The Use of NMACA Approach in Building a Secure Message Authentication Code

Khaled S. Alghathbar and Alaaeldin M. Hafez

***Abstract*[1]—** Integrity and authentication of long-term stored information are important issues that should be considered in secure storage systems. Digital archived information may include different types of objects with different representation, such as, documents, images and database tables. Authenticity of such information should be verified, especially when it transferred through communication channels. Authentication verification techniques are used to verify that information in the archive is authentic and has not been unintentionally or maliciously altered. In addition to detecting malicious attacks, integrity checks also identify data corrupted information. Message authentication code (MAC) algorithms are keyed hash functions whose specific purpose is message authentication. In most cases, MAC techniques use iterated hash functions, and those techniques are called iterated MACs. Such techniques usually use a MAC key that is used as an input to the compression function, and is involved in the compression function *f* at every stage. A wide range of authentication techniques use un-keyed hash functions, which are known as modification detection codes (MDCs). MD4, MD5, SHA-1 and RIPEMD-160 are some of many. Recently, powerful new attacks on hash functions such MD5 and SHA-1, among others, suggest introducing more secure hash functions. In this paper, we propose a new MAC methodology that uses an input MAC key on the compression function, to permute the order of message words and shifting operation in the compression function. The new methodology can be used in conjunction with a wide range of modification detection code techniques. Using MD5 algorithm as a model, a new MD5-MAC algorithm is presented. The MD5-MAC algorithm uses the MAC key in building the hash functions by defining the order for accessing source words and defining the number of bit positions for circular left shifts.

***Keywords*—** Message Authentication Code, Modification Detection Code, Information Security, Cryptography, Hashing, MD5

I. INTRODUCTION

The integrity and authenticity of information transmitted or stored in an unreliable medium is considered as a prime necessity in modern environments of open communications [13, 18, 25]. Authentication is defined as the verification of the source of data, and that this data came from the claimed source [2, 14, 30]. Usually integrity comes in conjunction with the authentication algorithm. Integrity is assuring that, what the sender has transmitted, the receiver has received, and there is no accidental or intentional unauthorized modification of the transmitted data [6, 29, 33].

Compared to a large number of modification detection code algorithms (MDC) [6, 12, 15], many of them are block-cipher based. Those with relatively short MAC bit lengths (e.g., 32-bits for MAA [8]) or short keys (e.g., 56 bits for MACs based on DES-CBC [3]) may still offer adequate security. This is depending on the computational resources available to adversaries, and the particular environment of application. Many iterated MACs can be described as iterated hash functions. In this case, the MAC key is generally part of the output transformation; it may also be an input to the compression function in the first iteration, and be fed to the compression function at every stage. An upper bound on the security of MACs should be considered, based on an iterated compression function, which has n bits of internal chaining variable, and is deterministic (i.e., the m-bit result is fully determined by the message). The MAC forgery is possible using $O(2^{n/2})$ known text-MAC pairs. The most commonly used MAC algorithms are based on block ciphers that make use of cipher-block-chaining (CBC) [1, 4, 7, 10]. While CBC-MAC is secure for messages of a fixed number of blocks t, additional measures are required if variable length messages are allowed. RIPE-MAC [17] is a variant of CBC-MAC, producing 64-bit MACs. The internal encryption function E being either single DES or two-key triple-DES, respectively, requiring a 56- or 112-bit key k. The compression function uses a non-invertible chaining after padding, a final 64-bit length-block, giving bit length of original input, is appended; final output block is encrypted. A common suggestion is to construct a MAC algorithm from a modification detection Codes (MDC) algorithm, by simply including a secret key k as part of the MDC input [6, 10, 28, 33]. A concern with this approach is that implicit but unverified assumptions are often made about the properties that MDCs have; in particular, while most MDCs are designed to provide one-wayness [28, 32, 34]. Even in the case where a one-way hash function precludes recovery of a secret key that is used as a partial message, this does not guarantee the infeasibility of producing MACs for new inputs [9, 18]. A more conservative approach for building a MAC from an MDC is to make the MAC compression function depend on k, implying the secret key be involved in all intervening iterations; this provides additional protection in the case that weaknesses of the underlying hash function becomes known. Such technique is employed using MD5. It provides performance slightly slower to that of MD5. A MAC technique alternatively can be based on cyclic redundancy codes [15, 31].

Manuscript received January 17, 2009: Revised version received April 30, 2008. This work was supported by the Center of Excellence in Information Assurance, King Saud University.

K. S. Alghathbar is with the Center of Excellence in Information Assurance and the College of Computer and Information Sciences, King Saud University, Riyadh 11543, KSA +96650-529-5367, fax: +9661-467-5423; e-mail: kalghathbar@ksu.edu.sa.

A. M. Hafez is with College of Computer and Information Sciences, King Saud University, Riyadh 11543, KSA +96650-529-5367, fax: +9661-467-5423; e-mail: ahafez@ksu.edu.sa.

Powerful new attacks on the hash functions MD5 and SHA-1, among others [12, 18, 19, 25, 28], have suggested introducing a new and presumably more secure hash function [12, 18]. In [28], an attack on MD5 has been introduced that finds collisions of MD5 in about 15 minutes to an hour computation time. The attack is a differential attack, where exclusive-ors are not used as a measure of difference. Instead of that, it uses modular integer subtraction as the measure. The same attack could find collisions of MD4 in less than a second. In [19], the attack was applied to SHA-0 and all variants of SHA-1. The attack could find real collisions of SHA-0 in less than 2^{39} hash operations. The attack was implemented on SHA-1, and collisions could be found in less than 2^{33} hash operations.

In [1], we have proposed a new approach that is based on the message authentication code approach NMACA. In NMACA a secret key K is used to determine the order of using message words. Different algorithms such as, MD5 and SHA-1, can be adopted for the approach implementation. In NMACA-MD5, the 128 bit secret key K is used to determine the MD5 algorithmic steps. The access order for message words and the shift amounts in the distinct rounds are determined by K. Based on the experimental results, the algorithm is almost as fast as the MD5 and is as robust as MD5. In the performed experiments, the proposed algorithm has been compared to MD4, MD5, and MAC-MD5 algorithms. The results have shown that the speed performances are comparable. Two measures are considered, the confession and diffusion measures, where number of bits changed in cipher code due to slight changes in the input message and/or MAC key, are used as measures of performance. Those measures illustrate the avalanche effect of the underlying hash function. Percentage of change in a 512 bit message according to changes in number of bits shows that the new function NMACA-MD5 has a high avalanche effect. This is due to its high randomness.

In this paper, the NMACA approach which is based on the message authentication code approach NMACA is explored. In this approach, a secret key K is used to form a fast robust MAC algorithm. Different algorithms such as, MD5 and SHA-1, can be adopted for the approach implementation. Depending on the technique used, rather than using the secret key K as the initial chaining values to the algorithm, K is used to determine the access order for message words. In NMACA-MD5, a 128 bit key K is used to determine the MD5 algorithmic steps, rather than using the key as the initial chaining values to the algorithm. K is used to determine the access order for message words and to determine the shift amounts in the distinct rounds.

The rest of this paper is organized as follows: in section 2, we give related works in the area of message authentication coding. In section 3, we give the requirements needed for MAC techniques. In section 4, we give the problem definition. In section 5, we present the experimental results. The paper is concluded in section 6.

II. RELATED WORK

There are two major approaches to implement authentication/integrity mechanisms, the use of digital signature and the use of message authentication code [20, 26, 33].

In digital signature approach, public key cryptography is used, which uses a public key and a private key. A sender signs a message digitally by computing a hash function (or checksum) over the data, and then encrypts the hash function value using the private key. The encrypted hashing value is sent to the receiver accompanied with the data. The receiver would verify the authenticity of the received data by recalculating the hash value and decrypting the transmitted hashing value using the public key. The two hash values are compared, if matched then the message is authentic and came from the claimed sender.

In Message Authentication Code (MAC), a shared secret key is used instead of the private key. There are several ways to provide authentication/integrity by using the secret key [21, 24]. The main two are Hash-Based Message Authentication Codes (HMAC), and Encryption-Based Message Authentication Codes.

In HMAC, a strong hash function algorithm, such as MD5 or SHA1, is used to create a hashing value over the data and the embedded secret key. Different HMAC algorithms use different embedding strategies. At the receiver side, the same hash function is applied on the concatenated data and key. The authenticity and integrity of the received data is assured by matching the hashing value of the received message with the re-calculated hashing value.

In Encryption-Based Message Authentication Codes, a combination of hashing and encryption is used [2, 14]. A hashing value is calculated over the data using the hashing algorithm. The encryption algorithm is used to encrypt the hashing value using the secret key. At the receiver side, the hashing value is recalculated, and using the secret key, the sent hashing value is decrypted. The authenticity and integrity of the received data is assured by matching the recalculated hashing value with the decrypted hashing value.

Many algorithms have been proposed for mechanisms that provide integrity checks based on a secret key [6, 12, 21, 27]. Those mechanisms are known as message authentication codes (MACs) mechanisms. Message authentication codes are used with a secret key in order to authenticate transmitted or stored information. Typically those mechanisms use a cryptographic hash functions in conjunction with secret keys to guarantee that authenticity of information. Generally, MAC mechanisms should fulfill the following features:

- The used hash function should be available, well defined and mostly doesn't need much modification.
- The performance of the hash function should not be changed due to the use in the proposed mechanisms.
- Secret Keys should be handled in a simple way.
- The cryptographic analysis of the authentication mechanism should be based on reasonable hash function assumptions.

- The hash function of the MAC mechanism should be easily replaced with other hash functions if necessary.

III. REQUIRMENTS FOR MAC'S

In designing a MAC technique, some requirements should be satisfied. For a message X with cipher code Y, the requirements could be summarized as follows:

- It is infeasible to find another message X' with the same cipher code Y.
- A cipher code Y should be uniformly distributed
- A cipher code Y should depend equally on all bits of the message

The above requirements are used in assessing the security of a MAC function, where the different types of attacks should be considered. In the first requirement, message replacement attacks are considered, where the attacker does not know the MAC key, and is still able to construct a new message X' to match a given cipher code Y. The second requirement is concerned with preventing a brute-force attack based on certain input X. The last requirement satisfies the condition that the MAC algorithm should not be weak with respect to certain parts of the message.

A. Requirements for Hash Functions

The purpose of a hash function is to produce a fixed length code that represents an input, which could be a message, an image, a file, or any other block of data. A good hash function should satisfy some requirements that make it is extremely difficult to find two input messages with the same hash code, and there is no obvious way to relate the message to its hash code. Hash functions and block ciphers have some similarities. A hash function h should satisfy the following requirements.

- h can be applied on different messages X with different sizes.
- The output of h produces a fixed-length output Y.
- Easiness of computing Y=h(X) for any message X.
- One-way hash function or preimage resistance, where for any specified output, it is computationally infeasible to find any input that is hashed to that output, or if given Y_1 is the hash value of an unknown X_1, then it is computationally infeasible to find X_2 such that $Y_1=h(X_2)$.
- Weak collision resistance or 2nd-preimage resistance, where for any specified input X_1 , such that $Y_1=h(X_1)$, it is computationally infeasible to find another input X_2 such that $Y_1=h(X_2)$.
- Strong collision resistance, where for any two distinct inputs X_1 and X_2, it is computationally infeasible to have $h(X_1) = h(X_2)$.

B. Attacks on hash functions

For a given hash function h, the security of h is the complexity of applicable attacks, and the storage complexity of the attacks.

- For a message X_1 and n-bit hash function h(x), finding an input X_2 colliding with X_1 could be done by picking a random message X_2 and checking if $h(X_1) = h(X_2)$. Assuming the hash value is uniformly distributed, the probability of finding a match is 2^{-n}. Such case is known as a birthday attack which allows colliding pairs of messages X_1, X_2 with $h(X_1) = h(X_2)$, to be found in about $2^{n/2}$ operations, and negligible memory. An n-bit hash function has ideal security if given a hash output Y, producing each of a preimage and a 2nd-preimage requires approximately 2^n operations; and producing a collision requires approximately $2^{n/2}$ operations.
- A MAC key could be determined by exhaustive search. For a known text-MAC pair (X, Y), an attacker may compute the n-bit cipher code on that text under all possible key values, and check the results against the input pair. For a k-bit key space, the number of such trails is 2^k MAC operations, in which case $1+2^{k-n}$ candidate keys remain. Assuming the MAC behaves as a random mapping, it can be shown that one can expect to reduce this to a unique key by testing the candidate keys using just over t/n text-MAC pairs. Ideally, a MAC key (or information of cryptographically equivalent value) would not be recoverable in fewer than 2^k operations. As a probabilistic attack on the MAC key space distinct from key recovery (note that for a k-bit key and a fixed input), a randomly guessed key will yield a correct (n-bit) MAC with probability $\sim 2^{-k}$ for $k < n$.
- MAC forgery is defined as producing any input-MAC code (X, Y) without having the knowledge of the key K. For an MAC algorithm with n bits, guessing successively a MAC code Y for a given input X, or guessing a preimage X for a given MAC output code Y, has probability 2^{-n}, same as for an MDC. A difference here, however, is that guessed MAC code cannot be verified without the knowledge of the (X, Y) pairs –either by knowledge of the key K, or an algorithm that provides MAC codes Y's for given inputs X's. In such case, an attacker could be able to produce new pairs (X, Y) with at most probability less than Max $(2^{-k}; 2^{-n})$.
- For the preimage attacks and the second preimage attacks, an attacker, with large storage, can, off line, computes a large number of hash function (X, Y) pairs and could trade off the off line computation for subsequent attack time. For an n-bit hash value with 2^n possible values, if 2^r randomly selected (X, Y) pairs are stored, the attacker could decrease the number of possible computations to 2^{n-r}. For comparable values of n and r, such case could find a preimage in a possible number of trials, e.g., if n =64 and r=34, the probability of finding a preimage is increased from 2^{-64} to 2^{-30}. Similarly, the probability of finding a second preimage is increased to 2^r times its original value when 2^r input-output pairs are off line computed and stored in memory.

- For a hash function h with n-bit hash codes, at least 2^{80} operations are acceptably beyond computational feasibility. For a one-way hash functions, n >=80 is one of the needed requirements. Off-line attacks require at most 2^n operations. If off line computations are used, the number of computations could be reduced. For a collision resistant hash functions, n >= 160 is considered as a needed requirements. For a MAC, a cipher code of length n bits; n >= 64, and a MAC key of 64-80 bits are considered sufficient for most applications. A MAC key should be regularly changed. If a MAC key remains in use, off-line attacks may be possible given one or more (X, Y) pairs. For a suitable MAC algorithm, preimage, 2nd-preimage resistance, and collision resistance should follow directly from lack of knowledge of the key. For such attacks, security should depend only on the MAC key size.

In [17], Shannon defined the entropy h(p) as

C. Attacks on key selection

$$h(p)= - \sum p_i \log_2(p_i)$$

h(p) is a measure of the number of uncertainty bits associated with symbols i that has probability p_i. Key guessing attacks can be determined by measuring how bad a key distribution is. This is done by calculating key K entropy. Entropy of K; E(K), is the number of real bits of information in K. Key K can be determined in $2^{E(K)}$guesses, and

$$E(K)= - \sum p_K \log_2(p_K)$$

p_k is the probability of key K. Guessability is a related measure that is used to measure how the key can be guessed. Many measures are used in calculating guessability. One of those measures is the expected number of guesses required to get the correct key. Dictionary attacks brute force attacks strategies are considered as common strategies that are used in getting the right key. In brute force attacks, all key symbols are guessed in random order. In dictionary attacks, those symbols that deemed more probable are guessed first. A modified version of the dictionary attacks is called the optimal dictionary attack. In optimal dictionary attack, symbols are guessed in decreasing order of probability. After each symbol guessing, those symbols produced by the source are relabeled so that the first symbol in order has probability p_1 is the most likely and the sequence of other symbols i with probability p_i are ordered in a non-increasing order. The expected number of guesses is

$$G(p)= - \sum i p_i$$

The average value of G(p) is

$$(n+1)/2$$

This is the number of guesses needed to correctly guess the key K from n equally likely possibilities. To relate G(p) with h(p), we define a measure H(p) as

$$H(p)= (2^{h(p)} +1)/2$$

Experimentally [16], the relationship between G(p) and h(p) is

$$0.7\ H(p) \le G(p) \le H(p)$$

In this work, in addition to the analytical comparisons done in section 4, two more measures are used to justify the strength of MAC hash functions, namely, the confusion and the diffusion. The confusion measure is defined as the complexity of the relation between the key and the hash value, while the diffusion is defined as the relationship between the input message and the hash value.

D. Confusion and Diffusion

The confusion of a hash function h is evaluated by recognizing changes in the hash value with respect to a slight change in the MAC key. For message X and MAC key K, the generated hash value Y is defined as a function h of both X and K. If K is slightly changed to K_1, the generated hash value should be changed to Y_1, and Y_1 is completely unrelated to Y. In other words, it is highly inapplicable to find a relation between $Y = h(X,K)$ and $Y_1 = h(X,K_1)$ where K and K_1 are different, even if difference between the both of them is very little. A hash function h is considered highly confused if the generated hash values for slightly different MAC keys, even in one bit, then the generated hash values are completely different. Since the probability of having one bit of the hash value flipped is ½, then, assuming the distribution of such change is uniformly distributed, then half of the randomly located bits, would be flipped. If the hash function has a weak confusion property, one can easily use brute force attacks to determine the MAC value. In such case, searching in the search space will be computationally acceptable. By changing few bits in K, then measure the relative or normalized number of changed bits in Y and Y_1, α , where

α =*Normalized number of bits changed in (h(X,K), h(X,K$_1$))*

By comparing α with a threshold value α_{min} , $0 \le \alpha_{min} \le 1$, if $\alpha \ge \alpha_{min}$, then the hash function h has a high confusion value with respect to MAC key K.

Similar to the confusion measure, the diffusion of a hash function h is observed by getting changes in the hash value with respect to slight changes in input message X. For message X and MAC key K, the generated hash value Y is defined as a function h of both X and K. If X_1 is a slightly changed message from X_1, the generated hash value is changed to Y_1. Ideally Y_1 should be completely unrelated to Y. In other words, it is highly inapplicable to find a relation between $Y = h(X, K)$ and $Y_1 = h(X_1, K)$ where X and X_1 are different. This could happen even if the difference between the X and X_1 is quite little, say one bit. A hash function h has a good diffusion if for slightly different messages, could be one bit difference, the generated hash values are completely different. Using the same argument we used in the confusion

measure, the probability of having one bit of the hash value flipped is ½, then, assuming the distribution of such change is uniformly distributed, then half of the randomly located bits, would be flipped. If the hash function has a weak diffusion property, one can easily use brute force attacks to determine the message X. If few bits of X are changed, the diffusion measure is the relative or normalized number of changed bits in Y; β, where

β =*Normalized number of bits changed in* $(h(X,K), h(X_1,K))$

By comparing β a threshold β_{min}, $0 \le \beta_{min} \le 1$, if $\beta \ge \beta_{min}$, then the hash function h has a high diffusion value with respect to the input message X.

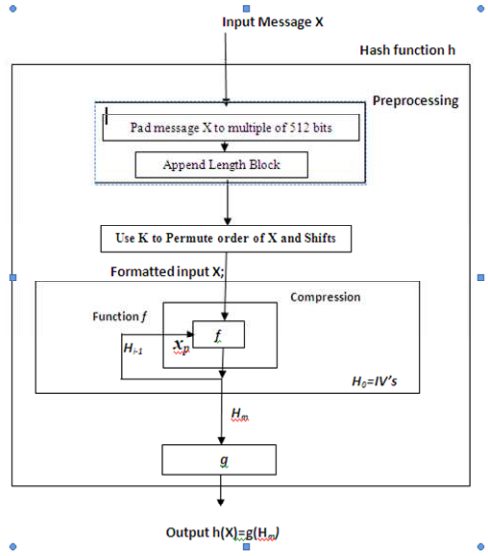
IV. The NMACA Approach

In this section, the NMACA approach is demonstrated on the MD5 technique. The new technique is called the NMACA-MD5 message authentication code algorithm. NMACA-MD5 takes as an input, a message X of arbitrary length and a secret 128 key. The produced cipher code is a 128-bit MAC. It is conjectured that it is computationally infeasible to produce two messages having the same MAC, or to produce any message having a given pre-specified target MAC. NMACA-MD5 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. NMACA-MD5 is more conservative in design than previous well known algorithms. The details of the NMACA-MD5 algorithm will be described in section 4.1.

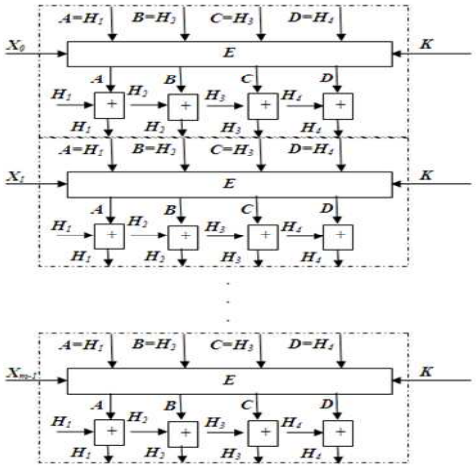
The structure of the proposed algorithm is illustrated in figure 1.

A. The NMACA-MD5 message authentication code algorithm

A more conservative approach to build a MAC from an MDC is to arrange the MAC compression procedure such that it depends on secret key k, implying that k is involved in all intervening iterations. This provides an additional protection in case when weaknesses of the underlying hash function becomes known. Algorithm MD5-MAC is such a technique. MD5-MAC is obtained from MD5 by replacing the four 32-bit IV's of MD5 by the secret key K [1]. We propose the new NMACA-MD5 by involving the key K in the different steps of MD5 algorithmic, rather than using the key as the initial chaining values to the algorithm. K is used to determine the access order for message words and to determine the shift amounts in the distinct rounds. In the proposed algorithm, only two changes are done to the MD5 algorithm. Both algorithms use the same initial chaining values (IV's), apply the same hash functions, and use the same padding process to adjust the length of the transmitted message. The two differences between the MD5 and the NMACA-MD5 techniques are the order of message words, and the order of values that are used in circular left shifts on the four auxiliary functions. The 128-bit secret key K is divided into two parts, each of 64 bits. The first 64 bits are used to rearrange the order of the message words. Those 64 bits of K are divided into four divisions; each has 16 bits. Each of the 16 bits of K is used to rearrange a block of 16 words (32-bit words) of the input message. Starting from a vector of a predefined order of accessing each of 16 words, the order of such vector is changed according to the related part in K. The other 64 bits of K are used to define the numbers of circular left shifts used in the four auxiliary functions. Using the same key related permutations, the order of circular shifts in each of the four auxiliary functions is defined by using one of the 16-bits of K.



(a) General NMACA-MD5



(b) Detailed NMACA-MD5

Figure 1. NMACA-MD5

The proposed algorithm is as fast as MD5 and is as robust as MD5. The algorithm is depicted below.

Algorithm NMACA- MD5

INPUT: bit string x of arbitrary bit length b ≥ 0 and128-bit key K.
OUTPUT: 128-bit hash-code of x.

a) Define four 32-bit initial chaining values (IVs): *h1=0x67452301,h2= 0xefcdab89, h3 = 0x98badcfe, h4=0x10325476.*

b) Four auxiliary functions are defined, such that each takes as input three 32-bit words and produce, as an output, a 32-bit word. They apply the logical operators *and*, *or*, *not* and *xor* on the input bits.
f(B, C, D)= (B and C) or (not(B) and D)
g(B, C, D)= (B and D) or (C and not(D))
h(B, C, D)= (B xor C xor D)
i(B, C, D)= (C xor (B or not(D))

c) Define additive 32-bit constants:
y[j] = first 32 bits of binary value abs(sin(j+1)), $0 \le j \le 63$, where j is in radians and "abs" denotes absolute value.

d) Use secret key:

d.1) Define order for accessing source words by using the secret key K:
z[0::15] = [Permutation P_O of the 1st 16 bits of K, P_0 : {0, 1, …, 15}
z[16::31] = [Permutation P_1 of the 2nd 16 bits of K, P_1 : {16, 17, …, 31}→ $\{O_i \mid 16 \le O_i \le 31\}$],
z[32::47] = [Permutation P_2 of the 3rd 16 bits of K, P_2 : {32, 33, …, 47}
z[48::63] = [Permutation P_3 of the 4th 16 bits of K, P_3 : {48, 49, …, 63}

d.2) Define the number of bit positions for left shifts (rotates) by using the secret key K:
s[0::63] = [Permutation P_s of the last 64 bits of K, P_s : {0, 1, …, 63} → $\{O_i \mid 0 \le O_i \le 63\}$],

e) Preprocessing:
Pad x such that its bit length is a multiple of 512, as follows. Append a single a bit of 1, then append (r−1) bits of 0's, r=512-(b+64) mod 512, r≥ 0. Finally append the least significant 32 bits of b, then append the next least significant 32bits of b, i.e., the last 64 bits of b with least significant word first. Let m be the number of 512-bit blocks in the resulting string (b + r + 64 = 512m =32 x 16m). The formatted input consists of 16m 32-bit words: $x_0\ x_1 \ldots x_{16m-1}$. Initialize: $(H_1; H_2; H_3; H_4) \leftarrow (h_1; h_2; h_3; h_4)$.

f)* Processing*:
For each i from 0 to m − 1, copy the i^{th} block of 16 32-bit words into temporary storage: $X[j] = x_{16i+j}$; $0 \le j \le 15$, then process these as below in four 16-step rounds before updating the chaining variables:
(*initialize working variables*)

$(A;B;C;D) \leftarrow (H_1; H_2; H_3; H_4)$.

(*Round 1*)
For j from 0 to 15 do the following:
$t \leftarrow (A + f(B;C;D) + X[z[j]] + y[j])$,
$(A;B;C;D) \leftarrow (D; t \leftarrow s[j];B; C)$.

(*Round 2*)
For j from 16 to 31 do the following:
$t \leftarrow (A + g(B;C;D) + X[z[j]] + y[j])$,
$(A;B;C;D) \leftarrow (D; t \leftarrow s[j];B; C)$.

(*Round 3*)
For j from 32 to 47 do the following:
$t \leftarrow (A + h(B;C;D) + X[z[j]] + y[j])$, $(A;B;C;D) \leftarrow$ $(D; t \leftarrow s[j];B; C)$.

(*Round 4*)
For j from 48 to 63 do the following:
$t \leftarrow (A + k(B;C;D) + X[z[j]] + y[j])$,
$(A;B;C;D) \leftarrow (D; t \leftarrow s[j];B; C)$.

(*update chaining values*)
$(H1;H2;H3;H4) \leftarrow (H1+A;H2+B;H3+C;H4+D)$.

g) ***Completion***
The final MAC value is the concatenation: H1||H2||H3||H4 (with first and last bytes, the low- and high-order bytes of H1, H4, respectively).

V. EXPERIMENTAL RESULTS

In this section, experiments results are shown, that are used to show the performance of MD5 and NMACA-MD5 in terms of speed and avalanche effect. We have run our experiments on a 2.4 GHz machine, with 4 GB of RAM and running windows Vista. Messages of up to 1Mbytes are used. From the results obtained in figure 2, we have found that, the speed performance of the NMACA-MD5 algorithm is comparable to the speed of the MD5 algorithm. Actually, such performance is predictable. The difference between the performance of MD5 and NMACA-MD5 comes from three different factors,

- The initial time needed in calculating the order of message words.
- The initial time needed in calculating the number of circular left shifts
- The time needed for circular left shifting. Depending on the key K, the time needed for circular left shifting in NMACA-MD5 could be less or more than the time needed in MD5.

The first two factors are fixed for all messages regarding its length, and the third factor depends on the message length and should have some effect on the performance of NMACA-MD5.

Also, the experimental results have shown that by using different bit changes in the MAC key K, we have found that the confusion effect of the proposed approach is almost 50%, which is considered an indicator of the quality of the hash function used. The results is the same as in the MD5 approach, which means the use of NMACA with MD5 has not degraded the quality of the MD5 approach. The same applies to the diffusion effect.

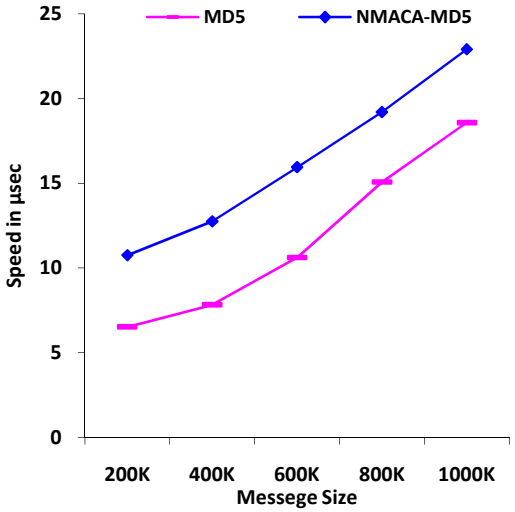
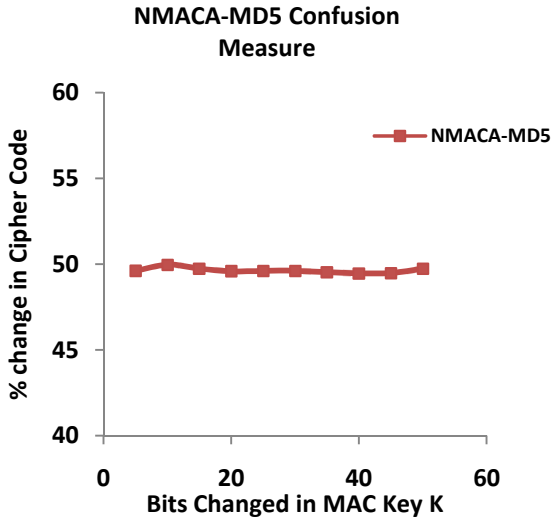


Figure 2. Speed performance of MD5 and NMACA-MD5

Changed Bits (In MAC Key)	NMACA-MD5
5	50.26%
10	50.39%
15	50.18%
20	49.89%
25	49.94%
30	50.05%
35	49.96%
40	49.91%
45	49.57%
50	49.96%

(a)

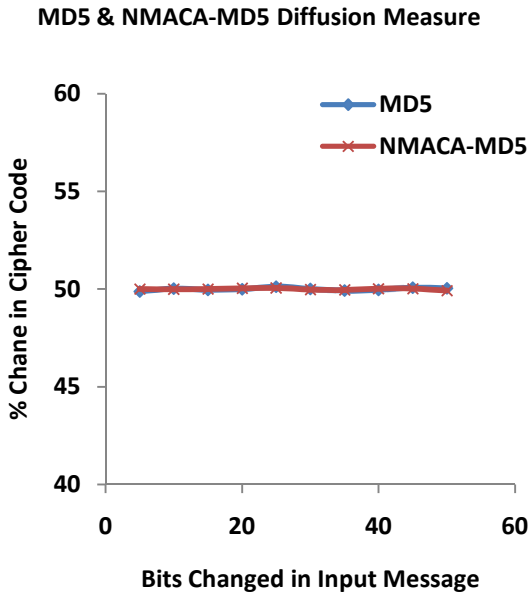


(b)

Figure 3. Confusion Measure or the Avalanche Effect of MAC key K

Change Bits (In Input Message)	MD5	NMACA-MD5
5	49.86	50.019
10	49.08	49.991
15	50.01	50.013
20	49.43	50.036
25	49.58	50.056
30	49.9	49.971
35	50.07	49.955
40	50.19	50.031
45	49.66	50.034
50	49.61	49.933

(a)



(b)

Figure 4. Diffusion Measure or the Avalanche Effect of Message X

VI. CONCLUSIONS

We have proposed a new approach that is based on the message authentication code approach NMACA. In NMACA, using any of the MDC techniques, such as, MD5 and SHA-1, a secret key K is involved in all steps of applying the used MDC technique. In this work, we have adopted the MD5 technique to demonstrate the proposed NMACA approach; we call it NMACA-MD5. In NMACA-MD5, a 128 bit secret key K is used to determine the MD5 algorithmic steps. The access order of message words and the shift amounts of the distinct rounds are determined by K. Based on our experimental results, NMACA-MD5 has been compared to MD5. The results have shown that the speed performance results of both implementations are almost the same. That is due to the fact

that the only extra overhead needed for NMACA-MD5 is the processing time of defining the reordering of message words and circular left shifts. The diffusion and confusion effects are studied by studying the effect of changes in a message bits (diffusion) and MAC key bits (confusion). Those two measures illustrate the avalanche effect of the underlying hash function. The experimental results have shown that, by using different bit changes in the input message and the MAC key K, we have found that the confusion and diffusion effects of the proposed approach are almost 50%, which is considered an indicator of the quality of the hash function used.

References

[1] K. Alghathbar, A. Hafez, F. B. Muhaya, and H. M. Abdalla, "NMACA: A Novel Methodology for Message Authentication Code Algorithms", 8th WSEAS Int. Conf. on Telecommunications and Informatics (TELE-INFO '09), Istanbul, Turkey, May 30 - June 1, 2009.

[2] T. Aura, P. Nikander, and J. Leiwo, "DOS-resistant authentication with client puzzles", volume 2133 of Lecture Notes in Computer Science, pages 170–177. Springer-Verlag, Berlin, Germany, 2001.

[3] S. Bellovin, "An Issue With DES-CBC When Used Without Strong Integrity", Proceedings of the 32nd IETF, Danvers,MA, April 1995.

[4] J. Black and P. Rogaway, "CBC MACs for arbitrary-length messages: The three-key constructions", Advances in Cryptology - CRYPTO '2000 Proceedings, Volume 1880 of Lecture Notes in Computer Science (Mihir Bellare, ed.), pp. 197-215. Springer-Verlag, 2000.

[5] H. Chien, T. Hsu, and Y. Tang, "Fast Pre-authentication with Minimized Overhead and High Security for WLAN Handoff", WSEAS TRANSACTIONS on COMPUTERS, No. 2, Vol. 7, pp. 46-51, February 2008.

[6] M. D. Corner and B. D. Noble, "Zero-Interaction Authentication," in IEEE/ACM MOBICOM, pp. 1–11, September 2002.

[7] B. Coskun, N. Memon, "Confusion/Diffusion Capabilities of Some Robust Hash Functions", CISS 2006: Conference on Information Sciences and Systems, March 22-24, 2006, Princeton, NJ.

[8] D. W. Davies, "The Message Authenticator Algorithm (MAA) and its Implementation", National Physical Laboratory, Teddington, Middlesex TW11 0LW, UK, 1988. http://www.cix.co.uk/~klockstone/maa.htm

[9] O. Elkeelany, M. M. Matalgah, K. Sheikh, M. Thaker, G. Chaudhary, D. Medhi, and J. Qaddour, "Perfomance Analysis Of IPSEC Protocol: Encryption and Authentication," in IEEE Communication Conference (ICC), pp. 1164–1168, May 2002.

[10] D. B. Faria and D. R. Cheriton, "DoS and Authentication in Wireless Public Access Networks," pp. 47–56, September 2002.

[11] J. Fridrich, "Robust bit extraction from images," in ICMCS '99, Florence,Italy, June 1999.

[12] S. Haber, P. Kamat, and K. Kamineni, "A content integrity service for digital repositories," Open Repositories, Southampton, UK, January 2008.

[13] S. Haber and P. Kamat,"A Content Integrity Service For Long-Term Digital Archives", the IS&T Archiving 2006 Conference, 23-26 May 2006, Ottawa, Canada.

[14] S. Haber and W. S. Stornetta, "How to time-stamp a digital document," Journal of Cryptology, 1991.

[15] T. Karygiannis and L. Owens, "Wireless Network Security 802.11, Bluetoothand Handheld Devices," National Institute of Technology, Special Publication, pp. 800–848, November 2002.

[16] D. Malone and W. Sullivan, "Guesswork is not a substitute for entropy", In Proceedings of the Information Technology and Telecommunications Conference, October 2005.

[17] J. Menezes and P. C. van Oorschot, "Handbook of Applied Cryptography", CRC Press, 1997.

[18] Y. Matsunaga, A. Merino, T. Suzuki, and R. H. Katz, "Secure Authentication System for Public WLAN Roaming," pp. 113–121, 2003.

[19] M. K. Mihcak and R. Venkatesan, "New iterative geometric methods for robust perceptual image hashing," in Proceedings gf the Digital Rights Management Workshop, November 2001.

[20] L. Y. Por, X. T. Lim, M. T. Su, and F. Kianoush, "The Design and Implementation of Background Pass-Go Scheme Towards Security Threats", WSEAS TRANSACTIONS on INFORMATION SCIENCE & APPLICATIONS, No. 6, Vol. 5, pp. 943-952, June 2008.

[21] B. Preneel and P. Oorschot, "MDx-MAC and building fast MACs from hash functions," Lecture Notes in Computer Science, Volume 963, pp. 1-14. Springer-Verlag, 1995.

[22] C. E. Shannon, "Communication theory of secrecy systems," Bell System Technical Journal, vol. 28, pp. 656–715, October 1949.

[23] G. Skinner, "Making A CASE for PACE: Components of the Combined Authentication Scheme Encapsulation for a Privacy Augmented Collaborative Environment", WSEAS TRANSACTIONS on COMPUTERS, No. 6, Vol. 7, pp. 630-639, June 2008.

[24] S. Song and J. JaJa, "ACE: A Novel Software Platform to Ensure the Integrity of Long Term Archives", Proceedings of the Archiving 2007 Conference, May 2007, Washington, DC.

[25] M. Stevens, A. Lenstra, and B. deWeger, Chosen-prefix collisions for MD5 and colliding X.509 certificates for different identities, EUROCRYPT 2007 (Moni Naor, ed.), LNCS, vol. 4515, Springer, 2007, pp. 1–22.

[26] Q. Sun, J. Apostolopoulos, C. Chen, S. -Fu Chang, "Quality-Optimized and Secure End-to-End Authentication", Proceedings of the IEEE, Vol. 96, No. 1, January 2008

[27] R. Venkatesan, S. Koon, M. Jakubowski, and P. Moulin, "Robust image hashing," in Proc. IEEE Int. Conf. Image Processing, 2000.

[28] X. Wang and H. Yu. “How to break MD5 and other hash functions.” In Ronald Cramer, editor, Advances in Cryptology – EUROCRYPT 2005, Volume 3494 of Lecture Notes in Computer Science, 2005.
[29] X. Wang, Y. L. Yin, and H. Yu. “Finding collisions in the full SHA-1.” In Victor Shoup, editor, Advances in Cryptology — CRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, 2005.
[30] C. Wright, R. Spillane, G. Sivathanu, and E. Zadok, “Extending ACID Semantics to the File System”, ACM Transactions on Storage (TOS), May 2007.
[31] E. Zadok, R. Iyer, N. Joukov, G. Sivathanu, and C. Wright, On Incremental File System Development”, ACM Transactions on Storage (TOS), May 2006.
[32] Y. Zahur and T. A. Yang, “Wireless LAN Security and Laboratory Designs,” Journal of Computing Sciences in Colleges, vol. 19, pp. 44–60, January 2004..
[33] IETF, “PPP EAP TLS Authentication Protocol,” RFC 2716, October 1999.
[34] “The Keyed-Hash Message Authentication Code (HMAC)”, Federal Information Processing Standards Publication 198, March 2002.