Implementation of MDC-2 with AES-hash Function

Nway Htet Myat, Myat Su Win
University of Computer Studies, Taung-ngu, Myanmar
nwayko83@gmail.com, myattsuro@gmail.com

Abstract

Cryptographic hash functions are a third type of cryptographic algorithm. They take a message of any length as input, and output a short, fixed length hash which can be used in (for example) a digital signature. As digital signature technology becomes more widely understood and utilized, many countries worldwide are competitively developed their own signature standards for their use and applications. Hash functions are the mathematical algorithms that transform arbitrary length sequences of bits into a hash result of a fixed, limited length. And they are used in many cryptographic protocols. Currently, hash functions based on block cipher used in many of the cryptographic applications. This paper will propose a hash function which is based on AES algorithm and it is the modified version of MDC-2 algorithm to perform efficiently the iteration of compression function for message with huge length.
Keywords: hash function, AES, DES, MDC-2, message digest.

1. Introduction

Hash function is a fundamental tool in Information Security. In its simplest from a hash function is an algorithm that takes an input of any size and outputs a fixed length "hash code" that is, in some sense, difficult to predict in advance. That is, the output of the hash function serves as a digital finger-print for the input and should be the same each time the same message is hashed. Hash functions are used to help provide data integrity in Message Authentication Codes (MACs), to produce message digests for use with digital signature schemes and to produce Manipulation Detection Codes (MDCs) in entity authentication and key establishment schemes.

For a hash function to be secured it is required to be one-way and collision resistant. The one-way property can be achieved if it is easy to generate the message digest of a message but, is hard to determine the original message when the digest of it is known. On the other hand, collision resistance can be attained if it is hard to find two different messages, having same message digest as output. Apart from these requirements, the hash function should be accepting a message of any size as input and computation of the message digest must be fast and efficient. Signing the message digest rather than the message often improves the efficiency of the process because the message digest is usually much smaller than the message. [1, 4]

2. Background Theory

In this section, cryptographic hash function and modification detection code (MDC-2) are presented as background theory.

2.1 Cryptographic Hash Function

Cryptographic hash functions play a fundamental role in modern cryptography. Hash functions take a message as input and produce an output referred to as a hash-code, hash-result, hash-value, or simply hash. Cryptographic hash functions are used for data integrity and message authentication.

The basic idea of cryptographic hash functions is that a hash-value serves as a compact representative image (sometimes called an imprint, digital fingerprint, or message digest) of an input string, and can be used as if it were uniquely identifiable with that string.

Hash functions are used for data integrity in conjunction with digital signature schemes, where for several reasons a message is typically hashed first, and then the hash-value, as a representative of the message, is signed in place of the original message. A distinct class of hash functions, called message authentication codes (MACs), allows message authentication by symmetric techniques. MAC algorithms may be viewed as hash functions which take two functionally distinct inputs, a message and a secret key, and produce a fixed-size (say n-bit) output, with the design intent that it be infeasible in practice to produce the same output without knowledge of the key. MACs can be used to provide data integrity and symmetric data origin authentication, as well as identification in symmetric-key schemes. [1, 7]

Sometimes a MAC is called a keyed hash function, but then one has to use for an MDC the artificial term un-keyed or keyless hash function. According to their properties, the class of MDC's

will be further divided into one-way hash functions (OWHF) and collision resistant hash functions (CRHF). [4, 5]

The hash function will be denoted with h, and its argument, i.e., the information to be protected with M. The image of M under the hash function h will be denoted with $h(M)$ and the secret key with K.

2.1.1 Types of Hash Function

At the highest level, hash functions may be split into two classes:

- **Unkeyed Hash Functions**, whose specification dictates a single input parameter (a message) and
- **Keyed Hash Functions**, whose specification dictates two distinct inputs, a message and a secret key.

2.1.2 Properties of Hash Function

A cryptographic hash function must be able to withstand all known types of cryptanalytic attack. As a minimum, it must have the following properties:

- **Preimage resistance -** Given a hash h it should be hard to find any message m such that h = hash (m). This concept is related to that of one-way function. Functions that lack this property are vulnerable to preimage attacks.
- **Second preimage resistance -** Given an input m_1 it should be hard to find another input m_2 — where $m_1 \neq m_2$— such that hash (m_1) = hash (m_2). This property is sometimes referred to as weak collision resistance, and functions that lack this property are vulnerable to second preimage attacks. [8]
- **Collision resistance -** It should be hard to find two different messages m_1 and m_2 such that hash (m_1) = hash (m_2). Such a pair is called a cryptographic hash collision, a property which is sometimes referred to as strong collision resistance. It requires a hash value at least twice as long as that required for preimage-resistance, otherwise collisions may be found by a birthday attack. [1,2,4]

2.2 Modification Detection Codes (MDC)

A move from general properties and constructions to specific hash functions is now made, and in this section the subclass of unkeyed hash functions known as modification detection codes (MDCs) is considered. From a structural viewpoint, these may be categorized based on the nature of the operations comprising their internal compression functions.

2.2.1 Single-length MDCs

Single-length hash functions based on block ciphers make use of the following predefined components:

- a generic n-bit block cipher E_K parameterized by a symmetric key K;
- a function g which maps n-bit inputs to keys K suitable for E (if keys for E are also of length n, g might be the identity function); and
- a fixed (usually n-bit) initial value IV , suitable for use with E.

2.2.2 Double-length MDCs

MDC-2 and MDC-4 are manipulation detection codes requiring 2 and 4, respectively, block cipher operations per block of hash input. They employ a combination of either 2 or 4 iterations of the Matyas-Meyer-Oseas (single-length) scheme to produce a double-length hash. When used as originally specified, using DES as the underlying block cipher, they produce 128-bit hash-codes. The general construction, however, can be used with other block ciphers. [1]

2.3 MDC-2 (DES-based)

MDC-2 makes use of the following pre-specified components:

- Use DES algorithm as the block cipher E_K of bit length n = 64 parameterized by a 56-bit key K;
- Two functions g and $\tilde{g}$ which map 64-bit values U to suitable 56-bit DES keys as follows. For U = $u_1u_2 \ldots u_{64}$, delete every eighth bit starting with u_8, and set the 2nd and 3rd bits to '10' for g, and '01' for $\tilde{g}$:

g (U) = u_1 1 0 $u_4u_5u_6u_7u_9u_{10} \ldots u_{63}$
$\tilde{g}$ (U) = u_1 0 1 $u_4u_5u_6u_7u_9u_{10} \ldots u_{63}$

MDC-2 is illustrated in Figure 1. (E = DES cipher encryption)

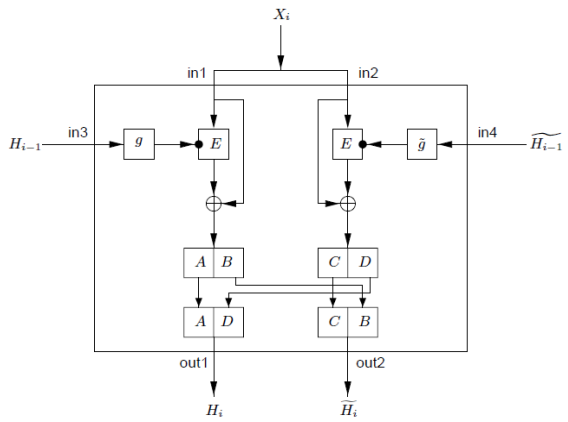

Figure 1. MDC-2 hash function

INPUT: string x of bit length r = 64t for t ≥ 2.
OUTPUT: 128-bit hash-code of x.

1. Partition x into 64-bit blocks x_i: $x = x_1 x_2 \ldots x_t$.
2. Choose the 64-bit non-secret constants IV, $\widetilde{IV}$ (the same constants must be used for MDC verification) from a set of recommended prescribed values. A default set of prescribed values is (in hexadecimal):
   IV = 0x5252525252525252,
   $\widetilde{IV}$ = 0x2525252525252525.
3. Let || denote concatenation, and C_i^L , $\widetilde{C}_i^R$ the left and right 32-bit halves of C_i. The output is $h(x) = H_t \| \widetilde{H}_t$. [1, 3]

Figure 2. MDC-2 hash function algorithm

3. System Implementation

The proposed hash function *h* is designed as iterative processes which hash arbitrary length inputs (any text) by processing successive fixed-size blocks of the input, as illustrated in Figure 3. A hash input *x* of arbitrary finite length is divided into fixed-length 256-bit blocks x_i. This preprocessing typically involves appending extra bits (padding) as necessary to attain an overall bit length which is a multiple of the block length 256, and often includes a block or partial block indicating the bit length of the unpadded input. Each block x_i then serves as input to an internal fixed-size hash function *f (Modified MDC-2)*, the compression function of *h*, which computes a new intermediate result of bit length *n* for some fixed *n*, as a function of the previous *n*-bit intermediate result and the next input block x_i. Letting H_i denote the partial result after stage *i*, the general process for an iterated hash function with input $x = x_1 x_2 \ldots x_t$ can be modeled as follows:

$H_0 = IV$; $H_i = f(H_{i-1}, x_i)$, $1 \le i \le t$; $h(x) = g(H_t)$

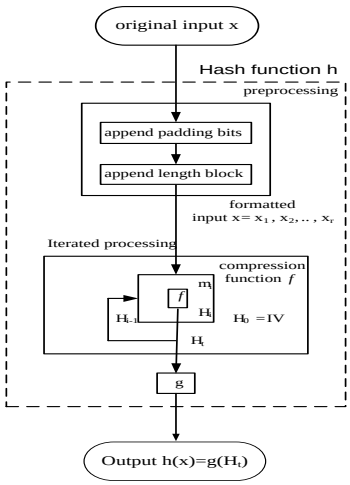

Figure 3. Overview of the proposed system

H_{i-1} serves as the *n*-bit chaining variable between stage $i - 1$ and stage *i*, and H_0 is a pre-defined starting value or initializing value *(IV)*. An optional output transformation *g* is used in a final step to map the *n*-bit chaining variable to an *m*-bit result $g(H_t)$; g is often the identity mapping $g(H_t) = H_t$.

3. 1 Modified MDC-2 with AES

Modified MDC-2 makes use of the following pre-specified components:

- Use AES algorithm as the block cipher E_K of bit length n = 128 parameterized by a 128-bit key K.
- Two functions g and $\widetilde{g}$ which map 128-bit values U to suitable 128-bit AES keys as follows. For $U = u_1 u_2 \ldots u_{128}$, set the 2nd and 3rd bits of each byte to '10' for g, and '01' for $\widetilde{g}$:

  g (U)= u_1 1 0 $u_4 u_5 u_6 u_7 u_8 u_9$ 1 0 $u_{12} \ldots u_{128}$
  $\widetilde{g}$ (U)= u_1 0 1 $u_4 u_5 u_6 u_7 u_8 u_9$ 1 0 $u_{12} \ldots u_{128}$

Modified MDC-2 is illustrated in Figure 4. E = AES cipher encryption

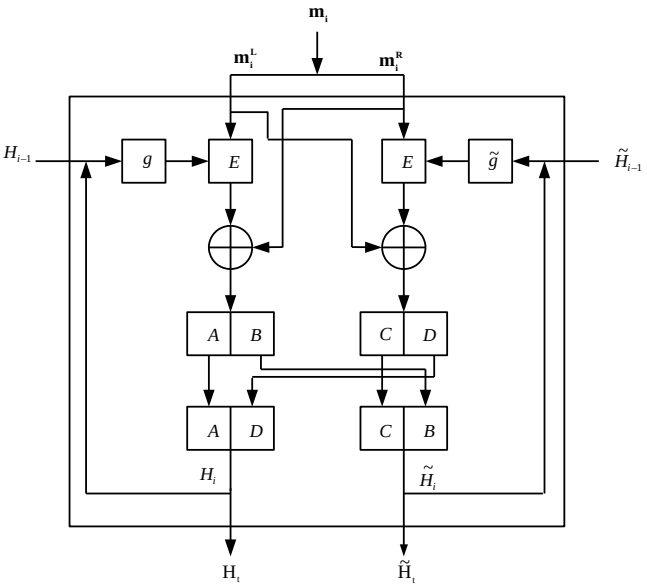

Figure 4. Modified MDC-2 hash function

INPUT: string x of bit length r = 128 t for t ≥ 2.
OUTPUT: 256-bit hash-code of x.

1. Partition x into 256-bit blocks x_i: $x = x_1 x_2 \ldots x_t$.
2. Choose the 128-bit non-secret constants *IV*, $\widetilde{IV}$ (the same constants must be used for MDC verification) from a set of recommended prescribed values. A default set of prescribed values is (in hexadecimal):
   IV = 0x5252525252525252,
   $\widetilde{IV}$ = 0x2525252525252525.

3. Let || denote concatenation, and C_i^L , $\widetilde{C}_i^R$ the left and right 64-bit halves of C_i. The output is h(x) = H_t || $\widetilde{H}_t$.

Figure 5. Modified MDC-2 hash function algorithm

4. Experimental Result

The Figure 6 shows the analysis result of Modified MDC-2 Hash Algorithm. In this analysis, the hash value was produced 10,000 times for 10,000 random messages and 10,000 random key. And compare these 10,000 hash value. The system found no redundancy. So, the accuracy of this hash function is 100%.

Sr No	Message	Key	Hash Value	Status
9968	ICXFTPST	AFRXXMGKMZKT	[illegible]	OK
9969	QCKJK	YDGPH	d9fbfa3febb44448be6ddd602d3dbd55f57802f23844c46d42d3a...	OK
9970	VYGWWIDJPAV	AAWMGLZX	457b4da83ad4a3c9aea9a5d5ca736325b676f21db726affba0d1...	OK
9971	MSITVASLNWHXR	JBIQO	2b5205a7f19615d9dc9ded19536ef2f32116ec1c318cc9073a4df...	OK
9972	TOBHSAWO	LINPQT	b8c4b416d94180b8d056b09804aa201418e34b5b8fc5e56630e...	OK
9973	GFPEKHEJ	ZZXZPWL	6b8ab898f882cedd4db7e05d7538900c1dc81bf2e4f557cf01584...	OK
9974	IOTXKOAAE	LQCYD	fd15fc3a15e21d40bc1a7af039578dd8245d078d1c8b6f540d88e...	OK
9975	UMSHPGJZ	VKYFCT	75876730dcf9ce75c81650b1b927f5b8ee6192392a7acbe1db83...	OK
9976	KYHHPUW	EFAEGHCN	4a67707b0b9e38e3687b8061aece08d71c5098f1621df074b12c...	OK
9977	FBMBVAKOYT	OTIHZBWDPOC	fc758011fc09983e4ca5754d75bc6d68f19a97dfe2aedad6fb4154...	OK
9978	PXVLYYLWBV	VERHOHBHTZHFIV	9918f38f939016c716b43812d01c166b6fe55d4830aa86214052...	OK
9979	GZTHLUW	KILKM	bf967c7259c83fde819aa9223422edb2f36f1b4836f42668f714e4...	OK
9980	SHVWYNU	AOWHGA	fc62930973a6bbbb7425857121fee0a4be7d7a386b817765668f...	OK
9981	VSNHVXXJVH	TJVKDLVDWEZ	9854fd02d3bbb77004c33dd3c78a1df89ec02d8d7c569623cf530...	OK
9982	WROSTEEXQZDH	OKDREX	8874db9e1a23a5626113d5620ecea5dbfa0e63da0aa10680633...	OK
9983	EMTHOQAVXMRYHB	IKGPWVXYAUGU	40205d9fa1f42fc76d9ded2233f315a9873f9ae1ff8f590fb74c153e...	OK
9984	RLXMLVDIKA	ASUZBXEJ	9743b83310cf04e75d09bfb9a28c0725c144e6ce53bcd2d5e4f0d...	OK
9985	YJCHOQDZEBWACF	BAQYCLQBINIECB	1918d1e89d65dc99f5f072cc5a4f115fff2ad89b40fbee15d9e2092...	OK
9986	NCXGJHHJYPYT	FRDTKPOQXVOZH	78fe8c4052e66999a0255de2841a0aa63ec2c7be667c9cc327d2...	OK
9987	OEJDSSY	QQTEDSRLVRWRFU	568eee0af81fd9137a8248fc32836922dec62fbac499c553c0767...	OK
9988	PXICBYHBCUXVE	JPXAOSF	4e6e7ada4fafe12a5875c53244ebea3b0d83b08ec452e7ca8ff9b...	OK
9989	THIBKPENX	XIEXHECD	3ad8db826b5efe0dbe2d06f179e9272b3473265d35fe383b2125...	OK
9990	LOGMYEDM	VFCJZUAIPIOFL	1e5c295ba924b5f429c99a09ba805f1901824902a680839fe0ed...	OK
9991	SBNEMLIAWRMJ	YIJXZYO	d03337802af87eb65f556adef97bb5866156fec752c7a070849dd...	OK
9992	SHHUSMB	VDCFE	05d12b756ec7806e60fa0b361340592cb7985d634e9401221edf...	OK
9993	ZJMBOLYNEX	WHMIGWAFSQ	a0b2bd51a033c8ca7ae9b27a8c55d0a2a834230ee7014271e61...	OK
9994	WWZXVKYNNUH	WQJEVMYFY	9a9bd1a1c2feffef839f58967cfa17cffaebf8e74a718b18e83b28c1...	OK
9995	FRSFUZURI	HTNPCYWG	9d8d381234a7cc8b7eb0df6a67c3787b90048a83d7fc593dc2fe1...	OK
9996	HTDCDKLFRLIJ	WEEHUGER	d78b08e6f687d274b8afcb3aa1774b46a0c86f29bf12012f1f8e56...	OK
9997	XIIRV	XQXAOAPYG	c082eef8051c5ea547bcd79c75d6c0d2fd2ff46b076ae9bf57ff384...	OK
9998	XBBJXZIDJ	HEIRGVYIDK	d79b31a89f4b6a2d738d2fd49331c09a203f6a7e54c99d6405a5...	OK
9999	POWBJNFEUWVY	PTTEIXJD	8e03531224fc02b17d40c210b2320217021 8fdc2ed9895649297...	OK
10000	NXNUJBCVI	QXXNBCXASVJ	97d1cae0681b82cf9389ddb8632ecbed29ef632685a3c172e150...	OK

Figure 6. Analysis result of modified MDC-2 hash function

Table 1. Numbers of rounds for MDC-2 and modified MDC-2

Message Length (number of bits)	Number of rounds	
	MDC-2 (DES)	Modified MDC-2 (AES)
N	⌈N/64⌉	⌈N/256⌉
256	4	1
512	8	2
1024	16	4
2048	32	8

Table 1 shows the possible number of rounds for MDC-2 hash function and proposed Modified MDC-2 hash function. The MDC-2 hash function take 64-bits block at each round of its processes. The Modified MDC-2 hash function take 256-bits block of message at one round and it is split as two 128-bits blocks for internal process of each round. This is obviously decreasing the number of rounds needed to process.

Modified MDC-2's Rounds = MDC-2's Rounds / 4

Input Message: Cryptographic hash functions are a third type of cryptographic algorithm. They take a message of any length as input, and output a short, fixed length hash which can be used in (for example) a digital signature. As digital signature technology becomes more widely understood and utilized, many countries world-wide are competitively developing their own signature standards for their use and applications. Hash functions are the mathematical algorithms that transform arbitrary length sequences of bits into a hash result of a fixed, limited length.

Size: 4400 bits
Number of Blocks: 18

User Key: This is test for AES key.
AES Key: 546b1cd7383fd6a5a522bcba53803d45
G: 544b5cd7585fd6c5c542dcda53c05d45
G bar: 342b3cb7383fb6a5a522bcba33a03d25

Left Encryption: 119bea5f8df43ebe4dbe98e3968bb906
Right Encryption: dbebce695684dddd6dafc00385b37997
Register AB: 62f3ca39f89a5dca24d1f690b6eacb63
Register CD: 9899b7f922ebbaaf0cdfa86ae69311f6
G: 62f3ca39f89a5dca0cdfa86ae69311f6
G bar: 9899b71922ebbaaf24d1f690b6eacb63

.
.
.

Left Encryption: 21fd8dd0ec5f6101426fd0c02d7a3fa7
Right Encryption: d056f1fb476ebfb84f6582ee537d7d34
Register AB: 21fd8dd0ec5f6101426fd0c02d7a3fa7
Register CD: b538968f2f40bfb84f6582ee537d7d34
G: 21fd8dd0ec5f61014f6582ee5 37d7d34
G bar: b538968f2f40bfb8426fd0c02d7a3fa7

Hash Value: 21fd8dd0ec5f61014f6582ee537d7d34b538
968f2f40bfb8426fd0c02d7a3fa7

Figure 7. Implementation of modified MDC-2 hash function with huge length message

The Figure 7 is implementation of modified MDC-2 hash function with huge length message. In this process, input message is text and then message length has 4400 bits and 18 blocks. The hash value is 21fd8dd0ec5f61014f6582ee53 7d7d34b538968f2f40 bfb8426fd0c02d7a3fa7. In this process, modified MDC-2 takes 18 rounds and original MDC-2 takes 69 rounds for this message.

As a result, it can be concluded the larger the message size, the more decrease in number of rounds needed to process. Since the time complexity has not

considered in this current work, it can be hoped that the reducing in number of rounds may less the overall processing time.

5. Conclusion

The only weakness of MDC-2 is its block cipher. DES finally and definitively proved insecure. This casts some doubts on the security of MDC-2. This system shows a modified method for construction of hash functions based on block ciphers such as AES which is more secure than all others. And, AES cipher takes byte by byte (8-bits by 8-bits) in its process and DES cipher takes bit by bit in its process. So, the Modified MDC-2 Hash Function, that is MDC-2 with AES, is an efficient Hash Function. An effort is made the hash function MDC-2 algorithm which is modified to perform efficiently the iteration of compression function for message with huge length. The proposed system can prove that there is no redundancy upon testing 10000 times. Moreover, this hash function was designed to attain better security than original MDC-2 algorithm.

The system can be extended in several different ways. The system can be extended by changing the way of internal processes of the system to make more secure or faster. And can change the output length of the hash function in easier manner. Moreover, the block cipher can be replaced by other block cipher.

6. References

[1] Alfred J. Menezes, Paul C. Van Oorschot and Scott A. Vanstone, Handbook of Applied Cryptography, Chapter 9: "Hash functions and data integrity". CRC Press, 1997.

[2] Ilya Mironov. "Hash functions: Theory, attacks, and applications". Accessed on- http://research.microsoft.com/users/mironov/papers/hash_survey.pdf. Last accessed on 15th of December 2006.

[3] Lars R. Knudsen , Florian Mendel , Christian Rechberger , and Søren S. Thomsen, "Cryptanalysis of MDC-2".

[4] Murali Krishna Reddy Danda, "Design and Analysis of Hash Functions", 2007.

[5] Ralph C. Merkel Xerox PARC, "One Way Hash Function and DES".

[6] http://en.wikipedia.org/wiki/cryptography

[7] http://www.garykessler.net/library/crypto.html

[8] http://www.openbsd.org/crypto.html