Secure Hash Algorithm for Public Key Digital Signature Schemes

P.Rizwan Ahmed
Head of the Department
Department of Computer Applications and
PG Department of Information Technology
Mazharul Uloom College, Ambur.
rizwanphd@gmail.com

C. Ashok Kumar
Mazharul Uloom College, Ambur.
ashok_mca2000@yahoo.com

***Abstract*- Hash functions are the most widespread among all cryptographic primitives, and are currently used in multiple cryptographic schemes and in security protocols. This paper presents a new Secure Hash Algorithm called (SHA-192). It uses a famous secure hash algorithm given by the National Institute of Standard and Technology (NIST).The basic design of SHA-192 is to have the output length of 192.The SHA-192 has been designed to satisfy the different level of enhanced security and to resist the advanced SHA attacks. The security analysis of the SHA-192 is compared to the old one given by NIST and gives more security and excellent results as shown in our discussion. In this paper the digital signature algorithm which is given by NIST has been modified using the proposed algorithms SHA-192. Using proposed SHA-192 hash algorithm a new digital signature schemes is also proposed. The SHA-192 can be used in many applications such s public key cryptosystem, digital signcryption, message authentication code, random generator and in security architecture of upcoming wireless devices like software defined radio etc.**

***Keywords* - Data integrity, hash algorithm, digital signature, pre-image, message digest, message authentication**

I. INTRODUCTION

Cryptographic hash function plays an important role in the world of cryptography. They are employed in many applications for digital signatures, message authentication data integrity and key derivation. Secure Hash Algorithm (SHA-1) specifies which generates condensed of message called message digest. Hash functions takes a message of variable length as input and produce a fixed length string as output referred to as hash code or simply hash of the input message. The basic idea of cryptographic hash function is use of hash code as compact and non ambiguous image of message from which latter cannot be deduced. The term non ambiguous refers to the fact that the hash code can be as it was uniquely identifiable with the source message. For this reason it is also called as digital finger print of the message.

The hash functions [1, 2, 3] are classified into keyed and unkeyed hash function; the keyed hash functions are used in the Message Authentication Code (MAC) whose specification are dictates two distinct inputs a message and a secret key. The unkeyed hash function have there categories hash function based on block ciphers, modular arithmetic and customized hash function. The hash functions have one-way property; given n and an input M, computing (M)=n, must be easy and given n. it is hard to compute M such that H(M)=n. The type of attacks [1] are the collision attack (find two message M=M' with H(M)=H(M'), the preimage attacks (given a random value Y, find a message M with H(M)=y) and the second preimage attack (given a message M, find a message M-M' with H(M)=H(M')). The SHA-1 is required for use with the digital signature algorithm as specified in Digital Signature Standard (DSS) and whenever a secure hash algorithm is required. Both the transmitter and intended receiver of a message in computing and verifying a digital signature uses the SHA-1. It is necessary to ensure the security of digital signature algorithm, when a message of any length is input, the SHA produces m bits output called Message Digest (MD) [4, 5, 6]. The MD is then used in the digital signature algorithm. Signing the MD using the private key rather than the message often improved efficiency of the process because the MD is usually much smaller than the message. The same MD should be obtained by the verifier using the user public key when the received version of the message is used as input to SHA.

In the recent years much progress has been made in the design of practical one-way hashing algorithms which is efficient for implementation by both hardware and software. Noteworthy work includes the MD family which consist of three algorithms MD2, MD4, MD5 [2, 4], the federal information processing standards for secure hash proposed by NIST [5]for the past few years NIST designed the SHA family which produce 160, 256, 384 and 512bit [6, 7, 8, 9]. SHA-1 which produces message digest of 160bits long was the best established of existing SHA hash functions and employed in several widely used security application and protocols. It has been identified that security flaws in SHA-1 in 2004 [4, 10], namely that a possible mathematical weakness might exist indicating that stronger hash function would be desirable. The aim of this research is to design a secure one-way hashing algorithm of 192bit to enhance the security and resist to advanced attacks such as preimage, second preimage and collision attacks. Certain modifications are introduced in the existing SHA-1 algorithm to improve the strength of security.

The maximum security depends on the length of message digest generated by the hash functions which is limited by the size of input to the algorithm. It also shows how the modification is done with satisfying the properties like compression, preimage resistance, and collision resistance. The simulation results show that proposed scheme provides better security than the existing one. The simulated results of proposed SHA-192 are analysed and used to generate modified DSA.

II. GENERAL MODEL OF PROPOSED SHA-192

The proposed SHA-192 algorithm is similar to SHA-1 algorithm. The SHA-192 algorithm is similar in structure except that it has one an extra 32-bit word, say F. The elementary functional block processing of message is shown in Figure 1. It is similar to SHA-1 message digest function, but it is little slower to execute and presumably more secure. It produces a 192 bit message digest as opposed to the 160 of the SHA-1.

The proposed SHA-192 algorithm has three processing steps: pre-processing, iterated processing and output transformation. The pre-processing step involves padding, parsing the padded message into m bit block and setting initial values to be used in iterated processing. The iterating process has eighty steps in all and in each step there is a elementary function which calculates a message digest every time and sends it to the next step. And moreover there is no secure hash algorithm that gives a message digest size of greater than 160 bits and less than 256 bits, so we have proposed a new hash algorithm that undergoes a significant change in the elementary function of the secure hash algorithm and also gives us a message digest of length 192 bits.

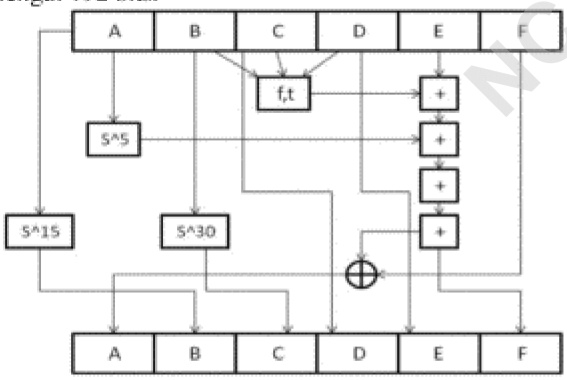

Figure 1. Proposed SHA-192 elementary function

The modified SHA uses the padding algorithm, breaking the message into 512 blocks and adding the length as a 64 bit number at end. The output transformation is used in a final step to map the n bit to variable length s m bits results called the MD. The following operations are used in the processing and all of these acts on 32 bit word.

Λ-;AND ¬¬ ; NOT ; V- OR ; +XOR , +: mod2 ^32 addition

R –Right shit by n bits

The word size and the number of rounds are same as that of SHA-1. The key characteristics of SHA -192 algorithm is shown in Table 1.

Table 1. Key characteristics of SHA -192 algorithms

Name	Block Size (Bits)	Word Size (Bits)	Output Size (Bits)	Round
SHA-0	512	32	160	80
SHA-1	512	32	160	80
SHA-192 (PROPOSED)	512	32	192	80

In order to increase the security aspects of the algorithm the number message digest should be increased .To achieve this first, number of chaining variables used initially is increased by one. Due to this number of bits generated by message digest is considerably increased, which makes SHA-1 more complex in breaking than before. The modified structure of SHA-1 algorithm is given in Figure 1.

III. SHA-192 PRE PROCESSING

Pre-processing is the step used to prepare the message before the SHA-192 processing step. This contains the three steps: padding, parsing the padded message into blocks and setting initial hash values.

- *Padding the Message:* The purpose of padding is to ensure that the padded message is multiple of 512. If the length o the message M, is l bits it is append the bit 1 to the end of the message followed by k zero bits, where k is smallest, non negative solution to the equation l+1+k≡448 mod 512. To this append the 64bit block that is equal to the number l written in binary.
- *Parsing the Padded Message:* Parse the message M into N 512 bits of blocks M1, M2……, MN. Each of the Mi parsed into 16, 32bit words Mi0, Mi1,……Mi15. The message blocks are processed one at a time, beginning with the initials hash values called message digest buffer.

Setting Initial Hash Values: Before the hash function begins, the initial hash value H0 must be set. The hash is 192bits used to hold the intermediate and final results. The hash can be represented as six 32bit words registers A,B,C,D,E,F: A=67452301, B=EFCDAB8.
C=98BADCF, D=10325476, E=C3D2E1F0, F=40385172

IV. SHA-192 PROCESSING

The processing step depends upon expanded message block and compression function. In order to increase the security level of the algorithm the size message digest produced should be increased .To achieve this first, number of chaining variables used initially is increased by 32bits. Due to increase in input value the number of bits generated as message digest is also considerably increased. Secondly the changes have been introduced in round function. In this the number of XOR operations performed is increased in order to make it more complex and in turn make it more secure. The number of times the round function being called is increased. And the shifting of some of the chaining variable by 15bits and 30bits in each round. By moving the last block bits to the front followed by other consecutive bits will increase the randomness in bit change in the next successive routines.

The SHA-192 hash computation uses functions and constants previously defined EXOR operation is performed after pre-processing is completed each message block is processed in order using the following steps:

1. *For i=1 to N { prepare the message schedule{};*
   Wt = Mti 0<t<15
   ROTL {Wt-3+Wt-8+Wt-14+Wt-16}, 16<t<79
2. *Initialize the six working variables A, B, C, D, E, F with (i-1)st hash value*
3. *For t=0 to 79*
   {T =ROTL5 (A) + F (B, C, D) +E + Kt + Wt
   L =ROTL5 (A) + F (B, C, D) +E +F + Kt+ Wt
   E= D
   D= C
   C= ROTL30 (B)
   B= ROTL 15 (A)
   F=T
   A=L
   }
4. *Compute the intermediate hash value H(i):*

H0(i) = A + H0(i-1) (1)
H1(i) = B + H1(i-1) (2)
H2(i) = C + H2(i-1) (3)
H3(i) = D + H3(i-1) (4)
H4(i) = E + H4(i-1) (5)
H5(i) = F + H5(i-1) (6)

The output transformation step is modular summation used to map the final output of the single compression function of n bits to the output length.

V. EXPERIMENTAL RESULTS AND ANALYSIS

The hashing algorithms SHA-1 and a newly proposed SHA-192 were tested based on the security and time needed to generate message digests for the text data and images. The algorithm has been tested using a system with a pentium 4 processor and 512MB of RAM. Based on the simulation results, it was found that proposed SHA-192 needs more time to generate a message digest when compared with SHA-1 because the message digest generated by the pprosed algorithm longer than the SHA-1.

For a simple message M="abc," the 8bit ASCII message "abc" has length e=24bits and it is given by "01100001 01100010 01 100011". It is padded with a one "1," so 24+1+k=448 mod 512 and then k=423 zero bits. Append the 64bit block that is equal to the number 1 written in binary "0000….011000" and then its length are 512bits added message which is shown in Table 2.

Table 2. SHA-192 pre processing.

a	b	c	Padded Bits	423 Zeros	64-Bits
01100001	01100010	01100011	1	00000….0	0…..1

The "abc" message has a single (N=1) 512bit block, parse the 512 bits into 16,32 bit words. $M^{(1)}M^{(2)}$,.....$M^{(15)}$, (in hexadecimal): 61626380 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000018.

These 16, 32 bit words are expanded to L 32bit words W, W_0, W_1, W_2 ....W_t. Followed by hash computation process. The MATLAB simulated test vectors for the SHA-192 are given.

Table 3 shows the hash value of test vector for different cases with small and the same string compared with the existing hash algorithm.

Table 3. Generation of hash value for test vector.

Hash	String	Hash Value(as Hex Byte String)
SHA-1	abc	a9993e36 4706816a ba3e2571 7850c26c 9cd0d89d.
SHA-192	abc	ba7816bf 8f01cfea 414140de 5dae2223 b00361a3 96177a9c

The sample text data say, "the quick brown fox jumps over lazy dog" is given as input first time and second the same data with slight modification i.e., by making small change in the word "dog" as "mog" is given as input next time. For this it generates the new message digest which is entirely differ from the previous one. It shows that for the same data it will produce different hash value. This is shown in Table 4.

The proposed algorithm is tested with sample digital image data. The algorithm is tested with given digital images to generate the hash value. Some sample digital images like bridge images Figure 2 and camera man image Figure 3 are taken and hash values are generated for these images. Similar as text data, images also generate different hash value every time. This shows that the proposed algorithm is highly secured The proposed algorithm can also be implemented in hardware [13] so that it can be used in transmission of medical image information system for providing authentication.

Table 4. Simulation results for text data.

Sample Data	Message Digest	Computation Time Inseconds
The Quick Brown Fox Jumps over Lazy Dog	2684DDEB 5518D609 EF7218CF 2344452E 2FD5E684 3F5B2678	4.4680
The Quick Brown Fox Jumps Over Lazy Mog	51D19353 1F2BEFCF 47FB6856 1BABEC86 9C760B1C 1B 3F 572C	4.4540

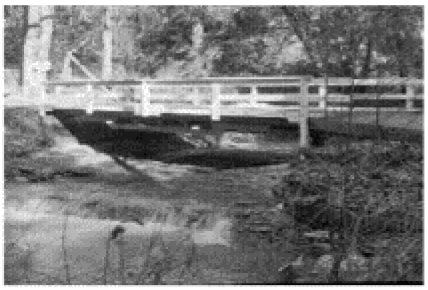

Figure 2. Bridge image.

Tables 5 and 6 Shows the hash computation time and message digest generated for different images taken.

Table 5. Hash computation SHA-1 and proposed SHA-192 algorithm for bridge image.

Hash Algorithm	Hashing Time in Seconds	Message Digest
SHA-1	25.063	7ABADFF3 7B769FF8 5F957149 37065866 63985BD7
SHA-192	27.703	9CCEB8C1 A43D689F D343C37B CF581229 5178B5B9 E7BA381A

Figure 3. Cameraman Image.

Table 6. Hash computation SHA-1 and proposed SHA-192 algorithm for cameraman image.

Hash Algorithm	Hashing Time in Seconds	Message Digest
SHA-1	25.563	7DE2E230 DA78B34E 432D15A3 D053F312 5AD540CD
SHA-192	29.570	5CBF9937 7CC7F5E4 A66F4DDE 30A04E2F 38AFD83C 7B2A855

VI. Security Analysis of Proposed Algorithm

Good hash functions have a strong ability to withstand all kinds of cryptanalysis and attacks that try to break the system such as brute-force analysis, statistical analysis and collision attacks. Cryptanalysis of secure hash algorithm focuses on the internal structure of the iterated compression function and is based on attempts to find efficient techniques for producing collisions for a single execution of the compression function. The best way to find a collision pair is by using the birthday attack. To measure the security strength of the SHA with 192 bit results, finding preimage or second old preimage attacks requires about bit operations and for collisions attack. The preimage and second image of proposed algorithm occurs at different rate which is greater than both SHA-1. Standard attacks on various hash functions are tabulated in Table 7.

Table 7. Attacks on standard hash function.

Hash Algorithm	Author	Type	Complexity	Year
SHA-0	Chabaud & Joux	Collision	2^61(theory)	1998
	Biham & Chen	Near-Collision	2^40	2004
	Biham et al.	Collision	2^51	2005
	Wang et al.	Collision	2^39	2005
SHA-1	Biham et al.	Collision	2^75	2005
	Wang et al.	Collision	2^63	2005
SHA-192 (Proposed)	None	None	None	None

VII. MODIFIED DIGITAL SIGNATURE STANDARD USING SHA192 ALGORITHM

The NIST has proposed the DSSA [11, 12] as the public standard for digital signature. The DSSA words. A Trusted Centre (TC) is assumed in DSSA and the role of TC are to select and use secure hash algorithm with the fixed output length, fixed structure and fixed additive constant publish system parameters for public usage.

Digital signatures are used to detect unauthorized modifications to data and to authenticate the identity of the signatory. In addition, the recipient of signed data can use a digital signature in proving to a third party that the signature was in fact generated by the signatory. This is known as non-repudiation since the signatory cannot, at a later time, repudiates the signature. The Digital Signature Algorithm (DSA) is described in following sections.

A. DSA Signature Generation

The signature of a message M is the pair of numbers r and s computed according to the equations below:

$$r = gk \bmod p) \bmod q \text{ and} \quad (8)$$

$$s = (k\text{-}I\ (SHA\text{-}192(M)+xr)) \bmod q \quad (9)$$

In the above, k-I is the multiplicative inverse of k, mod q, i.e., (k-I k) mod q=1 and 0<k-I< q. The value of SHA-192(M) is a 160-bit string output by the Secure Hash Algorithm specified. For use in computing s, this string must be converted to an integer. As an option, one may wish to check if r=0 or s=0. If either r=0 or s=0, a new value of k should be generated and the signature should be recalculated (it is extremely unlikely that r=0 or s=0 if signatures are generated properly). The signature is transmitted along with the message to the verifier.

B. DSA Signature Verification

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner. Let M', r', and s' be the received versions of M, r, and s, respectively, and let y be the public key of the signatory.

To verify the signature, the verifier first checks to see that 0<r'<q and 0<s'<q; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes:

w= (s') -1 mod q (10)
u1 = ((SHA-192(M')) w) mod q (11)
u2 = ((r') w) mod q (12)
v = (((g)u1 (y)u2) mod p) mod q (13)

If *v=r'*, then the signature is verified and the verifier can have high confidence that the received message was sent by the party holding the secret key *x* corresponding to y. For a proof that *v=r'* when *M'=M*, *r'=r*, and *s'=s*, If v does not equal *r'*, then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should *be considered invalid.*

C. Results and Discussions

The modified DSA algorithm using proposed secure hash algorithm is tested with sample data and the results are given below.

Hash message:
Input message string to hash:
> abc
The hashed message is A9993E36 4706816A BA3E2571 7850C26C
9CD0D89D, Generating primes p and q, p and q have been obtained:
p=EE72B63E166458A57AC6071F806B36C77258F864211E C55C20EF44
3380B0EE0032BE9D97F012549981DB79216EB60E764725 47A66F5F57
5AA9E63A9ACE67B77E6980985EAF81008F7C8C1C35BD A0E914801B4
E86D3145EA4CC0B12FA920C24C7A3F3F77366E2B3FA6 F677DF46D5
C5D266112F291014ED6E5781A9ACC6D7D58E7
q= 808DBCD4B8B85BFF6D7AE85EDE83B28641BEF4E9
For certification purposes, the seed and counter values are:
seed=1001111111100101110011001001110011100101111110100011011
1000100011000110000100000100100011001001111001101101100110101
1100110010101000100000111111110010000011101 00
counter = 504
DSA parameters: public key y, and the quadruplet [p, q, g, y] becomes the
users public identity. Public group g is:
D9E9E76297D15BBB4DC7F848F72DC0466C835B23513F7 DDBAE747
CFA607004D6BCF01C6566EC960314FD0E1EC92D04751A CCF7355C5
26B860A40761FD8F4A5DC8374BBFE15F8EC844486EEA AFBF327EB2
EC5C9CCBE03C4E187E50B2AF8CC740815CCBA8615CD DF78383347
F07C8A14D523A1A01FA7E9FA7D9C8B5E9AE4135328
Generating x, k and y, Private key x is
C9475C49F8236B839620B20FE32E1BF1AAF41403
Private session key k is:
6D5B301B3B5787E0D8AD5C3F45736154A27F7A47
Public key y is:
7254551CB0051FAAA618D3A2EA2D543A90D64F3FDD7F 90D7D8EE5
4892F582DF55CFBB442ED213281BC86107865E1FAD1BD D4C28B960
93DF394A0B7625F69FF43AD992FA9504100D0DF06F98D 3308E8B2D
C16821ED7F0AAE97CE7158E892E253A662B519CB3B8D8 B5B6FB467
DFA2D7FC50C531F67D5A8C8D91AEC7FE42DEFA9DE
Signing the message:
The next step is to sign the hashed message, and this produces the
signature pair [r,s]. The signature pair [r, s] in integer form is:
r=`766E76D00D99802E2B647D9E3BAF7FF82D1C7892`
s=`2F2E3DB99D0AA8DD4BF027F81A0EC5104650CA3D`
Verifying the signature:
Message [M,r,s] is sent to the verifier in an authenticated manner. The
verifier beginning the verification process: it generates value of v using
digital signature verification algorithm and generated the value of v. The
value of v is obtained as:
v=`766E76D00D99802E2B647D9E3BAF7FF82D1C7892`
The value of v and r are same. r=v: The signature has been verified!
>>

The proposed algorithm works efficiently in generating digital signature for any kind of message.

VIII. CONCLUSIONS

This paper presents a new secure hash algorithm based on static structure algorithm called Secure Hash Algorithm (SHA-192). It uses a famous secure hash algorithm (SHA-1) given by the National Institute of Standards and Technology (NIST). All NIST, SHA have a fixed structure, fixed constant and fixed output length. With this structure it can provide many choices for practical applications with different level of e-handed security to resist the advanced SHA attacks such as pre-image, second preimage and collision attacks. The security analysis of SHA-192 is compared to old SHA-1 given by the NIST and it gives enhanced security and excellent results. The proposed SHA-192 hashing algorithm has been observed to be

better than the already existing SHA-1 hashing algorithm in terms of the number of brute force attacks needed to break it and moreover it is fast when compared to the other secure hash algorithms. The proposed method has advantage of portability on mobile devices, which are currently embed Security enhancement of proposed system is more than the existing one but there the time delay is more since it generate 192bits of message digest. By using proposed SHA-192 a novel digital signature algorithm has also been proposed.

REFERENCES

[1] Alfred M., Oorschot P., and Vanstone S.,

[2] Handbook of Applied Cryptography, CRC press, 1997.

[3] Bruce S., Applied Cryptography: Protocols, Algorithms and Source Code in C, John Wiley and Sons, Canada, 1996.

[4] Stallings W., Cryptography and Network Security Principles and Practices, Prentice Hall Press Upper Saddle River, 2010.

[5] Goldwasser S., Micali S., and Rivest R., "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks," Journal on Computing, vol. 17, no. 2, pp. 281-308, 1988.

[6] Ilya M., "Hash Functions: Theory, Attacks, and Applications," in Proceedings of Microsoft Research, Silicon Valley Campus, pp. 1-22, 2005.

[7] National Institute of Science and Technology, "Secure Hash Standard," Federal Information Processing Standard 180-1, available at: http:/ http://www.itl.nist.gov/fipspubs/fip180-1.htm, last visited 1995.

[8] National Institute of Science and Technology, "Secure Hash Standard," Federal Information Processing Standard 180-2, available at: http://www.itl.nist.gov/fipspubs/fip180-1.htm, last visited 2002.

[9] National Institute of Standards and Technology, "Secure Hash Standard," "FIPSPUB180www.itLnist.gov/fipspub/fips180-1.html, last visited 2003.

[10] National Institute of Science and Technology, "Implementing Cryptography," NIST SP 800, available at: http://csrc.nist.gov/publications/ nistpubs/800-21-1/sp800-21-1_Dec2005.pdf, last visited 2005.

[11] "New European Schemes for Signatures, Integrity and Encryption Project," available at: http://www.cryptonessie.org, last visited 2000.

[12] Phan R. and Wagner D., "Security Consideration for Incremental Hash Function Based on Pair Blocking Chaining," in Proceedings of

[13] Computers and Security, USA, pp. 131-136, 2006.

[14] Rivest R., Shamir A., and Adleman L., "A Method for Obtaining Digital Signature and Public-Key Cryptosystems," Communication of The ACM, vol. 21, no. 2, pp. 120-126, 1978.

[15] Sklavos N., Alexopoulos E., and Koufopavlou O., "Networking Data Integrity: High Speed Architectures and Hardware Implementations,"

[16] The International Arab Journal of Information Technology, vol. 1, no. 0, pp. 54-59, 2003.

[17] United States Department of Commerce, National Bureau of Standards, Data Encryption Standard, Federal Information Processing Standards Publication, 1977.

[18] William S., Cryptography and Network Security, Principles and Practice, Prentice Hall of India, 2005.