Cryptographic Hash Functions

Kelvin W. Macharia
kwm14@student.london.ac.uk

Abstract. Cryptography is the science and technique of securing information and communications to guarantee that only those for whom the information is intended can understand and process it. Hashing is the process through which plaintext data of any length is mapped into a unique ciphertext of fixed length known as a hash. A function that does hashing is a hash function. All cryptographic hash functions are hash functions but not every hash function is a cryptographic hash function. This paper describes what cryptographic hash functions are, what security properties are expected of them and what attacks can be performed against them.

1. Introduction

A hash function is a mathematical algorithm that takes data of arbitrary length as input and maps it to a fixed length enciphered text as output. This output is called a message digest, a hash value, a hash code or simply a hash.

More formally, a hash function is a mathematical function $H : D \rightarrow R$, where the domain $D = \{0,1\}^*$ and $R = \{0,1\}^n$ for some n >= 1, that maps a numerical input value m of arbitrary length into a condensed numerical value output h of fixed length [1]. That is: $h = H(m)$.

A hash function that satisfies some additional requirements so that it can be used for cryptographic applications is known as a cryptographic hash function. These functions are essential constructs that have a variety of use cases. The main fields of their application are protection of stored passwords, message authentication, digital signatures, and therefore certificates.

Cryptographic hash functions are broadly classified into two classes: unkeyed hash functions also known as Manipulation Detection Code (MDC) or Message Authentication Code (MAC) with single a parameter – an input message – and keyed hash functions with two distinct inputs – an input message and a secret key. Generally, the term hash functions refers to unkeyed hash functions [8].

Some examples of cryptographic hash algorithms are:

- The **SHA (Secure Hash Algorithm)** family **-** published by the National Institute of Standards and Technology (NIST) as a U.S. Federal Information Processing Standard (FIPS) [2]. This family designates six different hash functions: SHA-0, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512 [2, 3]. The first four operate on 512-bit message blocks divided into 32-bit words and the last two on 1024-bit blocks divided into 64-bit words. Bitcoin, the original and largest cryptocurrency (at the time of writing), uses the SHA-256 hash function.
- The **MD (Message Digest)** family - comprises of MD2, MD4, MD5 and MD6 authored by Ronald Rivest for RSA security and was adopted as the Internet Standard RFC 1321 [4].
- **RIPEMD (RACE Integrity Primitives Evaluation Message Digest)** – a family of cryptographic hash functions based upon the design principles used in MD4 developed by Hans Dobbertin, Antoon Bosselaers, and Bart Preneel at the COSIC research group at the Katholieke Universiteit Leuven. RIPEMD-160 produces a hash digest of 160 bits (20 bytes).
- **Whirlpool** – designed by Vincent Rijmen and Paulo S. L. M. Barreto, this hash function based on a substantially modified version of the Advanced Encryption Standard (AES). Whirlpool produces a hash digest of 512 bits (64 bytes).
- **BLAKE** – a hash function submitted to the NIST hash function competition by Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and Raphael C.-W. Phan. It is based on Dan Bernstein's ChaCha stream cipher, but a permuted copy of the input block, XORed with round constants, is added before each ChaCha round.
- **Curl-P** – a hash function formerly used in IOTA Signature Scheme (ISS). IOTA is a cryptocurrency designed for use with the Internet of Things (IoT) and automotive ecosystems. ISS is based on Winternitz One-Time Signatures but unlike traditional Winternitz, in IOTA users sign the hash of a message. Thus, the security of ISS relies on its cryptographic hash function, which was Curl-P-27.

2. Properties of cryptographic hash functions

A cryptographic hash function is expected to have the following properties that guarantee its effectiveness and security:

- **One-way function (pre-image resistance)** – this property requires that for a hash function H if given any hash value h, it is computationally infeasible to find an input m such that $H(m) = h$. In other words, it must be easy to compute on every input but extremely difficult to invert given the image of a random input.

- **Target collision resistance (2nd pre-image resistance)** – this property requires that given a hash function *H* and any input *m*, it should be computationally infeasible to find another input *m'* such that *m' ≠ m* and *H(m) = H(m')*.
- **Collision resistance** - this property requires that given a hash function *H*, it should be computationally infeasible to find two inputs *m* and *m'* such that *m ≠ m'* and *H(m) = H(m')*. Due to the fixed size of hash values compared to the much larger – and arbitrary – size of inputs, collisions are expected to exist in hash functions. However, they must be computationally intractable to find.
- **Deterministic** – this property requires that a hash function *H* should consistently map a given input *m* to a hash value *h*. It should also be public and computable.
- **Avalanche effect** – this property requires that a change in just one bit of the input data should result in a large change in the output. This "diffusion" ensures that any inference about the input from the output is infeasible thus this property is also sometimes defined as unruliness.
- **Hash speed** – an ideal property of a cryptographic hash function is its ability to operate at a reasonable speed. In many situations, a hashing algorithm should compute hash values rather quickly. However, it's worthwhile to note that faster is not always better or more secure.

3. Attacks on cryptographic hash functions

Attacking a cryptographic hash function implies breaking one of its security properties. For example, breaking pre-image resistance means that an attacker can create a message that hashes to a specific hash [5]. Attacks on hash functions may focus on either the structure of the hash function or on the algorithm of the compression function used to condense arbitrary size input into a fixed size hash value.

Over years, a significant number of cryptographic hash functions have been broken and proven to be vulnerable to security attacks. The main target of these attacks is the collision resistance of hash functions. For instance, in August 2004 collisions were found in several then-popular hash functions, including MD5 [6].

A hash function can be termed as "broken" when, immaterial of the computation feasibility of that effort, a lower number of its evaluations compared to the brute force attack complexities and strengths estimated by its designer are used to overcome at least one of its properties. For instance, consider that it requires 2^{90} evaluations to find a collision for a 256-bit hash function. The hash function is broken because this factor is less than the 2^{128} evaluations required by the Birthday attack despite the intractability of the computational complexity [7].

Attacks on hash functions can be classified into two broad categories – Brute Force attacks and Cryptanalytical attacks. Figure 1 below illustrates this classification:

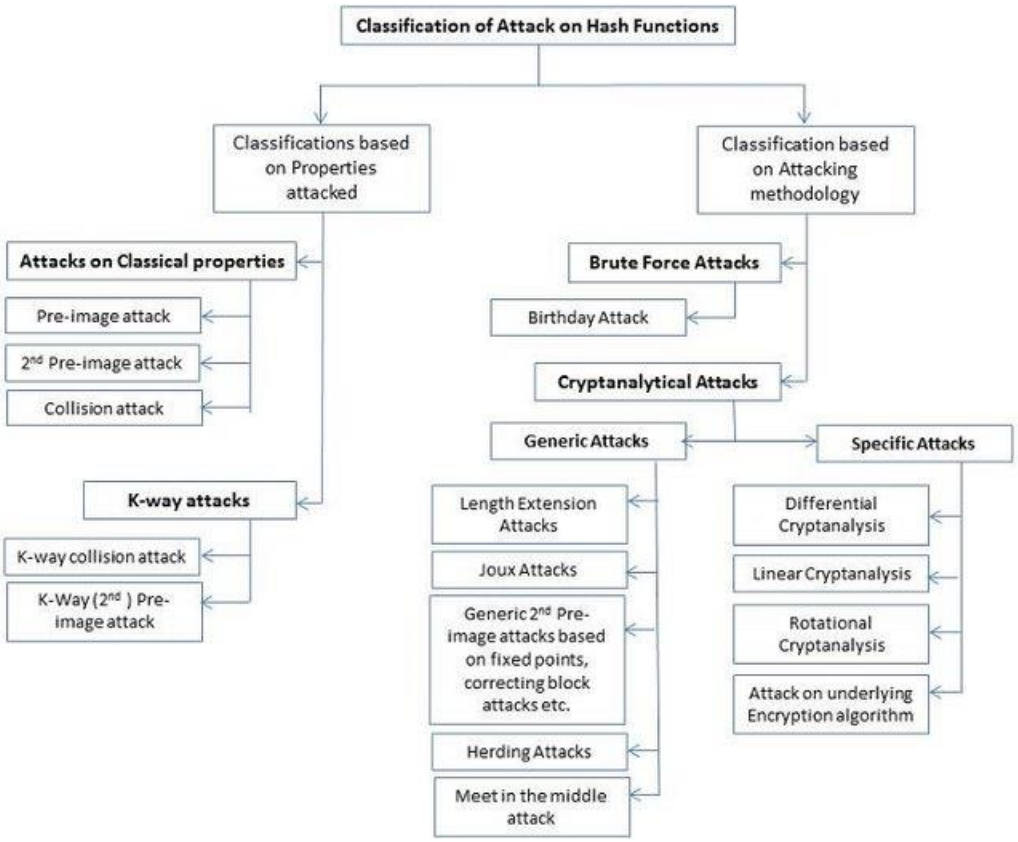

Fig. 1 Classification of attacks on Hash Functions

3.1 Brute Force attacks

A brute force attack, also known as an exhaustive search, is a trial-and-error based approach in which an attacker uses a set of predetermined inputs (guesses) against an algorithm while analysing the output for a possible match. It is the equivalent of trying every key on a key ring and eventually finding the right one. Brute force attacks work on all hash functions independent of their structure or any other working details [8].

The strength and security of a hash function – and the difficulty of Brute Force attacks – relies solely on the size of its output hash value. For a hash output of length n, the effort required to resist various classical brute force attacks can be expressed as follows:

- **One-way function inversion (pre-image resistance) attack** – the effort required to find an input m which maps to h by H given a challenge h equals 2^n because for a given n-bit hash h of the hash function H, an attacker

would evaluate H with every plausible input m until the desired output hash value h is obtained.

- **2nd pre-image resistance attack** – the effort required to find two inputs m and m' that are mapped to the same output by H equals 2^n. In this variation of the brute force attack, an attacker would evaluate the hash function H with every possible input $m' \neq m$, for a given input m, until the hash value $h = H(m)$ is obtained.
- **Collision attack** – the effort required for a given hash function H, *to* find two inputs m and m' such that $m \neq m'$ and $H(m) = H(m')$ equals $2^{n/2}$ as on average, they would have to try 2^{n-1} (i.e., $2^n / 2$) inputs to find one whose hash value matches. However, in what is referred to as a Birthday attack that is based on the Birthday Paradox, a chosen plain text attack is possible in which case the effort required for a collision in a hash function equals $2^{n/2}$ as opposed to 2^{n-1}. [9]

Further extensions of these classic Brute Force attacks have been studied by various authors. These include: the **K-way collision attack for K >= 2** whose aim is to find K different inputs m^i such that $H(m^i) = \ldots = H(m^k)$ [5] and the **K-way 2nd pre-image resistance attack for K >= 1** where given an input m, a hash value h, and a hash function H such that $h = H(m)$, the aim is to find K different inputs m^i with $H(m^i) = h$ and $m^i \neq m$.

3.2 Cryptanalytical attacks

Cryptanalysis is the study of ciphertext, ciphers and cryptosystems with the aim of identifying weaknesses or leaks of hidden aspects of cryptosystems that are useful in obtaining meaning of encrypted information without access to the secret key typically required to do so. Cryptanalysis of hash functions focuses on their underlying structure and/or the algorithm of the compression function.

Hash functions must be efficient to be used in information processing tasks such as computation of digital signatures. Effectiveness is achieved by designing them in the iterative mode of operation where a function that accepts fixed length input is iterated until an arbitrary length input is processed completely [7]. This iterative structure was independently proposed by Ivan Damgård [10] and Ralph Merkle [11] at Crypto ’89. It is referred to as the Merkle- Damgård construction and is applied in the design of most hash functions in use today.

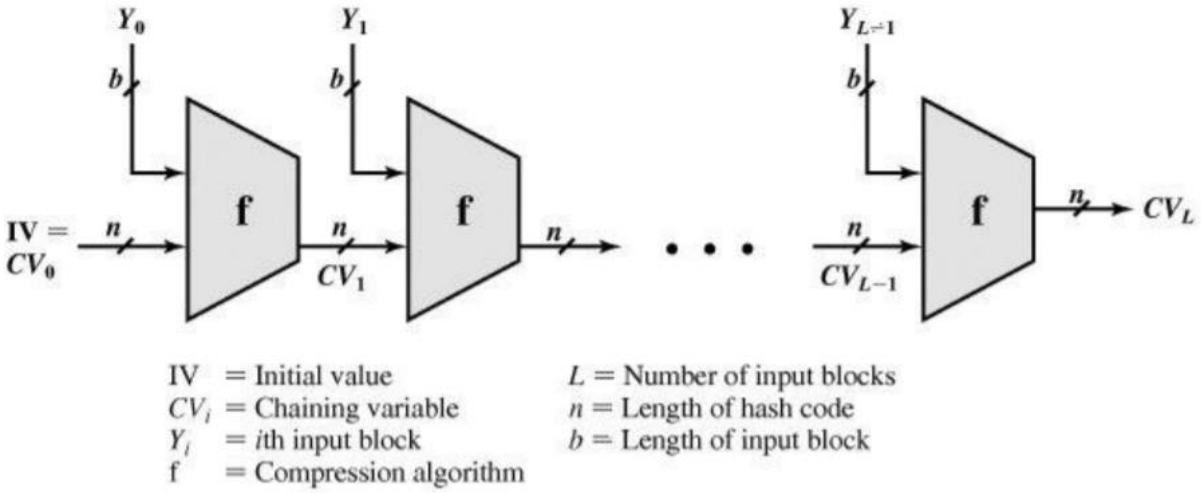

Fig. 2 General structure of most hash algorithms (IJERT [12])

The hash function takes an input message and partitions it into *L* fixed size blocks of *b* bits each. The final block can be padded to *b* bits if necessary and may also include the value of the total length of the input to the hash function which makes the job of an attacker more difficult. They must either find two messages of equal length that hash to the same value or two messages of differing lengths that, with each of their respective lengths appended, hash to the same value [12].

In their respective papers, Damgård [10] and Merkle [11], provide theorems showing that if there exists a fixed-length input collision resistant compression function $f : \{0, 1\}^b \times \{0, 1\}^t$ then one can design a variable-length input collision resistant hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^t$ by iterating that compression function [7]. In other words, if the compression function is collision resistant, then so is the resultant iterated hash function. Thus, if the compression function is vulnerable to any attack, then so is the iterated hash function but the converse of this result is not necessarily true in general.

Cryptanalytical collision finding algorithms and attacks may be classified as either single or multi block attacks depending on whether the attack uses one compression function (one block) or more than one iteration of the compression function (more than one block) in finding collisions or pre-images [8]. In his thesis, Gauravaram [7] further classifies cryptanalytical attacks on hash functions into generic and specific attacks.

Generic attacks are those that work on a general hash function construction. These attacks are applicable even if the underlying compression algorithm is replaced by some abstract oracle [8]. For example, attacks on the Merkel-Damgård construction that work on all hash functions designed using the approach are generic attacks. Length extension attacks, Joux's multi-collision attacks, multi (2nd) pre-image attacks such as the one based on fixed points, correcting block attacks, herding attacks, and meet in the middle attacks are examples of generic cryptanalytical attacks.

Specific attacks are those that apply differential [19], linear [20,21] or rotational [22] cryptanalysis or that work on specific hash functions or the algorithm underlying its compression function such as the collision attacks on the specific hash function of MD4 [13], MD5 [14,15], SHA-0 [16,17], and SHA-1 [16,18]. If the hash function's underlying compression function is implemented using the encryption algorithm, then the weaknesses in the encryption algorithm – the encryption algorithm may have complementation property or weak keys or fixed points – can be exploited to attack hash functions.

4. Conclusion

Cryptanalytical attacks on hash functions, just like with encryption algorithms, seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Hash functions are however practically easier to attack than encryption algorithms because the attacker does need to assume any secrets and the maximum computational effort required to attack the hash function is only upper bounded by the attacker's resources and not user's gullibility. This is not the case with block ciphers where the maximum practical count of executions of the block algorithm is limited by how much computational effort the attacker can get the user to do [7].

The measure of resistance of a hash algorithm to cryptanalysis is based upon a comparison of its strength to the level of effort required for a brute-force attack. That is, an ideal hash algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

References

[1] Rompay, B. V. (2004) "Analysis and Design of Cryptographic Hash functions, MAC algorithms and Block Ciphers". Ph.D. thesis, Leuven, Belgium: Electrical Engineering Department, Katholieke Universiteit.

[2] National Institute of Standards and Technology (NIST). (1995) “Secure Hash Standard”. FIPS Publication 180-1. Google Scholar

[3] National Institute of Standards and Technology (NIST). (2002) “Secure Hash Standard”. FIPS Publication 180-2. Google Scholar

[4] Rivest, R. (1992) “The MD5 Message-Digest Algorithm”. Request for Comments: 1321, Network Working Group. RFC 1321

[5] Lucks, S. (2004) “Design Principles for Iterated Hash Functions”. Cryptology ePrint Archive, Report 2004/253, pp. 253. https://eprint.iacr.org/2004/253

[6] Wang, X. Feng, D. Lai, X. Yu, H. (2004) “Collisions for Hash Functions MD4, MD5, HAVAL-128, and RIPEMD”. Jinan250100, China: The School of Mathematics and System Science, Shandong University. https://eprint.iacr.org/2004/199.pdf

[7] Gauravram, P. (2003) “Cryptographic Hash Functions: Cryptanalysis, design and applications”. Ph.D. thesis, Brisbane, Australia: Faculty of Information Technology, Queensland University of Technology.

[8] Rajeev, S. Geetha, G. (2012) “Cryptographic Hash Functions: A Review”. International Journal of Computer Science Issues, ISSN (Online): 1694-0814. Vol 9. 461 - 479. https://www.researchgate.net/publication/267422045_Cryptographic_Hash_Functions_A_Review

[9] Bellare, M. Kohno, T. (2004) "Hash Function Balance and Its Impact on Birthday Attacks". In EUROCRYPT, pp.401-418.

[10] Damgård, I. (1989) “A design principle for hash functions”. In Gilles Brassard, editor, Advances in Cryptology: CRYPTO 89, volume 435 of Lecture Notes in Computer Science, pages 416-427. Springer-Verlag.

[11] Merkle, R. (1989) “One way hash function and DES”. In Gilles Brassard, editor, Advances in Cryptology: CRYPTO 89, volume 435 of Lecture Notes in Computer Science, pages 416-427. Springer-Verlag.

[12] Kumar, C. K. Suyambulingom, C. (2012) “Cryptographic of high Security Hash Functions”. International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 1 Issue 3. https://www.ijert.org/research/cryptographic-of-high-security-hash-functions-IJERTV1IS3074.pdf

[13] Wang, X. Lai, X. Feng, D. Chen, H. Yu, X. (2005) "Cryptanalysis of the Hash Functions MD4 and RIPEMD". In EUROCRYPT, pp.1-18.

[14] Wang, X. Lai, X. Feng, D. Yu, X. (2004) "Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD". IACR Cryptology ePrint Archive, pp. 199.

[15] Wang, X. Yu, X. (2005) "How to Break MD5 and Other Hash Functions". In EUROCRYPT, pp. 19-35.

[16] Biham, E. Chen, R. Joux, A. Carribault, P. Lemuet, C. Jalby, W. (2005) "Collisions of SHA-0 and Reduced SHA-1". In EUROCRYPT, pp.36-57.

[17] Wang, X. Yu, H. Yin, Y. L. (2005) "Efficient Collision Search Attacks on SHA-0". In CRYPTO, pp.1-16.

[18] Wang, X. Yin, Y. L. Yu, H. (2005) "Finding Collisions in the Full SHA-1". In CRYPTO, pp.17-36.

[19] Biham, E. Shamir, A. (1991) "Differential Cryptanalysis of DES-like Cryptosystems". Journal of Cryptology, Vol. 4, No. 1, pp. 3-72.

[20] Bakhtiari, S. Safavi-Naini, R. Pieprzy, J. (1995) "Cryptographic Hash Functions: A Survey". Technical Report 95-09, Department of Computer Science, University of Wollongong

[21] Matsui, M. (1993) "Linear Cryptanalysis methods for DES Cipher". In EUROCRYPT, pp. 386-397.

[22] Khovratovich, D. Nikolic, I. (2010) "Rotational Cryptanalysis of ARX". In FSE, pp.333-346.

[23] Muller F. (2004) "The MD2 Hash Function Is Not One-Way". In: Lee P.J. (eds) Advances in Cryptology - ASIACRYPT 2004. ASIACRYPT 2004. Lecture Notes in Computer Science, vol 3329. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-30539-2_16