Optimal Covering Codes for Finding Near-Collisions

Mario Lamberger[1] and Vincent Rijmen[1,2]

[1] Institute for Applied Information Processing and Communications
Graz University of Technology, Inffeldgasse 16a, A–8010 Graz, Austria.
[2] Dept. of Electrical Engineering ESAT/COSIC, K. U. Leuven and Interdisciplinary Institute for BroadBand Technology (IBBT), Kasteelpark Arenberg 10, B–3001 Heverlee, Belgium.
mario.lamberger@iaik.tugraz.at

Abstract. Recently, a new generic method to find near-collisions for cryptographic hash functions in a memoryless way has been proposed. This method is based on classical cycle-finding techniques and covering codes. This paper contributes to the coding theory aspect of this method by giving the optimal solution to a problem which arises when constructing a suitable code as the direct sum of Hamming and trivial codes.

Keywords: Hash functions, memoryless near-collisions, covering codes, direct sum construction, digital expansions

1 Introduction

The field of hash function research has developed significantly in the light of the attacks on some of the most frequently used hash functions like MD4, MD5 and SHA-1 (cf. [5,6,7,22,23]). As a consequence, academia and industry started to evaluate alternative hash functions, *e.g.* in the SHA-3 initiative organized by NIST [16]. During this ongoing evaluation, not only the three classical security requirements *collision resistance, preimage resistance* and *second preimage resistance* are considered. Researchers look at (semi-)free-start collisions, near-collisions, etc. Whenever a 'behavior different from that expected of a random oracle' could be demonstrated, the hash function is considered suspect, and so are weaknesses that are demonstrated only for the compression function and not for the full hash function.

Coding theory has entered the stage of hash function cryptanalysis quite early where an integral part in the attack strategies is based on the search for low-weight code words in a linear code (*cf.* [1,3,18] among others). In this paper, we want to elaborate on a newly proposed application of coding theory to hash function cryptanalysis. In [13], it is demonstrated how to use covering codes to find near-collisions for hash functions in a memoryless way. We also want to refer to the recent paper [10] which look at similar concepts from the viewpoint of locality sensitive hashing.

A. Biryukov, G. Gong and D. R. Stinson (Eds.): SAC 2010, LNCS 6544, pp. 187–197, 2011
The original publication is available at http://dx.doi.org/10.1007/978-3-642-19574-7_13

The rest of the paper is organized as follows: In Section 2 we review some basic definitions and well known generic algorithms. Section 3 gives the main idea on how to use covering codes to find near-collisions for hash functions. Section 4 shows how to construct suitable codes for the method of Section 3. Section 5 finally presents the optimal solution to a problem that arises in Section 4 which asks how to construct a code of given length and covering radius that can be used to find near-collisions. Finally, we conclude in Section 6.

2 Collisions and Near-Collisions of Hash Functions

In all of the following, we will work with binary values, where we identify $\{0,1\}^n$ with $\mathbb{Z}_2^n$. Let "+" denote the n-bit exclusive-or operation. The Hamming weight of a vector $v \in \mathbb{Z}_2^n$ is denoted by $\mathrm{w}(v) = \#\{i \,|\, v_i = 1\}$ and the Hamming distance of two vectors by $\mathrm{d}(u,v) = \mathrm{w}(u+v)$. The Handbook of Applied Cryptography defines *near-collision resistance* as follows:

Definition 1 (Near-Collision Resistance [15, page 331]). *It should be hard to find any two inputs m, m^* with $m \neq m^*$ such that $H(m)$ and $H(m^*)$ differ in only a small number of bits:*

$$\mathrm{d}(H(m), H(m^*)) \leq \epsilon. \tag{1}$$

For ease of later use we also give the following definition:

Definition 2. *A message pair m, m^* with $m \neq m^*$ is called an ϵ-near-collision for H if (1) holds.*

The definitions suggest that a hash function for which an efficient algorithm is known that allows an attacker to construct near-collisions, can no longer be considered to be ideal. Above that, for several designs, near-collisions for the compression function can be converted to collisions for the hash function (*e.g.* [14,23]).

Collisions are of course a special case of near-collisions where the parameter $\epsilon = 0$. The generic method for finding collisions for a given hash function is based on the *birthday paradox* and attributed to Yuval [24]. This birthday attack requires approximately $2^{n/2}$ hash function computations and a table of the same size. In practice, memory is expensive relative to computation, and memoryless algorithms are given the preference over algorithms with large memory requirements. There are well established cycle-finding techniques (due to Floyd, Brent, Nivasch, *cf.* [2,12,17]) that remove the memory requirements from an attack based on the birthday paradox (see also [20]). These methods work by repeated iteration of the underlying hash function where in all of these applications the function is considered to behave like a random function.

Let S_ϵ be the set $\{x \in \mathbb{Z}_2^n \,|\, \mathrm{w}(x) \leq \epsilon\}$. An approach analogous to the classical birthday attack to find ϵ-near-collisions is to start with an empty table and randomly select messages m^j, compute $H(m^j)$ and check whether or not the entry $(H(m^j) + \delta, m^*)$ is present in the table, $\forall \delta \in S_\epsilon$ and an arbitrary m^* If no

match is found, then we add $(H(m^j), m^j)$ to the table and select a new message. Proceeding in this manner, the expected number of messages that we need to hash and store before we find an ϵ-near-collision is about:

$$\frac{2^{n/2}}{\sqrt{\sum_{i=0}^{\epsilon} \binom{n}{i}}}. \tag{2}$$

We see that, depending on ϵ, finding ϵ-near-collisions is clearly easier than finding collisions.

3 Memoryless Near-Collisions based on Coding Theory

In [13], the question is raised whether or not the above mentioned cycle-finding techniques are also applicable to the problem of finding near-collisions. We now give a short account of the ideas of [13].

Since Definition 2 includes collisions as well, the task of finding near-collisions is easier than finding collisions. The goal is now to find a generic method to construct near-collisions more efficiently than the generic methods to find collisions.

The first straight forward approach is to apply the cycle-finding algorithms to the problem of finding near-collisions is a projection based approach. The basic idea is to fix ϵ bit positions in $\mathbb{Z}_2^n$ and define the map π to set the bits of an $x \in \mathbb{Z}_2^n$ to zero at these ϵ positions. Then, we can apply a cycle-finding algorithm to the map $\pi \circ H$ which can find an ϵ-near-collision in a memoryless way with a complexity of about $2^{(n-\epsilon)/2}$. This can be even improved by setting $2\epsilon + 1$ bits to zero, since the probability is still $\frac{1}{2}$ that a collision for $\pi \circ H$ is an ϵ-near-collision, thus improving the complexity to about

$$2^{(n-1)/2-\epsilon}. \tag{3}$$

A drawback to this solution is of course that we can only find ϵ-near-collisions of a limited shape (depending on the fixed bit positions), so only a fraction of all possible ϵ-near-collisions can be detected, namely

$$\frac{2^{\epsilon}}{\sum_{i=0}^{\epsilon} \binom{n}{i}}. \tag{4}$$

To circumvent this drawback, there is the following nice solution. The idea is to replace the projection π by a more complicated function g. The choice for g is to be the decoding operation of a certain *covering code* $\mathcal{C}$. These codes are concerned with the so called *covering radius*

$$R(\mathcal{C}) = \max_{x \in \mathbb{Z}_2^n} \min_{c \in \mathcal{C}} \mathrm{d}(x, c)\,, \tag{5}$$

that is, R is the smallest radius such that the union of Hamming spheres $B_R(c)$ around the codewords of $\mathcal{C}$ cover the whole space $\mathbb{Z}_2^n$. If the minimum distance

between codewords is the parameter of importance, then we speak of error-correcting codes. Covering codes are a well researched topic in coding theory, see for example the monograph [4] for a thorough introduction.

The decoding function g of a covering code with covering radius R groups possible outputs of the hash function H into classes where for every two elements x, y in such a class we have $\mathrm{d}(x, y) \leq 2R$. Now instead of iterating the function on the output space of the hash function, one does so on the function over classes represented by their canonical members. If two different inputs are mapped into the same class, it corresponds to an ϵ-near-collision with $\epsilon = 2R$. In other words, we apply the cycle-finding algorithms mentioned in Section 2 to the function $g \circ H$ under the assumption that H acts as a random function. The main idea can thus be summarized as follows:

Theorem 1 ([13]). *Let H be a hash function of output size n. Let $\mathcal{C}$ be a covering code of the same length n, size K and covering radius $R(\mathcal{C})$ and assume there exists an efficiently computable map g satisfying*

$$\begin{aligned} g : \mathbb{Z}_2^n &\to \mathcal{C} \\ x &\mapsto c \quad \textit{with} \\ \mathrm{d}(x, c) &\leq R(\mathcal{C}). \end{aligned} \tag{6}$$

Then, we can find $2R(\mathcal{C})$-near-collisions for H with a complexity of about $\sqrt{K}$ and with virtually no memory requirements.

In order to be efficient, the task is thus to find a code $\mathcal{C}$ with K as small as possible. However, also the computability of the function g defined in (6) plays a crucial role. An evaluation of g should be efficient when compared to a hash function call. The actual task is thus to find a code $\mathcal{C}$ with given length n and covering radius R, such that the size of $\mathcal{C}$ is as small as possible and decoding can be done efficiently.

A general bound for the size of a covering code is the so called *Sphere Covering Bound*, which states that if a code $\mathcal{C}$ of length n and covering radius $R = R(\mathcal{C})$ exists, the size $K(n, R)$ must be

$$K(n, R) \geq \frac{2^n}{\sum_{i=0}^{R} \binom{n}{i}}. \tag{7}$$

An extensive amount of work in the theory of covering codes is devoted to derive better bounds and to construct codes achieving these bounds (*cf.* [4,19,21]).

4 Direct Sum Constructions of Covering Codes

In the rest of this paper, we will restrict ourselves to linear covering codes in $\mathbb{Z}_2^n$. We will use the notation $\mathcal{C} = [n, k]R$ for a code of length n, dimension k (that is, the size $K = 2^k$) and covering radius R. It is well known that *perfect codes* are good covering codes, non-trivial examples in the binary case are the Hamming

codes $\mathcal{H}_i$ for $i \geq 1$ which are $[2^i - 1, 2^i - 1 - i]1$ codes and the $[23, 12]3$ Golay code. For other lengths n, no non-trivial perfect codes exist.

Fortunately, the bitlength of a hash value is very often of the form $n = 2^i$. For these lengths, explicit and almost explicit results in the case when the covering radius is $R = 1$ and $R = 2$ are known:

Proposition 1 *Let $n = 2^i$ for $i \geq 1$ and let $k(n, R)$ be the minimum dimension k such that an $[n, k]R$ code exists. Then,*

$$k(2^i, 1) = 2^i - i \tag{8}$$

$$k(2^i, 2) \in \{2^i - 2i, 2^i - 2i + 1, 2^i - 2i + 2\}. \tag{9}$$

For a proof of (8) we refer to [21], (9) is shown in [11]. One basic construction principle when deriving such bounds is the so called *direct sum* of linear codes (*cf.* [11]):

Lemma 1 *For linear codes $\mathcal{C}_1 = [n_1, k_1]R_1$ and $\mathcal{C}_2 = [n_2, k_2]R_2$, the direct sum of $\mathcal{C}_1$ and $\mathcal{C}_2$ is defined as*

$$\mathcal{C}_1 \oplus \mathcal{C}_2 = \{(c_1, c_2) \,|\, c_1 \in \mathcal{C}_1, c_2 \in \mathcal{C}_2\}.$$

Then, $\mathcal{C}_1 \oplus \mathcal{C}_2$ is a linear code with parameters $[n_1 + n_2, k_1 + k_2]R_1 + R_2$.

For (linear) codes of length $n \leq 33$ and for relatively small covering radii R we can find extensive tables of the values of $k(n, R)$ (see *e.g.* [4] or [9]). However, since the length of the code is determined by the output size of the hash function, we will deal with lengths n that are significantly larger than 33.

In [13], a direct sum construction was proposed that would be applicable for all lengths n and covering radii R and only combines Hamming codes and the trivial codes $\mathbb{Z}_2^q$. This construction looks as follows. Let the numbers $N_i = 2^i - 1$ denote the lengths of the Hamming codes $\mathcal{H}_i$ for $i = 1, 2, \ldots$. Let $\mathcal{D} = \{0, 1, \ldots, R\}$ be the set of digits. We denote by

$$\mathcal{X} := \left\{ x = \sum\nolimits_{i \geq 1} d_i N_i \,|\, d_i \in \mathcal{D}, d_i \neq 0 \text{ for finitely many } i \right\} \tag{10}$$

the set of possible expansions in the base $(N_i)_{i \geq 1}$ and with digits in $\mathcal{D}$. Furthermore, we also introduce

$$\mathcal{X}_n := \{x \in \mathcal{X} \,|\, x \leq n\}. \tag{11}$$

For ease of notation, we will denote by $d \cdot \mathcal{H}_i$ the direct sum of d copies of $\mathcal{H}_i$. Then, the following optimization problem can be formulated:

Problem 1 *Let n and $R < \lfloor \frac{n}{4} \rfloor$ be given. Find an expansion*

$$x = \sum\nolimits_{i \geq 1} d_i N_i \in \mathcal{X}_n \tag{12}$$

that additionally satisfies the following properties:

$$\sigma(x) := \sum\nolimits_{i\geq 1} d_i = R, \tag{13}$$

$$e(x) := \sum\nolimits_{i\geq 1} d_i \cdot i \rightarrow \ max. \tag{14}$$

Then, the code

$$\mathcal{C} = \bigoplus_{i\geq 1} d_i \cdot \mathcal{H}_i \ \oplus \mathbb{Z}_2^{n-x} \tag{15}$$

has length n, covering radius R and the dimension of the code is

$$k = n - \sum_{i\geq 1} d_i \cdot i\,.$$

From Lemma 1 it follows that the code (15) has length $x + (n - x) = n$, and that the covering radius is exactly R. For the dimension of the code we get

$$\begin{aligned} k &= \sum\nolimits_{i\geq 1} d_i(N_i - i) + n - \sum\nolimits_{i\geq 1} d_i N_i \\ &= n - \sum\nolimits_{i\geq 1} d_i \cdot i. \end{aligned}$$

Clearly, the dimension is minimal if $\sum_{i\geq 1} d_i \cdot i$ is maximal.

5 The Optimal Solution for Problem 1

We now give a complete solution of the optimization question formulated in Problem 1. We start by stating the main theorem.

Theorem 2. *Let n, R be given as in Problem 1. Define*

$$\ell := \left\lfloor \log_2\left(\frac{n}{R} + 1\right)\right\rfloor \quad \text{and} \quad r := \left\lfloor \frac{n - R(2^\ell - 1)}{2^\ell}\right\rfloor. \tag{16}$$

Then, the expansion

$$x = (R - r)(2^\ell - 1) + r(2^{\ell+1} - 1) \tag{17}$$

satisfies all conditions (12), (13) *and* (14). *The resulting code*

$$\mathcal{C} = (R - r) \cdot \mathcal{H}_\ell \ \oplus r \cdot \mathcal{H}_{\ell+1} \ \oplus \mathbb{Z}_2^{n-x} \tag{18}$$

has dimension $k = n - R \cdot \ell - r$ and is an optimal solution subject to this construction.

The proof of Theorem 2 is split into two auxiliary results for the expansions $x \in \mathcal{X}_n$ which satisfy (13) and (14) which will be shown in Proposition 2 and Proposition 3. Remember that for a given expansion $x \in \mathcal{X}_n$ we denote by $e(x) = \sum_{i\geq 1} d_i \cdot i$ the value of the expansion, which has to be maximized.

Proposition 2 *Assume $x^* \in \mathcal{X}_n$ is an optimal expansion satisfying* (13) *and* (14)*. Then we have*

$$R\,\ell \leq e(x^*) < R(\ell+1). \tag{19}$$

Proof. The left hand side of the inequality is easy to see, since from (16) we know that ℓ is chosen in such a way that

$$R(2^\ell - 1) \leq n < R(2^{\ell+1} - 1).$$

So $x = R(2^\ell - 1) \in \mathcal{X}_n$ is a valid expansion and $e(x) = R\,\ell$.

For the right hand side, we assume there is an expansion $x^* \in \mathcal{X}_n$ with $e(x^*) \geq R(\ell+1)$, that is, we have

$$\sum_{i\geq 1} d_i(2^i - 1) \leq n, \tag{20}$$

$$\sum_{i\geq 1} d_i = R, \tag{21}$$

$$\sum_{i\geq 1} d_i \cdot i \geq R(\ell+1). \tag{22}$$

Since $d_i \geq 0$ and because of (21) we can interpret the sequence

$$\left(\frac{d_1}{R}, \frac{d_2}{R}, \dots\right)$$

as a discrete probability distribution for a random variable X. In other words, we consider X with $\mathbb{P}(X = i) = \frac{d_i}{R}$ for $i \geq 1$. In this light, we can read (22) as an inequality for the expected value $\mathbb{E}(X) \geq \ell + 1$. Now Jensen's inequality (cf. [8]) states, that for a convex function ϕ we have

$$\mathbb{E}(\phi(X)) \geq \phi\left(\mathbb{E}(X)\right). \tag{23}$$

Applying this to the random variable X from above and the convex function $\phi(x) = 2^x - 1$ we can derive

$$\sum\nolimits_{i\geq 1} \tfrac{d_i}{R}(2^i - 1) = \mathbb{E}(\phi(X)) \geq \phi(\mathbb{E}(X)) \geq \phi(\ell+1) = 2^{\ell+1} - 1.$$

After multiplying this inequality by R, we end up with

$$x^* \geq R(2^{\ell+1} - 1) > n,$$

by the definition of ℓ. This contradiction proves the proposition. ∎

The last proposition specifies the interval in which $e(x^*)$ must lie for an optimal solution x^*. The next proposition will give the explicit solution.

Proposition 3 *For given n and R, the expansion*

$$x^* = (R-r)(2^\ell - 1) + r(2^{\ell+1} - 1)$$

with ℓ and r as in (16) *is optimal, that is, it reaches the maximal value $e(x^*) = R\ell + r$. Depending on n and R, this optimum can also be attained by other expansions than x^*.*

Proof. We want to address first, that in general there is not a unique optimal expansion. This is easy to see since for certain values of n and R it might occur, that

$$x^{**} = (2^{\ell-1} - 1) + (R - r - 2)(2^\ell - 1) + (r+1)(2^{\ell+1} - 1) \leq n$$

holds. This expansion also has

$$e(x^{**}) = \ell - 1 + (R - r - 2)\ell + (r+1)(\ell+1) = R\ell + r.$$

For example in the case $n = 160$ and $R = 2$ we would have $x^* = 2(2^6 - 1) = 126$ and $x^{**} = (2^5 - 1) + (2^7 - 1) = 158$ which both are ≤ 160 and have $e(x^*) = e(x^{**}) = 12$.

By definition in (16), we see that $r \in \{0, \ldots, R-1\}$. Actually, r is chosen in such a way that

$$(R-r)(2^\ell - 1) + r(2^{\ell+1} - 1) \leq n < (R-r-1)(2^\ell - 1) + (r+1)(2^{\ell+1} - 1). \quad (24)$$

Thus, Proposition 2 states that $R\ell + r$ would be a possible optimal value for $e(x^*)$. Let us now assume that there exists an expansion $x' \in \mathcal{X}_n$ such that $e(x') > R\ell + r$. Let $e(x') = R\ell + r + \delta$ with $0 < \delta < R - r$. Then, another expansion x'' having $e(x'') = e(x')$ is

$$x'' = (R - r - \delta)(2^\ell - 1) + (r + \delta)(2^{\ell+1} - 1). \quad (25)$$

Now we will show that any x' with $e(x') = e(x'') = R\ell + r + \delta$ satisfies $x' \geq x''$.

Let us consider the digits of the expansion (25) as units, that is, we have a total of R digits (because of (21)) and these digits are distributed at positions ℓ and $\ell + 1$. Any other x' satisfying $e(x') = e(x'')$ can be seen to result from x'' by moving the digits of the expansion of x'' to other positions.

To be even more specific we can describe a "unit move" by S^i_j which denotes the process of moving one digit of x'' at position i to $i+1$ and simultanously moving one digit from position j to $j-1$ (see also Fig. 1). Any such step has the property of maintaining the value $e(x'')$. Starting from x'', i and j can only be ℓ or $\ell + 1$. In this case only the moves S^ℓ_ℓ, $S^{\ell+1}_\ell$ and $S^{\ell+1}_{\ell+1}$ make sense, since $S^\ell_{\ell+1}$ would be redundant by leaving x'' unchanged. In general, we only have to consider moves S^i_j with $i \geq j$ since any move with $i < j$ results in a previous configuration of digits. (This can be shown by induction over the number of non-zero digits of a given expansion in $\mathcal{X}$.) When applying a unit move S^i_j with $i \geq j$ to x'', the digit going from i to $i+1$ increases the value of the resulting

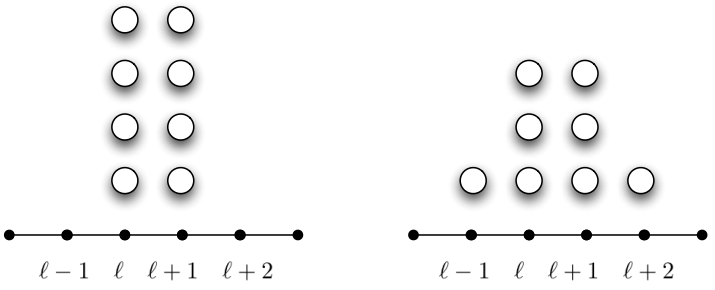

Fig. 1. Example of the unit move $S_\ell^{\ell+1}$ with $R = 8$

expansion x' by $2^{i+1} - 1 - 2^i + 1 = 2^i$ whereas the digit going from j to $j-1$ decreases the value of x' by 2^j. Therfore, the overall change of one unit move is $2^i - 2^j$ which is always non-negative, since we assumed $i \geq j$. Since any x' with $e(x') = e(x'')$ results from x'' by a series of unit moves S_j^i with $i \geq j$, we can therefore deduce that we have $x' \geq x''$. But $x' \geq x''$ also implies $x' > n$ because

$$x'' \geq (R - r - 1)(2^\ell - 1) + (r + 1)(2^{\ell+1} - 1) > n$$

by (24) and since $\delta > 0$. This proves the proposition. ∎

We conclude this section by giving a table which compares the dimension of the codes (18) with the projection based approach and the complexity of a table-based near-collision search for several values of ϵ and n.

Table 1. For given $\epsilon \in \{2, 4, 6, 8\}$, the table compares the base-2 logarithms of the complexity of the standard table-based approach (2), the projection based approach (3) and our construction (15) for $n = 128, 160$ and 512.

	$n = 128$			$n = 160$			$n = 512$		
ϵ	(2)	(3)	(15)	(2)	(3)	(15)	(2)	(3)	(15)
2	57.5	61.5	60.5	73.2	77.5	76.5	247.5	253.5	251.5
4	52.3	59.5	58.0	67.7	75.5	74.0	240.3	251.5	248.0
6	47.8	57.5	56.0	62.8	73.5	71.5	233.8	249.5	245.0
8	43.8	55.5	54.0	58.5	71.5	69.5	227.7	247.5	242.0

Remark 1. It is of course natural to compare the codes resulting from Theorem 2 with other well known codes. Binary BCH codes with parameters e, m (see [4, Sect. 10.1]) are algebraic codes of length $n = 2^m - 1$, dimension $k \geq n - m \cdot e$ and minimum distance $d \geq 2e + 1$. For $e \in \{2, 3\}$ the exact covering radius of $BCH(e, m)$ is known, namely $R(BCH(2, m)) = 3$ and $R(BCH(3, m)) = 5$. If

we now consider for example $\mathcal{B}_1 = BCH(2,7) \oplus \mathbb{Z}_2$ and $\mathcal{B}_2 = BCH(2,9) \oplus \mathbb{Z}_2$, we see that $\mathcal{B}_1$ is a $[128, 114]$ $R = 3$ code and $\mathcal{B}_2$ has parameters $[512, 494]$ $R = 5$, so their dimension is higher than that of (15).
One possible way to achieve an improvement is to move from the direct sum construction of Section 4 to blockwise or amalgamated direct sum constructions (cf. [4]).

6 Conclusion

In this paper, we have solved a problem concerning the direct sum of Hamming codes and trivial codes $\mathbb{Z}_2^q$. This problem arose in the context of a newly proposed method that allows us to find near-collisions for a cryptographic hash function H in a memoryless way by applying the standard cycle-finding algorithms to the composition of the decoding operation of a covering code and the hash function H. This method is then able to find ϵ-near-collisions for H with $\epsilon = 2R$ where R is the covering radius of the underlying code. The efficiency of the method is determined by the size of this code. The question of finding the right combination of Hamming and trivial $\mathbb{Z}_2^q$ codes has been translated into an optimization problem for digital expansions and this has been solved by Theorem 2.

Acknowledgements

The authors wish to thank the anonymous referees, Florian Mendel and René Struik for valuable comments and discussions. The work in this paper has been supported in part by the European Commission under contract ICT-2007-216646 (ECRYPT II), in part by the Austrian Science Fund (FWF), project P21936 and in part by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy).

References

1. E. Biham and R. Chen. Near-Collisions of SHA-0. In M. K. Franklin, editor, *CRYPTO*, volume 3152 of *LNCS*, pages 290–305. Springer, 2004.
2. R. P. Brent. An improved Monte Carlo factorization algorithm. *BIT*, 20(2):176–184, 1980.
3. F. Chabaud and A. Joux. Differential Collisions in SHA-0. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *LNCS*, pages 56–71. Springer, 1998.
4. G. Cohen, I. Honkala, S. Litsyn, and A. Lobstein. *Covering codes*, volume 54 of *North-Holland Mathematical Library*. North-Holland Publishing Co., Amsterdam, 1997.
5. C. De Cannière, F. Mendel, and C. Rechberger. Collisions for 70-Step SHA-1: On the Full Cost of Collision Search. In C. M. Adams, A. Miri, and M. J. Wiener, editors, *Selected Areas in Cryptography*, volume 4876 of *LNCS*, pages 56–73. Springer, 2007.

6. C. De Cannière and C. Rechberger. Finding SHA-1 Characteristics: General Results and Applications. In X. Lai and K. Chen, editors, *ASIACRYPT*, volume 4284 of *LNCS*, pages 1–20. Springer, 2006.
7. H. Dobbertin. Cryptanalysis of MD4. *J. Cryptology*, 11(4):253–271, 1998.
8. W. Feller. *An introduction to probability theory and its applications. Vol. I.* Third edition. John Wiley & Sons Inc., New York, 1968.
9. Gerzson Kéri. Tables for bounds on covering codes. `http://www.sztaki.hu/~keri/codes/`. accessed 2010/05/17.
10. D. Gordon, V. Miller, and P. Ostapenko. Optimal hash functions for approximate matches on the n-cube. *IEEE Trans. Inform. Theory*, 56(3):984 –991, 2010.
11. R. L. Graham and N. J. A. Sloane. On the covering radius of codes. *IEEE Trans. Inform. Theory*, 31(3):385–401, 1985.
12. D. E. Knuth. *The art of computer programming. Vol. 2.* Addison-Wesley Publishing Co., Reading, Mass., third edition, 1997. Seminumerical algorithms, Addison-Wesley Series in Computer Science and Information Processing.
13. M. Lamberger, F. Mendel, V. Rijmen, and K. Simoens. Memoryless Near-Collisions via Coding Theory. `http://asiacrypt2009.cipher.risk.tsukuba.ac.jp/rump/slides/13_NC-talk.pdf`, Dec. 2009. (short talk) presented at the ASIACRYPT 2009 rump session.
14. F. Mendel and M. Schläffer. On Free-Start Collisions and Collisions for TIB3. In P. Samarati, M. Yung, F. Martinelli, and C. A. Ardagna, editors, *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009*, volume 5735 of *LNCS*, pages 95–106. Springer, 2009.
15. A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1996.
16. National Institute of Standards and Technology (NIST). Cryptographic Hash Project, 2007. `http://www.nist.gov/hash-competition`.
17. G. Nivasch. Cycle detection using a stack. *Inf. Process. Lett.*, 90(3):135–140, 2004.
18. N. Pramstaller, C. Rechberger, and V. Rijmen. Exploiting Coding Theory for Collision Attacks on SHA-1. In N. P. Smart, editor, *IMA Int. Conf.*, volume 3796 of *LNCS*, pages 78–95. Springer, 2005.
19. R. Struik. An improvement of the Van Wee bound for binary linear covering codes. *IEEE Transactions on Information Theory*, 40(4):1280–1284, 1994.
20. P. C. van Oorschot and M. J. Wiener. Parallel Collision Search with Cryptanalytic Applications. *J. Cryptology*, 12(1):1–28, 1999.
21. G. J. M. van Wee. Improved sphere bounds on the covering radius of codes. *IEEE Transactions on Information Theory*, 34(2):237–245, 1988.
22. X. Wang, Y. L. Yin, and H. Yu. Finding Collisions in the Full SHA-1. In V. Shoup, editor, *CRYPTO*, volume 3621 of *LNCS*, pages 17–36. Springer, 2005.
23. X. Wang and H. Yu. How to Break MD5 and Other Hash Functions. In R. Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 19–35. Springer, 2005.
24. G. Yuval. How to swindle Rabin? *Cryptologia*, 3(3):187–191, 1979.