Tatra Mt. Math. Publ. **47** (2010), 67–80
DOI: 10.2478/v10127-010-0031-8

TATRA MOUNTAINS
Mathematical Publications

REMARKS ON GÖDEL'S CODE AS A HASH FUNCTION

Michal Mikuš* — Petr Savicky**

ABSTRACT. In this paper we analyze a simple hash function introduced in a popular book *PopCo* by Scarlett Thomas that is based on well known Gödel's numbering function. The numbering function is very efficient for practical use, however it is widely used in foundations of logic and computability theory. We show that the properties of the suggested hash function (computing the hash as a "shorter digest" of the *long* Gödel's number code) are not sufficient for cryptography. We introduce two ways how to construct meaningful collisions and in special cases also second-preimages. Further we propose a simple improvement of this hash function which prevents the simplest of the attacks, however it does not eliminate the other attacks.

1. Introduction

It is not a very rare occasion to glimpse a piece of mathematics in a popular film or book. Sometimes the piece is so twisted that it hardly resembles the truth, but it was not the case with the book *PopCo* by S. Thomas, where a function used as a checksum of secret message was defined. The function was a hash function in the general sense, e.g., from [2] and in this paper we examine its cryptographical properties.

Although the function cannot meet the criteria for effectiveness and we show it actually has no collision resistance and it can still be considered a good classroom example of a simple hash function.

2010 Mathematics Subject Classification: 14G50, 94A60.
Keywords: Gödel numbering function, hash function, integer relation algorithm, rational reconstruction.
* This material is based upon the work supported under the grant VEGA 1/3115/06 and the grant NIL-I-004 from Iceland, Lichtenstein and Norway through the EEA Financial Mechanism and the Norwegian Financial Mechanism.
** The second author acknowledges the support of the Grant Agency of the Czech Republic under the grant number P202/10/1333 and by Institutional Research Plan AV0Z10300504.

2. Gödel's hash function

According to the text in *PopCo*, we provide a formal definition of the hash function.

Definition 2.1. Let Σ be an alphabet and let $\phi : \Sigma \to \{1, \ldots, |\Sigma|\}$ represent an ordering of symbols of Σ.

We consider the regular 26-letter lower case English alphabet without space as Σ. Every message is converted into this alphabet before computing its hash code. We define ordering ϕ of the symbols (letters) of the underlying alphabet according to their frequencies so that the most frequent symbol gets number 1 and the least frequent symbol gets number 26. Arbitrary alphabets can be used instead, however, the ordering of the symbols has to be defined.

Definition 2.2. Let $M \in \Sigma^*$ be an arbitrary message. Denote the length of M by L and let $\{p_i\}_{i=1}^{L}$ be a list of different prime numbers. We define code of M as

$$c(M) = \prod_{i=1}^{L} p_i^{\phi(a_i)},$$

where a_i is ith prime and a_i is ith letter of M. The hash code $h(M)$ of M then consists of n most significant digits of $c(M)$.

The original hash function uses the list of the first L primes as $\{p_i\}_{i=1}^{L}$. We also investigate a variant, where $\{p_i\}_{i=1}^{L}$ is the list of the first L odd primes.

Remark 2.3. The code of the message is the same as the standard Gödel's numbering function. Note that the ordering of symbols according to frequency (the smallest exponents for most frequent symbols) leads to the lowest computational complexity in typical cases.

The parameter n represents the length of the hash code. It is easy to see that a longer parameter leads to a greater complexity of cryptanalysis and therefore to better resistance of the hash function. Since this function is too slow for use in practice and its properties are mainly of academic importance, short digests are sufficient for the presentation of the analyzed properties. Therefore we restrict ourselves therefore to the same length as suggested by S. Thomas in [1] $n = 10$.

Example 2.4. Let us compute the digest of a short message using the parameters above.

Frequency table (English, decreasing frequency):

e t a o i n s r h d l u c m f y w g p b v k x q j z [1]

Message:

$M =$ 'articles'.

Compute the code:

$$\begin{aligned} c(M) &= 2^3 \times 3^8 \times 5^2 \times 7^5 \times 11^{13} \times 13^{11} \times 17^1 \times 19^7 \\ &= 207346230290196365983579463986334514327755094 00, \\ h(M) &= 2073462302. \end{aligned}$$

Computing the frequency table depends on a choice of a sample of texts and so the frequency tables from different sources may differ slightly.

2.1. Implementation issues

As the original function in [1] has used but a pocket-calculator for the computation of the hash, there were essentially two options how to implement this function namely either to compute the precise result using the long arithmetic and cut n most significant digits at the end, or to simulate the pocket calculator and its precision. We assumed that the old calculator had the display limited to 10 decimal digits and the arithmetic precision limited to 20 digits, that means, the least significant digits could have been discarded after each operation.

Both methods were implemented and they typically provide the same hash code using reasonable parameters. Let $c(M)$ be the product defined above and let $c'(M)$ be the same product computed using floating point arithmetic with k significant decimal digits with rounding the middle cases to an even last digit. Using the analysis from [10] for a sequence of multiplications and computing a power of exact numbers, we obtain the following bound for the relative error

$$\left| \ln \frac{c(M)}{c'(M)} \right| < 2L\,\epsilon\ ,$$

where $\epsilon = 5 \cdot 10^{-k}$. In our experiments, the actual error was bounded by $\sqrt{2L}\,\epsilon$. Although this error estimate cannot be rigorously proven, it may be explained by the stochastic error analysis, see [11], which seems to be suitable for our simple situation, if $L \ll 10^k$.

If the error is significantly smaller than 10^{-n}, we typically get the correct $h(M)$ using $c'(M)$ instead of $c(M)$. For example, if $k = 17$ and $L = 7000$, then

$$\sqrt{2L}\,\epsilon \leq 6 \cdot 10^{-15}$$

and we typically obtain the correct first 10 digits. On the other hand, an error cannot be excluded. The number $c(M)$ may be close to the numbers between

[1] from http://www.math.cornell.edu/∼mec/2003-2004/cryptography/subs/frequencies.html.

which the nth decimal digit changes. Then, even if the relative error of $c'(M)$ is much less than 10^{-n}, the nth most significant digits of $c(M)$ and $c'(M)$ may differ.

3. Cryptanalysis

In this section we provide cryptanalysis of the hash function from Definition 2.2. Namely, we describe a way how to create second preimages and meaningful colliding messages. We also consider a related problem of preimage resistance.

3.1. Collision search

Collision resistance is the most important property of hash functions. It is also a property that requires the least effort to break—thanks to the birthday paradox [3].

Definition 3.1. For two arbitrary messages

$$M = a_1 a_2 \dots a_{L_1} \quad \text{and} \quad M_2 = b_1 b_2 \dots b_{L_2}$$

let us define the vector

$$(c_1, c_2, \dots, c_L) \text{ by } c_i = \phi(b_i) - \phi(a_i), \text{ where } L = \max(L_1, L_2).$$

If one message is shorter, we append necessary number of empty characters ε at the end of this message and define $\phi(\varepsilon) = 0$. We will call the vector $(c_1, c_2, \dots, c_L)$ as difference vector of M_2 from M.

Remark 3.2. Note that for English letters we have $c_i \in \{-26, \dots, 26\}$.

3.1.1. Simple collisions

In the following, we will describe the first of the devised attacks for finding collisions of the analyzed hash function. It is based on the fact that the result is computed in decimal representation and that 10 is product of $p_1 = 2$ and $p_3 = 5$. This means every message with the difference vector equal to $(a, 0, a, 0, 0 \dots, 0)$ will result to 10^a times greater code and exactly the same hash code providing that the original code was greater than 10^n.

Some examples of such collisions are:

`bags - paws`, `bait - tact`, `dust - hunt`, `banana - havana`, `backed - masked`, etc.

For a sufficiently long code one has to append the same suffix to these words.

This fact leads to an extremely simple way of constructing collisions; however, it can be also easily prevented. By omitting 2 from the prime sequence we will

be unable to get message codes multiplied by 10 and the computation of the hash code will be only slightly different with no significant impact on the time complexity.

3.1.2. Another type of collisions

Here we describe the more general method of constructing colliding messages for the considered hash function. This method applies also to the case, when only odd primes are used, as suggested in Section 3.1.1. Let M be an arbitrary message with code $c(M)$. Our algorithm is based on a simple observation that if we create a new message with the difference vector $(0, \ldots, 0, 1, -1, 0, \ldots, 0)$, the resulting code will be [2]

$$c(M) \times p_i/p_{i+1}.$$

The ratio of these hash codes is p_i/p_{i+1} and if the ratio lies in the interval $[1 - 10^{-n}, 1 + 10^{-n}]$, then the codes are likely to have the same digits on the n most significant places. In such a case, the hash codes will be equal and that means a collision. We will investigate the difference vectors, which represent a potential collision in this sense. Although the condition that the ratio is in the interval $[1 - 10^{-n}, 1 + 10^{-n}]$ is only a necessary condition for a collision, we will call such difference vectors a collision for simplicity. The number of nonzero components of a difference vector, which is the number of changed positions, will be called the length of the collision.

The simplest way how to get such a ratio is to select two primes with such property, however, as this means

$$p_i > 2 \cdot 10^n,$$

it would result in an extremely long message for an even 10-digit digest. Another way is to combine more positions in the difference vector and search for a ratio close to 1. We suggest the following procedure for this purpose.

Algorithm 1

1. Select T, the length of the lists of numbers to be sorted in each iteration.
2. Select K and L — the length of message (equal to the length of the sequence of primes available).
3. Compute at least T ratios of the form $(p_{i+j}/p_i)^k$, where $1 \leq j \leq K$, $4 \leq i \leq L - j$ and $1 \leq k \leq K/j$ and form a list of the obtained ratios together with the corresponding primes and exponents.
4. Repeat the following steps 5 to 8 until a required number of collisions is found.

[2] Similarly the difference vector $(0, \ldots, 0, a, -a, 0, \ldots, 0)$ corresponds to the ratio $(p_i/p_{i+1})^a$ between the codes.

5. Sort the ratios in ascending order. If there are pairs of ratios, whose difference is of the order of the rounding error, compare their vectors of primes and exponents and if they are the same, remove the repeated occurrences of the same number.
6. If there is a ratio in the interval $[1, 1 + 10^{-n}]$, then store the primes and exponents (collision found).
7. Choose $\delta > 0$, which is as small as possible among those, which lead to at least T different ratios in the next step. See the calculation below, which suggests a value of δ to be chosen.
8. Divide the ratios in the list into groups defined by intervals between the numbers of the form $1 + \delta i$, where $i \geq 0$ is an integer. If $\delta = 10^{-d+1}$, then the groups consist of the numbers with equal decimal expansion up to d digits. For each group, compute the ratio between all pairs of numbers in this group, which are larger than 1. For each pair of consecutive groups, compute the ratio of each number in the group containing larger numbers divided by each number in the group containing smaller numbers. Form a list of all the obtained ratios together with the corresponding primes and exponents as the input for the next iteration of the step 5.

The algorithm needs parameters L, K and T. The number of ratios in the first iteration will be at most

$$(L-4)\sum_{j=1}^{K}\lfloor K/j\rfloor.$$

Consequently, for the initialization of the algorithm, we need

$$(L-4)\sum_{j=1}^{K}\lfloor K/j\rfloor \geq T.$$

The number T influences the complexity of each iteration, since we have to sort a list of at least T numbers. On the other hand, the larger T is used, the smaller number of iterations will be needed and, consequently, shorter collisions will be found.

The list of ratios produced in the step 8 may contain ratios, which are exactly equal to some other ratios in the list and will be eliminated in the next iteration of the step 5. In order to obtain a feasible value of δ, we first choose a value δ as described in the next two paragraphs. This value guarantees that the step 8 produces at least T ratios disregarding the question, whether they are equal or not. Then, if the actual number of different ratios was lower than T, we heuristically increased δ so that the number of different ratios is at least T. The number of repeated ratios was quite low, so this approach was sufficiently efficient.

Assume, the list of ratios, which is the input to the step 7, has length T' and δ is such that the list is split into g groups. If s_i is the size of ith group, the number of all computed ratios is

$$\sum_{i=1}^{g-1} s_i s_{i+1} + \sum_{i=1}^{g} \binom{s_i}{2} \geq \sum_{i=1}^{g} \binom{s_i}{2}.$$

Since $\binom{s}{2}$ is a convex function of a real variable s and

$$\frac{1}{g} \sum_{i=1}^{g} s_i = \frac{T'}{g},$$

we have

$$\frac{1}{g} \sum_{i=1}^{g} \binom{s_i}{2} \geq \binom{T'/g}{2}.$$

Consequently, the number of ratios obtained in the step 8 is at least

$$g\binom{T'/g}{2} = \frac{1}{2} T' \left(\frac{T'}{g} - 1 \right).$$

This is at least T, if

$$g \leq \frac{(T')^2}{2T + T'}. \tag{3.1}$$

Let the largest ratio in the list be $1+\varepsilon$. In order to split the list into g groups, we may choose $\delta = \varepsilon/g$, where g is the largest integer, which satisfies (3.1). The new ratios obtained for the next iteration belong to the interval $[1, 1+\delta]$. Since $\delta = \varepsilon/g$, the number of groups g controls how close to 1 are the ratios in the next iteration. The upper bound on g given by (3.1) is most restrictive, if $T' = T$, when (3.1) is equivalent to $g \leq T/3$. Hence, if we want to have, for example, $g \geq 10$, we need $T \geq 30$. In this case, the precision of the collisions obtained from the ratios in the list increases at least by one digit in each iteration. Since sorting T numbers is an efficient operation also for lists of larger size, it is better to set T to a larger number. For example, if we set $T = 3 \cdot 10^5$, we may choose $g = 10^5$ and the precision increases at least by 5 digits in each iteration. Since each iteration doubles the length of the candidate collisions, smaller number of iterations leads to shorter collisions.

Example 3.3. Let us assume that for $n = 10$ we set $T = 1200$ and $L = 450$, $K = 3$. We used 1300 ratios with $j = 1$ at the beginning (the ratios are of the form $(p_{i+1}/p_i)^k$). The following were the smallest ratios we obtained after the step 5:

Ratio	Primes	Exponents
1.00063151247237	3169, 3167	1, -1
1.00064123116383	3121, 3119	1, -1
1.00066688896299	3001, 2999	1, -1

Since there were no collisions, we used $\delta = 10^{-3}$ to group the ratios according to 4 decimal places [3]. As the distribution of the ratios was not uniform, we obtained approx. 9500 ratios after the steps 7 and 8, all in interval $[1, 1.0002)$.

The following ratios are the first four ratios obtained after sorting in the second iteration of the step 5:

Ratio	Primes	Exponents
1.00000000007863	2113, 2111, 3169, 3167	2, -2, -3, 3
1.00000000025439	1429, 1427, 2143, 2141	2, -2, -3, 3
1.00000000094084	811, 809, 1621, 1619	1, -1, -2, 2
1.00000000173916	661, 659, 1321, 1319	1, -1, -2, 2

The first row actually has a ratio smaller than $1 + 10^{-10}$ so we found a pattern for collision which indicates four places where to change the symbols of the message by ± 2 and ± 3. Next, we selected precision 7 decimal places ($\delta = 10^{-6}$) and repeated the steps 7 and 8. After sorting the 82000 ratios, we found 59 more collision patterns which required 8 places for change in the message. The following table shows the first three of them:

Ratio	Primes	Exponents
1+1.27e–12	2647, 2633, 769, 761, 3191, 3187, 1663, 1657	2, -2, -1, 1, -3, 3, 1, -1
1+1.38e–12	1879, 1877, 1301, 1297, 2153, 2143, 1307, 1303	3, -3, -1, 1, -2, 2, 3, -3
1+2.94e–12	1613, 1609, 859, 857, 1933, 1931, 2087, 2083	1, -1, -1, 1, -2, 2, 1, -1

In the next iteration we would find more collision patterns, but we stopped the process here. It should be noted, that the goal of Algorithm 1 is to find quickly some collision patterns and not to find all collisions that can be found. Its running time is very small, with these settings it took negligible time to compute the above collisions.

Meaningful collisions and second-preimages

Every collision pattern found by our algorithm is independent of the message. It only selects a few places to change in an arbitrary message so that the new message will likely have the same hash code. This gives us a straightforward

[3] So the above three ratios were in the first group.

method to construct second-preimages. Unfortunately, these second preimages will probably have no meaning—we cannot change the text on a few places and expect it to be still meaningful.

For collision search, however, we can influence both of the messages. We can use a simple dictionary to search for words that can be changed on subsequent positions by $(+k, -k)$ and still have a similar meaning. Then these words can be placed to the positions corresponding to the collision pattern and the rest of the message can be filled up so that the entire text is meaningful. We demonstrate this method on the following examples:
An example of collision corresponding to the simplest collision pattern:

Ratio	Primes	Exponents
1.00000000007863	2113, 2111, 3169, 3167	2, -2, -3, 3

The changed positions are:

$$317\text{th by } -2, \quad 318\text{th by } +2, \quad 447\text{th by } +3 \quad \text{and} \quad 448\text{th by } -3.$$

Message no. 1:
An influx of party drugs has killed two teenagers but she has a suspect handsome rich newcomer a published author seeking smalltown atmosphere To build her case she moves closer to Mac She risks everything for her investigation even when it means letting her guard down and falling for her suspect Mac Callan lives and breathes for undercover work She still hears his words said in a ***wa****rm tone innocent His last mission ended in near disaster and he has chance to prove his value She decided and started to pack her things her movements were* ***so****ft and quick*

Message no. 2:
An influx of party drugs has killed two teenagers but she has a suspect handsome rich newcomer a published author seeking smalltown atmosphere To build her case she moves closer to Mac She risks everything for her investigation even when it means letting her guard down and falling for her suspect Mac Callan lives and breathes for undercover work She still hears his words said in a ***fi****rm tone innocent His last mission ended in near disaster and he has chance to prove his value She decided and started to pack her things Her movements were* ***de****ft and quick*

We show also an example of collision corresponding to a more complicated collision pattern:

Ratio	Primes	Exponents
1.00000000001781	463, 461, 2789, 2777, 3001, 2999, 3041, 3037	1,-1,-1, 1,-2, 2, 1,-1

The numbers of changed positions are: 88, 89, 403, 404, 429, 430, 434, 435.

Message no. 1:
Ten people working at CGE went down in the plane in the Atlantic nine of them were accompanied by a child or ***fa****ther the bitter outcome was a consequence of one of the normally happy rituals of corporate life they were on the flight as a reward for winning a competition for top sales performance a linguet had been at cge for ten years linguets career started in banks moving between strasbourg and the paris region before joining the company our family suffered a fatal blow said c linguet* ***mo****ther of two very well known ba****nk****ers* ***fa****ther of those young men ...*

Message no. 2:
Ten people working at CGE went down in the plane in the Atlantic nine of them were accompanied by a child or ***mo****ther the bitter outcome was a consequence of one of the normally happy rituals of corporate life they were on the flight as a reward for winning a competition for top sales performance a linguet had been at cge for ten years linguets career started in banks moving between strasbourg and the paris region before joining the company our family suffered a fatal blow said c linguet* ***fa****ther of two very well known ba****rb****ers* ***mo****ther of those young men ...*

3.1.3. Modified search for collisions using sorting

Algoritm 1 for searching collisions described in Section 3.1.2 searches collisions of restricted form. In particular, it searches only collisions, which may be obtained as a ratio between two approximate collisions from the previous iteration. In this section, we describe a variant of Algorithm 1, which constructs an initial list of products of primes as large as possible given the available computing power, sorts the list and looks for close numbers in this sorted list. The initial list of products is constructed with no restriction on the values of these products.

The input parameters of the following Algorithm 2 are L, K and B, where L is the maximal length of the message, K is the maximal exponent (in absolute value) and $2B$ is the maximal length of a collision.

Algorithm 2

1. Let A be the set of vectors of the length B of exponents defined as a difference of two Cartesian products $A = \{-K, \ldots, K\}^B \setminus (\{0\} \times \{-K, \ldots, K\}^{B-1})$.
2. For each $i = 1, \ldots, L - B + 1$ and $(\alpha_1, \ldots, \alpha_B) \in A$, calculate $\ln(p_i^{\alpha_1} \ldots \ldots p_{i+B-1}^{\alpha_B}) = \alpha_1 \ln(p_i) + \cdots + \alpha_B \ln(p_{i+B-1})$. Form a list from all logarithms of this form, which are positive, and attach the corresponding i and $(\alpha_1, \ldots, \alpha_B)$ to each of them.

3. Sort the list and find consecutive pairs of its elements, whose difference is less than 10^{-n}. Each of such pairs determines a collision of length at most $2B$.

Note that this algorithm may be obtained from Algorithm 1 by replacing its step 3 by creating a larger set of possible half-collisions and then running only one iteration to find close-enough ratios.

The set A contains all vectors α from $\{-K, \ldots, K\}^B$ with nonzero first component. Moreover, the set A is symmetric in the sense that $\alpha \in A$ if and only if $-\alpha \in A$. Consequently, in the step 2, exactly one half of the computed logarithms will be positive. It is possible to reduce the computation time by computing logarithms only for exponent vectors α, with a positive first component. In this case, we include in the list either the computed logarithm for α itself, if it was positive, or the logarithm for the exponents $-\alpha$ otherwise.

Using this approach with $L = 450$, $K = 5$ and $B = 3$, we obtained the following collisions.

Error	Primes	Exponents
6.829054e–11	2339, 2341, 3119, 3121	-3, 3, 4, -4
7.862933e–11	2111, 2113, 3167, 3169	-2, 2, 3, -3
7.88554 e – 11	2711, 2713, 2719, 2729, 2731	-3, 5, -3, 3, -2
5.173761e–13	2833, 2837, 2843, 2851, 2857, 2861	-1, 3, -4, 4, -3, 1

3.1.4. Searching collisions using LLL algorithm

Let $\{i_1, \ldots, i_m\} \subseteq \{1, \ldots, L\}$ be a set of indices of m primes. The problem to find exponents α_j for $j = 1, \ldots, m$ such that

$$|\alpha_1 \ln p_{i_1} + \cdots + \alpha_m \ln p_{i_m}| < \delta \tag{3.2}$$

for a small $\delta > 0$ is related to the problem of detecting exact integer relations between real numbers on the basis of their numerical value known to a limited precision. Algorithms used for numerical detection of exact integer relations actually look for approximate integer relations (3.2) with δ derived using the known bound on the precision of the input numbers. Algorithms searching for integer relations between given real numbers may be found, for example, at [6]. We have chosen LLL algorithm and its implementation in C++ library for computational number theory LiDIA, see [9].

Consider the matrix

$$\begin{pmatrix} \ln p_{i_1} & \varepsilon & 0 & \ldots & 0 \\ \ln p_{i_2} & 0 & \varepsilon & \ldots & 0 \\ \ldots & & & & \\ \ln p_{i_m} & 0 & 0 & \ldots & \varepsilon \end{pmatrix}. \tag{3.3}$$

An integer combination of the rows of this matrix with coefficients α_j for $j = 1, \ldots, m$ has the form

$$\left(\sum_{j=1}^{m} \alpha_j \ln p_{i_j},\ \varepsilon\alpha_1,\ \ldots,\ \varepsilon\alpha_m \right)$$

and the square of its Euclidean norm is

$$\left(\sum_{j=1}^{m} \alpha_j \ln p_{i_j} \right)^2 + \ \varepsilon^2 \sum_{j=1}^{m} \alpha_j^2 \,.$$

In order to find collisions, we consider the solutions of (3.2), where $\delta = 10^{-n}$. Similarly, as in the previous approaches, we impose a bound on the exponents α_j. In this case, we require $|\alpha_j| \leq K$, where $K = 5$. In order to find solutions of (3.2) with the given parameters, we look for integer combinations of the rows of the matrix (3.3), which have the absolute value of the first component smaller than δ and the absolute value of the coefficients α_j at most K. If we choose ε such that εK is roughly of the order 10^{-n}, then such a solution is represented by a vector of the above form and has a small Euclidean norm. In order to find integer combinations of the rows, which have a small norm, we used LLL algorithm to compute a reduced basis of the lattice generated by the rows of the matrix (3.3). The reduced basis typically consists of rows, which have Euclidean norm smaller than the original vectors. The obtained basis does not need to be the optimal basis from the point of view of the norm of its elements, but it approximates such a basis in a sense described for example in [8].

Let S be the maximum length of the collisions, which we are looking for. In our calculations, it appeared to be advantageous to use sets of primes of size m larger than S and look for solutions, which have at most S nonzero coefficients α_j.

The choice of the parameter ε influences the solutions, which are obtained. If ε increases, the method prefers solutions with smaller coefficients α_j and possibly also smaller number of nonzero coefficients on the cost of a larger error of the solution. Decreasing ε has the opposite effect and the method prefers more exact solutions with, possibly, larger coefficients. In our calculations, we used values of ε from the sequence $\varepsilon = 10^{-14}, 2 \cdot 10^{-14}, 5 \cdot 10^{-14}, 10^{-13}, \ldots, 10^{-8}$, which was chosen so that extending the sequence in any direction would likely not allow to find further solutions.

The sets of primes, for which we used the above approach, were chosen as a union of two disjoint intervals of consecutive primes, each of the length B, where B is a parameter.

For technical reasons, we used only even values of B and in order to limit the number of tested sets, we did not use all $L - B + 1$ intervals of the length B, but only those, which are obtained as follows. The sequence $1, \ldots, L$ of indices

of the used primes was divided into disjoint intervals of the length $B/2$ starting from 1 and a possible remainder of the length smaller than $B/2$. In this way, we obtained $\lfloor 2L/B \rfloor$ intervals of the length $B/2$, namely $[1, B/2], [B/2 + 1, B], [B + 1, 3/2B], \ldots$ We consider unions of each two consecutive intervals in this sequence, which yields $\lfloor 2L/B \rfloor - 1$ intervals of the length B. Some of the obtained intervals overlap, but at most in $B/2$ elements.

Each pair of disjoint intervals of the length B described in the previous paragraph was used to form a set of indices of size $2B$ by taking union. Hence, we considered $(\lfloor 2L/B \rfloor - 2)(\lfloor 2L/B \rfloor - 3)/2$ sets of size $m = 2B$.

We used $B = 4, 6, 8, 10$ and the maximum length of the required solutions was $S = 9$. The number of solutions of different lengths, which we obtained, are presented in the following table.

Length	Number of solutions found
4	2
5	6
6	50
7	446
8	2547
9	10600

The following list presents the best two solutions found for each of the length between 5 and 8 and the best solution for the length 9. For the length 4, the LLL algorithm found the same two solutions as the Algorithm 2.

Error	Primes	Exponents
4.517e–12	2111, 2153, 2161, 2273, 2287	-3, 2, 2, 1,-2
3.187e–11	859, 863, 1279, 1283, 1297	-4, 4,-1, 3,-2
1.527e–13	1187, 1193, 2371, 2377, 2383, 2389	4,-4,-1,-5, 5, 1
1.769e–13	7, 2381, 2383, 2389, 2393, 2399	4,-1, 2,-3, 3,-2
8.24e–14	277, 281, 311, 313, 1747, 1801, 1823	3,-3,-1, 1,-2, 4,-2
1.529e–13	1607, 1609, 1621, 1657, 1663, 1777, 1801	-1, 2,-1, 2,-2,-1, 1
1.463e–15	5, 7, 17, 881, 907, 911, 937, 941	4,-2,-1,-4, 1,-2, 1, 4
8.873e–15	277, 293, 313, 317, 1201, 1223, 1231, 1277	1,-4, 4,-1, 2, 1,-1,-2
5.158e–17	673, 677, 691, 719, 727, 2447, 2467, 2503, 2521	4,-3, 4, 5,-4, 4,-5, 1,-5

4. Conclusions and open questions

In this paper we considered a simple hash function inspired by a popular book [1] and provided its cryptanalysis from the point of view of collision resistance and second preimage resistance. We found a weakness in the collision resistance in such a way that we were able to produce meaningful collisions. The weakness seems to be applicable also to the second preimage construction,

but finding meaningful second preimages remains an open question for further research. Another open question is the strength of the preimage resistance. The hash function has a very clear structure and we think that the preimage resistance can be broken.

REFERENCES

[1] THOMAS, S.: *PopCo.* Fourth State, Edinburgh, 2004.

[2] MENEZES, A.—VAN OORSCHOT, P. C.—VANSTONE, S. A.: *Handbook of Applied Cryptography.* CRC Press Ser. on Discrete Math. Appl., CRC Press, Boca Raton, FL, 1997.

[3] WIENER, M. J.: *Bounds on Birthday Attack Times.* IACR Cryptology ePrint Archive, Report 2005/318, 2005, `http://eprint.iacr.org/2005/318`.

[4] *Gödel,* Wikipedia article, `http://en.wikipedia.org/wiki/Goedel`.

[5] *Integer relation,* Wikipedia article, `http://en.wikipedia.org/wiki/Integer_relation`.

[6] SHOUP, V.: *A Computational Introduction to Number Theory and Algebra.* Cambridge University Press, Cambridge, 2005.

[7] SHOUP, V.: *NTL: A Library for doing Number Theory,* `http://www.shoup.net/ntl`.

[8] YAP, CH. K.: *Fundamental Problems in Algorithmic Algebra.* Oxford University Press, Oxford, 2000.

[9] *LiDIA, a C++ Library for Computational Number Theory,* `http://www.cdc.informatik.tu-darmstadt.de/TI/LiDIA/`.

[10] KNUTH, D.: *The Art of Computer Programming, Vol. 2: Seminumerical Algorithms* (3rd ed.), Addison-Wesley Co., Reading, Massachusetts, 1997.

[11] BARLOW, J. L.—BAREISS, E. H.: *On roundoff error distributions in floating point and logarithmic arithmetic,* Computing **34** (1985), 325–347.

Received September 25, 2009

Michal Mikuš
Department of Applied Informatics and Computing Technologies
Faculty of Electrical Engineering and Information Technology
Ilkovičova 3
SK–812-19 Bratislava
SLOVAKIA
E-mail: michal.mikus@stuba.sk

Petr Savicky
Institute of Computer Science
Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2
CZ–182-07 Prague 8
CZECH REPUBLIC
E-mail: savicky@cs.cas.cz