University of Technology, Graz

Master Thesis

Differential cryptanalysis with SAT solvers

Author:
Lukas Prokop,
BSc BSc

Supervisor:
Dipl.-Ing. Dr.techn.
Florian Mendel

A thesis submitted in fulfillment of the requirements for the master's degree in Computer Science

at the

Institute of Applied
Information Processing and
Communications

April 6, 2016

Abstract

Hash functions are ubiquitous in the modern information age. They provide preimage, second preimage and collision resistance ensuring data and origin integrity. They are used as cryptographic primitives in protocols and applications.

In August 2006, Wang et al. showed efficient attacks against several hash function designs including MD4, MD5, HAVAL-128 and RIPEMD. With these results differential cryptanalysis has been proven useful to break collision resistance in hash functions. Over the years advanced attacks based on those differential approaches have been developed.

In this thesis we encode differential attack settings as SAT problems. SAT solvers are utilized to solve the problem revealing actual message collisions for a defined message difference.

Keywords: hash function, differential cryptanalysis, MD4, collision resistance, satisfiability, SAT solver

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis dissertation.

(digital signature)

Acknowledgements

First of all I would like to thank my academic advisor for his continuous support during this project. Many hours of debugging were involved in writing this master thesis project, but thanks to Florian Mendel, this project came to a release with nice results.

I would also like to thank Maria Eichlseder for her great support. Her unique way to ask questions brought me back on track several times. Mate Soos supported me during my bachelor thesis with SAT related issues and his support continued with this master thesis in private conversations.

Also thanks to Roderick Bloem and Armin Biere who organized a meeting one year before submitting this work defining the main approaches involved in this thesis.

And finally I am grateful for the support by Martina, who also supported me during good and bad days with this thesis, and my parents which provided a prosperous environment to me to be able to be where I am today.

Thank you!

All source code is available at lukas-prokop.at/proj/megosat and published under terms and conditions of Free/Libre Open Source Software. This document was printed with LuaLaTeXand Linux Libertine Font.

List of Figures

List of Tables

Contents

I

Chapter 1

Introduction

Hash functions are used as cryptographic primitives in many applications and protocols. They take an arbitrary input message and compute a hash value. Input message and hash value are considered as byte strings in a particular encoding. The hash value is of fixed length and satisfies several properties which make them useful to enable data and origin integrity.

Todo 1. *explain our attack approach using terminology used in the following*

1.1 Preliminaries

Definition 1 (Hash function)**.** A *hash function* is a mapping $h : X \to Y$ with $X = \{0,1\}^*$ and $Y = \{0,1\}^n$ for $n \in \mathbb{N}_{\geq 1}$.

- Let $x \in X$. $h(x)$ is called *hash value of* x.
- Let $h(x) = y \in Y$. y is called *preimage of* y.

One example showing the use of hash functions as primitives is PKCS #5 specified in RFC 2898 [3]. Section 5.2 specifies PBKDF1 and PBKDF2 using an arbitrary pseudorandom function to derive password-based keys. Hash algorithms can be used as those pseudorandom functions. Given a minimum iteration count of 1000, as defined in section 4.2, yields the additional requirement that fast computation of hash values for given $x \in X$ is desirable.

A hash function has to satisfy the following security requirements:

Definition 2 (Preimage resistance)**.** Given $y \in Y$. A hash function h is preimage resistant iff it is computationally infeasible to find $x \in X$ such that $h(x) = y$.

Definition 3 (Second-preimage resistance)**.** Given $x \in X$. A hash function h is second-preimage resistant iff it is computationally infeasible to find $x_2 \in X$ with $x \neq x_2$ such that $h(x) = h(x_2)$. x_2 is called *second preimage.*

Definition 4 (Collision resistance)**.** A hash function h is collision resistant iff it is computationally infeasible to find any two $x \in X$ and $x_2 \in X$ with $x \neq x_2$ such that $h(x) = h(x_2)$.

As far as hash functions accept input strings of arbitrary length, but return a fixed size output string, existence of preimages and collisions is unavoidable [10]. However, good hash functions make it very difficult to determine collisions or preimages.

1.2 Cryptanalysis of Hash Functions

In August 2004, Wang et al. published results at Crypto'04 [13] which revealed that MD4, MD5, HAVAL-128 and RIPEMD can be broken practically using differential cryptanalysis. On an IBM P690 machine, an MD5 collision can be computed in about one hour using this approach. Collisions for HAVAL-128, MD4 and RIPEMD were found as well. Patrick Stach's `md4coll.c` program [11] implements Wang's approach and can find MD4 collisions in few seconds on my Thinkpad x220 machine powering a .

Due to the birthday paradox, a collision attack has a generic complexity of $2^{\frac{n}{2}}$ whereas pre-image and second pre-image attacks have a generic complexity of 2^n.

Todo 2. *discuss their approach in more detail, introduce basics of diff crypt-analysis (w/o notation)*

1.3 Satisfiability

Definition 5. A *boolean function* is a mapping $h : X \to Y$ with $X = \{0, 1\}^n$ for $n \in \mathbb{N}_{\geq 1}$ and $Y = \{0, 1\}$.

The following definition gives three basic boolean functions:

Definition 6. *AND* is a boolean function mapping $X = \{0, 1\}^2$ to 1 if all values of X are 1. *OR* is a boolean function mapping $X = \{0, 1\}^2$ to 1 if any value of X is 1. *NOT* is a boolean function mapping $X = \{0, 1\}^1$ to 1 if the single value of X is 0. All functions return 0 in the other case.

Definition 7. A *truth table* unambiguously defines a boolean function by enlisting the evaluated truth value for all possible sets of inputs.

Table 1.1 shows truth tables for `AND`, `OR` and `NOT`.

v_1	v_2	$f(v_1, v_2)$
1	1	1
1	0	0
0	1	0
0	0	0

(a) AND

v_1	v_2	$f(v_1, v_2)$
1	1	1
1	0	1
0	1	1
0	0	0

(b) OR

v	$f(v)$
1	0
0	1

(c) NOT

Figure 1.1: Truth tables for AND, OR and NOT

In the following we discuss how boolean functions are related to general computations:

Definition 8. An *algorithm* is a step-wise set of instructions to solve a problem. An algorithm transforms a given input to some output. Turing machines can be utilized as computational model which is conjectured to be equivalent to Lambda Calculus.

Todo 3. *go on*

Theorem 1. Every algorithm can be represented as boolean function.

Todo 4. *go on*

1.4 Satisfiability

Todo 5. *intro*

Definition 9. A boolean function is *satisfiable* iff there exists at least one input $x \in X$ such that $h(x) = 1$.

1.5 Thesis Outline

This thesis is organized as follows:

In Chapter 1 we discussed the basic properties and fundamentals of the tools in discussion including hash functions and SAT solvers.

In Chapter 2 we introduce the MD4 hash function and discuss possible approaches in differential cryptanalysis.

In Chapter 3 we discuss SAT solving and potential approaches to speed up SAT solvers for cryptographic problems.

In Chapter 4 we show results of our work and discuss its implications.

Chapter 2

Differential cryptanalysis

2.1 MD4

MD4 is a cryptographic hash function originally described in RFC 1186 [7], updated in RFC 1320 [8] and obsoleted by RFC 6150 [12]. It was invented by Ronald Rivest in 1990 with properties given in Table 2.1. Since 1995 [2] successful attacks have been found to break collisions, preimage and second-preimage resistance in MD4; including but not limited to [9] and [5]. A Python 3 implementation derived from a previous Python version is available at github [6].

block size	512 bits	namely variable `block` in RFC 1320 [8]
digest size	128 bits	as per section 3.5 in RFC 1320 [8]
internal state size	128 bits	namely variables A, B, C and D
word size	32 bits	as per section 2 in RFC 1320 [8]

Table 2.1: MD4 hash algorithm properties

A short summary of MD4's design is given: Once padding and length extension has been applied, the message is split into 512-bit blocks (i.e. 16 32-bit words). Four state variables A, B, C and D are initialized with hexadecimal values:

[**A**] `01234567` [**B**] `89abcdef` [**C**] `fedcba98` [**D**] `76543210`

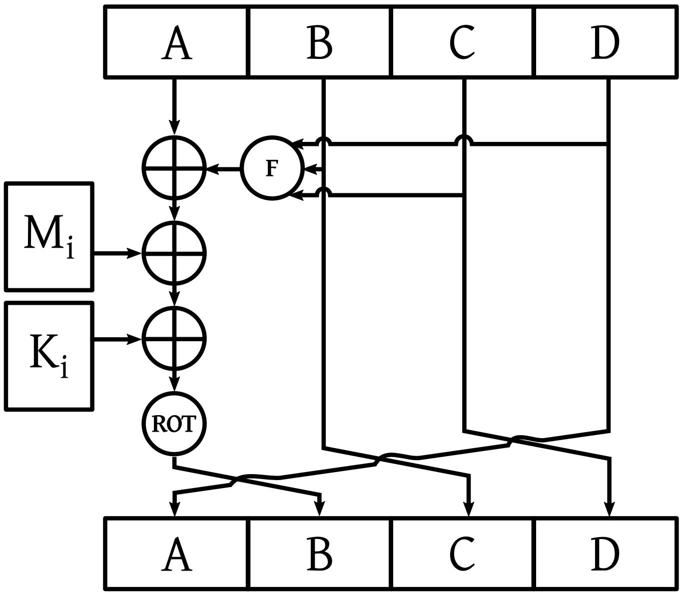

Figure 2.1: MD4 round function

To process one block, three auxiliary functions are defined:

$$F(X,Y,Z) = (X \wedge Y) \vee (\neg X \wedge Z) \tag{2.1}$$
$$G(X,Y,Z) = (X \wedge Y) \vee (X \wedge Z) \vee (Y \wedge Z) \tag{2.2}$$
$$H(X,Y,Z) = X \oplus Y \oplus Z \tag{2.3}$$

2.2 Differential notation

2.3 Addition example

2.4 Differential path

Chapter 3

Satisfiability

“What idiot called them logic errors rather than bool shit?”

3.1 Basic SAT solving techniques

3.2 SAT solvers in use

3.3 Encodings

Chapter 4

Results

4.1 Benchmark results

4.2 Related work

4.3 Conclusion

Appendices

.1 Testcases

Figures ??, ??, ?? and ?? show testcases used to test performance measures.

.2 Hardware setup

In the following we introduce two hardware setups which were used to run our testcases. The first setup is referred to as "Thinkpad x220" throughout the document whereas the second setup is referred to as "Cluster".

i		$\nabla S_{i,0}$	$\nabla S_{i,1}$	$\nabla S_{i,2}$
-4	A:	01100111010001010010001100000001		
-3	A:	00010000000110010010101000111010		
-2	A:	10011000101110101101110011111110		
-1	A:	11101111110011011010101110001001		
0	A:	x-------------------------------	W:	---x----------------------------
1	A:	--------------------------------	W:	--------------------------------
2	A:	---------------------x----------	W:	x-------------------------------
3	A:	xxx-----------------------------	W:	--------------------------------
4	A:	------------------------------xx	W:	x-------------------------------
5	A:	-------xxxxxxxxxxxxxxxx-x-------	W:	--------------------------------
6	A:	x--------x-----------x-x-xxxx--x	W:	--------------------------------
7	A:	---------x-----x----------------	W:	--------------------------------
8	A:	---------x-------x-x-x----------	W:	x-------------------------------
9	A:	-----------------x---x----------	W:	--------------------------------
10	A:	------x-----x--xxx-xxx----------	W:	--------------------------------
11	A:	x----------------xxx-x----------	W:	--------------------------------
12	A:	---x--x-------------------------	W:	x-------------------------------
13	A:	--------------------------------	W:	--------------------------------
14	A:	-x------------------------------	W:	--------------------------------
15	A:	x-x----------x------------------	W:	--------------------------------
16	A:	-xxx----------------------------		
17	A:	--------------------------------		
18	A:	--------------------------------		
19	A:	x-------------------------------		
20	A:	x-------------------------------		
21	A:	--------------------------------		
22	A:	--------------------------------		
23	A:	--------------------------------		
24	A:	--------------------------------		
25	A:	--------------------------------		
26	A:	--------------------------------		
27	A:	--------------------------------		
28	A:	--------------------------------		
29	A:	--------------------------------		
30	A:	--------------------------------		
31	A:	--------------------------------		
32	A:	x-------------------------------		
33	A:	--------------------------------		
34	A:	--------------------------------		
35	A:	--------------------------------		
36	A:	--------------------------------		
37	A:	--------------------------------		
38	A:	--------------------------------		
39	A:	--------------------------------		
40	A:	--------------------------------		
41	A:	--------------------------------		
42	A:	--------------------------------		
43	A:	--------------------------------		
44	A:	--------------------------------		
45	A:	--------------------------------		
46	A:	--------------------------------		
47	A:	--------------------------------		

Table 1: TODO description

i		$\nabla S_{i,0}$	$\nabla S_{i,1}$	$\nabla S_{i,2}$
-4	A:	01100111010001010010001100000001		
-3	A:	00010000001100100101010001110110		
-2	A:	10011000101110101101110011111110		
-1	A:	11101111110011011010101110001001		
0	A:	????????????????????????????????	W:	---x----------------------------
1	A:	????????????????????????????????	W:	--------------------------------
2	A:	????????????????????????????????	W:	x-------------------------------
3	A:	????????????????????????????????	W:	--------------------------------
4	A:	????????????????????????????????	W:	x-------------------------------
5	A:	????????????????????????????????	W:	--------------------------------
6	A:	????????????????????????????????	W:	--------------------------------
7	A:	????????????????????????????????	W:	--------------------------------
8	A:	????????????????????????????????	W:	x-------------------------------
9	A:	????????????????????????????????	W:	--------------------------------
10	A:	????????????????????????????????	W:	--------------------------------
11	A:	????????????????????????????????	W:	--------------------------------
12	A:	??????????????------------------	W:	x-------------------------------
13	A:	??????????????------------------	W:	--------------------------------
14	A:	??????????????------------------	W:	--------------------------------
15	A:	??????????????------------------	W:	--------------------------------
16	A:	???x----------------------------		
17	A:	?-------------------------------		
18	A:	?-------------------------------		
19	A:	?-------------------------------		
20	A:	x-------------------------------		
21	A:	--------------------------------		
22	A:	--------------------------------		
23	A:	--------------------------------		
24	A:	--------------------------------		
25	A:	--------------------------------		
26	A:	--------------------------------		
27	A:	--------------------------------		
28	A:	--------------------------------		
29	A:	--------------------------------		
30	A:	--------------------------------		
31	A:	--------------------------------		
32	A:	x-------------------------------		
33	A:	--------------------------------		
34	A:	--------------------------------		
35	A:	--------------------------------		
36	A:	--------------------------------		
37	A:	--------------------------------		
38	A:	--------------------------------		
39	A:	--------------------------------		
40	A:	--------------------------------		
41	A:	--------------------------------		
42	A:	--------------------------------		
43	A:	--------------------------------		
44	A:	--------------------------------		
45	A:	--------------------------------		
46	A:	--------------------------------		
47	A:	--------------------------------		

TABLE 2: TODO description

i		$\nabla S_{i,0}$	$\nabla S_{i,1}$	$\nabla S_{i,2}$
-4	A:	01100111010001010010001100000001		
-3	A:	00010000001100100101010001110110		
-2	A:	10011000101110101101110011111110		
-1	A:	11101111110011011010101110001001		
0	A:	????????????????????????????????	W:	---x----------------------------
1	A:	????????????????????????????????	W:	--------------------------------
2	A:	????????????????????????????????	W:	x-------------------------------
3	A:	????????????????????????????????	W:	--------------------------------
4	A:	????????????????????????????????	W:	x-------------------------------
5	A:	????????????????????????????????	W:	--------------------------------
6	A:	????????????????????????????????	W:	--------------------------------
7	A:	????????????????????????????????	W:	--------------------------------
8	A:	????????????????????????????????	W:	x-------------------------------
9	A:	????????????????????????????????	W:	--------------------------------
10	A:	????????????????????????????????	W:	--------------------------------
11	A:	????????????????????????????????	W:	--------------------------------
12	A:	????????????????????????????????	W:	x-------------------------------
13	A:	????????????????????????????????	W:	--------------------------------
14	A:	????????????????????????????????	W:	--------------------------------
15	A:	????????????????????????????????	W:	--------------------------------
16	A:	????????????????????????????????		
17	A:	????????????????????????????????		
18	A:	????????????????????????????????		
19	A:	????????????????????????????????		
20	A:	????????????????????????????????		
21	A:	--------------------------------		
22	A:	--------------------------------		
23	A:	--------------------------------		
24	A:	--------------------------------		
25	A:	--------------------------------		
26	A:	--------------------------------		
27	A:	--------------------------------		
28	A:	--------------------------------		
29	A:	--------------------------------		
30	A:	--------------------------------		
31	A:	--------------------------------		
32	A:	x-------------------------------		
33	A:	--------------------------------		
34	A:	--------------------------------		
35	A:	--------------------------------		
36	A:	--------------------------------		
37	A:	--------------------------------		
38	A:	--------------------------------		
39	A:	--------------------------------		
40	A:	--------------------------------		
41	A:	--------------------------------		
42	A:	--------------------------------		
43	A:	--------------------------------		
44	A:	--------------------------------		
45	A:	--------------------------------		
46	A:	--------------------------------		
47	A:	--------------------------------		

TABLE 3: TODO description

i		$\nabla S_{i,0}$	$\nabla S_{i,1}$	$\nabla S_{i,2}$
-4	A:	01100111010001010010001100000001		
-3	A:	00010000001100100101010001110110		
-2	A:	10011000101110101101110011111110		
-1	A:	11101111110011011010101110001001		
0	A:	????????????????????????????????	W:	????????????????????????????????
1	A:	????????????????????????????????	W:	????????????????????????????????
2	A:	????????????????????????????????	W:	????????????????????????????????
3	A:	????????????????????????????????	W:	????????????????????????????????
4	A:	????????????????????????????????	W:	????????????????????????????????
5	A:	????????????????????????????????	W:	????????????????????????????????
6	A:	????????????????????????????????	W:	????????????????????????????????
7	A:	????????????????????????????????	W:	????????????????????????????????
8	A:	????????????????????????????????	W:	????????????????????????????????
9	A:	????????????????????????????????	W:	????????????????????????????????
10	A:	????????????????????????????????	W:	????????????????????????????????
11	A:	????????????????????????????????	W:	????????????????????????????????
12	A:	????????????????????????????????	W:	????????????????????????????????
13	A:	????????????????????????????????	W:	????????????????????????????????
14	A:	????????????????????????????????	W:	????????????????????????????????
15	A:	????????????????????????????????	W:	????????????????????????????????
16	A:	????????????????????????????????		
17	A:	????????????????????????????????		
18	A:	????????????????????????????????		
19	A:	????????????????????????????????		
20	A:	????????????????????????????????		
21	A:	--------------------------------		
22	A:	--------------------------------		
23	A:	--------------------------------		
24	A:	--------------------------------		
25	A:	--------------------------------		
26	A:	--------------------------------		
27	A:	--------------------------------		
28	A:	--------------------------------		
29	A:	--------------------------------		
30	A:	--------------------------------		
31	A:	--------------------------------		
32	A:	x???????????????????????????????		
33	A:	--------------------------------		
34	A:	--------------------------------		
35	A:	--------------------------------		
36	A:	--------------------------------		
37	A:	--------------------------------		
38	A:	--------------------------------		
39	A:	--------------------------------		
40	A:	--------------------------------		
41	A:	--------------------------------		
42	A:	--------------------------------		
43	A:	--------------------------------		
44	A:	--------------------------------		
45	A:	--------------------------------		
46	A:	--------------------------------		
47	A:	--------------------------------		

TABLE 4: TODO description

Type model	Thinkpad Lenovo x220 tablet, 4299-2P6
Processor	Intel i5-2520M, 2.50 GHz, dual-core, Hyperthreaded
RAM	16 GB (extension to common retail setup)
Memory	160 GB SSD
L3 cache size	3072 KB

TABLE 5: Thinkpad x220 Tablet specification [4]

Processor	Intel Xeon X5690, 3.47 GHz, 6 cores, Hyperthreaded
RAM	192 GB
L3 cache size	12288 KB

Table 6: Cluster node nehalem192go specification [1]

Index

Bibliography

[1] Intel Corporation. *Intel Xeon Processor X5690 (12M Cache, 3.46 GHz, 6.40 GT/s Intel QPI) Specifications.* URL: http://ark.intel.com/products/52576/Intel-Xeon-Processor-X5690-12M-Cache-3_46-GHz-6_40-GTs-Intel-QPI (visited on 04/05/2016).

[2] Hans Dobbertin. "Cryptanalysis of MD4". In: *Journal of Cryptology* 11.4 (1998), pp. 253–271. ISSN: 1432-1378. DOI: 10.1007/s001459900047. URL: http://dx.doi.org/10.1007/s001459900047.

[3] B. Kaliski. *PKCS #5: Password-Based Cryptography Specification Version 2.0.* RFC 2898. The Internet Engineering Task Force, 2000, pp. 9–11. URL: https://tools.ietf.org/html/rfc2898#section-5.2 (visited on 03/29/2016).

[4] Lenovo Group Ltd. *ThinkPad X220 Tablet (4299) - Onsite (2011).* URL: http://www.lenovo.com/shop/americas/content/pdf/system_data/x220t_tech_specs.pdf (visited on 04/05/2016).

[5] Yusuke Naito et al. "Improved Collision Attack on MD4". In: (2005), pp. 1–5. URL: http://eprint.iacr.org/.

[6] prokls. *MD4 in pure Python 3.4.* URL: https://gist.github.com/prokls/86b3c037df19a8c957fe.

[7] Ronald Rivest. *The MD4 Message Digest Algorithm.* RFC 1186. The Internet Engineering Task Force, 1990, pp. 1–18. URL: https://tools.ietf.org/html/rfc1186.

[8] Ronald Rivest. *The MD4 Message-Digest Algorithm.* RFC 1320. The Internet Engineering Task Force, 1992, pp. 1–20. URL: https://tools.ietf.org/html/rfc1320.

[9] Yu Sasaki et al. "New Message Difference for MD4". In: (2007), pp. 1–20. URL: http://www.iacr.org/archive/fse2007/45930331/45930331.pdf.

[10] Martin Schläffer and Elisabeth Oswald. "Searching for differential paths in MD4". In: *Fast Software Encryption.* Springer. 2006, pp. 242–261.

[11] Patrick Stach. *MD4 collision generator.* URL: http://crppit.epfl.ch/documentation/Hash_Function/Fastcoll_MD4/md4coll.c (visited on 04/05/2016).

[12] S. Turner and L. Chen. *The MD4 Message Digest Algorithm*. RFC 6150. The Internet Engineering Task Force, 2011, pp. 1–10. URL: https://tools.ietf.org/html/rfc6150 (visited on 03/15/2016).

[13] Xiaoyun Wang et al. “Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD.” In: *IACR Cryptology ePrint Archive* 2004 (2004), p. 199.