Int. j. inf. tecnol. (October 2022) 14(6):3099–3108
https://doi.org/10.1007/s41870-022-00940-x

ORIGINAL RESEARCH

Light weight hash function using secured key distribution technique for MANET

Srividya Ramisetty[1] · **K. P. Vyshali Rao**[1]

Received: 4 January 2022 / Accepted: 8 April 2022 / Published online: 19 July 2022

Abstract Mobile adhoc network has stringent requirements of light weight computations with aid of keeping track with the processing power and speed, without compromising data security counterpart. This paper talks about the employment of SHA algorithms with less computational complexity retaining the distinguished security standards. XOR and padding logic are used in the design that is put forth. It also contributes to lesser consumption of logic resources, by maintaining the defined speed. Security of data is ensured with various versions of SHA algorithms. Results are acquired for each of the hash functions. Frequency analysis along with swapping technique is done in order to measure the decode ability for input message or key reconstruction.

Keywords MANET · Cryptanalysis · Cryptography · PKI · SHA · HMAC

1 Introduction

The resource constrained decentralized wireless networks demands for highly secured data communication and the requirement poses several challenges in designing efficient systems. In addition to incorporation of basic data security, the design is also intended to be receptive to possible vulnerable attacks [1]. In terms of limited usage of resource MANET design is complicated in ensuring data security [2]. Hash function is used in MANET routing protocol design, in order to prevent attacks like black hole attack. This significantly shows the applicability of Hash function in adhoc networks. Multiple secure hash algorithm (SHA) are used such as 1 and 2 in adopting the security technique [3]. Use of Hash function sets a trade off in its hash code length over performance of the routing protocol. The hash code bit length decides the performance factor of the MANET, generally higher the bit length greater the performance and is effectively demonstrated using Zone routing protocol. The effective storage requirement would set the limitation in mobile networks with the increased bit length.

The use of hash code with various bit code length and the respective protocols matter, so as to keep track with the performance [4, 5]. The Hash function implementation over hybrid network protocol is remarkable in terms of greater speed in software environment, over any other stream cipher encryption model unlike DES.

Data integrity and authentication plays a prime role in the field of wireless Adhoc network. It explores the technical implications in terms of point to point authentication and also end to end authentication. At each node point to point authentication ensures more security over any attacks, while end to end authentication does not consider possible attacks at close proximity.

2 Literature Survey

Singh and Garg provided an insight into how to select a particular hash function based on specific problem [6]. Their paper listed the applications for which the Hash tables are not suitable. They recommended a design of efficient collision resolution strategy for Hash

✉ Srividya Ramisetty
srividya.ramisetty@gmail.com

K. P. Vyshali Rao
raovyshali@gmail.com

[1] CMR Institute of Technology, Bengaluru, Karnataka, India

table creation. Various Hash functions with respect to suitable applications are also discussed. The MANET application is in requirement of information authentication and verification hence cryptographic Hash function suits the targeted environment.

Latest cryptanalysis hash function predominantly used in real time environments are discussed by Aru and Suresh. The main frame work considered in designing the hash is SHA [7]. This paper not only focuses on security issue over SHA-256 but also concentrates on the speed issue. FORK 256 is proposed and implemented. RIPEMD-family compression functions which involve complex operations are also discussed here. The new technique FORK 256 works on 128 bit output and requires 2^{128} operations to achieve the symmetric cryptography. It inherently increases the computation time and assures the SHA-256 level of performance. Although the implementation is efficient over SHA-256 in terms of addition, bitwise and shifting operation, the design complexity is exceptionally high. The challenging task in mobile adhoc network is that, it demands low battery consumption applications to be designed and incorporated.

Mukundan and team have made great research in designing the light weight Hash function [8] which would be applicable to battery critical applications. Their work explains the tradeoff set between security, costs and performance, in designing the hash functions. Also the paper thoroughly discusses several reported lightweight hash algorithms and their applicability.

Keccak [9] is one of the well-known one-way encryption codes, which is used to secure the data in network communication applications. This paper incorporates padding, permutation and parallelism techniques in order to optimize the design in terms of FPGA resource utilization [9]. This paper also states that the Keccak is SHA-3 standard and is authenticated by NIST. This is capable of encrypting 512 bits on FPGA. The proposed method uses Padding and permutation logic and works on 512 bits with reduced area and high throughput and is demonstrated.

Gene Tsudikt discussed the message authentication using one way encryption [10]. The paper suggests secret suffix and prefix algorithms based on MD4. There by design the unique protocol in securing data in addition to checking message integrity. Also this is not vulnerable to message substitution attack. However this technique is too old in the context of MANET.

Chankasame et al. proposed Chaos based key Hash generation for MANET environment. The paper says, traditional message authentication codes and routing protocols do not address the security issue completely [11]. But it is effectively addressed using the Chaos based hash key concept. The proposed concept realizes the nonlinearity for generating high iterated values of ASCII input messages. It offers robustness and randomness over wide genre of parameter spaces. With greater degree of collision this performs well over MD5 and SHA-1.

Dilli and Reddy [2] discussed the tradeoff between Hash code performance and its length. They related this tradeoff in context with routing protocol used in MANET applications.

Alosaimy et al. proposed an approach for message authentication, the NMACA (message authentication code), in network communication [12]. NMACA is derives concepts from SHA-1 and produces 160 bit output for 160 bit key. According to the literature, the design is conservative and requires less size substitution table. The design has similarity with SHA-1 in rearranging the message by initial chaining value and padding. But they differ in the order of message words and order of the values used in circular left shifts. The draw back associated with this approach is the preprocessing time which increases computation time of the algorithm.

Wang et al. [13] discussed the verification of SHA-3 algorithms such as Keccack and RHash and produced bugs in terms of protocol consistency that are associated with software verification.

Liu et al. [14] proposed a hash algorithm using hyperchaotic Lorenz system. This system served as a sponge function to engross input message via multiple parameters time-varying perturbation. The algorithm designed can generate 256, 512, 1024 or longer hash value through parameter switcher. Experimental evaluation and collision analysis demonstrated the function's resistance to differential attack and second pre-image attack. The proposed hash function can be applied in identification, information integrity and figure signature. But hash functions based on chaos has low performance. 4D hyper chaotic system used here on intel Core-i7 16 GB RAM has 0.697Gbps speed which could be reduced further when compared to other such existing systems.

Patel et al. proposed a deep learning-based cryptocurrency price prediction scheme for financial institutions. Records here are secured using efficient cryptographic algorithms such as secure Hash algorithm 2 (SHA-2) and Message Digest 5 [15]. Blockchain technology was made us of to make the transactions, immutable, transparent, secure and traceable. In this paper, a long short-term memory and gated recurrent unit-based hybrid cryptocurrency prediction scheme is proposed, which focuses on only two cryptocurrencies, namely Litecoin and Monero. From the errors of prediction, it was evident that proposed scheme has proved to be better than the LSTM network. An open issue and research challenge here was, as technology is continuously evolving and with it the computing power is increasing day by day with a decrease in time consumed. These advances affect the prices of different currencies

differently and add to the already volatile nature of prices. So,a method is required which can provide utmost data security with lesser computational power involved.

Rupa et al. proposed a blockchain technology based solution to improve the security and privacy of virtual circuit (VC) based device data [16]. Technical information was stored in cloud wherein Pentatope based elliptic curve cryptography and SHA were used to ensure data privacy. This methodology proposed to help protect data from stalkers, plaintext as well as ciphertext attacks, but, it was limited only to UAV. It would be ideal to expand the scope of the research presented here to other applicable areas of IoT and MANET's outside of UAV.

Khan et al. discussed a novel chaos and compressive sensing-based image encryption algorithm [17]. All the experimental and simulation results reveal that the key space is large enough and the key is highly sensitive. Therefore, the scheme requires large space which is not suitable for nodes in MANET. Since nodes in MANET have limited resources.

Choi et al. presented a SHA-3 implementation using Parallel Thread eXecution (PTX) inline assembly, which could make full use of the coarse-grained registers and arithmetic instructions for the optimum performance in graphic processing units environment [18]. This proposed SHA-3 could be applied to authentication processors and hash function-based cryptographic algorithms. But this method considered only CUDA stream for validation of the same.

Alzahrani et al. proposed a methodology for implementing algorithms on dataflow platforms. The proposed methodology was applied to obtain a novel dataflow multicore computing model for the secure hash algorithm-3 [19]. Only MATLAB models were developed to verify and validate the correctness of the computing model that this may not to greater extent be helpful in securing data in MANETs.

Yang et al. proposed compact hardware implementation of a SHA-3 core for wireless body sensor networks. Due to the limited resources of sensors in WBSNs, a lightweight implementation of SHA-3 is needed. The implementation of SHA-3 here consists of a reliable logic structure, random access memory, and an enhanced finite state machine [20]. The sensor area of the proposed SHA-3 implementation was evaluated and compared with other recently proposed hardware implementations of SHA-3. The compact hardware FPGA implementation of SHA-3 is more suitable for deployment in the sensors of the WBSN and may not work efficiently in MANET environment.

3 Existing systems

In literature its stated that SHA-256 HMAC algorithm was used in correlation with the Hybrid protocol for MANETs [4]. The sender uses signing algorithm and its private key to sign the message, there by hashing input bits, and the message with signature is sent to receiver. Verification algorithm at the receiver end is used to authenticate the message which requires verification key. Verification algorithm either accepts the message or rejects the message based on shared public key. Hash function is used for digital signature before sending the message at the receiver. However the approach fails for diverse network applications because of inconsistency in handling proactive and reactive systems [4].

Many researchers analyzed MANET about its ability of vulnerability for attacks. There are many attacks reported in the literature [21]. One of the types of attacks not easily diagnosable in mobile network environment is Worm hole attack. Prevention of this genre of attack requires granular network security design and the paper uses Hash base compression function with respect to RREQ packet. RREQ packet is used in order to mitigate the false intermediate node attacks to the destination node. The paper does not focus on any specific Hash function. It implements SHA-1 as default, only with RREQ which has advantages in terms of data security. But it has limitation in terms of bit length over other counter methods such as SHA-2, SHA-3 and so on.

Literature studies state that secured routing protocol is incorporated and uses the message authentication technique to protect the data packets transmitted [5]. The technique essentially deals with modifying routing information attacks, black hole and impersonation attacks. Key distribution and sharing overheads caused in asymmetric cryptography technique is minimized using the key count based key distribution. At the time of deployment, using key pre distribution technique, at each node same group of shared keys are stored as values in key table from K_0 to K_{n-1} (in proposed method, value of the n is 10 i.e., 10th message block is being transmitted). Keys for authentication are selected according to the value of the key count field in the control packet. Security is ensured along with the performance with this technique because of Nested MAC. And the different key is selected at each node based on the key count which is very difficult to crack by any crypt analyzer.

However this paper does not recommend any specific secure hash algorithm. It predominantly incorporates hashing technique with MD-5 which ensures security at certain level. The performance and security is even more

scaled and fine-tuned by using the most efficient hash algorithms applicable for the MANET environment.

The limitations associated with MANET such as resource constrains and energy starvation, draws towards the advent of designing light weight functioning modules [22]. It is extended even to the incorporated data security modules. The paper also suggests rigorous analysis on designing or selecting light weight functions without trading off the security threats. The paper suggests symmetric key paradigm in which encryption keys are preloaded in nodes. This reduces the key distribution and exchange overhead. The proposed algorithm has stages such as preprocessing, permutation and concatenation. It uses bit level logical operations there by using minimal computational cycles i.e. 3273 and storage overhead of 2079 bytes. It is exceptional compared to other full-fledged SHA techniques.

Few researchers incorporated SHA-1 technique in AODV protocol while routing packets. It powers the features of data integrity and authentication, over black hole and grey hole attacks [3, 4, 23–25]. The technique is very common and there is scope of fine tuning the performance with advanced hashing algorithms with additional security having more bit lengths. The report confronts that need of Public key base is eliminated and are not practically implemented.

Dual authentication technique [26] is proposed in MANET which accomplishes the additional security. One authentication is for authentication of routing packets and the other is used in preventing routing information being analyzed. The paper proves that proposed technique is efficient over digital signature method for its resource overhead and expensive design. This uses the PKI based distribution and sharing. Two hash values for both integrity and authentication is used. These values also known as MD5 checksum generate 128 bit output. Their Implementation is based on the assumption of absence of colluding in between the nodes. However the key distribution technique is not featured by any advantages in terms of resource utilization and overhead.

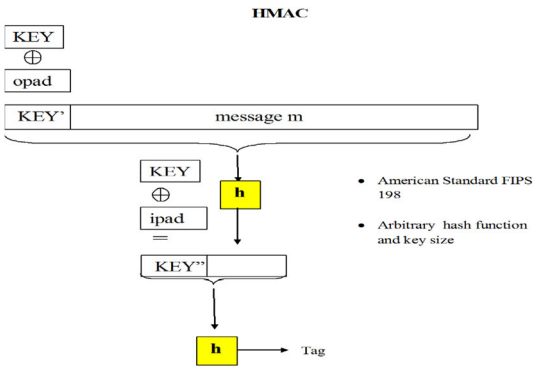

Fig. 1 Block diagram of Hash message authentication code with innerpad and outerpad key with hash flag generated

4 Proposed system

The objective of the design is to achieve high security in exchange of data between two nodes in infrastructure less adhoc-networks. The main objective of this work is to investigate the light weight hash algorithm, effective key distribution at the defined security standards. The data message is propagated through hash function block to obtain a fixed value output, along with the key.

The hash function used is one-way hash and is designed only with padding and XOR operation. MANET environment demands battery critical applications with higher data security and least cost. The main purpose of hash is to ensure data authentication and integrity. The hash function SHA-512 has its remarks in terms of advantages over security. It is also recognized as latest version of secure hash algorithm. In SHA-512 as the message digest length increases the security level also linearly increases. The put forth design is be formed of hash algorithm, message block and key table blocks respectively.

MANET environment is expected to have all the inputs mentioned above and features incorporated in each node. In order to ensure extended security, the nodes are characterized by a key count for the transmitted message. Each node consists of key count and hash algorithm to check

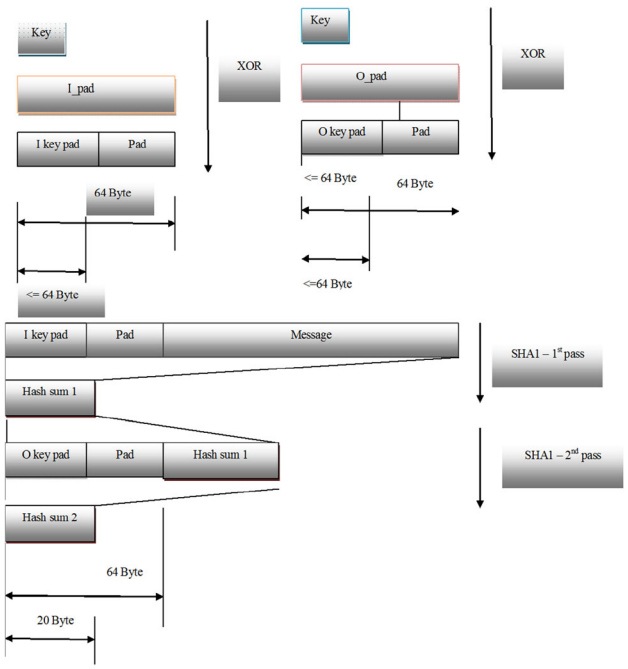

Fig. 2 Hash message authetication code block diagram (HMAC SHA-1)

integrity and authentication. The key used for hashing plays a vital role in ensuring security and various attacks mentioned in literature survey.

The dedicated key table is designed in such a way that, each key is generated based on the key count obtained from the sequence number of message transmitted. Figure 1. depicts block diagram of Hash message authentication code with Innerpad and Outerpad key and with Hash Flag generated.

Each key for the key count is considered to be unique. But generating new key for each message block to be transmitted involves more processing because of redundant, recursive and pseudo random logic used extensively in maintaining the uniqueness. This contributes to greater processing power and delay. Hence the idea is to use same key followed by bit stuffing operation thereby reduce the resources in maintaining uniqueness. The bit stuffed key is used for creating hash function.

5 Implementation

The design proposed, leverages lightweight hash algorithm in order to incorporate data security with the aid of preventing malicious attacks on the unorganized distributed MANET architectures. Using hash design message digest is calculated in each node. The message digest is appended to actual message and forwarded to either the destination node or an intermediate node.

In next node, the message integrity and authentication is checked. It is done by generating new message digest at that particular node with the selection of a particular key based on the value of key count in current node. If the message digest generated in current node matches with message digest received, the received message is accepted or else rejected. The value of key count is incremented upon successful match and the computation repeats on each node in various iterations. The implementation of proposed design consists of following modules.

- Hash message authentication design.
- Key count table.
- Key distribution.

5.1 The Hash message authentication design

It is used for both data integrity and authentication. SHA-1, SHA-256, SHA-384 and SHA-512 are implemented for the hash design. The hash output size varies based on the above specified techniques. This technique uses a key that is divided into inner and outer keys respectively. As shown in Fig. 2.

The message to be transmitted is hashed using inner key initially and the output of resulting function is hashed with the outer key. The message is divided into blocks and in accordance with the block size, there by compressing the input. The HMAC equation is given below in Eq. 1.

$$HMAC(K,m) = H(\left(k' \oplus opad\right)||H(\left(k' \oplus ipad\right)||m)) \quad (1)$$

where H is a cryptographic hash function, K is the secret key, m specifies message to be authenticated, K' is secret key, derived from the original key K, ‖ specifies concatenation, + specifies exclusive or XOR, opad is the outer padding and ipad is the inner padding.

The implementation of hash uses, key length. The length of key is compared with the message block. If key length is greater than the message, hash is computed for that key. In case key length is lesser than the message length, key is padded and then hash is computed. Outer padding and inner padding are successively accomplished followed by XOR operation in accordance with the message block size. The message block is XORed with inner key and the result is again XORed with the outer key. Finally, resulting in a final hash value, this is depicted using Fig. 3.

The Fig. 4. shows hash output for 128 bit key and message, which uses SHA-1 technique.

5.2 Key count table

The key count table is generated based on the number message blocks transmitted. Key count starts from the message 0 and terminates at message n. The Key count

```
Blocksize =

    64

hash =

C15787647D315BF30D7BA704C31A721171817AA1

ans =

C15787647D315BF30D7BA704C31A721171817AA1

Blocksize =

    64

hash =

78E5B922675BBD3BFC1D571CABADA6FAC5F5C8D9

ans =

78E5B922675BBD3BFC1D571CABADA6FAC5F5C8D9

>>
```

Fig. 3 SHA-1 HASH Code generation based on the input block size 64

```
Message_s =

00112233445566778899aabbccddeeff

>> key_hex = ('000102030405060708090a0b0c0d0e0f'; '111102030405060718090a0b0c0d0eff')

key_hex =

    '000102030405060708090a0b0c0d0e0f'
    '111102030405060718090a0b0c0d0eff'

>> HP = 0

HP =

     0

>> Kfirst = mod(HP, 2)

Kfirst =

     0

>> Kfirst = key_hex(1,:)

Kfirst =

    '000102030405060708090a0b0c0d0e0f'


Blocksize =

    64


hash =

DAFF17FD5A266F18CB10D6D516AD55393D091E5C


ans =

DAFF17FD5A266F18CB10D6D516AD55393D091E5C
```

Fig. 4 SHA-1 output based on the first key and first node keycount value

Table 1 Key count value for each message

Message block	Key count
0	0
1	1
2	2

value starts from the value 1 and ends at 4, if the number of message blocks transmitted is 4. Initially the Key count value is assigned a zero value and is incremented by the value one when the message is to be transmitted. The below Table 1 shows Key count value for each message at sender node.

5.3 Key distribution

Key distribution is done effectively in between the nodes, which are responsible for calculating hash values. First and Second pair of keys is used. Every node will have the First and Second key updated in the control packet. Hash is computed based on the selected key. The key is selected based on the Key count value. If the Key count value is 0, then First key is selected to calculate the First key and if Key count value is 1 s key is calculated for hash computation.

HP indicates Key count value

$$\text{KFirst} = \text{K}\ (\text{HP mod N}), \tag{2}$$

$$\text{KSecond} = \text{K}\ [(\text{HP} + 1)\ \text{mod N}]. \tag{3}$$

The proposed method Hash Generation in HMAC is presented below.

Step 1: Intermediate Digest Hash (KFirst ‖ Message);

Step 2: Final Digest Hash (KSecond ‖ Intermediate Digest).

The terms used in algorithms are:

i. HP is the temporary variable for storing Key count value.
ii. KFirst is the first key in HMAC.
iii. KSecond is the second key in HMAC.
iv. H (M) hash code of M.
v. Hr (m) hash code of received message M.
vi. N is number of sub keys in key.

6 Experimental results

The message of 32 hexa decimal string, pair of shared keys each of length 32 hexadecimal, Key count equal to 2 are considered. The message digest is calculated two times in each node. The first value is calculated along with KFirst and the input message, the second hash value is calculated along with the first hash value and the second key. The same computation is repeated in node 2. Finally, message digest which is nothing but the second hash value in both first and second node is compared. If both the values are

```
Message_s =

00112233445566778899aabbccddeeff

>> key_hex = ('000102030405060708090a0b0c0d0e0f'; '111102030405060718090a0b0c0d0eff')

key_hex =

    '000102030405060708090a0b0c0d0e0f'
    '111102030405060718090a0b0c0d0eff'

>> HP = 1

HP =

     1

>> Ksec = mod(HP, 2)

Ksec =

     1

>> Ksec = key_hex(2,:)

Ksec =

    '111102030405060718090a0b0c0d0eff'


Blocksize =

    64


hash =

5ADAFA64F23F7B4DB4F90246CAAEE9CC7BF9859B


ans =

5ADAFA64F23F7B4DB4F90246CAAEE9CC7BF9859B
```

Fig. 5 SHA-1 output based on the second key and second node keycount value

```
Kfirst =

     1


Blocksize =

    64


hash =

DAFF17FD5A266F18CB10D6D516AD55393D091E5C


Blocksize =

    64


hash =

5ADAFA64F23F7B4DB4F90246CAAEE9CC7BF9859B


tf =

     1

The 128 bit message is given by
00112233445566778899aabbccddeeff
The Shared pair of First Key of 128 bit is
    '000102030405060708090a0b0c0d0e0f'

The Shared pair of Second Key of 128 bit is
    '111102030405060718090a0b0c0d0eff'

The sha-1 output for the First Key in Sending node is
DAFF17FD5A266F18CB10D6D516AD55393D091E5C
The sha-1 output for the Second Key is in Sending node is
5ADAFA64F23F7B4DB4F90246CAAEE9CC7BF9859B
The sha-1 output for the First Key in receiving node  is
DAFF17FD5A266F18CB10D6D516AD55393D091E5C
The sha-1 output for the First Key in receiving node  is
5ADAFA64F23F7B4DB4F90246CAAEE9CC7BF9859B
The Message Authentication is correct and is Accepted after comparision!
```

Fig. 6 Message digest generated at the second node for comparison with the first node along with the integrity check

equal then the message is accepted or else it is rejected. Here SHA-1, SHA-256, SHA-384 and SHA-512 hash algorithm are considered and implemented.

6.1 SHA-1

Figure 4 shows input message and KFirst which is having 16 values at node 0. The key_hex is the key table which is holding two 16 hexa value keys. Based on the key count which is 0 and as per the key distribution technique using modulus function the Kfirst is selected. The number of keys considered is 2. Length of hash value obtained from input message and KFirst is 40 and is shown in Fig. 4. below.

Figure 5 shows calculation of hash value for previous hash value and the second key.

Figure 6 shows an input message, KFirst, KSecond, hash output at node 1 and hash output at node2 and comparison result of the same, hash values at node 1 and 2 are equal hence the message is accepted. The length of the hash value is 64.

```
tf =

     1

The 128 bit message is given by
00112233445566778899aabbccddeeff
The Shared pair of First Key of 128 bit is
    '000102030405060708090a0b0c0d0e0f'

The Shared pair of Second Key of 128 bit is
    '111102030405060718090a0b0c0d0eff'

The sha-1 output for the First Key in Sending node is
4AC47690FF87EE5F0F3EC32B11F8BA25181907B9F0F4E4609B8F74612E3EE97A
The sha-1 output for the Second Key is in Sending node is
1621F281E6EEB10956D2886D9EE09A476CB811D3EDF4CF2A2E2C4B247AE0ODB9
The sha-1 output for the First Key in receiving node  is
4AC47690FF87EE5F0F3EC32B11F8BA25181907B9F0F4E4609B8F74612E3EE97A
The sha-1 output for the First Key in receiving node  is
1621F281E6EEB10956D2886D9EE09A476CB811D3EDF4CF2A2E2C4B247AE0ODB9
The Message Authentication is correct and is Accepted after comparision!
```

Fig. 7 SHA-256 output at node 1, 2 for first and second keys

```
tf =

    1

The 128 bit message is given by
00112233445566778899aabbccddeeff
The Shared pair of First Key of 128 bit is
   '000102030405060708090a0b0c0d0e0f'

The Shared pair of Second Key of 128 bit is
   '111102030405060718090a0b0c0d0eff'

The sha-1 output for the First Key in Sending node is
37A5430AD8073DD8EDCCA4FC6E79A25E1499BAF09E15E9AAA2BF6302D5B798D6FE15688C6B0D94A68DA6EE4313786DEF
The sha-1 output for the Second Key is in Sending node is
66E7CED0D63375F7ECOE2A2C2ADC37B7471CBDD7A78F72E9BCCA0A168CDBF30DC71FOB2CCCOD6353D2787D6B109EDE76
The sha-1 output for the First Key in receiving node  is
37A5430AD8073DD8EDCCA4FC6E79A25E1499BAF09E15E9AAA2BF6302D5B798D6FE15688C6B0D94A68DA6EE4313786DEF
The sha-1 output for the First Key in receiving node  is
66E7CED0D63375F7ECOE2A2C2ADC37B7471CBDD7A78F72E9BCCA0A168CDBF30DC71FOB2CCCOD6353D2787D6B109EDE76
The Message Authentication is correct and is Accepted after comparision!
```

Fig. 8 SHA-384 output at node 1, 2 for first and second keys

6.2 SHA-256

Figure 7 shows the input message, KFirst, KSecond, hash output at node 1 and hash output at node 2 and comparison result of the same, hash values at node 1 and 2 are equal hence the message is accepted.

6.3 SHA-384

Figure 8 shows an input message, KFirst, KSecond, hash output at node 1 and node2 and comparison result of the same. Hash values at the node 1 and 2 are equal hence the message is accepted. The hash value is 96.

6.4 SHA-512

Using SHA-512, simulation result shows that Hash values at the node 1 and 2 are equal hence the message is accepted. The length of the hash value is 128.

The decode logic is designed so as to decode original message digest for SHA-1, SHA-256, SHA-384 and SHA-512 algorithms respectively. It extensively checks for the frequency analysis, most repeated letters in the string. The logic predominantly swaps most common letters in any string i.e. T and E. It is done in a way to make a meaning full message and make an attempt to trace out the similarity index. The below specified input message and key is considered for the below analysis and is iterated on all the discussed SHA algorithms.

The input message considered is as follows:

"00112233445566778899aabbccddeeff".

Key considered is as follows:

"1111020304050607080 90a0b0c0d0eff".

Message digest varies with the SHA algorithms used. Message digest would be an input to the decode logic which generates two new messages out of which one is based on the frequency analysis and another is based on swapping with the letter T and E. The output messages are compared with original message, hash message and key for the match.

Figure 9 depicts the result of decode logic for SHA-1, the message digest is of length 20 and hexa string is as follows:

"DAFF17FD5A266F18CB10D6D516AD
55393D091E5C"

```
Warning: message is short (less than 50 characters). Frequency analysis is unlikely to decode message. Attempt with a longer message.

analysis =

[a b c d e f g h i j k l m n o p q r s t u v w x y z]
[3 1 2 6 1 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

the most likely letters to be correct are "E" and "T".
 Look out for "T-E" which is likely the word "the", or an individual letter, most likely "A".
 If you find a potential mistake you may swap two letters. e.g TOE THH ->swap O and H->THE TOO

message =

EATT7TEA6T8OIOEE1AE59E9NO

would you like to swap two letters? (''T''/''N'')y

ans =

EATT7TEA6T8OIOEE1AE59E9NO

>> X= 'TAEE7ETA6E8OTT1AT59T9NO'

X =

TAEE7ETA6E8OTT1AT59T9NO
```

Fig. 9 Comparative analysis of SHA-1 in predicting the Keys and messages from the hash value

and it is very small i.e. "EATT7TEA6T8OI0EE 1AE59E9NO".

Upon swapping the letters T and E in above message, the result is as follows:

"TAEE7ETA6E80TT1AT59T9NO"

It neither matches with any of the above strings.

Figure 10 depicts decode analysis with SHA-256. This hash algorithm generates the following message digest:

"4AC47690FF87EE5F0F3EC32B11F8BA2518
1907B9F0F4E4609B8F74612E3EE97A"

Upon applying the decode logic it generates most repeated characters. Swapping mechanism is used to reconstruct the original message. The new decode message generated is:

"OI79EE7TTEETI2A1EAO5897AEET69A
E41TTT7O"

And the swapping message is given as follows:

"OI79EETTEI2A1TAO5897ATTE69AT41EEE7O"

Neither the message digest nor Eqs. 1 and 2 match the above two messages from decode logic.

Figure 11 shows the decode analysis with SHA-384 hash algorithm and generates a message digest as given below:

"37A5430AD8073DD8EDCCA4FC6
E79A25E1499BAF09E15E9AAA2BF6302D5B7
98D6FE15688C6B0D94A68DA6EE4313786DEF".

Upon applying decode logic, it generates the most repeated characters and uses swapping mechanism to rebuild the original message.

The new decode message generated is

"7E40ET03TTATNNEONA9E5A49IEO9A
5AEEEIO32TI9TOA58NIT4E8TEAA338TAO".

and swapping message is given by

"7T40TE03EEAENNTONA9T5A49ITO9
A5ATTTIO32EI9EOA58NIE4T8ETAA338EAO".

Neither the message digest nor Eqs. 1 and 2 match the above two messages from decode logic.

The Fig. 12, shows decode analysis with SHA-512 hash algorithm, generating the message digest as follows:

"1FDFC7C9F5FD538D9FF05AA43B82CE8
6F36DA7424175ADB1965BEFA9305E
EBA993E488AD9817AAC30F1BA79D7881
17166C2DD1E085518E245CD2403F0C04F730".

```
analysis =

[a b c d e f g h i j k l m n o p q r s t u v w x y z]
[3 4 2 0 7 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

the most likely letters to be correct are "E" and "T".
 Look out for "T-E" which is likely the word "the", or an individual letter, most likely "A".
 If you find a potential mistake you may swap two letters. e.g TOE THH ->swap O and H->THE TOO

message =

OI79EE7TTEETI2A1EAO5897AEET69AE41TTT7O

would you like to swap two letters? (''Y''/''N'')y

ans =

OI79EE7TTEETI2A1EAO5897AEET69AE41TTT7O

>> X='OI79EETTEI2A1TAO5897ATTE69AT41EEE7O'

X =

OI79EETTEI2A1TAO5897ATTE69AT41EEE7O
```

Fig. 10 Comparative analysis of SHA-256 in predcting the Keys and messages from the hash value

```
analysis =

[a b c d e f g h i j k l m n o p q r s t u v w x y z]
[10 4 4 9 9 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

the most likely letters to be correct are "E" and "T".
 Look out for "T-E" which is likely the word "the", or an individual letter, most likely "A".
 If you find a potential mistake you may swap two letters. e.g TOE THH ->swap O and H->THE TOO

message =

7E4OETO3TTATNNECNA9E5A49IEO9A5AEEEIO32TI9TOA58NIT4E8TEAA338TAO

would you like to swap two letters? (''Y''/''N'')y

ans =

7E4OETO3TTATNNECNA9E5A49IEO9A5AEEEIO32TI9TOA58NIT4E8TEAA338TAO

>> x='7T4OTEO3EEAENNTCNA9T5A49ITO9A5ATTTIO32EI9EOA58NIE4T8ETAA338EAO'

x =

7T4OTEO3EEAENNTCNA9T5A49ITO9A5ATTTIO32EI9EOA58NIE4T8ETAA338EAO
```

Fig. 11 Comparative analysis of SHA-384 in predcting the Keys and messages from the hash value

Upon applying the decode logic it generates most repeated characters and uses swapping mechanism to rebuild original message.

The new decode message generated is:

```
analysis =

[a b c d e f g h i j k l m n o p q r s t u v w x y z]
[10 5 7 10 7 11 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]

the most likely letters to be correct are "E" and "T".
 Look out for "T-E" which is likely the word "the", or an individual letter, most likely "A".
 If you find a potential mistake you may swap two letters. e.g TOE THH ->swap O and H->THE TOO

message =

EAEOOEEA3AEE5TT3N2OI6E6AT447TAN95NIET35IINT9I8TA87TTO0ENT9A8176OAAI858I4OA43EO4E3

would you like to swap two letters? (''Y''/''N'')y

ans =

EAEOOEEA3AEE5TT3N2OI6E6AT447TAN95NIET35IINT9I8TA87TTO0ENT9A8176OAAI858I4OA43EO4E3

>> x='TATOOTTA3ATT5EE3N2OI6T6AE447EAN9SN1TE35IINE9I8EA87EEO0TNE9A8176OAAI858I4OA43TO4T3'

x =

TATOOTTA3ATT5EE3N2OI6T6AE447EAN9SN1TE35IINE9I8EA87EEO0TNE9A8176OAAI858I4OA43TO4T3
```

Fig. 12 Comparative analysis of SHA-512 in predcting the Keys and messages from the hash value

"EAEOOEEA3AEE5TT3N2OI6E6AT447TAN9
5NIET35IINT9I8TA87TTO0ENT9A8176OAAI
858I4OA43EO4E3"

And the swapping message is given by following string:

"TATOOTTA3ATT5EE3N2OI6T6AE447EAN95
NITE35IINE9I8EA87EEO0TNE9A8176OAAI85
8I4OA43TO4T3",

Both the above messages from decode logic does not match with the Eq. 1, 2 and the message digest.

7 Conclusion and future scope

The SHA algorithms discussed uses only XOR and padding operations in order to generate a message digest. It consumes very less computations and latency. Effectively shared pair of key is distributed based on key count with less overhead of key distribution compared to asymmetric counterpart. Thereby they optimize the design in terms of processing power. Also boost the system speed with defined data security standards which are most vulnerable to possible attacks. The message digest is calculated for all the SHA versions discussed above].

In general sense, intensity of vulnerability depends on the length of message. As the message digest length increases, possibility and number of iterations required to decode the message would be exponentially difficult.

The SHA-384 and SHA-256 algorithms are more remarkable in generating increased length message digest which is hardly possible to decode from the message

digest. The frequency analysis is done which will strengthen the above statement as proof of concept.

The hash design is even more optimized in terms of light weight computation by using neural network concept and advanced mathematical functions which can increase the computational speed. The scope is even extended by incorporating the different vulnerability techniques and its predictability measured using different randomness test. Furthermore, it can also be tested using various CUDA streams.

References

1. Min Z, Jiliu Z (2009) Cooperative black hole attack prevention for mobile ad hoc networks. IEEE. https://doi.org/10.1109/IEEC.2009
2. Dilli R, Reddy PCS (2016) Trade-off between length of the Hash code and performance of hybrid routing protocols in MANETs. IEEE
3. Aware AA, Bhandari K (2014) Prevention of black hole attack on AODV in MANET using hash function. IEEE
4. Hizbullah Khattak N (2013) A hybrid approach for preventing black and gray hole attacks in MANET. IEEE
5. Patel A, Patel N, Patel R (2015) Defending against wormhole attack in MANET. In: 2015 fifth international conference on communication systems and network technologies
6. Singh M, Garg D (2008) Choosing best hashing strategies and hash functions. IEEE
7. Selvakumar AL, Ganandhas CS (2009) The evaluation report of SHA-256 crypt analysis hash function. IEEE
8. Mukundan PM, Manayankath S, Srinivasan C, Sethumadhavan M (2016) Hash-one: a lightweight cryptographic hash Function. IET Inf Secur 10(5):225–231
9. Patil MA, Karule PT (2015) Design and implementation of Keccak hash function for cryptography. IEEE
10. Tsudikt G (1992) Message authentication with one-way hash functions. IEEE
11. Chankasame W, San W (2015) A chaos-based keyed hash function for secure protocol and message authentication in mobile ad hoc wireless networks. In: Science and Information Conference, July 28–30, 2015. London
12. Alosaimy R, Alghathbar K, Hafez AM, Eldefrawy MH, Kim T et al (eds) (2010) NMACA approach used to build a secure message authentication code. In: SecTech/DRBC 2010, CCIS 122, pp 290–298. Springer, Berlin
13. Wang D, Jiang Y, Song H, He F, Gu M, Sun J (2017) Verification of implementations of cryptographic hash functions, vol 5, pp 2169–3536
14. Liu H, Kadir A, Liu J (2019) Keyed hash function using hyper chaotic system with time-varying parameters perturbation. IEEE Access. https://doi.org/10.1109/ACCESS.2019.2896661
15. Patel MM, Tanwar S, Gupta R, Kumar N (2020) A deep learning-based cryptocurrency price prediction scheme for financial institutions. J Inf Secur Appl. https://doi.org/10.1016/j.jisa.2020.102583
16. Rupa C, Srivastava G, Gadekallu TR, Maddikunta PKR, Bhattacharya S (2020) Security and privacy of UAV data using blockchain technology. J Inf Secur Appl. https://doi.org/10.1016/j.jisa.2020.102670
17. Khan JS, Kayhan SK (2021) Chaos and compressive sensing based novel image encryption scheme. J Inf Secur Appl. https://doi.org/10.1016/j.jisa.2020.102711
18. Choi H, Seo SC (2021) Fast implementation of SHA-3 in GPU environment. IEEE Access. https://doi.org/10.1109/ACCESS.2021.3122466
19. Alzahrani A, Gebali F (2018) Multi-core dataflow design and implementation of secure hash algorithm-3. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2799802
20. Yang Y, He D, Kumar N, Zeadally S (2018) Compact hardware implementation of a SHA-3 core for wireless body sensor networks. IEEE Access. https://doi.org/10.1109/ACCESS.2018.2855408
21. Ravilla D, Putta CSR (2015) Implementation of HMAC-SHA256 algorithm for hybrid routing protocols in MANETs
22. Luo J, Fan M, Ye D (2008) Black hole attack prevention based on authentication mechanism. IEEE
23. Arya KV, Rajput SS (2014) Securing AODV routing protocol in MANET using NMAC with HBKS technique. IEEE
24. Chowdhury AR, Das Bit S (2015) LMAC: a lightweight message authentication code for wireless sensor network. IEEE
25. Bhatia T, Verma AK (2013) Security issues in manet: a survey on attacks and defense mechanisms. IJARCSSE 3(6)
26. Raja L, Periasamy PS (2016) Dual authentication hashing for security enhancement in MANET. Circ Syst 7:350–359