

Mitchell Grout

COMP520-18Y (HAM)

This report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computing and Mathematical Sciences with Honours (BCMS(Hons))  
at The University of Waikato.

Automated Searching for  
Differential Characteristics   
in SHA-2

This report is submitted in partial fulfilment of the requirements for the degree of Bachelor of Computing and Mathematical Sciences with Honours (BCMS(Hons)) at The University of Waikato.

© 2018 Mitchell Grout

Abstract

The abstract should provide a summary of the key aspects of the report. It should mention important details such as data sources, methods and the main results. It should be standalone: as abstracts are often read independently of the full report. It should fit on this one page. Do not include references: as reference lists may not be present when reading the abstract. Two or three paragraphs is a typical length/structure.

It is a long-established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

It is a long-established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

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 them. 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 has given me the confidence to research this difficult fiend 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 this topic.

To Dr. Steven Miller, your statistical expertise has helped me to find the most appropriate ways to test my results, and to ensure that there is no room for misinterpretation of my 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 a success; it has been a pleasure to work with you all.

Contents

[List of Figures i](#_Toc524265896)

[List of Tables ii](#_Toc524265897)

[1. Introduction 3](#_Toc524265898)

[1.1 SHA-2 3](#_Toc524265899)

[1.2 Motivation 3](#_Toc524265900)

[1.3 Aim 3](#_Toc524265901)

[1.4 Importance 3](#_Toc524265902)

[1.5 Overview 3](#_Toc524265903)

[2. Background 4](#_Toc524265904)

[2.1 Hash Functions 4](#_Toc524265905)

[2.1.1 Generalized Birthday Paradox 5](#_Toc524265906)

[3. Design 6](#_Toc524265907)

[4. Results 7](#_Toc524265908)

[5. Discussion 8](#_Toc524265909)

[6. Conclusion 9](#_Toc524265910)

[References 10](#_Toc524265911)

[Appendices 11](#_Toc524265912)

[A: Additional Screenshots 11](#_Toc524265913)

# List of Figures

[Figure 2.1 Screenshot of the navigation structure of www.cs.waikato.ac.nz 7](file:///G:\My%20Drive\Dept%20Admin\programme%20convenor\report%20template\report-template.docx#_Toc514922191)

[Figure 3.1 Another screenshot 18](file:///G:\My%20Drive\Dept%20Admin\programme%20convenor\report%20template\report-template.docx#_Toc514922192)

# List of Tables

Table 2.1 Summary of Figures and Tables 9

Table 2.2 Another table 9

# Introduction

## Context

Cryptographic hash functions are a cryptographic primitive, alongside other primitives such as encryption algorithms and random number generators. As primitives, they are primarily used within other applications to help perform particular tasks, such as authentication or secure storage. Over the years, cryptographic hash functions have found a use in secure password storage, file integrity checks, and proof of work systems.

https://en.wikipedia.org/wiki/Comparison\_of\_cryptographic\_hash\_functions

The SHA-2 family of cryptographic hash functions is a collection of functions originally standardized by NIST as a Federal Information Processing Standard document, FIPS180-2. This family includes functions like SHA-256 and SHA-512, as well as lesser known variants such as SHA-224 and SHA-386. As they are part of the same family, all SHA-2 functions share the same internal structure, with the only difference being the size of registers, constants, and block sizes.

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

## Motivation

As cryptographic hash functions, SHA-2 sees use in security-critical areas such as file integrity checks, password storage, and identification.

## Aim

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

## Importance

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

## Overview

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

# 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. We often denote this with the signature for some , where , is all vectors over of length , and is all arbitrary vectors over . The input of such a function 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 viewed as some block of memory, while the hash value is viewed as some -bit integer.

A common application of hash functions is when a ‘fingerprint’ needs to be associated with some value. A typical example would be a hash table. This is a generalization of an array, in which the table key can be any arbitrary type. To allow for this, the hash value of the key is taken, and used to perform the usual integer-based lookup in an array.

class HashTable<Key,Value>

{

protected Value[] m\_arr = new Object[13];

public Value get(Key key) { return m\_arr[key.getHashCode() % m\_arr.length()]; }

public void set(Key key, Value value) { m\_arr[key.getHashCode() % m\_arr.length()] = value; }

}

HashTable<> table = new HashTable<String, String>();

table.set(“Foo”, “Bar”);

table.get(“Foo”); // returns “Bar”

Due to their definition, all hash functions admit collisions. A collision is a pair of messages satisfying , . We will refer to this pair as a colliding pair. This is a consequence of the domain of the hash function being infinite, while the codomain is finite. In applications such as hash tables, a collision is a non-issue. In the case of a collision, we can resize the underlying array, add the colliding values to a linked list, or simply add an offset to the hash value. However, in security-critical applications, in which the fingerprint is assumed to be unique, this can cause issues.

### Properties

General hash functions have very lax properties. By definition, a hash function must be able to take an arbitrary input and produce some fixed length output. Furthermore, as pure functions, they must be entirely deterministic, meaning repeated calls with the same input should always produce the same fingerprint, regardless of any external state. Although not a strict requirement, many hash functions have the following three properties:

* Fast: It should be quick to execute the hash function for any given message.
* Uniformly distributed: Each element in the codomain should have an approximately equal chance of occuring
* Continuity: If two inputs , differ by only a small value, then so should ,

These conditions are primarily focused around non-security critical applications, and are important for fast performance, specifically with regards to applications such as indexing.

## Cryptographic Hash Functions

A cryptographic hash function is a special type of hash function, with properties specifically tailored for security. However, many of the properties of ordinary hash functions are not desirable for cryptographic hash functions, so additional structure is required.

### Properties

A cryptographic hash function is defined to have three inherent properties:

* Fast: It should be quick to execute the hash function for any given message.
* Uniformly distributed: Each element in the codomain should have an approximately equal chance of occuring
* Discontinuity: Also referred to as the avalanche effect, if two inputs , differ by only a small value, then , should differ by a large value. Typically if , differ by a single bit, then statistically 50% of the bits of , should be different.

Furthermore, a cryptographic hash function must satisfy three key resistances:

* 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 is a colliding pair.
* Collision resistance: It is infeasible to find a colliding pair.

By infeasible, we mean that there is no better method than to search the input space of by brute force. If has an -bit hash value, then we specifically mean there is no algorithm that runs faster than . An interesting exception however is collision resistance, which is .

### Generalized Birthday Paradox

Suppose we have a room full of people; what is the probability that at least two people in that room share the same birthday? If we have 367 people in the room, then by the pigeonhole principle, there is surely at least one pair of people with the same birthday. Let denote the probability that of people, at least one pair shares a birthday. It then follows that:

To determine when it is statistically likely that a pair of people will share a birthday, we determine when :

Hence we only require people in a room to have at least a 50% chance of two people sharing the same birthday as each other.

We can generalize this to any uniform distribution. If each event has a probability of , then we must sample inputs in order to find a matching pair with at least a probability of 50%. We can this generalized results to finding collisions.

Suppose is a cryptographic hash function with an -bit hash value. It is then uniformly distributed on its codomain. As such, each element in the codomain has a probability of of occurring. By the generalized birthday paradox, in order to find two inputs with the same output with at least a probability of 50%, we must have at least samples. This gives an upper bound on the complexity for breaking collision resistance.

## Construction

Due to the strength and complexity required by cryptographic hash functions, many standards for their construction have been developed over the past years. Cryptographic hash functions can be considered to be two distinct parts: a compression function, and construction scheme.

A compression function is a function with . We refer to the value as the block size, and as the digest size of the compression function. These functions are used as the core of a cryptographic hash function, and as such, must have all the properties required of a cryptographic hash function.

### Davies-Meyer

Constructing a compression function is typically not easy. As such, there also exist many constructions for creating a compression function from other cryptographic primitives. One simple construction is the Davies-Meyer construction, which creates a compression function from a cryptographically secure block cipher. Let denote the encryption of a message with key . Let our message be broken into several parts, denoted . We can construct a compression function by setting , where is some well-known constant, and running this a desired number of times. The result of the compression function is then given as the final .

### Merkle-Damgård

To properly construct our hash function, we need some way of taking a message of arbitrary length, and running it all through our fixed-width compression function. The Merkle-Damgard construction allows this for any compression function. To begin, let be our message, and let be the padding. We can then take each and run it through the compression function with some to get , where is some well-known initial value.

## SHA-256

### Initial Values and Constants

### Functions

### Message Padding

### Compression Function

### Example Vectors

# Design

# Results

# Discussion

# Conclusion

References

Appendices

A: Additional Screenshots

This Appendix contains screenshots showing additional details of the new system outlined in Chapter 4. The clustering functionality is illustrated with a walkthrough of a scenario that takes NIWA weather data as input and outputs a new visualisation.