DOI: 10.14529/cmse170302

PREIMAGE ATTACK ON MD4 HASH FUNCTION AS A PROBLEM OF PARALLEL SAT-BASED CRYPTANALYSIS*

© 2017 I.A. Gribanova, O.S. Zaikin, I.V. Otpuschennikov,
A.A. Semenov

Matrosov Institute for System Dynamics and Control Theory SB RAS
(Lermontova st. 134, Irkutsk, 664033, Russia)
E-mail: the42dimension@gmail.com, zaikin.icc@gmail.com, otilya@yandex.ru,
biclop.rambler@yandex.ru

Received: 04.05.2017

In this paper we study the inversion problem of MD4 cryptographic hash function developed by R. Rivest in 1990. By MD4-k we denote a truncated variant of MD4 hash function in which k represents a number of steps used to calculate a hash value (the full version of MD4 function corresponds to MD4-48). H. Dobbertin has showed that MD4-32 hash function is not one-way, namely, it can be inverted for the given image of a random input. He suggested to add special conditions to the equations that describe the computation of concrete steps (chaining variables) of the considered hash function. These additional conditions allowed to solve the inversion problem of MD4-32 within a reasonable time by solving corresponding system of equations. The main result of the present paper is an automatic derivation of "Dobbertin's conditions" using parallel SAT solving algorithms. We also managed to solve several inversion problems of functions of the kind MD4-k (for k from 31 up to 39 inclusive). Our method significantly outperforms previously existing approaches to solving these problems.

Keywords: cryptanalysis, hash function, inversion problem, MD4, SAT, parallel computing, MPI.

FOR CITATION

Gribanova I.A., Zaikin O.S., Otpuschennikov I.V., Semenov A.A. Preimage Attack on MD4 Hash Function as a Problem of Parallel Sat-Based Cryptanalysis. *Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering.* 2017. vol. 6, no. 3. pp. 16–27. DOI: 10.14529/cmse170302.

Introduction

Hash function is a function which maps binary words of arbitrary length into binary words of fixed length. More precisely, a hash function is a total computable discrete function of the kind

$$\chi : \{0,1\}^* \to \{0,1\}^C, C = const. \tag{1}$$

By $\{0,1\}^k, k \in N$ we denote the set formed by all 2^k different vectors of length k. By $\{0,1\}^*$ we denote the set of all binary words of an arbitrary finite length.

Hash functions are used in various areas of computer science, for example, to speed up an access to large data sets. In cryptography and information security the range of issues that can be solved using hash functions is especially wide. Cryptographic hash functions meet additional requirement: corresponding functions should be hard to invert. In particular, not only the inversion problem (i.e. the problem of finding a preimage for a given hash value) but also the problem of finding collisions (i.e. the problem of finding an arbitrary pair of messages that give the same hash) should be hard.

[*] The paper is recommended for publication by the Program Committee of the International Scientific Conference "Parallel Computational Technologies (PCT) 2017".

In [1, 2] a method of finding collisions for hash functions from the MD family, widely used at that time, was presented. Further this problem was discussed in a number of papers, wherein various methods for constructing collisions were employed. The possibility of effective collision search for the MD family hash functions resulted in their exclusion from wide usage and replacement by other hash functions (mainly by the SHA family hash functions) in most cryptographic systems. However, today even for MD4 (the weakest hash function from the MD family) there are no successful results in solving the problem of its inversion. Moreover, there are no algorithms that would appear to be much more computationally stronger than the method proposed in [3] (one of the first papers in which the problem of MD4 inversion was studied).

The main idea of the attack proposed in [3] consists in considering the truncated variants of MD4 (with less than 48 steps of the hash value calculation) and some additional conditions to the equations, defining the corresponding function. In some cases this approach allows to solve the corresponding system of equations on a parallel computing system in reasonable time.

In the present paper we study the MD4 inversion problems using parallel algorithms for solving Boolean satisfiability problem (SAT). This paper is organized as follows. In Section 1 we present the necessary information concerning the algorithmic features of MD4 and the basic idea of H. Dobbertin's attack. In Section 2 we describe the foundations of the SAT-based cryptanalysis. We also consider the reduction of the MD4 inversion problem to SAT and give a brief description of parallel algorithms applied to the obtained SAT instances. Section 3 presents the results of computational experiments and comparison with the results obtained in previous works.

1. Structure of MD4 and Basic Idea of Dobbertin's Attack

The cryptographic hash function MD4 [4] was developed by R. Rivest in 1990. This function is one of the first examples of practical implementation of the Merkle-Damgard construction [5, 6]. The basic paradigm of the Merkle-Damgrad construction consists in the fact that a hash value is a result of a sequence of similar actions, which is written into a special register. Further this register is called a hash register. At the initial step the hash register is filled with some known value (Initial Value, IV). In case of MD4

$$IV = \{\texttt{0x67452310, 0xEFCDAB89, 0x98BADCFE, 0x10325476}\}. \tag{2}$$

At the next steps the hash register states are modified by mixing the current state with parts of the input message (which should be hashed). Like many other hash functions, MD4 works with input message divided into blocks with 512 bits in each. The value of MD4 is a binary 128-bit word. Basic primitives of MD4 are 32-bit words (as in many hash functions which were developed later): 512-bit block is divided into 16 32-bit words, 128-bit hash is divided into 4 32-bit words.

At the initial stage MD4 algorithm uses a special padding procedure. After that MD4 compression function f_{MD4} is applied to the obtained 512-bit block. The result of f_{MD4} is 128-bit hash of the considered block. Hereinafter we don't take padding procedure into account. Thus, we consider only the inversion problem of f_{MD4}. The process of calculation of this function is an iterative procedure which is divided into 48 steps. On each step the value of one hash register cell (filled with 32-bit word) is updated. Identifiers of these cells are called chaining variables. So each chaining variable takes values from $\{0, 1\}^{32}$. We denote the chaining variables by letters a, b, c, d. The process of hash calculation is divided into 3 rounds, each of them consists of 16 steps. In each round a certain round function is used, which operates with 32-bit variables. The calculation scheme of f_{MD4} is shown in Fig.

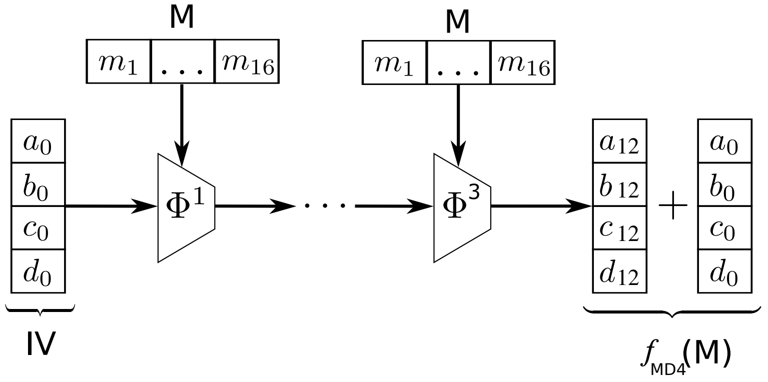

Fig. The calculation scheme of f_{MD4}

Let us briefly comment on Fig. By $M = m_1|\ldots|m_{16}$ we denote the 512-bit input block. By $\Phi^1 - \Phi^3$ the round transformations are denoted. For example, for the first 4 steps the corresponding transformations of chaining variables are as follows:

$$\begin{aligned} a_1 &= (a_0 + \phi^1(b_0, c_0, d_0) + m_1 + t_1) \lll s_a^1, \\ d_1 &= (d_0 + \phi^1(a_1, b_0, c_0) + m_2 + t_1) \lll s_d^1, \\ c_1 &= (c_0 + \phi^1(d_1, a_1, b_0) + m_3 + t_1) \lll s_c^1, \\ b_1 &= (b_0 + \phi^1(c_1, d_1, a_1) + m_4 + t_1) \lll s_b^1. \end{aligned} \tag{3}$$

In these formulas "+" stands for *mod* 2^{32} summation of the corresponding numbers, "$\lll s$" stands for the cyclic shift of a 32-bit word to s positions to the left. Constants t_i and $s^i_{a,b,c,d}$, $i \in \{1, 2, 3\}$ (here i is the round number) are known from the specification of the algorithm (for example, $t_1 = 0, t_2 = \texttt{0x5A827999}, t_3 = \texttt{0x6ED9EBA1}$).

In each round with number $i, i \in \{1, 2, 3\}$ all chaining variable are updated 4 times by applying the round function ϕ^i. The MD4 round functions are as follows:

$$\begin{aligned} &\phi^1(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z), \\ &\phi^2(X, Y, Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z), \\ &\phi^3(X, Y, Z) = X \oplus Y \oplus Z. \end{aligned} \tag{4}$$

Arguments of these functions are 32-bit words. All logical operations are performed component-wise over the corresponding vectors. The summation of the IV vector and the value of hash register at the end of the third round presented in Fig. is called a finalization stage.

By MD4-k we denote the hash function which corresponds to the execution of k-steps of MD4 algorithm applied to IV with finalization stage performed after these steps. For example, MD4-48 corresponds to the full-round version of MD4 hash function. An arbitrary function of the kind MD4-k, $k \in \{1, \ldots, 48\}$ is called the truncated version of MD4.

In [3] an algorithm aimed at inversion of MD4-32 using an ordinary PC was proposed. As a result, it was shown that 2-round version of MD4 hash function is not one-way. The basic idea of the attack is as follows. By analyzing the first two round functions it can be concluded that assignment of some chaining variables with some constant leads to finding the majority of words from the set $\{m_1, \ldots, m_{16}\}$ in a short time.

In particular, H. Dobbertin suggested to fix chaining variables values with some constant K at the steps with numbers 13, 17, 21, 25, 14, 18, 22, 26, 15, 19, 23, 27. Then the value of the chaining variable b at the 28-th step is varied. It should be noted that in this attack the value of variable b is recalculated at the 28-th step, but it isn't changed for the other 3 steps. Thus, if some value of b is determined, then the same value of this variable is also determined for the steps 29, 30, 31. The fixation of K and b at the steps mentioned above gives us an opportunity to derive the values of variables $m_1, m_2, \ldots, m_{12}$ and m_{16} of the input message. Their usage makes it possible to derive the values of several more chaining variables and, finally, the value of b at the 28-th step. The latter value can differ from the one fixed previously. In this case the attempt is considered to be unsuccessful and the sequence of actions described above is repeated for new b (in this case constant K may remain unchanged).

H. Dobbertin mentioned that in order to achieve successful results in the described attack one needs to make about 2^{32} attempts of selection of the value of b on the 28-th step. So the proper implementation of the presented algorithm can make it possible to achieve the successful result even on a weak PC.

2. Reduction of the MD4-k Inversion Problems to SAT and the SAT Version of Dobbertin's Attack

In this section we describe some new techniques for SAT-based cryptanalysis and apply them to the inversion problems of MD4 hash function.

SAT-based cryptanalysis is a relatively new direction in cryptanalysis implying the usage of algorithms for solving Boolean satisfiability problem (SAT) for the inversion of cryptographic functions. Let us remind that for SAT it is necessary for an arbitrary Boolean formula F to decide whether it is satisfiable or not, i.e. if there exists an assignment of Boolean variables from this formula that makes it TRUE. Using Tseitin transformations [7] the Boolean satisfiability problem for F can be reduced to SAT in the Conjunctive Normal Form (CNF) in polynomial time on the size of F description. Hereinafter by SAT we mean the problem of satisfiability of an arbitrary CNF.

According to the Cook theorem, a wide class of combinatorial problems can be effectively reduced to SAT, including the inversion problems of cryptographic functions: for a given image from a range of values of considered function to find a preimage from its domain (assuming that function is defined by known algorithm). Today there is a number of automatic translation systems designed for effectively construction of SAT encodings for inversion of cryptographic functions [8–11]. In all our computational experiments we use the Transalg system [11].

Once a SAT encoding for the inversion problem of the considered cryptographic function is built, the corresponding SAT instance is ready to be solved. A variety of algorithms can be used for this purpose. However, according to numerous computational experiments, CDCL-based SAT solvers [12] are better suited for inversion of cryptographic functions. The survey of algorithms and technologies underlying modern CDCL-solvers can be found in [13].

It should be noted that cryptanalysis problems in the form of SAT are usually extremely hard even for the best-known SAT-solvers (except for the inversion problems of some weak functions, e.g. the Geffe generator). Thus, for functions with serious cryptographic resistance additional considerations should be used (these considerations may arise from the algorithmic features of these functions). As usual, it is hard to avoid using parallel computations for the real attacks. Below we provide a brief description of the parallel SAT technologies, which we applied to the inversion problems of MD4-k hash functions.

Today there are two main approaches to the parallel solving of SAT instances: the portfolio approach and the partitioning approach [14]. Portfolio approach can be considered as a multi-threaded parallelism, while partitioning-approach is the large-block data parallelism.

According to the portfolio approach, multiple copies of a SAT solver are used and, roughly speaking, each of these copies goes through the common search space in different directions sharing accumulated information. In the partitioning approach a search space is divided into disjoint subdomains which are processed separately. Thus, the partitioning approach is better suited to solve hard SAT instances in distributed computing systems, while the portfolio approach shows the best results in multithreaded systems.

As already mentioned, apart from parallelism, it is often possible to accelerate the inversion of cryptographic functions by taking into account various features of these functions. For example, in order to achieve results in solving the SAT instances for finding collisions from MD family hash functions it is nessesary to add special conditions called differential paths on chaining variables to the corresponding SAT-encodings. The first successful attack of this kind has been shown in [15], where the authors have added to the SAT encodings conditions specifying the differential paths described in [1, 2].

A similar situation occurs with respect to the inversion problems of MD4-k functions. Dobbertin's conditions described above represent additional constraints which significantly reduce the search space. The following questions are: how successful these additional constraints are, and are there more effective ones? The answers to these questions are discussed in the present paper.

Before proceeding to the submussion of our results, let us note that SAT solvers can be considered as a means of "intellectual search". Consequently, the use of a SAT solver must be aimed primarily at the automation of a large number of similar operations. The technique of information preservation and non-chronological backtracking used in CDCL-solvers can provide essential reduction of the amount of calculation in comparison with the exhaustive search. Similar arguments were given in [15], where the authors emphasize that they have used the SAT approach primarily in order to automate the message modification phase forming the main part of the attack described in [1, 2].

In the Dobbertin's attack the search through all possible values of variable b corresponding to the 28-th step of the hash function is performed, wherein the value of K may be fixed for different values of b. In the attack proposed below we use the value $K = 0$ (as in [16]) and consider the inversion problem of 1^{128} hash, i.e. assuming that the hash value consists of 128 ones.

The main difference between our attack and Dobbertin's attack consists in the fact that we search through all possible variants of assignment of chaining variables with constant $K = 0$ using a parallel SAT solver. We do not use assignment of the variable b on the 28-th step assuming that SAT solver automatically adjusts the values of the unknown variables for a particular combination of chaining variables assigned with constant $K = 0$.

For the purpose of the automatic search through different combinations of chaining variables we use additional variables called switching variables. Let us describe the corresponding technique. By $C^* = C(f_{MD4-k}(M) = 1^{128})$ we denote CNF encoding of the inversion problem of f_{MD4-k} function in point 1^{128}. As in Dobbertin's attack first we consider the inversion problem of f_{MD4-32} function. In fact we need a procedure that will allow to quickly add to the C^* various combinations of chaining variables on the steps with numbers from 1 to 32 assigned with constant $K = 0$. By z^j we denote a 32-bit chaining variable of j-step. For z^j there are 32 corresponding Boolean variables $y_1^j, \ldots, y_{32}^j$. Let us consider the elementary conjunction:

$$R^j = \neg y_1^j \wedge \cdots \wedge \neg y_{32}^j. \qquad (5)$$

Obviously, R^j is equal to 1 if and only if variable z^j is assigned with constant $K = 0$. For each $j \in \{1, \ldots, 32\}$ let us consider a new Boolean variable s^j, which we associate with R^j by the following formula:

$$F^j = (\neg s^j \vee \neg y_1^j) \wedge \cdots \wedge (\neg s^j \vee \neg y_{32}^j). \quad (6)$$

Obviously, the substitution of $s^j = 1$ into F^j gives a formula which is logically equivalent to R^j. The substitution of $s^j = 0$ into F^j gives a constant of 1. We shall consider the following CNF:

$$\tilde{C} = C^* \wedge F^1 \wedge \cdots \wedge F^{32}. \quad (7)$$

According to the mentioned above, the substitution of any assignment $s \in \{0, 1\}^{32}$ of switching variables:

$$s^j = \alpha^j, \alpha^j \in \{0, 1\}, j \in \{1, \ldots, 32\}, \quad (8)$$

in $\tilde{C}$ gives the inversion problem of MD4-32 function in point 1^{128} with additional assignment with constant $K = 0$ of chaining variable z^j for which corresponding α^j takes value 1. Let us note that such assignment can be considered in application to the problem of the satisfiability of $\tilde{C}$ as an *assumption* and it is possible to apply incremental SAT technique [17] which allows to store and use the information obtained during processing of various assumptions.

3. Computational Experiments

In our experiments we used two SAT solvers. The first one is our parallel SAT solver PDSAT [18], which is based on the partitioning approach. This solver was designed especially for solving SAT instances that encode inversion problems of cryptographic functions. PDSAT is an MPI-program, in which there is one leader process, all the other are computing processes (each process corresponds to 1 CPU core). PDSAT works in two modes. In the estimation mode it searches for a decomposition set with good time estimations. A decomposition set is in fact a set of Boolean variables, for which all their possible assignments are generated. As a result, a family of SAT instances, where each of instances is simpler than the original problem, was obtained. In this mode we use the Monte Carlo approach and various optimization metaheuristics (simulated annealing, tabu search, etc.). In the solving mode PDSAT solves all SAT instances from a family obtained from a given decomposition set. CRYPTOMINISAT [10] is the second SAT solver which was used in our experiments. It is not designed for launching on an MPI cluster, so we launched it on a PC as a sequential program.

At the first stage we considered the MD4-31 inversion problem in point 1^{128}. We launched PDSAT in the solving mode on the decomposition set which consisted of 27 switching variables (see the previous section). Time limit of 0.01 second for each subproblem was used. PDSAT was launched on 5 nodes of the "Academician V.M. Matrosov" computing cluster of Irkutsk supercomputing center SB RAS[1]. Each node of this cluster consists of 2 16-core CPUs AMD Opteron 6276, so 160 CPU cores were used in total. All 2^{27} SAT instances were solved in 10 minutes 21 seconds, the processing of 4.88 % SAT instances was interrupted by time limit. As a result, 6 satisfying assignments (6 solutions of the considered inversion problem) were found. It should be noted that one of the assignments of switching variables with constant $K = 0$ was in fact the Dobbertin's conditions. Thus, we found these conditions in automatic mode.

At the second stage we considered the MD4-39 inversion problem, taking into account results from [16], where the authors involved 11 of the 12 Dobbertin's conditions. We constructed 3

[1] http://www.hpc.icc.ru

CNFs, in which we fixed the values of 11 switching variables corresponding to aforementioned 11 conditions, and assigned a different number of variables encoding the chaining variable corresponding to the omitted Dobbertin's condition. The value of this chaining variable was taken from one of the satisfying assignments of MD4-31 inversion problem. In the first CNF the first 8 bits of this chaining variable were assigned. In the second and third CNFs the first 16 and 32 bits of this chaining variable were assigned (respectively).

We launched CRYPTOMINISAT on a PC equipped with the i5-2410M CPU (1 core was used). The best result was obtained on the first CNF – a satisfying assignment was found in 15 minutes. We also employed PDSAT in the solving mode. The best result was obtained on the second CNF – the corresponding problem was solved in 5 seconds on 5 computing nodes (160 CPU cores in total). Thus, if we recalculate this time on the case of a sequential launch, we obtain the time of about 13 minutes. It should be noted, that in [16] this inversion problem was solved in about 8 hours on 1 CPU core. So our approach allowed to solve this problem much faster. In Tab. we show 4 input messages M found in our experiments by PDSAT and CRYPTOMINISAT. All these messages correspond to the hash value 1^{128}.

Table

Messages for the MD4-39 inversion problem in point 1^{128}

No.	**Message**
1	0xc7c08b1c,0xa57d8667,0xa57d8667,0x07e14fec, 0xa57d8667,0xa57d8667,0xa57d8667,0xa8cea698, 0xa57d8667,0xa57d8667,0xa57d8667,0x28e987ac, 0x4665c5f3,0x8c49173f,0xabc74a06,0x0cd9d788
2	0x40b2a2ff,0xa57d8667,0xa57d8667,0x2b010cef, 0xa57d8667,0xa57d8667,0xa57d8667,0xccfef2c3, 0xa57d8667,0xa57d8667,0xa57d8667,0xf041ded3, 0x5443c70c,0xaadd4c2b,0xe587e70e,0xe5bad382
3	0x40b26b1f,0xa57d8667,0xa57d8667,0xa87153ec, 0xa57d8667,0xa57d8667,0xa57d8667,0xbaee84bb, 0xa57d8667,0xa57d8667,0xa57d8667,0x6a228d63, 0x5960f23b,0x1915d72b,0xf2d3b064,0x7d85d6db
4	0xec3a2319,0xa57d8667,0xa57d8667,0xf87f9cee, 0xa57d8667,0xa57d8667,0xa57d8667,0x80a0aeb0, 0xa57d8667,0xa57d8667,0xa57d8667,0x51c2c922, 0x964933fa,0x0545c48c,0x98968391,0x783c0174

4. Related Work

As we mentioned above, the first successful example of the SAT approach application to the inversion of relevant cryptographic functions was presented in [15]. In that paper a SAT-based variant of attack by X. Wang et al. [1, 2] was suggested. The method used in [15] allows to find one-block collisions for MD4 relatively fast. To find two-block collisions for MD5 much more computational resources must be used. We significantly increased the efficiency of the approach proposed in [15]: the application of SAT encodings constructed by the Transalg system [11] allowed us to generate one-block collisions for MD4 approximately 1000 times faster. By applying modern multithreaded SAT solvers we managed to find two-block collisions for MD5 in reasonable time using one computing cluster node. As a result, a family of such collisions of special kind (with a large amount of most significant bits) was constructed [19].

The first SAT-based version of Dobbertin's attack was proposed in [16]. The corresponding chaining variables were assigned with constant $K = 0$. It should be noted that in that paper the Dobbertin's conditions were used in their original form – with no attempts to justify or derive them. Also in [16] one Dobbertin's condition was thrown out without any justification too.

In [16] it took about 8 hours on the Minisat solver to invert MD4-39 with the Dobbertin's conditions for chaining variables of the first two rounds. Thus, we can conclude that the effectiveness of the method we suggested is significantly higher than the one proposed in [16].

Conclusions and Future Work

In this paper, we managed to automatically synthesize conditions from Dobbertin's attack using parallel SAT solving algorithms. We also studied the inversion problem of the MD4-39 hash function. By applying parallel SAT algorithms this problem was solved faster than it was done in previous works. In the nearest future we plan to apply parallel SAT algorithms to inversion problems of some other hash functions (from the MD and SHA hash families).

This research was funded by Russian Science Foundation (project No. 16-11-10046). Oleg Zaikin and Ilya Otpuschennikov are partially funded by Council for Grants of the President of the Russian Federation (stipends SP-1184.2015.5 and SP-4751.2016.5 respectively).

References

1. Wang X., Lai X., Feng D., Chen H., Yu X. Cryptanalysis of the Hash Functions MD4 and RIPEMD. *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques. EUROCRYPT'05.* Berlin, Heidelberg: Springer-Verlag, 2005. pp. 1–18. DOI: 10.1007/11426639_1.

2. Wang X., Yu H. How to Break MD5 and Other Hash Functions. *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques. EUROCRYPT'05.* Berlin, Heidelberg: Springer-Verlag, 2005. pp. 19–35. DOI: 10.1007/11426639_2.

3. Dobbertin H. The First Two Rounds of MD4 are Not One-Way. *Fast Software Encryption / Ed. by Serge Vaudenay. Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 1998. vol. 1372. pp. 284–292. DOI: 10.1007/3-540-69710-1_19.

4. Rivest R.L. The MD4 Message Digest Algorithm. *Advances in Cryptology - CRYPTO'90, Proceedings / Ed. by Alfred Menezes, Scott A. Vanstone. Lecture Notes in Computer Science.* Springer, 1990. vol. 537. pp. 303–311. DOI: 10.1007/3-540-38424-3_22.

5. Damgård I.B. A Design Principle for Hash Functions. *Proceedings on Advances in Cryptology. CRYPTO '89.* New York, NY, USA: Springer-Verlag New York, Inc., 1989. pp. 416–427. DOI: 10.1007/0-387-34805-0_39.

6. Merkle R.C. A Certified Digital Signature. *Proceedings on Advances in Cryptology. CRYPTO '89.* New York, NY, USA: Springer-Verlag New York, Inc., 1989. pp. 218–238. DOI: 10.1007/0-387-34805-0_21.

7. Tseitin G.S On the Complexity of Derivation in Propositional Calculus. *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970.* Berlin, Heidelberg: Springer Berlin Heidelberg, 1983. pp. 466–483. DOI: 10.1007/978-3-642-81955-1_28.

8. Erkök L., Matthews J. High assurance programming in Cryptol. *Fifth Cyber Security and Information Intelligence Research Workshop, CSIIRW'09, Knoxville, TN, USA, April 13-15, 2009 / Ed. by Frederick T. Sheldon, Greg Peterson, Axel W. Krings [et al.].* ACM, 2009. pp. 60. DOI: 10.1145/1558607.1558676.

9. Janicic P. URSA: a System for Uniform Reduction to SAT. *Logical Methods in Computer Science.* 2012. vol. 8, no. 3. pp. 1–39. DOI: 10.2168/lmcs-8(3:30)2012.

10. Soos M., Nohl K., Castelluccia C. Extending SAT Solvers to Cryptographic Problems . *SAT / Ed. by Oliver Kullmann. Lecture Notes in Computer Science.* Springer, 2009. vol. 5584. pp. 244–257. DOI: 10.1007/978-3-642-02777-2_24.

11. Otpuschennikov I., Semenov A., Gribanova I., Zaikin O., Kochemazov S. Encoding Cryptographic Functions to SAT Using TRANSALG System. *ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August-2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016) / Ed. by Gal A. Kaminka, Maria Fox, Paolo Bouquet [et al.]. Frontiers in Artificial Intelligence and Applications.* IOS Press, 2016. vol. 285. pp. 1594–1595.

12. Marques-Silva J.P., Sakallah K.A. GRASP: A Search Algorithm for Propositional Satisfiability. *IEEE Trans. Computers.* 1999. vol. 48, no. 5. pp. 506–521. DOI: 10.1109/12.769433.

13. Marques-Silva J.P., Lynce I., Malik S. Conflict-Driven Clause Learning SAT Solvers. *Handbook of Satisfiability / Ed. by Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh. Frontiers in Artificial Intelligence and Applications.* IOS Press, 2009. vol. 185. pp. 131–153.

14. Hyvärinen A.E.J. Grid Based Propositional Satisfiability Solving. Ph. D. thesis, Aalto University, 2011.

15. Mironov I., Zhang L. Applications of SAT Solvers to Cryptanalysis of Hash Functions. *SAT / Ed. by Armin Biere, Carla P. Gomes. Lecture Notes in Computer Science.* Springer, 2006. vol. 4121. pp. 102–115. DOI: 10.1007/11814948_13.

16. De D., Kumarasubramanian A., Venkatesan R. Inversion Attacks on Secure Hash Functions Using SAT Solvers. *Theory and Applications of Satisfiability Testing - SAT 2007, Proceedings / Ed. by João Marques-Silva, Karem A. Sakallah. Lecture Notes in Computer Science.* Springer, 2007. vol. 4501. pp. 377–382. DOI: 10.1007/978-3-540-72788-0_36.

17. Eén N., Sörensson N. Temporal induction by incremental SAT solving. *Electr. Notes Theor. Comput. Sci.* 2003. vol. 89, no. 4. pp. 543–560. DOI: 10.1016/s1571-0661(05)82542-3.

18. Semenov A., Zaikin O. Algorithm for finding partitionings of hard variants of boolean satisfiability problem with application to inversion of some cryptographic functions. *SpringerPlus.* 2016. vol. 5, no. 1. pp. 1–16. DOI:10.1186/s40064-016-2187-4.

19. Bogachkova (Gribanova) I., Zaikin O., Kochemazov S., Otpuschennikov I., Semenov A., Khamisov O. Problems of search for collisions of cryptographic hash functions of the MD family as variants of Boolean satisfiability problem. *Numerical Methods and Programming.* 2015. vol. 16, no. 1. pp. 61–77. (in Russian)

УДК 004.056.55, 003.26 **DOI: 10.14529/cmse170302**

ПОИСК ПРООБРАЗОВ ХЕШ-ФУНКЦИИ MD4 КАК ПРОБЛЕМА ПАРАЛЛЕЛЬНОГО ЛОГИЧЕСКОГО КРИПТОАНАЛИЗА

© 2017 И.А. Грибанова, О.С. Заикин, И.В. Отпущенников, А.А. Семёнов

Институт динамики систем и теории управления
им. В.М. Матросова СО РАН (664033 Иркутск, ул. Лермонтова, д. 134)
E-mail: the42dimension@gmail.com, zaikin.icc@gmail.com, otilya@yandex.ru, biclop.rambler@yandex.ru

Поступила в редакцию: 04.05.2017

В статье исследуется задача обращения криптографической хеш-функции MD4, разработанной Р. Ривестом в 1990 году. Через MD4-k обозначается вариант данной функции, в которой параметр k обозначает количество шагов используемых для вычисления хеш-значения (при k=48 имеем полнораундовую версию MD4). В работах Г. Доббертина было показано, что хеш-функция MD4-32 не является односторонней, т.е. для нее может быть решена задача обращения. С этой целью к уравнениям, описывающим конкретные шаги алгоритма вычисления данной функции, были добавлены дополнительные условия на значения некоторых переменных сцепления (chaining variables). Эти дополнительные условия позволили за приемлемое время решить задачу обращения хеш-функции MD4-32 путем решения сооответствующей системы уравнений. Основным результатом представляемой статьи является автоматический вывод условий подобных условиям Доббертина (“Dobbertin’s conditions”) при помощи параллельных алгоритмов решения проблемы булевой выполнимости (SAT). Также с использованием данных алгоритмов были решены некоторые задачи обращения функции MD4-k для значений параметра k от 31 до 39 включительно. Стоит отметить, что предложенный метод существенно превосходит по эффективности описанные ранее подходы к решению данной проблемы.

Ключевые слова: криптоанализ, хеш-функции, задача обращения, MD4, SAT, параллельные вычисления, MPI

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Gribanova I.A., Zaikin O.S., Otpuschennikov I.V., Semenov A.A. Preimage Attack on MD4 Hash Function as a Problem of Parallel Sat-Based Cryptanalysis // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2017. Т. 6, № 3. С. 16–27. DOI: 10.14529/cmse170302.

Литература

1. Wang X., Lai X., Feng D., Chen H., Yu X. Cryptanalysis of the Hash Functions MD4 and RIPEMD // Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques. EUROCRYPT’05. Berlin, Heidelberg: Springer-Verlag, 2005. P. 1–18. DOI: 10.1007/11426639_1.

2. Wang X., Yu H. How to Break MD5 and Other Hash Functions // Proceedings of

the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques. EUROCRYPT'05. Berlin, Heidelberg: Springer-Verlag, 2005. P. 19–35. DOI: 10.1007/11426639_2.

3. Dobbertin H. The First Two Rounds of MD4 are Not One-Way // Fast Software Encryption / Ed. by Serge Vaudenay. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998. Vol. 1372. P. 284–292. DOI: 10.1007/3-540-69710-1_19.

4. Rivest R.L. The MD4 Message Digest Algorithm // Advances in Cryptology - CRYPTO'90, Proceedings / Ed. by Alfred Menezes, Scott A. Vanstone. Lecture Notes in Computer Science. Springer, 1990. Vol. 537. P. 303–311. DOI: 10.1007/3-540-38424-3_22.

5. Damgård I.B. A Design Principle for Hash Functions // Proceedings on Advances in Cryptology. CRYPTO '89. New York, NY, USA: Springer-Verlag New York, Inc., 1989. P. 416–427. DOI: 10.1007/0-387-34805-0_39.

6. Merkle R.C. A Certified Digital Signature // Proceedings on Advances in Cryptology. CRYPTO '89. New York, NY, USA: Springer-Verlag New York, Inc., 1989. P. 218–238. DOI: 10.1007/0-387-34805-0_21.

7. Tseitin G.S On the Complexity of Derivation in Propositional Calculus // Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970. Berlin, Heidelberg: Springer Berlin Heidelberg, 1983. P. 466–483. DOI: 10.1007/978-3-642-81955-1_28.

8. Erkök L., Matthews J. High assurance programming in Cryptol // Fifth Cyber Security and Information Intelligence Research Workshop, CSIIRW'09, Knoxville, TN, USA, April 13–15, 2009 / Ed. by Frederick T. Sheldon, Greg Peterson, Axel W. Krings [et al.]. ACM, 2009. P. 60. DOI: 10.1145/1558607.1558676.

9. Janicic P. URSA: a System for Uniform Reduction to SAT // Logical Methods in Computer Science. 2012. Vol. 8, No. 3. P. 1–39. DOI: 10.2168/lmcs-8(3:30)2012.

10. Soos M., Nohl K., Castelluccia C. Extending SAT Solvers to Cryptographic Problems // SAT / Ed. by Oliver Kullmann. Lecture Notes in Computer Science. Springer, 2009. Vol. 5584. P. 244–257. DOI: 10.1007/978-3-642-02777-2_24.

11. Otpuschennikov I., Semenov A., Gribanova I., Zaikin O., Kochemazov S. Encoding Cryptographic Functions to SAT Using TRANSALG System // ECAI 2016 - 22nd European Conference on Artificial Intelligence, 29 August – 2 September 2016, The Hague, The Netherlands - Including Prestigious Applications of Artificial Intelligence (PAIS 2016) / Ed. by Gal A. Kaminka, Maria Fox, Paolo Bouquet [et al.]. Frontiers in Artificial Intelligence and Applications. IOS Press, 2016. Vol. 285. P. 1594–1595.

12. Marques-Silva J.P., Sakallah K.A. GRASP: A Search Algorithm for Propositional Satisfiability // IEEE Trans. Computers. 1999. Vol. 48, No. 5. P. 506–521. DOI: 10.1109/12.769433.

13. Marques-Silva J.P., Lynce I., Malik S. Conflict-Driven Clause Learning SAT Solvers // Handbook of Satisfiability / Ed. by Armin Biere, Marijn Heule, Hans van Maaren, Toby Walsh. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009. Vol. 185. P. 131–153.

14. Hyvärinen A.E.J. Grid Based Propositional Satisfiability Solving. Ph. D. thesis, Aalto University, 2011.

15. Mironov I., Zhang L. Applications of SAT Solvers to Cryptanalysis of Hash Functions // SAT / Ed. by Armin Biere, Carla P. Gomes. Lecture Notes in Computer Science. Springer, 2006. Vol. 4121. P. 102–115. DOI: 10.1007/11814948_13.

16. De D., Kumarasubramanian A., Venkatesan R. Inversion Attacks on Secure Hash Functions Using SAT Solvers // Theory and Applications of Satisfiability Testing - SAT 2007, Proceedings / Ed. by João Marques-Silva, Karem A. Sakallah. Lecture Notes in Computer Science. Springer, 2007. Vol. 4501. P. 377–382. DOI: 10.1007/978-3-540-72788-0_36.

17. Eén N., Sörensson N. Temporal Induction by Incremental SAT Solving // Electr. Notes Theor. Comput. Sci. 2003. Vol. 89, No. 4. P. 543–560. DOI: 10.1016/s1571-0661(05)82542-3.

18. Semenov A., Zaikin O. Algorithm for Finding Partitionings of Hard Variants of Boolean Satisfiability Problem with Application to Inversion of Some Cryptographic Functions // SpringerPlus. 2016. Vol. 5, No. 1. P. 1–16. DOI:10.1186/s40064-016-2187-4.

19. Bogachkova (Gribanova) I., Zaikin O., Kochemazov S., Otpuschennikov I., Semenov A., Khamisov O. Problems of Search for Collisions of Cryptographic Hash Functions of the MD Family as Variants of Boolean Satisfiability Problem // Numerical Methods and Programming. 2015. Vol. 16, No. 1. P. 61–77. (in Russian)

Грибанова Ирина Александровна, аспирант, лаборатория 3.2, Институт динамики систем и теории управления имени В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Заикин Олег Сергеевич, к.т.н., с.н.с., лаборатория 3.2, Институт динамики систем и теории управления имени В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Отпущенников Илья Владимирович, к.т.н., н.с., лаборатория 3.2, Институт динамики систем и теории управления имени В.М. Матросова СО РАН (Иркутск, Российская Федерация)

Семёнов Александр Анатольевич, доцент, к.т.н., зав. лабораторией 3.2, Институт динамики систем и теории управления имени В.М. Матросова СО РАН (Иркутск, Российская Федерация)