International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 06 | June 2020 www.irjet.net p-ISSN: 2395-0072

Verification of Data Integrity using MD5 Hashing Function

Silveri Anvesh[1]

[1]*M.S, Dept. Of ECE Engineering, VEDAIIT, Andhra Pradesh, India.*

Abstract - *With the growth of the personal computers and the internet, computers are now integrated into our lives. The advancement of technology is leading to data threats and security issues. A hacker may gain access to our most secret information by breaking through our firewall and gaining access to the internals of our computing system even though he is physically present somewhere else. Compromised computers may leak our most private information: personal documents, pictures and movies, browsing history, chat history, bank account passwords, etc.*

A persistent problem for modern System-on-Chip is their vulnerability to code injection attacks. By tampering the memory content, attackers are able to extract secrets from the SoC and to modify or deny the SoC's operation. This work proposes secure, and fast method to check the integrated data is safe or not present in Memory. All applications that need high security should be immune to both physical and software attacks through the secure architectures. Memory integrity verification is a major and important issue while implementing secure processors. This paper proposed a method to ensure data integrity.

Key Words: Memory Integrity, data integrity, encryption, decryption.

1. INTRODUCTION

1.1 Data Integrity

Data integrity is the accuracy, completeness, and consistency of information. Data integrity also refers to the safety of data in regards to regulatory compliance and security. When the integrity of data is preserved, the information stored in memory will remain similar and reliable no matter how long it's stored or how frequent it is accessed. Data integrity also ensures that your data is safe from un-authenticated users or hackers. When the information which is sensitive is exchanged, the receiver must have the confirmation that the message has come intact from the correct sender and isn't modified by the hacker. There are two kinds of data integrity threats, they are passive and active.

1.1.1 Passive Threats

This type of threats exists because of changes in data. These data errors are likely to occur because of noise in an exceedingly communicating. Also, the information may get corrupted while the file is stored on a disk. Error correcting codes and straight forward checksums like Cyclic Redundancy Checks (CRCs) are accustomed detect the loss of information integrity. In these techniques, a digest of information is computed mathematically and appended to the information.

1.1.2 Active Threats

In this type, an attacker can alter the data with malicious data. In other Words, if data is without digest, it is going to be converted without notice. The system appends CRC to information for knowing any changes. At higher level of threat, attacker may alter data and replaces with other digest for altered data from exiting digest. This can be possible if the digest is computed using mechanisms such as CRC. Hashing and Encryption are extremely used in verifying the data integrity and for maintenance of security.

1.2 Hashing

Hash functions are very useful and exist in all security applications. A hash function is a mathematical function that converts a numerical input value to other coded numerical value. The input to the hash function is arbitrary length but output of hash function is always of same length.

Values delivered by a hash function is called message digest or hash values. The following picture shows hash function –

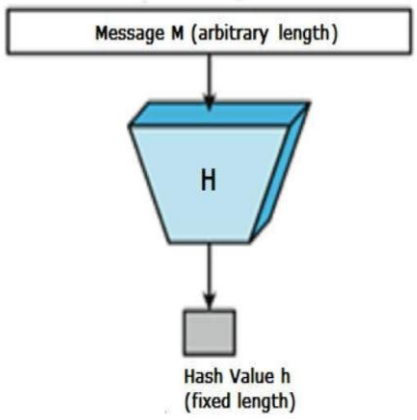

Fig -1: Hashing function

Features of Hash Functions:

The typical features of hash functions are –

Fixed Length Output (Hash Value)

- Hash function converts data of arbitrary length to a pre-defined length. This process is usually referred to as an hashing of the information.
- Generally, the hash value is way smaller than the input plain-text data, hence hash functions can be called as compression functions.
- Since a hash is a representation of a larger data, it is also referred to as a digest.
- Hash function with n bit output is called as an n-bit hash function. Popular hash functions will generate hash values between 160 and 512 bits.

Efficiency of Operation

- Generally for any hash function h with input x, computation of h(x) is a fast operation.
- Computationally hash functions are way faster than a symmetric encryption.

Properties of Hash Functions:

In order to be an effective cryptographic tool, the hash function is desired to possess following properties –

Pre-Image Resistance

- This property means it should be computationally hard to reverse a hash function.
- In simpler words, if a hash function h produced a hash value z, then it should be a difficult to figure out any input value that hashes to z.
- This property secures against an hacker who has a hash value and is willing to decoed the input data.

Second Pre-Image Resistance

- This property indicates given an input value and its hash value, it should be impossible to find a different input plain-text data with the identical hash value.
- In other words, if a hash function h for an input x produces hash value h(x), then it should be difficult to find other input value y such that h(y) = h(x).
- This property of hash function secures against an hacker who has an input value and its hash, and wants to substitute different value as legitimate value instead of original input value.

Collision Resistance

- This property means it should be very difficult to trace out two different inputs of any length that results in similar hash value. This property is also called as collision free hash function.
- In other words, for a hash function h, it's difficult to trace any two different inputs x and y so that h(x) = h(y).
- Since, hash function is compressing function with pre-defined hash length, it's possible for a hash function to occur collisions. This property of collision free only confirms that these collisions should be hard to trace out.
- This property makes it difficult for an hacker to trace two input values with the same hash value.
- If a hash function is collision-resistant then it is second pre-image resistant.

1.3 Encryption

Encryption is a way of altering information so that only authorized users can be able to understand the information. In technical terms, it is a process of converting plain-text data to cipher-text data. In other words, encryption takes readable data and changes it to the random data. Encryption uses an encryption key: a collection of mathematical values that both the sender and receiver of an encrypted message know.

Although encrypted data appears random, encryption proceeds in a exceedingly logical, predictable way, so a user receiving the encrypted data and in possession of the key used to encrypt the data can decrypt the information, turning it back into plain-text data. Truly secure encryption will be complex enough that a 3rd party is extremely unlikely to decrypt the cipher-text by brute force – in simpler words, by trial and error method.

The two popular methods of encryption are symmetric encryption and asymmetric encryption. Asymmetric encryption is additionally called as public key encryption.

In symmetric encryption, only one key is present, and all users use the similar key for encryption and decryption. In asymmetric, or public key, encryption, two keys are present - one key is for encryption, and a different key is for decryption. Either key can be used for either action, but data encrypted with the first key can't be decrypted with the same key, and vice versa. One key is kept private, while one key is shared to all, for anyone to use – hence the name public key.

1.4 Decryption

Decryption is the process of converting information that has been converted to unreadable using encryption back to its unencrypted form. In decryption, the system extracts and converts the garbled data and transforms it to texts and images that are easily understandable to the end user.

Decryption may be accomplished manually or automatically. It may also be performed with a set of keys or passwords. One of the main reasons for implementing an encryption-decryption system is for keeping data securely. As information travels over the internet, it can be accessed from unauthorized users. So, data is encrypted to reduce information loss. Some of the common items that are encrypted include email messages, text files, images, user data and directories. The person in charge of decryption receives a prompt or window in which a password may be entered to access encrypted information.

2. HASHING ALGORITHM – MD5

2.1 Design of Hashing Algorithm

Hashing is a mathematical function that operates on two fixed-size blocks of information to form a hash code. This hash function forms a part of the hashing algorithm.

The size of every data block varies based on the algorithm. Typically the block size varies from 128 bits to 512 bits. The following illustration figure1 shows hash function –

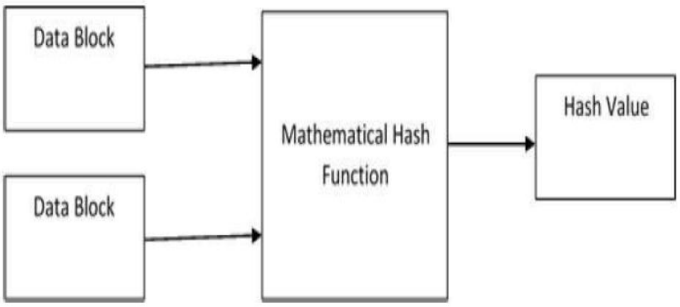

Fig -2: Design of Hash Value

Hashing algorithm involves rounds of above hash function like a block cipher. Each round takes an input of a fixed size, typically a combination of the most recent message block and the output of the last round.

This process is repeated for many rounds as required to hash the entire message. Schematic (figure2) of hashing algorithm is depicted in the following illustration –

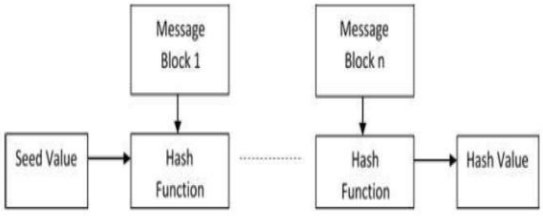

Fig -3: Schematic of Hashing algorithm

Since, the hash value of first message block becomes an input to the second hash operation, output of which changes the results of the third operation, and so on. This effect is named as an avalanche effect of hashing.

Avalanche effect produces different hash values for two messages that differ even by a single bit.

There's a difference among hash function and hash algorithm. The hash function generates hashes by taking inputs of two blocks of fixed-length binary data, where as Hashing algorithm is a method of using an hash function, specifying how the message will be broken up and how the results from previous message blocks are added together.

Cryptographic hash functions take data input and generates a pre-defined size result or digest. That result is called check some. It is not possible to trace an input from the result of hash function. One thing is that the hash functions are not encryption because we can't decrypt an input from the output.

Cryptographic hash Function is MD5 is one of the reputed hashing technique. MD5 creates a 128-bit message digest from the data input which is in 32 digits hexadecimal number. MD5 hashes are unique for different inputs irrespective of the size of the input. MD5 hashes looks like this.

When processor wants to store data in an external memory, which we assume as untrusted part, we have to make sure that data we wrote in the memory is not manipulated by the un-authenticated user, this is possible by using hashing technique.

Message Digest (MD) – A Hashing Function:

MD5 was most prefered and widely used hash function for few years.

- The MD comprises of hash functions MD2, MD4, MD5 and MD6.It is a 128-bit hash function.
- MD5 have been widely used in the security world to provide the integrity of the information.
- In 2004, collisions were found in MD5. An attack have been successfully hacked the confidential information only in an hour. This collision attack resulted in compromised MD5 and hence it is no longer recommended to be used for security.

MD5 generates a 128bit hash value from the input. The output must be unique from other hash values. Assume a b-bits message to digest. To digest this message 5 steps have to be followed. Professor Rivest used the first two steps to prepare the input for digestion by appending and padding its bits. In the third and fourth step he used few helper

functions which have 4 word buffers and 4 auxiliary functions which are already initialized.

2.2 Append Padding Bits

The first step is to extend the B-bits input so that the length of the message is equal to 448, modulo 512. In simpler words, message should be just 64-bits shy of being a multiple of 512 that is after padding the message+64 bit should be divisible by 512. (Message+64)/512 will have reminder 0. No matter the size of the message padding is always done. First a '1' bit is appended to the message and then a series of '0' bits have to be appended.

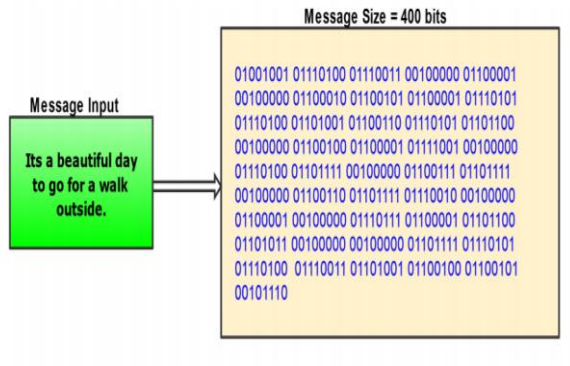

Fig -4: Message Structure

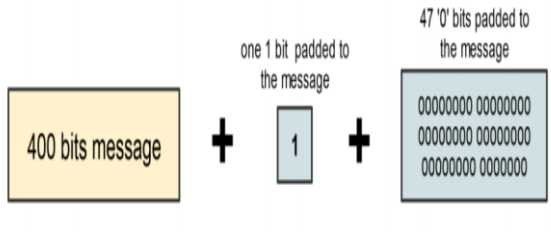

Fig -5: How bits are appended

2.3 Append Length

Now, take the original message and make 64-bit representation of the original B-bit message. We append this to the result of previous step. Now the message has length that is multiple by 512. Which itself is divisible by 16. The message is divided into blocks, each block of 512 bits each. Each 512 bits block have been divided into 16 words of 32-bits each. We denote the words as M[0.....N-1] where N is a multiple of 16.

2.4 MD5 Helper Functions

The Buffer:
MD5 uses a buffer which is made up of four words, each 32 bit long. These words are called A, B, C, D. They are initialized as
word A: 01 23 45 67
word B: 89 ab cd ef
word C: fe dc ba 98
word D: 76 54 32 10
The Table:

MD5 further uses a table K which has 64 elements. Element number i is indicated as Ki. The table is computed in advance to speed up computations. Elements are computed with a mathematical sin function:
$K_i = abs(sin(i + 1))*2^{32}$

2.5 Four Auxillary Functions

In addition MD5 uses 4 auxiliary functions that each takes as '3' 32-bit words as input and produce as '1' 32-bit word as output. They apply and, or, not and xor logical operators to the input bits.

F(X,Y,Z) = (X and Y) or (not(X)and Z)

G(X,Y,Z) = (X and Z) or (not(Z)and Y)

H(X,Y,Z) = (X xorYxor Z)

I(X,Y,Z) = Y xor (not(Z)or X)

2.6 Processing Blocks

The contents of the 4 buffers(A,B,C,D) are mixed with the words of input, using the four auxiliary functions(F,G,H,I).There are four rounds, each round involves 16 basic operations. One operation is illustrated in figure6 below:

The figure6 shows how an auxiliary function F is applied to the four buffers(A,B,C,D),using message word Mi and constant Ki. The item "<<<s" denotes a binary left shift by s bits.

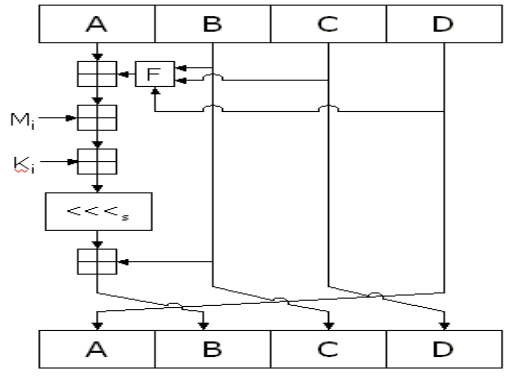

Fig -6: Application of auxillary function F

The Output

After all the rounds have been performed, the buffers A,B,C,D contain the MD5 digest of the input message.

3. AES ENCRYPTION AND DECRYPTION ALGORITHMS

The widely known and adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is six times faster than triple DES. There is a need to replace DES as its key size was too small. With increasing computing power, it absolutely was considered vulnerable against exhaustive key search attack. Triple DES was designed to beat this drawback but it absolutely was found slow.

The features of AES are as follows –

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide entire specification and design details
- Software implementable in C and Java

3.1 Operation Of AES

AES is an iterative method rather than Feistel cipher. It works on 'substitution–permutation network'. It consists of a series of linked operations, some of these involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations).

AES performs all the computations on bytes and not on bits. So, AES treats the 128 bits of a information block as 16 bytes. These 16 bytes are kept in four columns and four rows in a matrix form for processing –

In AES the amount of rounds is variable and depends on the length of the key whereas in DES it depends on key length. In AES has for 128-bit keys there are 10 rounds, 192-bit keys there are 12 rounds and 256-bit keys there are 14 rounds. Each of those rounds uses a unique 128-bit round key, which is generated from the particular AES key.

The schematic of AES structure is given in the following illustration –

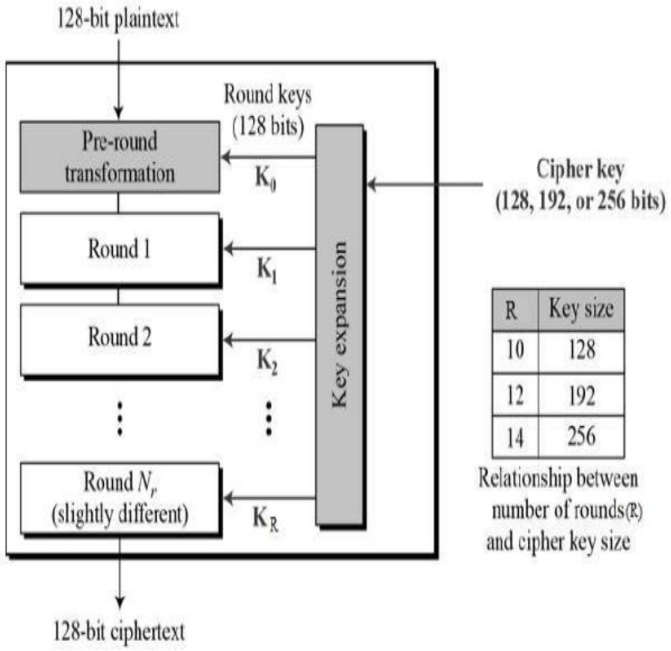

Fig -7: AES Structure

3.2 Encryption Process

Now, let's see the description of a typical round of AES encryption. Each round has four sub-processes. The first round process is as shown below –

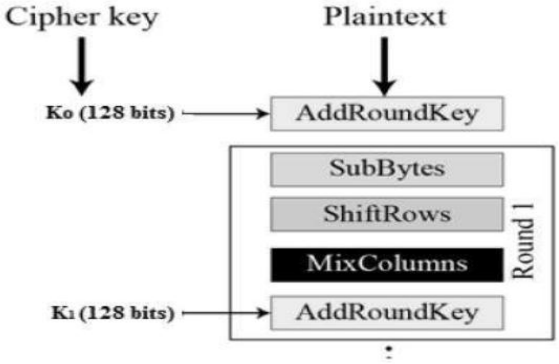

Fig -8: Encryption Process

Byte Substitution (SubBytes):

The 16 bytes of input are substituted by using a fixed table (S-box) given in design. The result is in the form of a matrix of four rows and four columns.

Shiftrows:

Every row of the matrix is shifted to the left. Any entry that 'fall off' are again re-inserted on the right side of row. Shift is done as follows –

- First row isn't shifted.
- Second row is shifted left by one (byte) position.
- Third row is shifted left to two positions.

- Fourth row is shifted left by three positions.
- The result is a new matrix with same 16 bytes but shifted accordingly with respect to each other.

MixColumns:

Every four byte column is now transformed using a special mathematical function. This function takes a four byte input of one column and outputs four new bytes, which replace the actual column. The result of this is an another new matrix consisting of 16 new bytes. It is to be noted that this step is not done in the last round.

Addroundkey:

The 16 bytes of the matrix are now taken as 128 bits and are XORed with 128 bits of the round key. If last round is being done then the output is the ciphertext. In other case, the resulting 128 bits are considered as 16 bytes and we begin another similar round.

3.3 Decryption Process

The decryption process of an AES ciphertext is similar to the encryption process but in the reverse order. Each round has four processes which are conducted in an inverted order –

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are inverse to each other, unlike for a Feistel Cipher, the encryption and decryption algorithms must be separately implemented, although they are very closely related.

3.4 AES Analysis

At present in cryptography, AES is widely used as it supports both hardware and software. Till today, no practical cryptanalytic attacks are registered against AES. And in addition to this AES has built-in flexibility of key length, which allows a degree of 'future-proofing' against progress in the ability to perform exhaustive key searches.

However, the AES security is assured only if it is correctly implemented and good key management is employed.

4. VERIFICATION OF MEMORY INTEGRITY

Memory Integrity Verification – Description:

- Our main intention is to check the integrity of the data which is present in the External Memory
- We assume, Memory as an un-trusted part. As the memory contains the important data, which is very sensitive, which should not be accessed by the un-authenticated users.
- There are many ways a hacker can corrupt or access the data, out of many I would like to verify whether the data present in the external memory is secure i.e., the un-authenticated user has not altered the data. For verifying it, Memory integrity is the best way.
- In the above block diagram, we can see the Message Digest5(MD5) function which is one among the famous hash functions and the AES Encryption and decryption block.
- Firstly, when processor wants to write data into the memory, it has to give address (to which subset of memory data has to be send)and data(actual information).
- When the processor sends the address information it will goes to the MD5 hash function through the address bus,(which we assume as 32bits) and then when data is sent by the processor, the Hash function using its algorithm converts the plain-text data to the hash Value, which is of 128bits(in case of MD5), those hash values will be stored in the hash table, In hash table the address and the corresponding hash value will be stored. Parallely the AES128 Encryption gets the 32bitWRDATA as the input, it converts the pain-text data to the Encrypted Data(which is reversible where as the Hash values generated can't be reversed) and the encrypted data will be stored in the external memory targeted by the processor.
- Now we have done with writing data into the External Memory, As the External Memory is Un-trusted, hackers can change data without the notice of the valid user. So to check that whether our information is secure or not we have to read the Data by giving the address which we want to check.
- By Giving the 32bitRDADDR it goes to External Memory and the data residing to that address will be selected and fetched from the External Memory, As the Data in External Memory is Encrypted it has to be decrypted so that it has to be Hashed by the MD5 again.
- When the Decrypted Data is Received by the MD5 it generates the 128bit Hash Value and sends to the Checker, Checker By receiving the Hash Value, it has to confirm whether the Hash value is similar to the Hash value generated for the WRDATA, so the checker polls the hashtable using the WRADDR , Hashtable by receiving the WRADDR will send the corresponding 128bitHashvalue which is stored in it.
- Then Checker after getting the Hash value will compare the two HashValues(of WRDATA and RDATA) Checker performs the XOR operation between two Hash values, When the two Hash values are same using the XOR operation it

Generates the zero as the output (which is error signal), which indicates that that no error is present and the Data written in memory is safe. If the two hash values are different then XOR Generates one, which implies Error signal asserts, which indicates that the data stored in the memory have been altered by the hacker.

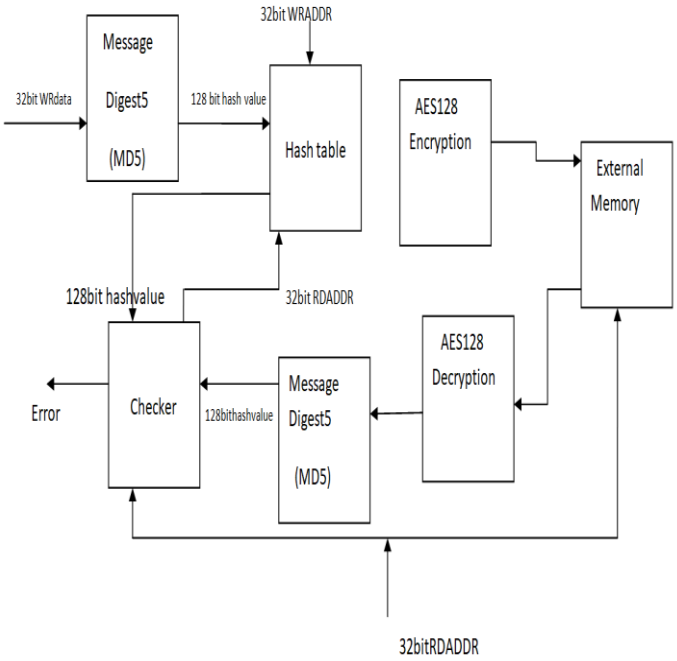

Fig -9: Verification of memory integrity - schematic

5. RESULTS

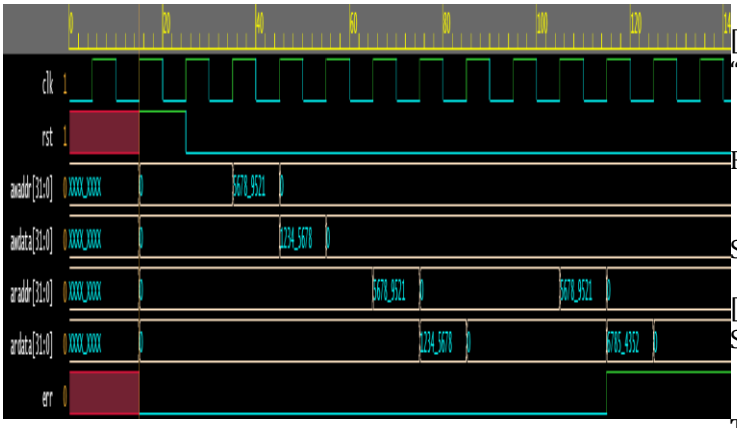

Fig -10: Result of MD5 hashing

Firstly, on resetting, I have passed a address through address bus, to where (in which segment of memory), I want to store data, then after sending data, I have passed data of 32 bits, that data will be encrypted through encryption algorithm and a hash value is created using a hash function, here as we are using MD5 it produces a hash value of 128 bits. In the memory the encrypted data was stored and in the hash table, the address and the corresponding hash value will be stored.

When we want to check the data integrity(security feature) , we have to read send a address using address bus, so that particular address location will be selected, and the encrypted data will be fetched from the memory, The data will be decrypted using the Decryption algorithm, and the hash value will be generated and compared with the hash value of the data, which was generated earlier for the same data, and does XOR operation, on both Hash values, if both are same “err” produces ‘0’ ,else produces ‘1’ .

In the Wave form, we can see two scenarios, where for the first reading from the memory, err is ‘0’ indicates, no memory changes, in the second reading, err is ’1’ indicates that data in the memory have been altered.

6. CONCLUSION

This paper discusses about the memory integrity verification of a SoC using MD5, which is one of the famous Hashing Functions. And here are two scenarios: In first scenario, the memory integrity is good, i.e., the write and read data matches. In the second scenario, the memory integrity failed as write and read data doesn’t match as the data in the memory was hacked by the hacker.MD5 Algorithms are useful because it is easier to compare and store these smaller hashes than to store a large text of variable length. The MD5 algorithm is a widely used algorithm for one way hashes that are used to verify without necessarily giving the original value. MD5 algorithms are widely used to check the integrity of the files.

REFERENCES

[1] Behrouz A. Forouzan, Debdeep Mukhopadhya, “Cryptography and Network Security” Second Edition,2011.

[2] William Stallings, “Cryptography and Network Security”, Fifth Edition”.

[3] Miles E. Smid and Dennis K. Branstad, “Data Encryption Standard past and furure” may-1998

[4] Dr Reinhard Wobst, “The Advanced Encryption Standard(AES): The Successor of DES” 2001.

[5] Douglas Stinson, Chapman & Hall/CRC, “Cryptography: Theory and Practice”.

[6] Charles Kaufman et al, “Network Security: Private Communication in a public word”

[7] Rivest, R. The MD5 message-digest algorithm. RFC 1321, 37 (April 1992).

[8] Zhang Shaolan, Xing Guobo, Yang Yixian, Improvement and Security Analysis on MD5 [J]. Computer Application, 2009, vol. 29(4):947-949.

[9] Xiaoling Zheng, JiDong Jin, Research for the Application and Safety of MD5 Algorithm in Password Authentication, 9th International Conference on Fuzzy Systems and Knowledge Discovery, 2012.

[10] H. Mirvaziri, Kasmiran Jumari, Mahamod Ismail, Z. Mohd Hanapi, A new Hash Function Based on Combination of Existing Digest Algorithms , The 5th Student Conference on Research and Development – SCOReD 2007, 11-12 December 2007, Malaysia.

[11] Abdullah, A. M., & Aziz, R. H. H. (2016, June). New Approaches to Encrypt and Decrypt Data in Image using Cryptography and Steganography Algorithm., International Journal of Computer Applications, Vol. 143, No.4 (pp. 11-17).

[12] Singh, G. (2013). A study of encryption algorithms (RSA, DES, 3DES and AES) for information security. International Journal of Computer Applications, 67(19).

[13] Gaj, K., & Chodowiec, P. (2001, April). Fast implementation and fair comparison of the final candidates for Advanced Encryption Standard using Field Programmable Gate Arrays. In Cryptographers' Track at the RSA Conference (pp. 84-99). Springer Berlin Heidelberg.