**Final Report**

COMP520-18Y (HAM)

**Automated Searching for  
Differential Characteristics in SHA-2**

*Mitchell Grout*

Supervisors:

Associate Professor Ryan K L Ko

Aleksey Ladur

Cameron Brown

30th August, 2018

**Abstract:** TODO

**Acknowledgements**

This project was only possible with the help of many members from the Faculty of Computing and Mathematical Sciences, and I would like to take this time to personally thank each and every one of you. I would firstly like to give my thanks to my supervisors, Associate Professor Ryan Ko, Aleksey Ladur, and Cameron Brown. Your suggestions, support, and advice have given me the confidence to research this difficult field and helped make this project as successful as it is.

I would like to thank Daniel Roodt of CROW for his expert advice on both cryptographic hash functions, as well as their cryptanalysis. You have provided an amazing amount of support and have helped me understand so much about cryptanalysis.

To Dr. Steven Miller, your statistical expertise has helped me to find the most appropriate ways to test my results, and ensure that there is no ambiguity in the results?

Finally, I would like to thank all of my fellow honours students, Christian Anderson, Daniel St. George, Christopher Chew, and Armajot Parmar, as well as all of my fellow CROWs. Your support and suggestions throughout my project has helped make this project both a success, and it has been a pleasure to work with you all.

Contents

[1. Introduction 6](#_Toc524210449)

[1.1. SHA-2 (Define terms) 6](#_Toc524210450)

[1.2. Motivation (Why is cryptanalysis important) 6](#_Toc524210451)

[1.3. Aim (Project statement) 6](#_Toc524210452)

[1.4. Importance (Literature Review) 6](#_Toc524210453)

[1.5. Overview (Description of remaining content in report) 6](#_Toc524210459)

[2. Background 6](#_Toc524210460)

[2.1. Hash Functions 6](#_Toc524210462)

[2.1.1. Properties 6](#_Toc524210463)

[2.2. Cryptographic Hash Functions 6](#_Toc524210464)

[2.2.1. Properties 6](#_Toc524210465)

[2.2.2. Broken Cryptographic Hash Functions 7](#_Toc524210471)

[2.3. MD5 7](#_Toc524210472)

[2.4. RIPEMD 7](#_Toc524210473)

[2.5. SHA-1 7](#_Toc524210480)

[2.6. Differential Cryptanalysis 7](#_Toc524210488)

[2.7. Optimization Algorithms 7](#_Toc524210489)

[2.7.1. Heuristics 7](#_Toc524210490)

[2.7.2. Random Search 7](#_Toc524210491)

[2.7.3. Genetic Algorithms 7](#_Toc524210492)

[3. Design 7](#_Toc524210493)

[3.1. Propagations 7](#_Toc524210497)

[3.1.1. Component-wise 7](#_Toc524210498)

[3.1.2. Function-wise 8](#_Toc524210499)

[3.1.3. Caveats 8](#_Toc524210500)

[3.2. Heuristics 8](#_Toc524210501)

[3.3. Random Search 8](#_Toc524210502)

[3.4. Genetic Algorithms 8](#_Toc524210503)

[4. Implementation 8](#_Toc524210507)

[5. Evaluation 8](#_Toc524210508)

[6. Conclusion 9](#_Toc524210509)

[7. References 10](#_Toc524210510)

[8. Appendices 11](#_Toc524210511)

[A. Project Source Code 12](#_Toc524210512)

# Introduction

## SHA-2 (Define terms)

Standardized under FIPS180-2 (currently FIPS180-4) by the NSA. Class of cryptographic hash functions superseding SHA-1. Used in a variety of applications such as digital signatures, proof of work, file integrity, and password storage.

## Motivation (Why is cryptanalysis important)

Many hash functions have come before SHA-2; many have been broken (MD5, RIPEMD, SHA-0, SHA-1). Hash functions are security-critical, we need assurance that a malicious actor cannot subvert them in a tractable amount of time / with a reasonable amount of computational power.

## Aim (Project statement)

Look at the existing framework that is differential cryptanalysis, determine problems or gaps of knowledge in the public sphere of research, and attempt to find solutions

## Importance (Literature Review)

Lack of public information on differential cryptanalysis of hash functions; most data is masters/PhD reports, or conference proceedings. Prior is indigestible in terms of content and sparse in terms of availability, latter does not have enough data and is not thorough in explanations. Difficult for new people to enter the field due to lack of knowledge.



## Overview (Description of remaining content in report)

Researched a whole host of current attacks on SHA-2 as well as past hash functions. Determined that differential cryptanalysis was promising, attempted to learn more about it. Fell flat when it came to finding any good material for it, especially delta-0. Working with SHA-2 is too extreme and difficult to manage; we create our own reduced SHA-2 variant which is both quicker to work with, and easier to understand. Attempt to use computational methods to find delta-0 for reduced variant with reasonable success. Hypothesis holds, but difficult to compare our actual fitness function to a “true” fitness due to tractability. Technique appears promising.

# Background



## Hash Functions

We define a hash function to be a pure mathematical function, which takes an arbitrary length input, and produces some fixed length output. The input to such functions is referred to as the message, and the output is referred to as the digest, or hash value associated with the message. Typically the input is some block of memory, while the output is typed as an integer. It is important to note that such functions are not unique mappings. As the input space for a hash function is arbitrary, and the output space is finite, there is theoretically infinitely many values and such that . This is true regardless of the construction.

A cryptographic hash function is a special class of hash functions which satisfy three key properties:

* Preimage resistance: Given a hash value , it is infeasible to find a message such that
* Second preimage resistance: Given a message with hash value , it is infeasible to find a second message such that , and
* Collision resistance: It is infeasible to find a message , such that , and

By infeasible, we mean that there is no better method than to brute-force search the input space of . One exception however is collision resistance;

If a cryptographic hash function can be shown to violate any of these three properties, typically by devising an algorithm that runs faster than brute force, we consider the hash function broken.

Hash functions are maps from . Effectively give a ‘fingerprint’ for the input ‘message’. Domain is infinite, codomain is finite, hence by extreme pigeonholing there must be at least one which map to the same output.

### Properties

Quick to compute; deterministic. Not required but sometimes assumed to be uniformly distributed, continuous, and non-invertible.

## Cryptographic Hash Functions

Strict subset of class of hash functions. Have stringent properties that make them more suitable for sensitive tasks.

### Properties

Uniformly distributed, non-continuous (avalanche effect), and the three resistances.

An ideal CHF cannot be constructed; it is a random oracle. No possible way to correlate an output with an input, and no possible way to correlate two inputs in any way; no better way to break the resistances than via brute force. Clearly not constructible. So, any CHF we construct cannot truly satisfy these properties, only get close to.



### Broken Cryptographic Hash Functions

A CHF is considered broken when at least one of its properties is violated.

## MD5

Was historically considered to be the first CHF up until 2004 (MD4 came before but was DOA). Broken by Wang et al using differential cryptanalysis.

## RIPEMD

Almost like running two MD4 instances in parallel and sharing data, arguably more complex than MD5. Also broken by Wang et al in the same year using differential cryptanalysis.



## SHA-1

Standardized by FIPS??? By NSA/NIST, broken by CWI/Google in 2017 as part of the SHAttered attack. Used differential cryptanalysis.



## Differential Cryptanalysis

A framework for breaking collision resistance (easiest of the three resistances arguably). Traces differences in inputs/outputs through each of the internal components. Aim is to find an output difference of 0 with high probability, find sufficient conditions for it to hold, and deal with contradictions.

## Optimization Algorithms

Finding delta-0 isnt obvious in many circumstances; can optimization algorithms be applied here?

### Heuristics

How can we determine the fitness of delta-0? Currently using zero/total and probability.

### Random Search

With no other information, can we guess-and-check values? This gives the added benefit of giving us an idea of what the search space ‘looks’ like, i.e. are there typically many ‘good’ solutions?

### Genetic Algorithms

# Design



## Propagations

Propagations refer to taking a set of input differences and determining the possible output differences, along with their frequency

### Component-wise

Let be the set of inputs (-dimensional) and be the set of outputs for a component given by . Say that is a difference. The propagation of is . We can count frequencies by determining how many times a particular term appears in this set. This is out of . Typically is too large to iterate over so random sampling is necessary in most non-trivial cases. Certain functions are linear in the sense that , i.e. they are -homomorphisms. In this case, , hence , so linear functions are constant on propagations. Such examples are the functions.

### Function-wise

A function is composed of many components, which will have inputs interlinked in various ways. All of these are ‘seeded’ by an initial difference, so the propagation itself is entirely deterministic. If we view our function as , propagation is as follows:

* Propagate the initial difference through to get a set
* For each element in , propagate it through to get a set
* Repeat until is obtained; this is the output.

### Caveats

Clearly this is an EXPSPACE problem. Along with issues with sampling, there are issues with actually propagating every possible value. As such we limit ourselves to ‘statistically significant’ results, i.e. output differences that have a frequency greater than some threshold probability.

As suggested by some papers, we should ensure that message expansion has a sufficient number of 0s in it [Schlaffer], and make sure our input differences only have bits set in the last few words.

## Heuristics

We need some way of determining how fit a particular value of is. We have two techniques; We can look at , which gives an indication of whether or not the number of zero trails dominates or not. Second technique is to look at the probability of any of these 0s actually being hit. Both heuristics have issues which will be explored later.

## Random Search

The first and simplest optimization algorithm in existence is random search, or guess-and-check. We randomly sample , attempt to propagate it, and then gauge its fitness.

## Genetic Algorithms

Assuming that the fitness function is typically not locally continuous, genetic algorithms will typically thrive.

- Approach to the problem – likely to refer to Background

- Overall Solution/Approach Design



# Implementation

- Technologies, libraries, schematics, data flow

# Evaluation

- Does it work, does it meet expectations/requirements

# Conclusion

- Drawing together outcomes and original plan. Future work.

1. References: Todo with Endnote
2. Appendices: Todo

# Project Source Code