technische universität
dortmund

Faculty for Computer Scinece

Bachelor's Thesis

Collsisions in Hash-Functions

Marco Bellmann

First Reviewer:
Prof. Dr. Coja-Oghlan

Second Reviewer:
Msc. Arnab Chatterjee

Dortmund, August 2023

Acknowledgements

Start

Contents

Chapter 1

Introduction

This thesis is about a deeper look on MD5. We take a closer look at the Master thesis of M. Stevens: a fast collisions finding algorithm [1]. The goal is to work out a more clear and understandable code, which is not necessarily faster, to reevaluate the code on modern systems and the difference to SHA1.

1.1 Hash Function

A *Hash Function* is something

1.2 Notation / Preliminaries

Pages

Chapter	Pages	%	content
1	2	5	Intro
2	5	14	Basic MD5
3	6	20	coll-theroy
4	9	30	colls
5	4	14	Othercolls
6	4	14	SHA1
7	1	3	Outro
A-D	0	0	Additions
sum	$\sim$30	$\sim$ 100	

Chapter 2

About MD5

2.1 Definition

MD5 stands for *Message-Digest Algorithm 5* since it generates a digest for any given text. MD5's output length is 128 Bit, represented in hexadecimal. Since the amount of possible text it close to and the amount of MD5s is limited by 32^{16}, some text may have the MD5. If two different inputs create have the same hash value, that is what we call a collision. MD5 is usually seen as 4 steps as seen in fig md5

2.2 Algorithm

As mention before, we can describe the MD5 hash algorithm in four steps (Did I?). The implementation of the padding is the least interesting, yet there are many mistakes one can make. To archive an adequate comparison we need build simply ignore the process of padding and pretend to expect the padding to be done, even thou we do it ourselves. The processing step is were the codes vary the most. Stevens works a lot with global variables. All in all his code is extremely optimized for performance. The MD5 sum is build up like:

1. padding
2. procssesing
3. md5 sum
4. output

Figure 2.1: MD5 algo

$$\begin{aligned}
F_t &= f\left(Q'_t, Q'_{t-1}, Q'_{t-2}\right) \\
T_t &= F_t + Q'_{t-3} + W_t \\
R_t &= RL\left(T'_t, RC_t\right) \\
Q_{t+1} &= Q_t + R_t
\end{aligned}$$

Figure 2.2: MD5 sum

2

3

4

5

Chapter 3

Collision Finding

3.1 About collisions

Hash functions have resistance:

1. First Preimage Resistance: for a hash function $h(m) = H$ the massage m is hard to find.
2. Second Preimage Resistance: for a given messages m_1 it is hard to find an m_2 with $m_1 \neq m_2$ and $h(m_1) = h(m_2)$.
3. Collision Resistance: two arbitrary messages m_1 and m_2 with $\neq m_2$ and $h(m_1) = h(m_2)$ are hard to find.

3.2 Differential Path

Explain the genereal idea of the bit conditions first. There is a part I want to talk about the diffeces of Wang and Steven. In this part I can tear the details of the differential paths apart.

Stevens starts with Wang's attack, which tries to find to pairs of blocks: (B_0, B_0') and (B_1, B_1') that $IHV = IHV'$, with the goal to create two massages M and M', with the same hash value:

$$\begin{tikzcd}
IHV_0 \arrow[r, "M_{(1)}"'] & \cdots \arrow[r, "M_k"'] & IHV_k \arrow[r, "B_0"'] & IHV_{k+1} \arrow[r, "B_1"'] & IHV_{k+2} \arrow[r, "M_{k+1}"'] & \cdots \arrow[r, "M_N"'] & IHV_N \\
= & & = & \neq & = & & = \\
IHV_0 \arrow[r, "M_{(1)}"'] & \cdots \arrow[r, "M_k"'] & IHV_k \arrow[r, "B_0"'] & IHV'_{k+1} \arrow[r, "B_1"'] & IHV'_{k+2} \arrow[r, "M_{k+1}"'] & \cdots \arrow[r, "M_N"'] & IHV_N
\end{tikzcd}$$

$$
\begin{aligned}
\delta F_t &= f\left(Q'_t, Q'_{t-1}, Q'_{t-2}\right) \\
\delta T_t &= \delta F_t + \delta Q'_{t-3} + \delta W_t \\
\delta R_t &= RL\left(T'_t, RC_t\right) - RL\left(T_t, RC_t\right) \\
\delta Q_{t+1} &= \delta Q_t + \delta R_t
\end{aligned}
$$

Figure 3.1: Add difference

The idea to manipulate a block B such that $Q_1 \ldots Q_{16}$ maintain their conditions and that $Q_1 7$ to some Q_k do not change at all. We try to make k as large as possible.

3.3 Bit Conditions

Again, try to keep it theoretical, give an overview what we want to do with the bit conditions and safe the details for the comparison between Steven and Wang.

Bit conditions describe the differential path on bits. We need the bit conditions to avoid a carry, so a manipulation in step t stays in step t and does not propagate beyond the 31st bit. We look at conditions and restrictions. The restrictions leads to conditions, which we calculate in the following. A restriction e.g. $\Delta T_2\,[\;31] = +1$ leads to conditions $Q_1\,[16]\,Q_2\,[16] = Q_3\,[15] = 0$ and $Q_2[15] = 1$. Notice, conditions are on $\Delta T_t[i]$ a state in md5-algorithm before the rotation and restrictions are on $Q_t[i]$ states of the md5-algorithm after the rotation.
We calculate the bit conditions by using the Add-Difference for two massage blocks containing tow blocks $N|M$ and $N'|M'$. The XOR-Difference is useful, too.

$$
\begin{aligned}
\delta X &= X' - X\,(mod 32) \;\text{ Add-Difference} \\
\Delta X &= X' \oplus X \text{ XOR-Difference} \\
\lambda\,[i] &= \text{ our guess for the ith bit: } X\,[i]
\end{aligned}
$$

$$
\text{if } \Delta X = \lambda \Rightarrow \delta x \text{ can be determined}
$$

For $\lambda\,[i]$ we only need to consider $i < 31$, since $X\,[31]$ as msb always creates a add difference of 2^{31}.

We calculate a δ for each f_t, Q_t, T_t and R_t for our add difference, to calculate Q_{t+1} . Additional we need the rotation constant RC for each t. In general we begin with the f_t since we want f_t to be in a particular state. Since we want to avoid a carries in our calculation

1. $t \in \{0, 1, 2, 3\}$:
   $Q_t = 0$ since here is no influence by an message and no calculation of f, there is nothing to change:

2. $t = 4$
   $\Delta T_4 = -2^{31}$, because we must not have a carry, we *lock* the last bit. Since $RL(T_4, RC_4) = RL(-2^{31}, 7) = -2^6$ and $\delta Q_4 = 0 \Rightarrow \delta Q_5 = -2^6$

for an example we can use the detailed calculations of Wang's or maybe used our own, for a first understanding Wang's might be better. Again, explain the bit conditions and not the differences between Steven's and Wang's Code.

t	$RC(t)$	$AC(t)$
0	7	
1	12	
2	17	
3	22	
4	7	
5	12	
6	17	
7	22	
8	7	
9	12	
10	17	
11	22	
12	7	
13	12	
14	17	
15	22	

Figure 3.2: Overview of each rounds constants

input

3.4 Second Preimage Resistance

Wang's attack of the MD5 bases on two algorithms. One processes the first and the other the second block of an arbitrary massage, without loss of generality (w.l.o.g.). We call it first block. The first algorithm finds to the given first input block's IHV another IHV so that both blocks have the same new IHV. Again it is important to notice that both blocks differ only a little form each other. In our approach we reconstructed the algorithm of Wang fitting our implementation of the *MD5-Algorithm* and the code of Steven's attack. We managed to provoke a second preimage resistance failure.

3.5 About Code

There are multiple approaches to find the second block. Stevens used in his code five. Four written by himself. They all have similarities, especially in the beginnings. The first *find-first-block* algorithm though, is the algorithm by Wang. His additional algorithms are "just" improvements, if the algorithm of Wang does not success. Since the similarities we only go in Wang's algorithm with a deeper view and explain some improvements eventually.
The question: "*If there are "better" algorithms by Stevens, why start with Wang's anyway?*" may come up, yet we have to clarify what better could. The algorithm of Wang is quite efficient, but also in very minimalistic. The quote I have to check is, *if the code by Wang finds a first block, it is in the most efficient way*, but it does not always find a first block were some is. So using the algorithm by Wang as a first search algorithm, may increases our performance. It is also an more easy way to implement working code, since the algorithms of Stevens tend to be more complex, but also building up on Wang's.
The first 16 Qs can be chosen arbitrary, as long as we fulfill the conditions. Stevens does this by generating really good random values. After this he alters the random values so thy fulfill conditions. We follow Stevens approach but generate "normal" random values and alter these so they fulfill the conditions Improvements for random number generation are possible.
Example for Stevens bit manipulation for Q_t with $t = 3$:

1. the values to set the zeros (bitwise-and with $0xfe87bc3f$)
2. the values to set the ones (bitwise-or with $0x017841c0$)
3. the new bit conditions
4. the old bit conditions

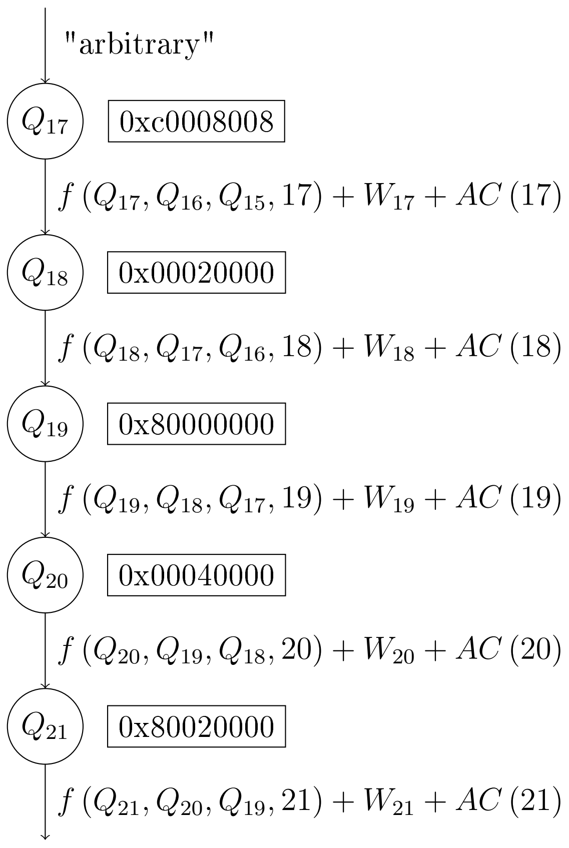
1. AND	11111110	10000111	10111100	00111111	$0xfe87bc3f$
2. OR	00000001	01111000	01000001	11000000	$0x017841c0$
3.	....	.1111...	.1....01	11......	
4.	....	....0...	....0...	.0......	

the & flips the 0 correct, the || flips the 1 correct
We now calculate the massage m_t for $t \in \{0, 6, \ldots, 15\}$. For this we use the reverse md5 function:

$$m_t = RR\left(Q_{t+1} - Q_t, RC_t\right) - f_t\left(Q_t, Q_{t-1}, Q_{t-2}\right) - Q_{t-3} - AC(t)$$

This function comes up by simply processing the MD5 sum 2.2 backwards. We try to pick a Q_{17} that $Q_{18}, \ldots, Q_{21}$ can be calculated with $Q_{17}, \ldots, Q_{15}$ and fulfill their conditions. We can solve this by looping about the calculation as long as the conditions are not fulfilled and then brake. Note: we may calculate the values in temp variable before pasting them into the actual Q.

For a good pick for Q_{17} we just generate a random value and force the conditions manual.


"arbitrary"
Q_{17}
0xc0008008
$f\left(Q_{17}, Q_{16}, Q_{15}, 17\right) + W_{17} + AC\left(17\right)$
Q_{18}
0x00020000
$f\left(Q_{18}, Q_{17}, Q_{16}, 18\right) + W_{18} + AC\left(18\right)$
Q_{19}
0x80000000
$f\left(Q_{19}, Q_{18}, Q_{17}, 19\right) + W_{19} + AC\left(19\right)$
Q_{20}
0x00040000
$f\left(Q_{20}, Q_{19}, Q_{18}, 20\right) + W_{20} + AC\left(20\right)$
Q_{21}
0x80020000
$f\left(Q_{21}, Q_{20}, Q_{19}, 21\right) + W_{21} + AC\left(21\right)$

6

Chapter 4

Differences between Steven and Wang

Here we want to look at the differences between the attempts and results of Steven and Wang and compare them. Maybe put in our own results and maybe an opinion. Important, this is not an conclusion. We need to phrase a concrete problem to answer the question if this project was successful.

2

3

4

5

6

7

8

9

Chapter 5

Othercolls

There is the birthday attack and it bases on the birthday paradox, this is something to write about. There is probably more. Just general things, maybe it is possible to test some of this.

2

3

4

Chapter 6

SHA1

Explain the *SHA1 - Algorithm.* Explain the difference to the *MD5 - Algorithm.* Pick up the concept of tunnels, explain why the tunnels are possible and why it is not in *SHA1*.

2

3

4

Chapter 7

Conclusion

Pick up: what was your intention? What did you discover? What did you expect and than put your results in context of with your expectations. How fast was your code? Was it about speed anyways?

Bibliography

Appendix A

Tables

+—+
|abc|
+—+

Appendix B

Code

a +=a;

Notes

B.1 Notation Helper and Ideas

$RR(X,Y) :\equiv X \circledast Y$
$RL(X,Y) :\equiv X \circledast Y$
$X \oplus Y :\equiv X"XOR"Y$

$$10000000000000000000000000000001 \circledast 7 \equiv 00000000000000000000000011000000$$

Notation

Stevens	Wang	Definition
$h(m) = H$		the hash H of some message m
$RL(X,Y)$	$ROTL^{Y}(X)$	cyclic left shift X by Y (mod 31)
$RR(X,Y)$	-	cyclic right shift X by Y (mod 31)
$RC(t)$	$S(t)$	rotation Constant of t
Block 1, Block 2	Block N, Block M	pair of first blocks for collisions finding
Block 0, Block 1	Block N, Block M	same pair but im Code

List of Figures

Algorithmenverzeichnis

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet sowie Zitate kenntlich gemacht habe.

Dortmund, den July 3, 2023

Muster Mustermann