding  $n$ -dimensional vectors having the coefficients of the polynomial as coordinates.

So we have the hash key

$$\mathbf{a} = (\mathbf{a}_1, \dots, \mathbf{a}_m) \in R^m,$$

where each  $\mathbf{a}_i$  is a polynomial with coefficients:

$$\begin{pmatrix} a_{i,1} \\ \vdots \\ a_{i,n} \end{pmatrix}$$

Therefore  $\mathbf{a}$  is equal to the matrix:

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{pmatrix} \in \mathbb{Z}_p^{n \times m}$$

In the same way an element  $\mathbf{b} = (\mathbf{b}_1, \dots, \mathbf{b}_m) \in D^m$  is a vector of  $m$  polynomials, each  $\mathbf{b}_i$  having the coefficients:

$$\begin{pmatrix} b_{i,1} \\ \vdots \\ b_{i,m} \end{pmatrix}$$

Therefore  $\mathbf{b}$  is equal to the matrix:

$$\begin{pmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{pmatrix} \in \mathbb{Z}^{n \times m}$$

So, using vectors the hash function  $\mathfrak{h}_{\mathbf{a}}(\mathbf{b})$  is:

$$\begin{aligned} \mathfrak{h}_{\mathbf{a}}(\mathbf{b}) &= \langle \mathbf{a}, \mathbf{b} \rangle = \langle (\mathbf{a}_1, \dots, \mathbf{a}_m), (\mathbf{b}_1, \dots, \mathbf{b}_m) \rangle = \\ &= \left\langle \begin{pmatrix} a_{1,1} & \cdots & a_{1,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{pmatrix}, \begin{pmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{pmatrix} \right\rangle, \end{aligned}$$

where each row of  $a_{i,\cdot}$  and  $b_{i,\cdot}$  represents the coefficients of the polynomials in front of  $x^{i-1}$ .

Now we can apply Algorithm 2 on  $\mathbf{a}$ . What we get as a result is the matrix

$$Y_0 = \begin{pmatrix} y_{1,1}^{(0)} & \cdots & y_{1,m}^{(0)} \\ \vdots & \ddots & \vdots \\ y_{m,1}^{(0)} & \cdots & y_{m,m}^{(0)} \end{pmatrix} \in \mathbb{Z}_p^{m \times m}$$

so that for all  $\mathbf{y}^{(0)} = (y_1^{(0)}, \dots, y_m^{(0)}) \in L(Y_0)$  we have that  $\mathbf{a} \cdot \mathbf{y}^{(0)} = \mathbf{0}$ .

That means that we have found vectors  $(\mathbf{y}_1^{(0)}, \dots, \mathbf{y}_m^{(0)})$  such that

$$\langle (\mathbf{a}_1, \dots, \mathbf{a}_m), (\mathbf{y}_1^{(0)}, \dots, \mathbf{y}_m^{(0)}) \rangle = \left\langle \begin{pmatrix} a_{1,1} & \cdots & a_{1,m} \\ a_{2,1} & \cdots & a_{2,m} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \cdots & a_{n,m} \end{pmatrix}, \begin{pmatrix} y_1^{(0)} & \cdots & y_m^{(0)} \\ 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix} \right\rangle = \mathbf{0}$$

Or otherwise stated:  $\mathbf{a}_1 \cdot y_1^{(0)} + \dots + \mathbf{a}_m \cdot y_m^{(0)} = \mathbf{0}$ . That is, we found constant polynomials:  $y_i^{(0)} \cdot x^0 \in R^m$  that lead to zero-sums. But these might have too large norms, so it might be difficult to find any  $\mathbf{b} \in D^m$  so that  $y^{(0)} + \mathbf{b} \in D^m$  as required in Lemma 4.1. So our aim is to find all possible  $m$ -dimensional vectors of polynomials that lead to zero-sums.

**Definition 4.2.** Let  $\mathbf{a}^x$  be the vector of polynomials of dimension  $m$ , such that

$$\mathbf{a}^x = (\mathbf{a}_1 \cdot x, \dots, \mathbf{a}_m \cdot x) \pmod{f}.$$

Analogously, let  $\mathbf{a}^{x^i}$  be the vector of polynomials of dimension  $m$ , such that

$$\mathbf{a}^{x^i} = (\mathbf{a}_1 \cdot x^i, \dots, \mathbf{a}_m \cdot x^i) \pmod{f}.$$

Now we can apply Algorithm 2 to the matrix  $\mathbf{a}^x \in \mathbb{Z}_p^{n \times m}$  as defined in Definition 4.2. As a result we get the matrix

$$Y_1 = \begin{pmatrix} y_{1,1}^{(1)} & \cdots & y_{1,m}^{(1)} \\ \vdots & \ddots & \vdots \\ y_{m,1}^{(1)} & \cdots & y_{m,m}^{(1)} \end{pmatrix} \in \mathbb{Z}_p^{m \times m}$$

And for all  $\mathbf{y}^{(1)} = (y_1^{(1)}, \dots, y_m^{(1)}) \in L(Y_1)$  we have that  $\mathbf{a} \cdot \mathbf{y}^{(1)} = \mathbf{0}$ . Or otherwise stated we will have:

$$\left\langle (\mathbf{a}_1, \dots, \mathbf{a}_m), \begin{pmatrix} 0 & \cdots & 0 \\ y_1^{(1)} & \cdots & y_m^{(1)} \\ \mathbf{0} \end{pmatrix} \right\rangle = \mathbf{0}.$$

That is, we found coefficients for polynomials in  $R$  of the form  $y_i^{(1)} \cdot x$  that lead to zero-sums.

We now apply Algorithm 2 to the matrix  $(\mathbf{a} \mathbf{a}^x) \in \mathbb{Z}_p^{n \times 2m}$ . As a result we get the matrix

$$Y_2 = \begin{pmatrix} y_{1,1}^{(0)} & \cdots & y_{1,m}^{(0)} & y_{1,m+1}^{(0)} & \cdots & y_{1,2m}^{(0)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{m,1}^{(0)} & \cdots & y_{m,m}^{(0)} & y_{m,m+1}^{(0)} & \cdots & y_{m,2m}^{(0)} \\ y_{1,1}^{(1)} & \cdots & y_{1,m}^{(1)} & y_{1,m+1}^{(1)} & \cdots & y_{1,2m}^{(1)} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{m,1}^{(1)} & \cdots & y_{m,m}^{(1)} & y_{m,m+1}^{(1)} & \cdots & y_{m,2m}^{(1)} \end{pmatrix} \in \mathbb{Z}_p^{2m \times 2m}$$

so that, for all  $\mathbf{y}^{(0)}, \mathbf{y}^{(1)} \in L(Y_2)$  and for all linear combinations we have

$$(\mathbf{a} \mathbf{a}^x) \cdot \begin{pmatrix} \mathbf{y}^{(0)} \\ \mathbf{y}^{(1)} \end{pmatrix} = \mathbf{0}.$$

Or otherwise stated we will have:

$$\left\langle (\mathbf{a}_1, \dots, \mathbf{a}_m), \begin{pmatrix} y_1^{(0)} & \cdots & y_m^{(0)} \\ y_1^{(1)} & \cdots & y_m^{(1)} \\ \mathbf{0} \end{pmatrix} \right\rangle = \mathbf{0}.$$

That is, we found coefficients for polynomials in  $R$  of the form  $y_i^{(1)} \cdot x + y_i^{(0)}$  that lead to zero-sums.

Analogously, we can apply Algorithm 2 to the matrix  $(\mathbf{a} \mathbf{a}^x \cdots \mathbf{a}^{x^{n-1}}) \in \mathbb{Z}_p^{n \times nm}$ . As a result we get the matrix

$$Y_3 = \begin{pmatrix} y_{1,1}^{(0)} & \cdots & y_{1,m}^{(0)} & \cdots & y_{1,nm-m}^{(0)} & \cdots & y_{1,nm}^{(0)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{m,1}^{(0)} & \cdots & y_{m,m}^{(0)} & \cdots & y_{m,nm-m}^{(0)} & \cdots & y_{m,nm}^{(0)} \\ y_{1,1}^{(1)} & \cdots & y_{1,m}^{(1)} & \cdots & y_{1,nm-m}^{(1)} & \cdots & y_{1,nm}^{(1)} \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ y_{m,1}^{(n-1)} & \cdots & y_{m,m}^{(n-1)} & \cdots & y_{m,nm-m}^{(n-1)} & \cdots & y_{m,nm}^{(n-1)} \end{pmatrix} \in \mathbb{Z}_p^{nm \times nm},$$

so that, for all  $\mathbf{y}^{(0)}, \dots, \mathbf{y}^{(n-1)} \in L(Y_3)$  and for all linear combinations we have

$$(\mathbf{a} \mathbf{a}^x \cdots \mathbf{a}^{x^{n-1}}) \cdot \begin{pmatrix} \mathbf{y}^{(0)} \\ \vdots \\ \mathbf{y}^{(n-1)} \end{pmatrix} = \mathbf{0}.$$

Or otherwise stated we will have:

$$\left\langle (\mathbf{a}_1, \dots, \mathbf{a}_m), \begin{pmatrix} y_1^{(0)} & \cdots & y_m^{(0)} \\ \vdots \\ y_1^{(n-1)} & \cdots & y_m^{(n-1)} \end{pmatrix} \right\rangle = \mathbf{0}.$$

That is, we found coefficients for all the polynomials in  $R$  of the form

$$y_i^{(n-1)} \cdot x^{n-1} + y_i^{(n-2)} \cdot x^{n-2} + \dots + y_i^{(0)}$$

that lead to zero-sums.

Now that we found the matrix  $Y_3$ , we want to look for vectors with possibly smaller norm. So that's why we can use an LLL- or a BKZ-algorithm<sup>1</sup> to reduce the given basis.

### 4.3 Algorithm for finding collisions

**Input:** An even integer  $n$ , an integer  $d \in \mathbb{N}, d \geq 1$ , an irreducible polynomial  $f$

**Output:** True - if a collision is found, False - otherwise

- 1 For the given polynomial compute the Expansion Factor  $\mathcal{E}$
- 2  $m, p = \text{Algorithm 1}(\mathcal{E}, d, n)$
- 3 Choose  $m$  random polynomials for the hash  $\mathbf{a}$
- 4 Compute  $\mathbf{a}^x, \mathbf{a}^{x^2}, \dots, \mathbf{a}^{x^{n-1}}$  as defined in Definition 4.2
- 5  $Y = \text{Algorithm 2}(\mathbf{a} \ \mathbf{a}^x \ \dots \ \mathbf{a}^{x^{n-1}})$
- 6 Reduce the basis  $Y$  using LLL- or BKZ-algorithm
- 7 **if** for a vector  $\mathbf{y} = (y_1, \dots, y_m)^T$  from the reduced basis, a vector  $\mathbf{b} = (b_1, \dots, b_m) \in D^m$  is found, such that  $\mathbf{y} + \mathbf{b} \in D^m$  **then**
- 8 |   **return** True
- 9 **end**
- 10 **else**
- 11 |   **return** False
- 12 **end**

**Algorithm 3:** Finding Collisions

## 5 Conclusion

In this bachelor thesis, we tried to investigate the security of the Hash function defined by Micciancio in [2]. For this, we first constructed an instance of the hash function in a low dimension, and showed that it was easy to find collisions. Moreover, we used the observations, made on this exact instance and later constructed an effective algorithm to find collisions in higher dimensions.

---

<sup>1</sup>LLL and BKZ are algorithms for reduction of lattice bases. For more information see: [3]

## References

- [1] J. Buchmann, R. Lindner, and M. Rückert. Creating a lattice challenge. <http://www.cdc.informatik.tu-darmstadt.de/~rlindner/latchall.pdf>, 2008.
- [2] V. Lyubachevsky and D. Micciancio. Generalized compact knapsacks are collision resistant. 2005.
- [3] C.P. Schnorr and M. Euchner. Lattice basis reduction: Improved practical algorithms and solving subset sum problems. 1993.

## A Appendix - Complete List of basic hashes used in 3.2

$\mathbf{b}_{1,1} = (1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{1,1}) = 10 + 2x$
$\mathbf{b}_{1,2} = (0, 1, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{1,2}) = 3 + 4x$
$\mathbf{b}_{1,3} = (0, 0, 1, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{1,3}) = x$
$\mathbf{b}_{1,4} = (0, 0, 0, 1, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{1,4}) = 80$
$\mathbf{b}_{1,5} = (0, 0, 0, 0, 1) \rightarrow \mathfrak{h}_a(\mathbf{b}_{1,5}) = 5 + 5x$
<hr/>
$\mathbf{b}_{2,1} = (x, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{2,1}) = 83 + 10x$
$\mathbf{b}_{2,2} = (0, x, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{2,2}) = 81 + 3x$
$\mathbf{b}_{2,3} = (0, 0, x, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{2,3}) = 84$
$\mathbf{b}_{2,4} = (0, 0, 0, x, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{2,4}) = 80x$
$\mathbf{b}_{2,5} = (0, 0, 0, 0, x) \rightarrow \mathfrak{h}_a(\mathbf{b}_{2,5}) = 80 + 5x$
<hr/>
$\mathbf{b}_{3,1} = (-x, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{3,1}) = 2 + 75x$
$\mathbf{b}_{3,2} = (0, -x, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{3,2}) = 4 + 82x$
$\mathbf{b}_{3,3} = (0, 0, -x, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{3,3}) = 1$
$\mathbf{b}_{3,4} = (0, 0, 0, -x, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{3,4}) = 5x$
$\mathbf{b}_{3,5} = (0, 0, 0, 0, -x) \rightarrow \mathfrak{h}_a(\mathbf{b}_{3,5}) = 5 + 80x$
<hr/>
$\mathbf{b}_{4,1} = (-1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{4,1}) = 75 + 83x$
$\mathbf{b}_{4,2} = (0, -1, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{4,2}) = 82 + 81x$
$\mathbf{b}_{4,3} = (0, 0, -1, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{4,3}) = 84x$
$\mathbf{b}_{4,4} = (0, 0, 0, -1, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{4,4}) = 5$
$\mathbf{b}_{4,5} = (0, 0, 0, 0, -1) \rightarrow \mathfrak{h}_a(\mathbf{b}_{4,5}) = 80 + 80x$
<hr/>
$\mathbf{b}_{5,1} = (x + 1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{5,1}) = 8 + 12x$
$\mathbf{b}_{5,2} = (0, x + 1, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{5,2}) = 84 + 7x$
$\mathbf{b}_{5,3} = (0, 0, x + 1, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{5,3}) = 84 + x$
$\mathbf{b}_{5,4} = (0, 0, 0, x + 1, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{5,4}) = 80 + 80x$
$\mathbf{b}_{5,5} = (0, 0, 0, 0, x + 1) \rightarrow \mathfrak{h}_a(\mathbf{b}_{5,5}) = 10x$
<hr/>

$$\begin{aligned}
\mathbf{b}_{6,1} &= (x-1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{6,1}) = 73 + 8x \\
\mathbf{b}_{6,2} &= (0, x-1, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{6,2}) = 78 + 84x \\
\mathbf{b}_{6,3} &= (0, 0, x-1, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{6,3}) = 84 + 84x \\
\mathbf{b}_{6,4} &= (0, 0, 0, x-1, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{6,4}) = 5 + 80x \\
\mathbf{b}_{6,5} &= (0, 0, 0, 0, x-1) \rightarrow \mathfrak{h}_a(\mathbf{b}_{6,5}) = 75 \\
\hline
\mathbf{b}_{7,1} &= (-x+1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{7,1}) = 12 + 77x \\
\mathbf{b}_{7,2} &= (0, -x+1, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{7,2}) = 7 + x \\
\mathbf{b}_{7,3} &= (0, 0, -x+1, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{7,3}) = 1 + x \\
\mathbf{b}_{7,4} &= (0, 0, 0, -x+1, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{7,4}) = 80 + 5x \\
\mathbf{b}_{7,5} &= (0, 0, 0, 0, -x+1) \rightarrow \mathfrak{h}_a(\mathbf{b}_{7,5}) = 10 \\
\hline
\mathbf{b}_{8,1} &= (-x-1, 0, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{8,1}) = 77 + 73x \\
\mathbf{b}_{8,2} &= (0, -x-1, 0, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{8,2}) = 1 + 78x \\
\mathbf{b}_{8,3} &= (0, 0, -x-1, 0, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{8,3}) = 1 + 84x \\
\mathbf{b}_{8,4} &= (0, 0, 0, -x-1, 0) \rightarrow \mathfrak{h}_a(\mathbf{b}_{8,4}) = 5 + 5x \\
\mathbf{b}_{8,5} &= (0, 0, 0, 0, -x-1) \rightarrow \mathfrak{h}_a(\mathbf{b}_{8,5}) = 75x
\end{aligned}$$