"A Modified Approach for Implementation of an efficient Padding Scheme in Digital Signature System"

Meenakshi Kaul [1], Dharmendra Choukse[2] and Umesh Kumar Singh[3]

[1] School of Computers and Electronics, IPS Academy, Indore, India
kaulmee@gmail.com
[2]Institute of Engineering & Science, IPS Academy, Indore, India
dharmendrachoukse@ipsacademy.org
[3] Institute of Computer Science, Vikram University, Ujjain, India
Umeshsingh@rediffmail.com

ABSTRACT

In order to provide secure transaction of documents over an insecure channel, Digital Signature Systems are made use of and Hash function is an eternal component of it. The requirement of devising an improved approach to reduce the impact of attacks in Cryptanalysis formed a driving force behind the emergence of changes in padding and parsing schemes used from time to time. However, it has been found that these schemes have not proved to be completely as efficient as this critical application demands. Therefore, there is always a scope for their improvement. The paper is organized into parts; in the first ones we give an overview of hash functions and a brief presentation of its use in Digital Signature. However the rest of parts are consecrated for our proposed improvement for padding structure and comparative results drawn from correlation coefficients obtained, finishing the paper by a conclusion and future extension of this work.

KEYWORDS

Digital Signature, Hash, Padding, Correlation.

1. INTRODUCTION

In modern society information has become a valuable commodity. It is often necessary to protect its confidentiality, which means that it should be infeasible for unauthorized people to learn the content. Cryptographic techniques have been used for many centuries to protect military and diplomatic secrets**.** Most of these techniques are encryption schemes that convert a message into a cryptogram by an invertible operation (encryption) depending on a small piece of secret information (the key).

The main focus of this paper is on the analysis and design of one particular category of algorithms: cryptographic hash functions. These are algorithms that take inputs of arbitrary length and produce as output a short string of bits. Their most important use is for the protection of data authenticity, but they are a versatile building block, used also in conjunction with digital signature schemes. The main idea of the use of a cryptographic hash function is to reduce the problem of protecting a message to the problem of protecting a short message (the hash result).[17][18]

The originator of the message M can make use of a cryptographic hash function h to compress his message into a short bit string h(M) (the hash result), and this value can be transmitted to the recipient via the authentic channel. The message itself is sent over the insecure channel .The recipient independently

hashes the message he receives (using the same cryptographic hash function) and accepts the message only if the result agrees with the value h(M). In order to cheat the system the adversary must create a modified message M', with the property that h (M') = h(M). For this he must use a one-way or collision-resistant hash function. [17][18]

A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h (that is, h = H (m)). A hash function H is said to be one-way if it is hard to invert, where ``hard to invert'' means that given a hash value h, it is computationally infeasible to find some input x such that H(x) = h. If, given a message x, it is computationally infeasible to find a message y not equal to x such that H(x) = H(y), then H is said to be a weakly collision-free hash function. A strongly collision-free hash function H is one for which it is computationally infeasible to find any two messages x and y such that H(x) = H(y). The hash value represents concisely the longer message or document from which it was computed; this value is called the message digest. One can think of a message digest as a "digital fingerprint" of the larger document.[17,18,19,20]

2. CRYPTOLOGY

It comprises two complementary fields of research: cryptography and cryptanalysis. A cryptographer is concerned with the development of new schemes or algorithms, providing security services such as confidentiality and authenticity. Data integrity is the property whereby data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source. Data origin authentication is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some (typically unspecified) time in the past. By definition, data origin authentication includes data integrity (information which has been modified effectively has a new source). Let us now consider a situation where an originator wants to send a message to a recipient over an insecure channel. For this setting the problem of authenticity is also called message authentication. It means that it should be possible to determine the source of the message and to assure that the message is not modified in any way during the transmission. The main idea of the use of a cryptographic hash function is to reduce the problem of protecting a (possibly long) message to the problem of protecting a short imprint of the message (the hash result).[22]

A cryptanalyst on the other hand is concerned with the development of attack methodologies that break a cryptographic algorithm, allowing, for example, unauthorized people access to secret information or the ability to forge documents. There is a strong relation between these two fields. We describe different types of hash functions, provide definitions and discuss the security provided by these algorithms. The use of hash functions in schemes for optimizing digital signature schemes is also overviewed.

2.1 Message authentication based on a hash function

Data integrity is the property whereby data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source. Data origin authentication is a type of authentication whereby a party is corroborated as the (original) source of specified data created at some (typically unspecified) time in the past. By definition, data origin authentication includes data integrity (information which has been modified effectively has a new source). Let us now consider a situation where an originator wants to send a message to a recipient over an insecure channel. For this setting the problem of authenticity is also called message authentication. It means that it should be possible to determine the source of the message and to assure that the message is not modified in any way during the transmission. The main idea of the use of a cryptographic hash function is to reduce the problem of protecting a (possibly long) message to the problem of protecting a short imprint of the message (the hash result).[22]

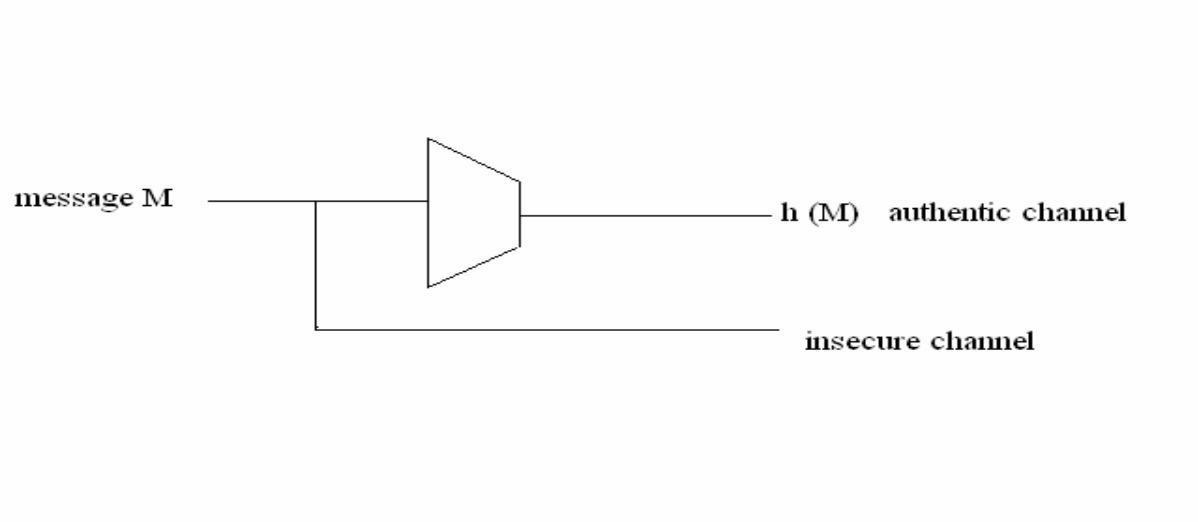


Fig 1: Message authentication using a hash function.

The originator of the message M uses a cryptographic hash function h to compress his message into a short bit string h (M) (the hash result), and this value is transmitted to the recipient via the authentic channel. The message itself is sent over the insecure channel. An active adversary may be able to modify the message M before it reaches the recipient. However, the recipient independently hashes the message he receives (using the same cryptographic hash function) and accepts the message only if the result agrees with the value h (M). In order to cheat the system the adversary must create a modified message M0, with the property that h (M0) = h (M). For this he must find a second preimage, which should be infeasible for a one-way or collision-resistant hash function. The system works by transferring the authenticity of the message to the authenticity of the hash result.

2.2 Authentication combined with encryption

The protection of integrity also depends on the inherent redundancy of the information. If the plaintext corresponding to a given ciphertext contains no redundancy, all decryptions of the ciphertext are meaningful. Therefore some form of redundancy is always needed for the protection of integrity. One can use a cryptographic hash function to add sufficient redundancy to information before encrypting it.

The originator of a message M uses a cryptographic hash function h to compute the hash result h(M). He appends this to the original message resulting in a string M□h(M). Next, he uses an encryption algorithm E based on a secret key K to encrypt this information, and he sends the ciphertext E_K(M□h(M)) over the insecure channel. The recipient, who also knows the secret key, decrypts the ciphertext. He separates the recovered message from the recovered hash result and checks the correspondence using the hash function h .For an adversary who does not know the secret key used in the encryption, it should be infeasible to change the ciphertext sent over the channel without disrupting the correspondence between the recovered message and the recovered hash result.

2.3 Optimization of digital signature schemes

Digital signature schemes are a different type of cryptographic primitive. They are used for the protection of authenticity but, in addition, they also offer the service of non-repudiation. This implies that it is impossible for an originator who sends an authenticated message, to dispute at a later time having sent

this message. Digital signatures are based on public-key cryptography. All users of a public-key cryptosystem have a public key, known to everyone else and linked by some mechanism to the correct identity. Every user also has a private key which should not be disclosed to anyone else (this may be assured when the private key is stored in tamper-resistant hardware).

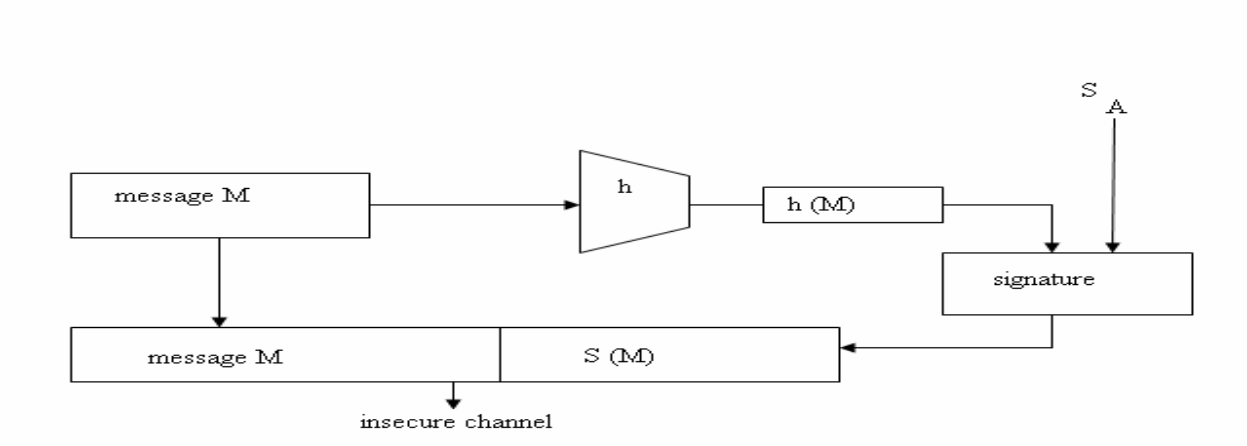


Figure 2: Digital signature scheme using a hash function.

The user A has a key pair (S_A, P_A) where S_A denotes his private (secret) key, and P_A denotes his public key. He first compresses his message M with the hash function h. The hash result h(M) is sent as input to the signature algorithm. This algorithm depends on the private key S_A and computes a value $sign S_A(h(M))$ which we denote in shorthand as s(M). The user then concatenates the signature s (M) to his message and sends the information M□s(M) over the insecure channel. The recipient of the signed message uses the signature s(M) as input to the verification algorithm. If the outcome of this is the same as the output of the verification algorithm, he accepts the message and signature as genuine. The use of a cryptographic hash function in a digital signature scheme has the advantage that the signature and verification algorithms have only short data strings to work with (the size is independent of the length of the message). This is important because public-key cryptography is much slower (several orders of magnitude) than conventional symmetric key algorithms or hash functions. Furthermore, the use of a hash function prevents attacks that exploit the algebraic structure of the message space in a signature scheme.

3. CRYPTOGRAPHIC HASH ALGORITHMS

The cryptographic operation that uses a private key to sign data does not deal directly with the data itself, but with a purportedly unique representation of this data, that has a predetermined fixed length, is short and therefore convenient to work with. This can be compared to a fingerprint as a purportedly unique, short and convenient representation of a human being. The process of creating such short representations of data is called "hashing". Unfortunately, because of the fixed length of the hash, there must exist pairs of different inputs that yield the same hash value. Good hash functions, however, have the property that finding such pairs is extremely difficult, even though they are guaranteed to exist. A "hash function" is a cryptographic function that takes as its input a bit string of any length, performs a deterministic algorithm with this input, and produces as output a bit string of fixed length. This output is called the "hash value" or the "hash" of the input. Other terms that can be found for "hash value" are "fingerprint" or "message digest".[18]

There is a long list of cryptographic hash functions, although many have been found to be vulnerable and should not be used. Even if a hash function has never been broken, a successful attack against a weakened variant thereof may undermine the experts' confidence and lead to its abandonment. For instance, in August 2004 weaknesses were found in a number of hash functions that were popular at the time,

including SHA-0 and MD5. This has called into question the long-term security of later algorithms which are derived from these hash functions — in particular, SHA-1 (a strengthened version of SHA-0). As of 2009, the two most commonly used cryptographic hash functions are MD5 and SHA-1. [13][15]However, MD5 has been broken; an attack against it was found in 2008. [1] SHA-0 and SHA-1 are members of the SHA family of hash functions developed by the NSA. In February 2005, a successful attack on SHA-1 was reported, finding collisions in about 269 hashing operations, rather than the 280 expected for a 160-bit hash function. In August 2005, another successful attack on SHA-1 was reported, finding collisions in 263 operations. Theoretical weaknesses of SHA-1 exist as well, [2][3] suggesting that it may be practical to break within years. New applications can avoid these problems by using more advanced members of the SHA family, such as SHA-2, or using techniques such as randomized hashing[4][5] that do not require collision resistance.

From a cryptographic point of view, collision resistance means that it is difficult to find two inputs that hash to the same output. By implication, the range of hash values must be large enough that a brute-force attack to find collisions is ``difficult". With a few assumptions, we can arrive at an estimate for the risk of a hash collision. We assume that the inputs to the hash function are random and uniformly distributed, and the output of the hash function is also random and uniformly distributed. Let n, be the number of input blocks, and let b, be the number of bits in the hash output. As a function of the number of input blocks, n, the probability that we will encounter one or more collisions is $1- (1- 2^{-b})^n$.

3.1 Attacks on SHA family

A hash function should satisfy the following requirements:
• (Practicality) computing the hash h (m) of any input m can be done efficiently,
• (Preimage resistance) given h, it is hard to compute an m such that h = H(m),
• (Second preimage resistance) given m, it is hard to compute m' such that $m \neq m'$ and yet H(m) = H(m'),
• (Collision resistance) it is hard to compute a collision for H, i.e. it is hard to compute m and m' such that $m \neq m'$ and yet H(m) = H(m').
Let us assume that we have a hash function which produces hashes with a length of k bits. Brute force methods simply try all possible inputs in a certain range, or try a large number of random inputs, until a (second) preimage or a collision are found. For (second) preimage resistance this requires an expected number of hash computations equal to 2k. For collision resistance the situation is different, due to the "birthday paradox", reducing the number of hash computations to approximately 2k/2. Cryptographers say that a hash function with hash length k provides only k/2 bits security. A cryptographic hash function is said to be "cryptographically strong" if no better methods are known than the above mentioned brute force type methods with the mentioned complexities.[18]
For a hash function for which L is the number of bits in the message digest, finding a message that corresponds to a given message digest can always be done using a brute force search in 2L evaluations. This is called a preimage attack and may or may not be practical depending on L and the particular computing environment. The second criterion, finding two different messages that produce the same message digest, known as a collision, requires on average only 2L/2 evaluations using a birthday attack. For the latter reason the strength of a hash function is usually compared to a symmetric cipher of half the message digest length. Thus SHA-1 was originally thought to have 80-bit strength. In terms of practical security, a major concern about these new attacks is that they might pave the way to more efficient ones. A collision for a hash function h means a pair x, y of different inputs such that h(x) =h(y).

3.2 Design Principles

SHA is based on MD4 algorithm, so the study of design of MDx hash is also covered here. The hash functions of the MDx-class are iterated constructions, This means that hashing is based on the iteration of

a compression function, taking a chaining variable and a message block as inputs and producing a new value for the chaining variable as output. An initial value is defined for the chaining variable, and the message to be hashed is first pre-processed by adding some padding bits and dividing it in blocks of equal length (b bits). A padding scheme is used which appends a single 1-bit to the message, followed by a number d of 0-bits, and finally a number of bits containing a representation of the length of the original message. [12]. Here d is chosen as the smallest number (possibly zero) for which the length of the padded message is a multiple of the block length b. The lengths of the chaining variable and the hash result are equal (c = n bits). Furthermore, the hash result is taken as the final value of the chaining variable, obtained after the last application of the compression function. The compression function is designed to be collision-resistant. From the iterated model and from the fact that the padding bits contain a representation of the length of the message, this would imply that the hash function is also collision-resistant .Hash functions from the MDx-class are word-oriented. This means that all data (chaining variable and message blocks) are divided into words of a specified length (w bits), and the compression function uses only operations on words of this length. MDx-class hash functions have a word length of w = 32 or w = 64 bits. The message to be hashed and the output from the algorithm are usually represented as strings of bytes. Therefore conversions must be made between strings of bytes and sequences of words (or vice-versa). For interoperable implementations on different processors, an unambiguous convention must be specified for these conversions. Consider a string of bytes b_i with increasing memory addresses i, and assume that we have to convert this to a sequence of 32-bit words. [14]
Each substring of four consecutive bytes
(e.g., b_0; b_1; b_2; b_3) is converted to a 32-bit word W as follows.
In a little-endian architecture the byte with the lowest memory address (b_0) is the least significant byte of the word:
$W = 2^{24} * b_3 + 2^{16} * b_2 + 2^8 * b_1 + b_0$.
In a big-endian architecture the byte with the lowest memory address is the most significant byte:
$W = 2^{24} * b_0 + 2^{16} * b_1 + 2^8 * b_2 + b_3$.
The compression function of MDx-class hash functions is of a sequential nature. This means that it consists of a large number of step operations that are executed sequentially (the result of a step must be known in order to proceed to the following step). Furthermore, the elementary step operations have a simple structure. For the parameters c (chaining length) and w (word length) the chaining variable consists of c = w words (of w bits each). The values of one or two of these words are updated in a step operation. Each step depends on one w-bit word of the message block that is being processed. Only simple operations on w-bit words are used. For a block length of b bits, the message blocks that are processed by the compression function consist of b/w words (of w bits each). Assume now that the compression function has a structure consisting of s sequential step operations. As mentioned above every step depends on one message word, so s of these words are needed in total. Here s > b/w for MDx-class hash functions. Therefore a procedure must be specified for expanding the b/w words of the message block input to a block of s words. Some hash functions use a very simple expansion where each of the b/w message words is used multiple times in a number of different steps (s/(b/w) times to be precise), but other hash functions have a more complex procedure for expansion, based on a linear code.[14]

3.3 Proposed new scheme

There are some flaws in the existing hash algorithms which are responsible for the cracking of these.It has been observed that we can have two messages with same hash value obtained by using same algorithm. Hence the property of hash functions being collision free is compromised, decreasing the reliability. So we need to make certain changes in them.In the padding scheme used, a number of zero's is used which can make the algorithm very predictable and vulnerable for collision.The existing hashing pattern comprises of appending a number of Zeroes to the original message, followed by a defined number of bits to show the length of message. This padding is done to allow the equal division of message into blocks,

which can de further put to parse message by using certain shift and Boolean operations on each bit.Now the study is concerned of finding if ,in addition to Zeroes ,a number of Ones can also be made use of in the padding pattern. The integrity of original message will not be effected in this case as a number of bits can still be made use of to show the length of message.

3.4 Algorithm

The last block must have minimum each character of 8 bits. So the minimum number of bits in the last block will always be 8 and the number of bits will be a multiple of 8.
Let there be 8 bits in the last block.
Now we divide them into two parts of 4 bits each.
Calculate decimal equivalent of these two parts, x and y respectively.
Apply a mathematical polynomial function to these x and y to get a bit stream of desired length, in our study we have used $x^2 + 3y^2$, as polynomial.
Append this bit stream to the last block after the end of message.
Repeat the process till we get the desired number of padding bits, equal to 448 mod 512.
Last 64 bits of last block indicate the length of message bits in the block, to know the start of padding bits.
This completes the padding process.

EXAMPLE

Let there be three characters in the last block with bit pattern as
01100001 01100010 01100011
Last block is of 01100011 bit pattern will be divided into two halves 0110 0011
The decimal equivalents are 6 and 3
Thus we have x = 6 and y = 3
Using polynomial $x^2 + 3y^2$, we obtain a result 63.So we have a bit pattern of 00111111, which is appended to the end of last block. This is repeated till we obtain 448 bits of padding. The last 64 bits indicate the length of message.
Also, a comparative study of various polynomials used in the structure has been done.
Now the whole message is divided into blocks of 512 bits each. The 512 bits blocks are divided into 16 words of 32 bits (16*32=512) each. Each 32 bits word is processed with the set of Boolean functions to obtain a result of 256 bits stored in 8 registers of 32 bits (8*32=256) each.
The Boolean functions used operate on variables x, y and z each of 32-bits.

$$f_t(x,y,z) = \begin{cases} (x \text{ OR } y) \text{ XOR } (x \text{ AND } (\text{NOT } z)) & \text{first 20 iterations} \\ (\text{NOT } x) \text{ XOR } (y) \text{ XOR } (\text{NOT } z) & \text{next 20 iterations} \\ (x \text{ OR } y) \text{ XOR } (x \text{ AND } z) \text{ XOR } (y \text{ OR } z) & \text{next 20 iterations} \\ (x) \text{ XOR } (\text{NOT } y) \text{ XOR } (z) & \text{next 20 iterations} \end{cases}$$

Further the padded message is processed by making use of one of the already existing algorithms. [13]
The registers used are
H_0 = 6a09e667
H_1 = bb67ae85
H_2 = 3c6ef372
H_3 = a54ff53a
H_4 = 510e527f
H_5 = 9b05688c

H_6 = 1f83d9ab
H_7 = 5be0cd19

The intermediate outputs of each processed word of 32 bits (16 * 32) is stored in registers a to h and then added up to make entry into registers H_0 to H_7.For processing of each block, we have 80 iterations.
For a 512 bit block made into 16 words of 32 bits each, we have words for first 16 iterations but for next 64 iterations, we take the rotated values of the first 16 words.
Initialize the five working variables, *a*, *b*, *c*, *d*, and *e*, with the $(i\text{-}1)^{th}$ hash value:

a = $H_0^{(i-1)}$
b = $H_1^{(i-1)}$
c = $H_2^{(i-1)}$
d = $H_3^{(i-1)}$
e = $H_4^{(i-1)}$
f = $H_5^{(i-1)}$
g = $H_6^{(i-1)}$
h = $H_7^{(i-1)}$

For *t* = 0 to 79:
{
$T = ROT^5(a) + f_t(b, c, d) + e + W_t$
h = g
g = f
f = e
e = d
d = c
$c = ROT^{30}(b)$
b = a
a = T
}
Where the function used is a combination of already mentioned Boolean operations above.
Compute the i^{th} intermediate hash value *H*(i)

$H_0^{(i)} = a + H_0^{(i-1)}$
$H_1^{(i)} = b + H_1^{(i-1)}$
$H_2^{(i)} = c + H_2^{(i-1)}$
$H_3^{(i)} = d + H_3^{(i-1)}$
$H_4^{(i)} = e + H_4^{(i-1)}$
$H_5^{(i)} = f + H_5^{(i-1)}$
$H_6^{(i)} = g + H_6^{(i-1)}$
$H_7^{(i)} = h + H_7^{(i-1)}$

After repeating steps ,the resulting 256-bit message digest of the message, *M*, is obtained as

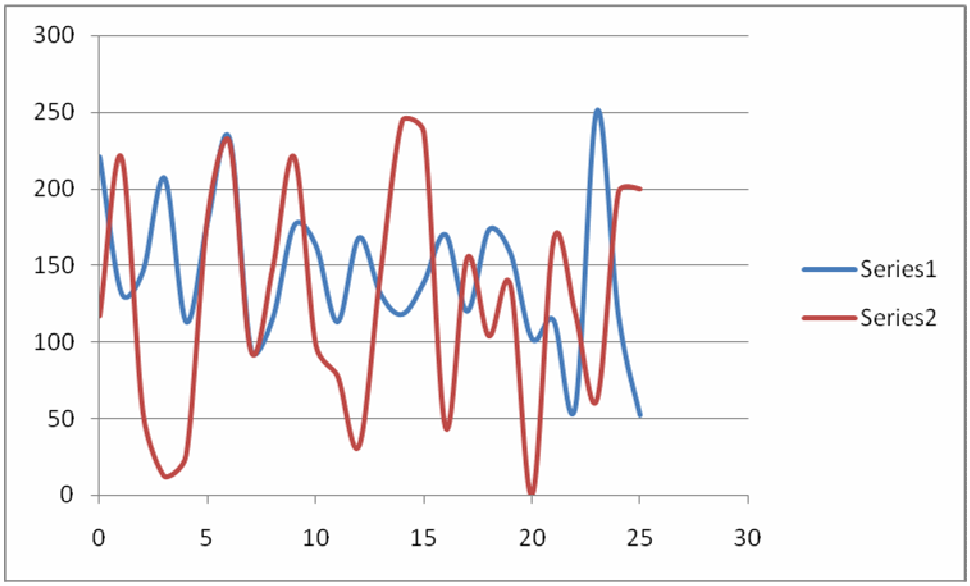
$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$

4. RESULTS

This work investigates the performance of the new proposed scheme and compares it with the original version.Certain significant modifications have been done on the hashed values obtained in order to compute the stability. Correlation Coefficients have been calculated by taking numerous outputs against certain changes in the message fed as input. As a very low correlation exists, so the proposed scheme has been justified to work against cryptanalysis. With enough hash values known, the message is still hard to be computed back.Table .1 lists the various coefficients obtained against different polynomials.

Table 1 Comparative Results

Polynomial	**Correlation Coefficient**
Original Scheme(no polynomial)	0.108581
$x^2 + y^2$	-0.22117
$x^2 + 2y^2$ (series 1)	0.382996
$x^2 + 4y^2$	-0.04547
$x^2 + 5y^2$	-0.04267
$x^2 + 6y^2$	0.166242
$x^2 + 7y^2$	0.179451
$x^2 + 8y^2$ (series 2)	0.0842
$x^2 + 9y^2$	-0.01232
$x^2 + 16y^2$	-0.09926



This graph shows that the input and output hash value exhibit almost no correlation, making the efficiency preserved in the critical applications.

5. CONCLUSION

The modified scheme of using ones along with zeroes has been evaluated to be efficient on the basis of comparison against the original by treating both on same basis. It has to be tested against other possible attacks and a better polynomial can be worked upon as a future scope of this study.

REFERENCES

[1] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, Benne de Weger, MD5 considered harmful today: Creating a rogue CA certificate, accessed March 29, 2009

[2] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu, Finding Collisions in the Full SHA-1

[3] Bruce Schneier, Cryptanalysis of SHA-1 (summarizes Wang et al. results and their Implications)

[4] Shai Halevi, Hugo Krawczyk, Update on Randomized Hashing

[5] Shai Halevi and Hugo Krawczyk, Randomized Hashing and Digital Signatures

[6] Licensing Declaration for US patent 6829355.. https://datatracker.ietf.org/ipr/858/. Retrieved 2008-02-17.

[7] Microsoft Corporation, Overview of Windows XP Service Pack 3

[8] Somitra Kumar Sanadhya and Palash Sarkar, New Collision attacks Against Up To 24-step SHA-2

[9] National Institute on Standards and Technology Computer Security Resource Center, NIST policy on Hash functions, accessed March 29, 2009.

[10] Somitra Kumar Sanadhya and Palash Sarkar, New Collision attacks Against Up To 24-step SHA-2

[11] Moni Naor and Moti Yung,"Universal One way Hash functions and their Cryptographic Applications",Proceedings of the Twenty First Annual ACM Symposium on TOC,Seattle,Washington May 15-17,1989,page 33-43

[12] NIST/NSA, "FIPS 180-2: Secure Hash Standard (SHS)", Aug. 2002 (change notice:February 2004).

[13] B. Preneel, "Analysis and design of cryptographic hash functions", PhD thesis, Katholieke University Leuven, 1993.

[14] R.L. Rivest, "The MD4 Message Digest Algorithm", Crypto'90,LNCS 537, Springer-Heidelberg, pp. 303–311, 1991.

[15] B. Van Rompay, "Analysis and design of cryptographic hash functions, MAC algorithms and block ciphers", PhD thesis, K. U. Leuven, Januvary 2004.

[16] J.-S. Coron, Y. Dodis, C. Malinaud, and P. Puniya,"Merkle-Damgård revisisted: How to construct a hash Function",in V. Shoup, editor, Advances in Cryptology CRYPTO 2005,

[17] S.G. Akl, "On the security of compressed encodings," Advances in Cryptology,Proc. Crypto'83, D. Chaum, Ed., Plenum Press, New York, 1984, pp. 209–230.

[18] I.B. Damg°ard, "Collision free hash functions and public key signature schemes,"Advances in Cryptology, Proc. Eurocrypt'87, LNCS 304, D. Chaum and W.L. Price, Eds., Springer-Verlag, 1988, pp. 203–216.

[19] D. Davies and W. L. Price, "The application of digital signatures based on public key cryptosystems," NPL Report DNACS 39/80, December 1980.

[20] D. Davies, "Applying the RSA digital signature to electronic mail," IEEE Computer,Vol. 16, February 1983, pp. 55–62.

[21] J. Pieprzyk and B. Sadeghian, "Design of Hashing Algorithms", LNCS, Springer-Heidelberg, ISBN-10: 0387575006, 1993.

[22] B. Preneel, "Analysis and design of cryptographic hash functions", PhD thesis, Katholieke University Leuven, 1993.

Biographical notes:

Meenakshi Kaul holds a B.E in Electronics and Telecommunication from IET, Devi Ahilya University,Indore-INDIA. She is currently Pursuing M.E in Computer Science from RGPV University Bhopal-INDIA. Her research interests include Network Security and Mobile Computing.

Dharmendra Choukse holds a M.Tech in Information Technology from Devi Ahilya University, Indore-INDIA. He is currently Pursuing Ph.D. in Computer Science From Institute of Computer Science, Vikram University, Ujjain-INDIA.and He is also currently Sr Software Engineer in Institute of Engineering & Sciences,IPS Academy, Indore-INDIA. A He served as Software Engineer in Choksi Laboratories ltd,Indore. His research interest includes network security, secure electronic commerce, client-server computing and IT based education.

Dr. Umesh Kumar Singh obtained his Ph.D. in Computer Science from Devi Ahilya University, Indore-INDIA. He is currently Reader in Institute of Computer Science, Vikram University, Ujjain-INDIA. He served as professor in Computer Science and Principal in Mahakal Institute of Computer Sciences (MICS-MIT), Ujjain. He is formally Director I/c of Institute of Computer Science, Vikram University Ujjain. He has served as Engineer (E&T) in education and training division of CMC Ltd., New Delhi in initial years of his career. He has authored a book on " Internet and Web technology " and his various research papers are published in national and international journals of repute. Dr. Singh is reviewer of International Journal of Network Security (IJNS), ECKM Conferences and various Journals of Computer Science. His research interest includes network security, secure electronic commerce, client-server computing and IT based education.