Volume : 2 | Issue : 4 | April 2013 ISSN - 2250-1991

Research Paper Mathematics

Construction Methods on Cryptographic Hash Functions

* C. Krishna Kumar ** Dr. C. Suyambulingom

* Sathyabama University, Chennai, India

** Professor, Dept. of Mathematics, TAU, Coimbatore, India

ABSTRACT

The design of hash functions are investigated under two subtopics which are compression functions and the construction methods. Compression functions are the core of the hashing algorithms and most of the effort is on the compression function when designing an algorithm. Moreover, for Merkle-Damgard hash functions, the security of the algorithm depends on the security of the compression function. Construction method is also an important design parameter which defines the strength of the algorithm. Construction method and compression function should be consistent

Keywords : metrics, software, quality, estimation

1.Construction Methods

construction methods of hash functions are discussed in details. Iterative hash functions still pace a big portion of all hash constructions. Therefore, iterative designs will be investigated deeply.These constructions include Merkle-Damgard(MD)1, HAIFA and some other alternative designs derived from Merkle-Damgard. Besides, sponge constructions is included. Sponge construction is a new and secure construction method which is becoming more popular.

1.1 Merkle-Damgård Construction

Merkle-Damgard construction is the most widely used hash construction method, which was designed by R.Merkle [50] and I.Damgard [33] independently in 1989. Most of the hash functions and all the standardized hash functions are build upon MD construction. MD construction is basically processed in three steps. First step is the padding step. The aim of the padding is to make the message length a multiple of message block length, m. The most widely used padding procedure is as follows: a '1' bit followed by a number of '0' bits and the bitwise notation of the message length are appended to the message. The number of '0' bits appended to the message are chosen so that the length of the message becomes a multiple of block length m. In general the maximum length of the message that can be processed by the hash function is 264 − 1. Therefore 64 bit space is provided for the length padding. Considering the appended '1' bit, at least 65 bits are appended to all messages regardless of the message length. If l is the length of the message, d, the number of '0' bits appended is the smallest positive root of the equation

$l + 65 + d \equiv 0 \pmod m$. Second step is dividing the padded message into m bit blocks $m_0 m_1 m_2 \dots m_{t-1}$. After this step, the chaining values are iteratively found by using a fixed, publicly known initialization vector, IV, and the message blocks: h0 = IV hi = f (h_{i-1}, m_{i-1}) i = 1, 2 . . , t

where f is the compression function of the hashing algorithm.

Generally ht is taken to be the hash value of the message. However an optional transformation

g(x) can be applied to ht to get the hash value as H(M) = g(ht).

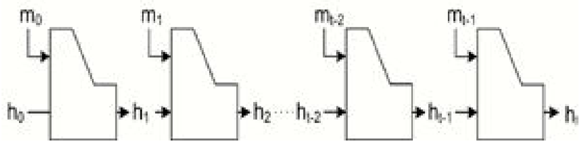

Figure 1.1: MD Structure

The iterative structure of MD construction enables arbitrary length messages to be processed easily by the hash functions. Also padding the message length and using a non-zero IV, which is called is called the MD − strengthening, increases the defeats or increases the complexity of various attacks. The most important property of MD structure is that the collision resistance property of the compression function is preserved [50, 33]. The MD structure has a security proof that states if the compression function used in the hash algorithm is collision resistant then the hash function itself is collision resistant [51]. Besides collision resistancy, it was believed that MD construction also preserves the preimage resistancy and second preimage resistancy of the compression function [52]. However there are several attacks in recent years against the second preimage resistency of the MD construction such as [53, 54, 55].

1.2 HAIFA Construction

MD structure is a provable collision resistant hash construction method. However, as computing power increases and new cryptanalytic tools are proposed, MD hash functions become weaker and more vulnerable to attacks. Biham and Dunkelman [52], fixing the flaws in MD structure, announced a new construction method HAIFA(HAsh Iterative FrAmework). HAIFA is an iterative structure which is based on Merkle-Damgard and uses additional tools to increase the security. The compression function f takes only message blocks mi and chaining values hi as inputs in MD hash functions. In HAIFA, compression function inputs are number of bits (or message blocks) hashed so far, b, and salt, s, in addition to message block and chaining value. So the chaining values can be expressed as hi = f (hi−1,mi−1, b, s). The number of bits, or message blocks, hashed so far is included as an input to prevent the fixed point attacks. In MD structure attacker can freely inject fixed points to find a second preimage as she needs. However, even if b is not mixed well in f , since the inputs to the compression functions are different at each iteration, attacker can use a fixed point very limited times. The other input, salt, is a precaution against attacks which has a precomputation phase. Some attacks have two phases as on-line and off-line phases. In the off-line phase attacker produces a number of structures which can be messages or chaining values using some weaknesses in the hash algorithm. After learning the pair (M, H(M)) she mounts the on line phase of the attack to produce a collision or second preimage. In HAIFA, attacker should know the salt to produce the structures but the salt is a random number which is sent with the pair (M, H(M)) as (M, S alt, H(M, S alt)) to the receiver. For security reasons the salt should be at least half length of the chaining value and should be random [52]. Moreover, in the HAIFA construction, there is a standardization of initial values for variable hash sizes. If a fixed IV is used for all hash sizes and if for some hash sizes hash values are truncated, then some bits of the hash values of distinct sizes will be equal. For example, if one needs a 224-bit hash value using a 256-bit hash function, he will produce the 256-bit hash value and truncate the final (or some other) 32-bits. Therefore, the 224 bits of the 256-bit and 224-bit hash values of a message will be common. HAIFA, uses different IVs for different hash values, so the truncation will not cause any problems. Another addition is padding the hash size. Usual padding adds a '1' bit, some '0' bits to make the padded message a multiple of message block length m bits and the bitwise notation of the length of the original message. New construction method concatenates the bitwise notation of hash size after length padding. This ensures that no two messages exist which have the same hash value without truncation for different hash sizes.

1.3 Sponge Construction

Sponge construction [56] is a new construction method for hash functions and stream ciphers. This construction can be built upon a function f which can be expressed as a random permutation or random function. If f is expressed as a random permutation, construction is called a P − sponge, otherwise, if it is expressed as a random function, construction is called a T − sponge. The main difference between the compression functions of MD structures and the f function in sponge construction is, unlike the compressing functions in MD or Haifa, f is a function which maps l bit input to l bit output. The construction is consisting of 2 phases: absorbing and squeezing. In the first phase, data is input to the sponge iteratively block by block and in the second phase output is given in the same manner. These two phases have similar procedures. In the absorbing phase, iteratively, message block is XORed or overwritten to the state and f is applied to this state. Then, next message block is processed and so on. After all the message blocks are input, second phase is applied. In the squeezing phase, some part of the state is output and f is applied to the state. Then, again some part of the state is output and f is applied until the desired hash size is achieved.

1.4 Some Other Hash Constructions

1.4.1 Wide Pipe and Double Pipe Construction

In [57] and [58], Lucks introduced a new concept: wide pipe hash functions. These hash functions are constructed so that the intermediate chaining values are w bits for an n-bit hash size with w > n. There is a final transformation at the end which reduces the w-bit final chaining value to n bits. The aim of the wide pipe construction is to increase the complexities of the attacks depending on chaining values. For example, for an n−bit hash H1 with n-bit chaining values, an attack which finds collisions for the compression function with complexity $2^{n/4}$, means a break for H1. However for an n-bit wide pipe hash H2 with w-bit chaining values where w = 2n, cost of this attack is same with exhaustive search cost and not a serious case as H1. Another approach, by Lucks, to widen the internal state is the double pipe hash functions. In this approach two parallel iterations are processed. These two iterations can be initialized with different IVs, can use different compression functions or can iterate the message blocks in different permutations. At the final step the outputs of the two iterations are mixed to get the hash value.

1.4.2 Prefix-Free Merkle-Damgård Construction

One of the problems of MD structure is about the randomness: although the underlying compression function is in differentiable from a random oracle, the hash function itself is not guaranteed to be in differentiable from a random oracle [59]. To solve this problem prefix-free encoding is suggested for iterated hash functions. Prefix-free encoding (or prefix encoding) is v applied to the message before padding process. The two of the suggested encoding functions are

- $g_1(M) = LM||m_0||m_1|| \cdots ||m_{t-1}$ where LM is the message length.
- $g_2(M) = 0||m_0||0||m_1||0|| \cdots ||0||m_{t-2}||1||m_{t-1}$ where 0 and 1 are single bits and mi is the (i + 1)th m − 1 bit block of the message M.

After encoding, the message is hashed as in MD structure and hash function becomes indifferentiable from a random oracle.

1.4.3 Enveloped Merkle-Damgård Construction

EMD is proposed by Bellare and Ristenpart in [60] which is a construction that preserves collision resistancy, preimage resistancy and pseudo-randomness of the compression function. The message blocks are iterated as MD up to the final message block. The final message block mt−1 and ht−1 are concatenated and taken as input to another iteration with a distinct IV. This operation is called enveloping.

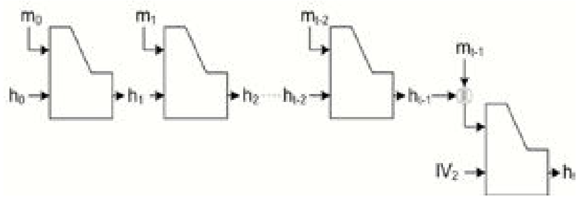

Figure : EMD Construction

1.4.4 RMX Construction

RMX is presented in [61] and [62] by Halevi and Krawczyk. The idea of RMX is randomization of the message before padding. In this method a random string, r, of length between smallest number of padding bits and message block length m is produced. From this random string r, three other parameters, r_0, r_1 and r_2, are produced: respectively by appending zero bits to r, repeating r as many times as necessary and taking some part of r. r_0 is prepended to the message, r1 is XORed with all the message blocks except padding and r_2 is XORed with the padding blocks. After obtaining the hash value, r is stored, or send to a receiver, with the hash value.

$h1 = f(IV, r_0)$ ...

$hi = f(h_{i-1}, r_1 \oplus m_{i-2})$ f or i = 2, 3, ..., t

$h_{t+1} = f(h_t, r_2 \oplus m_{t-1})$

$H(M) = h_{t+1}$

1.4.5 3C and 3C-X Constructions

3C and 3C − X are proposed by Gauravaram [63] which are similar to double pipe hash construction. In 3C construction, while message is iterated from the main line, there is an additional line which takes inputs from the main iteration line. Finally the outputs of two lines are mixed to get the hash value. The algorithm can be defined as follows

$h_1 = f(h_0, m_0)$

$z0 = h_1$

$hi = f(h_{i-1}, m_{i-1})$ i = 1, 2, . . . , t

$z_{i-1} = f(zi-2, PAD(hi))$ i = 2, . . . , t ht+1 = f (ht, PAD(zt−1)

H(M) = ht+1

where PAD(x) is padding x with 0 bits until it becomes m-bits.

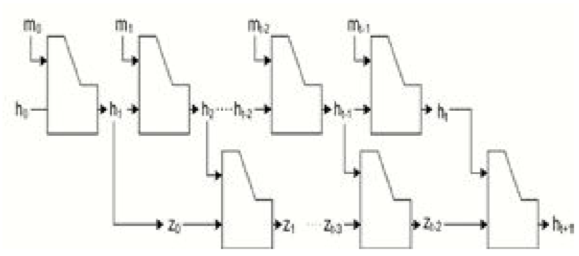

Figure: 3C Construction

The 3C−X construction is similar to 3C. The only difference is that the compression functions in the second iteration line, except the final compression function, are replaced with XOR operations. This way construction becomes lighter.

1.4.6 Dynamic Hash Function Construction
Dynamic hashing, by Speirs [64], is a collection of iteration lines where each line feeds the next line. This construction has a security parameter s which can be considered as salt. The message M is padded to be a multiple of m − n bits. A '1' bit followed by a number of '0' bits are appended to the message. Then 64 bits message length and 32 bits representation of s is concatenated. The number of lines, l, in the construction depends on hash size d. l is chosen such that n(l − 1) < d ≤ nl. Each line has an initial value, hi 0 which is derived from the line number and s as hi 0 = f (IV1, IV1||s||00..0||i − 1). The lines are interacting with each other. The output hjk of kth iteration of line j is concatenated to mk and input to the (k + 1) st iteration of line (j + 1) mod l, where l is the number of lines. At the end of iterations an "envelope" operation is done with another initial value hit +1 = f (IV2, hit ||00 · · · 0) and these chaining values are concatenated to get the hash value using the iterations

IVs, j = f (IV1, IV1||s||00 · · · 0||j − 1) f or j = 0, 1, . . . , l − 1,

h1, j = f (IVs, j,m0||IVs,(j−1) mod l) f or j = 0, 1, . . . , l − 1,...

$h_{i,j}$ = f (hi−1, j,mi−1||hi−1,(j−1) mod l) f or i = 2, . . . , t − 1, j = 0, 1, . . . , l − 1, ...

$h_{t,j} = f(IV_2, h_{t-1,j}||j)$ f or j = 0, 1, . . . , l − 1,

$H(M) = g(h_{t,0}||h_{t,1}|| \cdot\cdot\cdot ||h_{t,l-1},)$

where g(x) is a mapping from n · l bits to hash size.

CONCLUSION
Cryptographic hash functions have been used in many daily life applications. However, the knowledge about these tools are very little This paper is serving a as an extensive survey on hash functions. Starting from very basic definitions and properties, it covers design parameters and cryptanalytic attacks. In tis part types of construction methods are dealt with.

REFERENCES

[1] Xiaoyun Wang and Hongbo Yu. How to Break MD5 and Other Hash Functions. In EUROCRYPT, pages 19–35, 2005. | [2] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In EUROCRYPT, pages 1–18, 2005. | [3] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In CRYPTO, pages 17–36, 2005. | [4] Xiaoyun Wang, Hongbo Yu, and Yiqun Lisa Yin. Efficient Collision search Attacks on SHA-0. In CRYPTO, pages 1–16, 2005. | [5] Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of Applied Cryptography. CRC Press, 1996. | [6] R. Anderson. The Classification of Hash Functions. In IMA Conference in Cryptography and Coding,pages83–93, 1993. | [7] C.H. Meyer J. Oseas S.M. Matyas. Generating Strong One-Way Functions with ryptographic Algorithm. In IBM Techn. Disclosure Bull., Vol. 27, No. 10A, pages 5658–5659, 1985. | [8] S. Miyaguchi, M. Iwata, and K. Ohta. New 128-Bit Hash Function. In Proceeding of 4th International Joint Workshop on Computer Communications, pages 279–288, 1989. | [9] B. Preneel, R. Govaerts, and J. Vandewalle. Hash Functions Based on Block Ciphers: A Synthetic Approach. pages 368–378, 1993. | [10] D. Coppersmith M. Hyden S. Matayas C. Meyer J. Oseas S. Pilpel B. Brachtl and M. Schiling. Data Authentication Using Modification Detection Codes Based on a Public One-Way Encryption Function. In U. S. Patent Number 4,908,861, 1990. | [11] J. Black, P. Rogaway, and T. Shrimpton. Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions From PGV. In In Advances in Cryptology – CRYPTO 2002, pages 320–335. Springer-Verlag, 2002. | [12] Martijn Stam. Blockcipher Based Hashing Revisited. Cryptology ePrint Archive, Report http://eprint.iacr.org/2008/071.pdf, 2008. | [13] Lars Knudsen and Bart Preneel. Hash Functions Based on Block Ciphers and Quaternary Codes. In Advances in Cryptology ASIACRYPT, pages 77–90, 1996. | [14] Joan Daemen and Craig Clapp. Fast Hashing and Stream Encryption with Panama,. In Fast Software Encryption 1998, pages 60–74. Springer. | [15] Donghoon Chang, Kishan Chand Gupta, and Mridul Nandi. Rc4-hash: A New Hash | Function Based on RC4. In INDOCRYPT, pages 80–94, 2006. |