ISSN (Online) 2278-1021
ISSN (Print) 2319 5940

IJARCCE

International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

Implementation of Hybrid Cryptosystem using AES-256 and SHA-2 256 by LabVIEW

Israa H. Latif[1], Ergun Erçelebi[2]

Electrical and Electronics Dept., College of Engineering, Gaziantep University, Gaziantep, Turkey [1, 2]

Abstract: In This paper, the software implementation of the hybrid cryptosystem which consists of the Symmetric-key algorithm AES-256 and the secure hash algorithm SHA-2 256 is presented using the VI LabVIEW environment toolkit. The idea of the proposed Hybrid Cryptosystem is to use the SHA-256 bit as a key generation for AES-256 in order to improve the data security to a greater extent because it provides higher security in terms of complexity. The proposed hybrid cryptosystem is implemented using LabVIEW 2013, and from the simulation results obtained, we see the simplicity in modeling AES-256,SHA-256 and the complete hybrid cryptosystem, the results show two cases, the first case is how we can use the same input messages(plain text of AES-256 is the same as the input message to SHA-256), and the second case with different input messages (plain text of AES-256 is not the same as the input message to SHA-256). And from these results we see that the output results will be the same for the compelete hybrid cryptosystem in the two cases.

Keywords: Advanced Encryption Standard (AES-256), Cryptography, Hash function (SHA-2 256), key Generation, LabVIEW.

I. INTRODUCTION

Crypto-system is the dealing method with the security-related problems by encrypting the sending data and decrypting the receiving data. So, we can store sensitive and important information or transmit them across insecure networks in a safe way to be protected from unauthorized persons. Cryptography involves three distinct mechanisms: Asymmetric-key encipher mechanism (public key method), Symmetric-key encipher mechanism (secret key method), and Hashing Algorithms [1].

For asymmetric key cryptographic algorithms, there are two kinds of keys: a private key and a public key. The private key is kept secret while the public key is known to everyone. The public key is for encryption, and the private key is used to decrypt the encrypted message which can be only used by the corresponding private key. Examples on the asymmetric crypto system are the Rivest-Shmir–Adleman (RSA) and Elliptic curve cryptography (ECC).

For symmetric key cryptographic algorithm, it has a single key for both encryption and decryption processes. It can be classified into two groups, namely block and stream ciphers. The Data Encryption Standard (DES) and Advanced Encryption Standard (AES) are two examples of the symmetric block crypto system [2].

The Advanced Encryption Standard (AES) is a symmetric-key block cipher which is published by the National Institute of Standards and Technology (NIST) in December 2001.NIST announced that a draft of the Federal Information Processing Standard (FIPS) was available for public review and common in February 2001. Finally, in December 2001, AES was published as FIPS 197 in the Federal Register[2].

For Cryptographic hash functions, they are the mathematical operations that run on the digital data. A user can determine the integrity of any data by making a comparison between the computed “hash” to an expected hash value. The Security Hash Algorithm SHA-256 is a kind of cryptographic hash function that is designed by the NSA (U.S. National Security Agency)[8].

The most widely used hash functions are:

- MD2, MD4 and MD5 (Message Digest).
  – produce 128-bit digests.
  – final searches announced some weaknesses with these types.
- SHA-1 (Security Hash Algorithm1).
  – produce 160-bit digests.
- SHA-2 family (Security Hash Algorithm2).
  – Many Types: SHA-224, SHA-256, SHA-384 and SHA-512.
  – These produce digests of sizes 224, 256, 384 and 512 bits respectively.
- SHA-3 (Security Hash Algorithm 3).
  – it is very different and more secure than SHA-1 and SHA-2[3].

Finally, For LabVIEW (Laboratory Virtual Instrument Engineering Workbench) is an integrated development environment designed specifically for engineers and scientists. The LabVIEW is a graphical programming language (G) that uses a dataflow model, which make it different from other programming languages use sequential lines of textcodes that empower the user to write functional code using a visual layout that resembles your thought process. The VI (virtual instrument) is the

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

basic LabVIEW element. A VI has three main parts as shown in figure (1)
A. Front Panel: Interacts with the user.
B. Block Diagram: Contains the code.
C. Connector Pane: Method of connecting to other VI's [4].

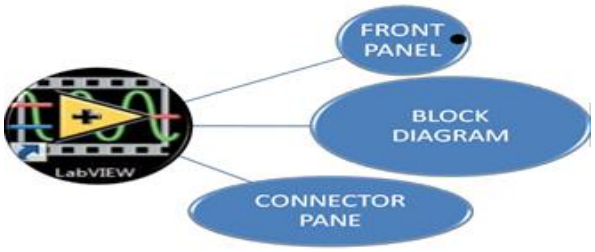

Figure (1): Main parts of LabVIEW

II. ADVANCED ENCRYPTION STANDARD (AES-256)

The AES algorithm is a block cipher algorithm, has s a fixed block size length of 128 bitswith three different keylengths of 128, 192, and 256 bits to encrypt and decrypt data in blocks of bits. The 128 bits means that the block that the AES operates on is a 4x4 square of bytes [5].

The AES parameters depend on the key length as shown in Table (1) below.

Table(1): The AES Parameters

Key size N_k (bits)	Plaintext block size (bits)	Number of rounds Nr (bits)	Round key size (bits)	Expanded key size (bits)
128	128	10	128	176
192	128	12	128	208
256	128	14	128	240

In this paper description, the key length of 256-bit is assumed, which is mainly used in top secret military applications to ensure the maximum level of security, but it costs many resourcesbecause it requires higher area implementations and longer processing time, while key sizes of 128-bit is sufficient for most of the commercial applications and 192-bit is also used in top secret military applications but it lower costs than 256-bit [5, 6].

The AES Encryption algorithm uses a round function that is composed of four different byte-oriented transformations: Byte substitution, Shifting rows, Mixing Column and Adding a Round Key [5].

And for The AES decryption algorithm uses a round function that is composed of four different byte-oriented transformations: Inverse Byte substitution, Inverse Shifting rows, Inverse Mixing Column and Adding a Round Key[5].As shown in figure (2).

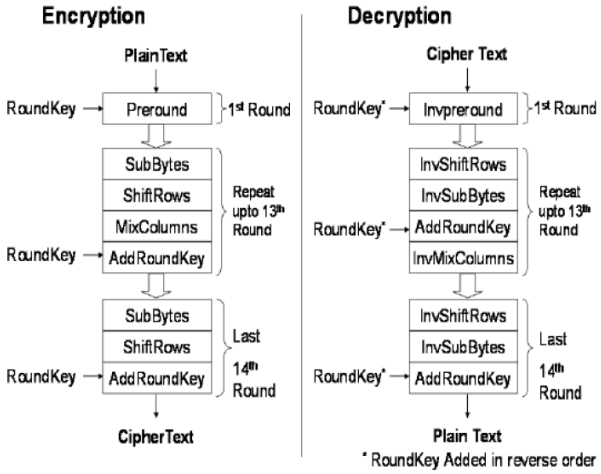

Figure (2): The Encryption and Decryption Rounds for AES-256

A. The Byte substitution and Inverse Byte substitution.
In Encryption process, the SubBytes step, each byte in the state matrix is replaced with a SubByte using an 8-bit data from the Rijndael S-Box. And for Decryption Process, in the Inverse SubBytes step, every byte in the cipher array is replaced with corresponding Inverse SubByte value.The SubByte operation will provide the non-linearity in the cipher. The S- Box used is derived from the multiplicative inverse over Galois Field (2^8),And for Decryption Process, the Inverse SubBytes arethe same routine as SubBytes, but uses the inverse S-Box[15].

B.The Shifting Rows and Inverse Shifting Rows.
In the encryption, the transformation ofShift Rows is a simple shifting transformation.The transformationis as following:
– 1^{st} row is unchanged.
– 2^{nd} row is shifted by one byte to the left.
– 3^{rd} row is shifted by two bytes to the left.
– 4^{th} row is shifted by three bytes to the left.

And for Decryption Process, the Inverse ShiftRows transformation is as following:
– 1^{st} row is unchanged.
– 2^{nd} row is shifted by one byte to the right.
– 3^{rd} row is shifted by two bytes to the right.
– 4^{th} row is shifted by three bytes to the right[5, 6].

C. The Mixing Column and Inverse Mixing Column.
In Encryption process, The MixColumns transformation operates on the State column-by-column, treating each column as a four-term polynomial. Each column is processed separately and each byte of a column is replaced by a value dependent on all 4 bytes in that column. Effectively a matrix multiplication in $GF(2^8)$ will use the prime poly $m(x) = x^8+x^4+x^3+x+1$

And in Decryption Process, The inverse MixColumn is performed in the same way of MixColumn but with different values in the array.

DOI 10.17148/IJARCCE.2017.6169

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

(Hint: In the last round of AES algorithm encryption and decryption there is no MixColomn transformation [2, 5].

D. Adding a Round Key.

In Encryption process, the bitwise exclusive-or (XOR) operation is performed between outputs from Mix Column and Round Key. The output of this round will be sent to the next round. Each round key consists of Nb words that are generated from the KeyExpansion routine. By inverting the encryption structure one can easily derive the decryption structure. However, the sequence of the transformations will be different from that in encryption [2, 6].

- W_0 , W_1 , W_2 and W_3 are Cipher Key
- W(i) = W(i-4) XOR g(W(i-3))
- W(i+1) = W(i-3) XOR W(i)
- W(i+2) = W(i-2) XOR W(i+1)
- W(i+3) = W(i-1) XOR W(i+2)
- Note: i should be a multiple of 4.
- W(i), W(i+1), W(i+2) and W(i+3) are collated together to generate the key for the next round. The function g() is clearly used only for generating W(i).

And for In the Decryption Process, There is no need of Inverse AddRoundKey in the decryption since XOR operation is inverse of itself[5, 6].Figure (3) shows the details of an AES round.

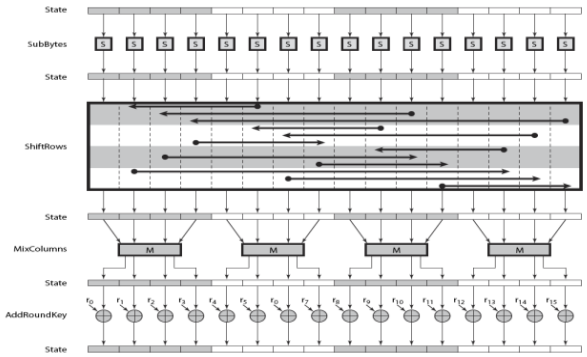

Figure (3): Details of an AES round.

III. THE SECURE HASH ALGORITHM SHA-2 256

SHA-256 can accept messages with arbitrary lengths up to 2^64-bit. The Hash computation produces a final digest message of 256- bits that depended on the input message, composed by multiple blocks of 512-bit each. Then this input block will be expanded and fedto the 64 cycles of the SHA-256 function in 32-bit words.

The SHA-256 algorithm basically consists of three stages: Message Padding and parsing, Message Expansion and Message Compression [9], SHA-256 works as follows:

At start the message is converted to its equivalent binary form, then it is padded with a one binary '1' and then padded with binary '0's until the block length ≡ 448 mod 512. Then the original message length (as a 64-bit binary number) will be added to the end of the padded message [16].

After that, the registers a, b, c, d, e, f, g and h are set to eight intermediate hash values (32-bit constants $H_0$to H_7) for the first message block, and to the intermediate hash value for the following blocks. Then, the 64 iterations of the compression function are completed, given by:

$T_1 = h + \Sigma_1(e) + Ch(e,f,g) + K_t + W_t$;

$T_2 = \Sigma_0(a) + Maj(a,b,c)$;

h = g; g = f; f = e;

$e = d + T_{1;}$

d = c; c = b; b = a;

$a = T_1 + T_{2;}$

Where:

Ch, Maj, Σ_0, and Σ_1 functions are independent of the round number, The Ch, Maj, Σ0, and Σ1 functions operate on 32-bit input words, and produce the 32-bit words given by

$Ch(X, Y, Z) = (X \wedge Y) \oplus (\neg X \wedge Z)$;

$Maj(X,Y,Z) = (X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$;

Σ0(X)= ROTR2(X) ⊕ROTR13(X) ⊕ROTR22(X);

Σ1(X)= ROTR6(X) ⊕ROTR11(X) ⊕ROTR25(X);

Where K_t and W_t are a constant and the message word which value depends upon the round number (t) [16].

Finally, the output of the registers is added to the previous intermediate hash value to give the new one. As shown in Figure (4-b) for an output of one round of the compression function.

The process of 'message scheduling' will take the original 512-bit message block as input and expands these 16 32-bit words into 64 words, one for every round of the compression function. This is done according to the following recurrence formula:

$W_t = \sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16}$;

As shown in figure (4-a),where the σ0 and σ1 functions operate on 32-bit input words, and produce the 32-bit words given by:

σ0 (X)= ROTR7(X) ⊕ROTR18(X) ⊕SHR3(X)

σ1 (X)= ROTR17(X) ⊕ROTR19(X) ⊕SHR10(X)

After all consecutive 512-bit message blocks have been hashed, the last intermediate hash value is the final overall hash value [16].

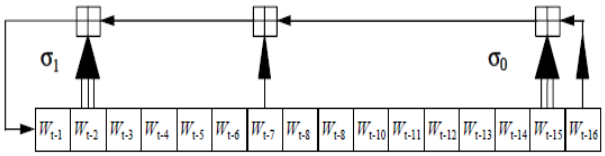

Figure (4-a): (W^t) message scheduled recurrence

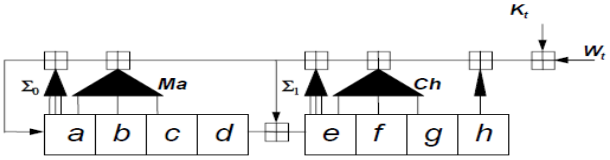

Figure (4-b): Hash computation, state register update function

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

IV. PROPOSED IMPLEMENTATION

The SHA-256 and AES-256 are implemented by using LabVIEW 2013, first the original message is processed via SHA-256 algorithm, then the code generated by SHA-256 is processed via AES-256 algorithm to give a very secure code (as a key for AES) that cannot be breakable easily.

A. The SHA-256 with LabVIEW2013.

For the secure hash algorithm SHA-256 stepsin LabVIEW2013;we will show the data flows from left to right. We take the string input from the Message textbox, convert it into a byte array, apply the SHA-256 hash to it and then output it into the Hash control. As shown in figure (5-a).

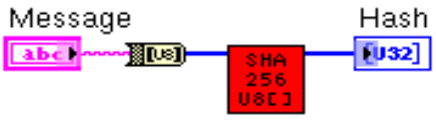

Figure (5-a): Block diagram of SHA-256

The red box "SHA 256 U8 []" contains the hash code,it accepts a byte array and outputs a32-bit array of unsigned integers.
In figure (5-b) shows the contents of "SHA 256 U8 []" block, where the prepared message is applied to SHA-256 block to function it.

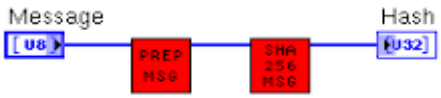

Figure (5-b): Block diagram of SHA-256

1) Message Preparation

The figure (5-c) shows the source code to the message preparation block. The first step is to add a single binary bit‘1’ to the message, by appending a 0x80 byte to the message array. Next step is to pad ‘0’s to the message with 0x00 bytes until the message block length= 448 mod 512. The working is continuous with bytes, so we will take the message length modulo 64 and then add enough padding bytes in order to make the message length =56 modulo 64. Finally, the length of the message will be appendedto a 64-bit unsigned integer andthe byte array is converted to an unsigned 32-bit integer array.

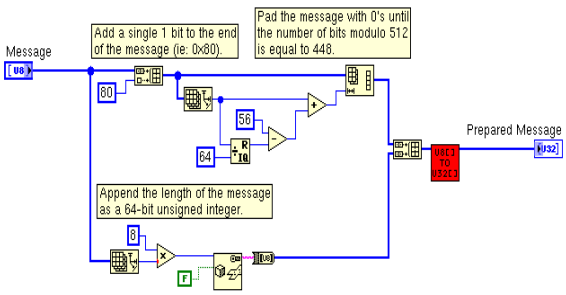

Figure (5-c): Message Preparation

2) 512-bit Chunk Hashing

In this step,we apply the hash function to the message in 512 bit chunks, which is simple loop works on 512 bit (16 x 32-bit) pieces of the message at a time. All the up and down arrows on the edge of the loop construct to indicate that the output of the previous round is used as the input to the next round. The whole SHA-256 compression function is done in the "SHA 256 BLK.".
As shown in figure (5-d)

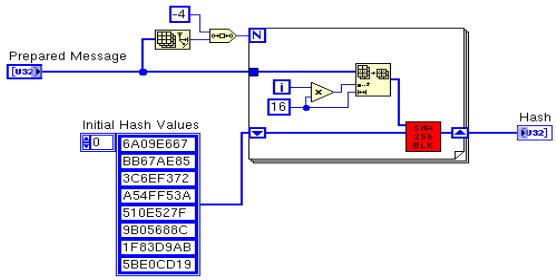

Figure (5-d): SHA-256 Hash Function and block contents

3) Block Hashing

Each round of the loop will execute the following SHA-256 hash block function with the output of one round used as the input to the next. This is the most complicated part of SHA-256.
At first, an array of 64 32-bit unsigned integers is declared, where the message block will be expanded in this working register with a variety of XOR functions, right shift and right rotate operations. The result of this will be the 512-bit block expanded into a 2048 bit space. As shown in figure (5-e).

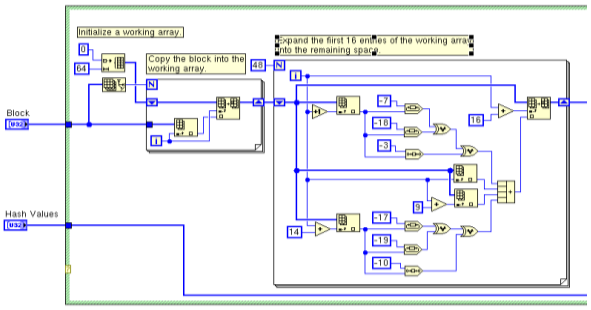

Figure (5-e): Initialize the SHA-256 Block

4) Round Constants

SHA-256 algorithm uses a group of round constants. These are numbers chosen to exhibit certain properties to make it difficult to architect an input to produce a desired output. An example of an invisible motive would be selecting constants that open a back door to the designer of the hashing algorithm. The SHA-256 series of hash functions that use square and cube roots of different small prime numbers as round constants.

5) Compression Loop

The next stage of the block hashing algorithm is to loop through the array that produced in the previous stage and

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

apply the round constants along with a variety of XOR and rotate right operations. As shown in figure (5-f).

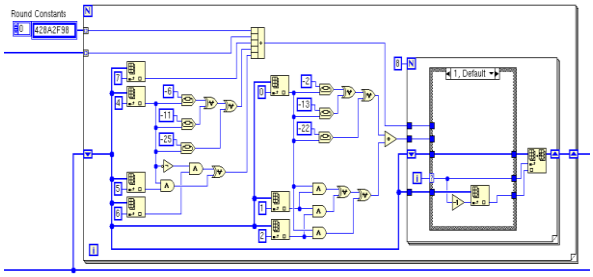

Figure (5-f): Compression Loop

6) Finalize Hash Values

At the last of SHA-256 operation is adding the results obtained from the compression loop to the previous round of hash values. Some implementations use a, b, c, d, e, f, g and h variables. As shown in figure (5-g).

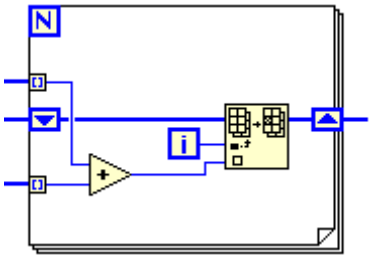

Figure (5-g): Finalize Hash Values

B. The AES-256 with LabVIEW2013.

The AES-256 steps in LabVIEW2013 are:

1- The start point of the code is the key generation that occurs on the server. Here we provide a secret key which then is expanded by using Rijndael's key schedule. The short key is expanded into a larger one. The 256-bit key is transformed into a 240-byte key. The key schedule will take a 4-byte subset of the key as a number of 32 bytes (256-bits) and an iteration count and send this data to the key schedule core, which returns 32 bytes (256-bits).As shown in figure(6-a)

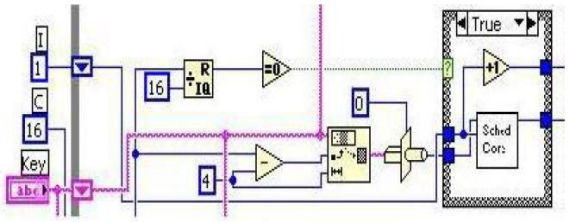

Figure (6-a): Key Generation in LabVIEW

(a) The first step for the key schedule core is performing a byte rotation on the key and splits the 32 bytes (256-bits) into respective bytes to send them later to have Rijndael S-box applied. This is used to apart the relationship between the key and thecipher text. As shown in figure (6-b).

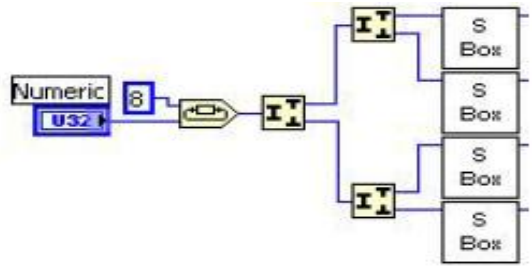

Figure (6-b): Block Implementation

(b) The next step in the core scheduler is the Rcon step. This takes the first byte of the output word (from the S-box) and performs an XOR on the byte with the result of the Rcon step, which is essentially an exponentiation of 2 to the iteration.As shown in figure(6-c).

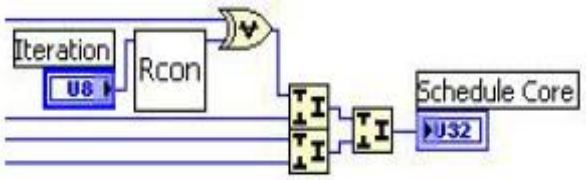

Figure (6-c): Modulo Operation in S Block Implementation

1- After the key has been generated, information about the key, the type of encryption is performed, as well as the key itself, needs to be transmitting to the scheduler. The key is transferred into a byte array, and then sent one byte at a time along with an interrupt to ensure the scheduler receives all the data in the correct order.
2- Then the server will send basic commands to the scheduler stepping the data through both the encryption and decryption processes before returning the data to the user.
3- The key is stored in a look-up-table (LUT) in order the server be able to perform the cryptanalysis. The LUT can be used to store different and many keys, which provide user adaptability.For this, we use a VI methodology. In spite that there are multiple cryptographic algorithms out there.
4- Once the key information has been provided from the server; the encryption can take place as soon as the host gives the signal. In the firststep of encryption, a round key is generated,which is derived from the key schedule. Then the iterations process will begin to perform the actual encryption.
5- Inthe first round of the encryption algorithm, each byte of the state array will be combined with the round key using a bitwise XOR. The bulk of the rounds will perform a non-linear substitution which replaces each byte with another byte according to a predefined LUT. Then each row of the state will be shifted cyclically.
6- After that a mixing operation will take place on the columns and combine the four bytes of each column, and then the round key will be added again. The final round is similar to the previous 'n'rounds, without the column mixing operation.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

7- The key expansion operation is used to produce the roundkeysfrom the cipherkey. The AES standard defines the key expansion operations on four byte words. A subkey is poised of four such words. The key expansion for AES-256 steps are:
- The first subkey is the cipher key itself.
- The next words are calculated repeatedly from the initial set of words using a simple XORfunction.
- In the AES standard there are operations are named RotWord and SubWord. In which a multiple a special transformation is used on the words. At start, the first byte ordering is changed by cyclic left shift, and then the SubBytes function is applied to all four bytes.

V. RESULTS

In the proposed implementation, As shown in figure (7),an input of arbitrary length is given to the SHA-2 module. A message digest of fixed length is generated which is 256-bits. This message digest is used in the encryption and decryption process as a key. As shown in the figure (7), after generating the message digest, it is given as a key for the encryption of the plain text which in turn generates a cipher text. Later by making use of same key, decryption is performed to retain back the original plain text. AES-256 itself is a strong security mechanism. Since SHA-256 is being used here along with the AES-256, this design ensures higher security since complexity of the design increases. Here the security is given in terms of complexity.

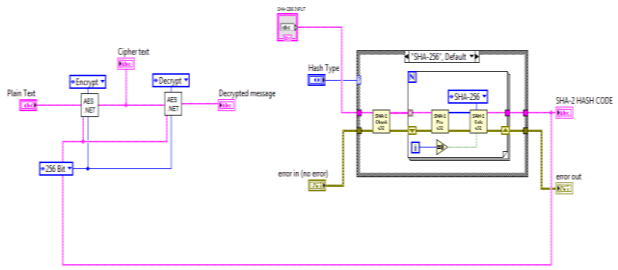

Figure (7): The Proposed model of AES-256 and SHA-256

We made two test cases as shown below:

FIRST CASE: The plaintext to the AES-256 is the same as the input message to SHA-256.

Plaintext to AES-256 ="gaziantep city*"
Cipher Text=3EF7 9267 D665 158E E946 1A10 6E98 37D1 A2CE A393 E872 2116 9EF3 E95F 62C2 A419
Decrypt message="gaziantep city*"
SHA-256Input ="gaziantep city*"
SHA-2Hash code=2FED 0DC5 6D5F E48D B23F EA45 77AD 43DF 62F3 D82E C1B3 58D7 31BA 00CC E013 9F23
I/P KEY=HASH CODE=
2FED 0DC5 6D5F E48D B23F EA45 77AD 43DF 62F3 D82E C1B3 58D7 31BA 00CC E013 9F23
As shown in figure (8-a)

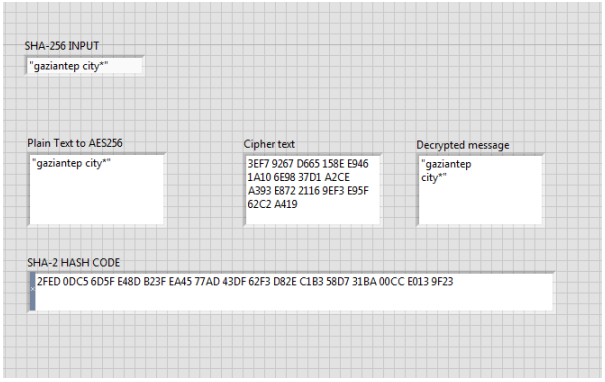

Figure (8-a): Test results of the first case.

SECOND CASE: The plaintext to the AES-256 is not the same as the input message to SHA-256.

Plaintext to AES-256 ="gaziantep city*"
Cipher Text=3649 042F 8078 BA99 49A1 9F13 ED57 7030 DDA7
1BED 8CF2 A0C7 E184 1E59 101C 2458
Decrypt message="gaziantep city*"
SHA-256Input =GOOD MORNING
SHA-2Hash code=E640 7EA3 71FA 0501 8A9B 0F00 60B5 83F1
A953 2873 A109 B1DF B31A C825 F291 8D95
I/P KEY=HASH CODE=
E640 7EA3 71FA 0501 8A9B 0F00 60B5 83F1 A953 2873 A109 B1DF B31A C825 F291 8D95
As shown in figure (8-b)

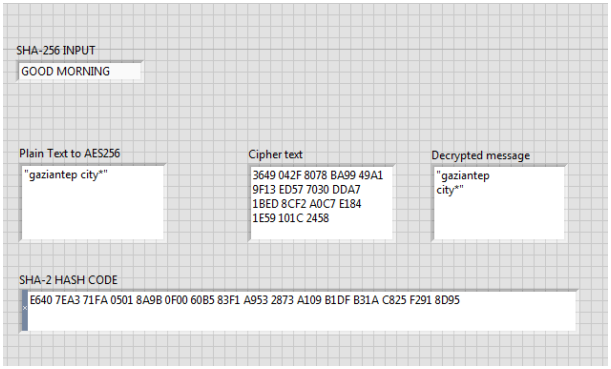

Figure (8-b): Test results of the second case.

VI. CONCLUSION

1- Regardless of the user's programming experience, LabVIEW makes user development fast and easy in the scientific community. So, by implementing the proposed hybrid cryptosystem which consists of AES-256 and SHA-256 algorithms, we notice the high speed of showing the output results and the flexibility in controlling the data flow through the cryptosystem.

2- From the test results, we notice that by changing the input message of SHA-256, the SHA-256 hash code (the key of AES-256) and the cipher text (the output of AES-256 encryption algorithm) will be changed,but the decrypted message (the output of AES-256 decryption algorithm) will keep the same.

IJARCCE
ISSN (Online) 2278-1021
ISSN (Print) 2319 5940
International Journal of Advanced Research in Computer and Communication Engineering
ISO 3297:2007 Certified
Vol. 6, Issue 1, January 2017

3- From the results obtained in the two cases, we noticed the ability of using uppercase and lowercase letters for the input messages for SHA-256.

4- By integrating the hashing algorithm SHA-256and the cryptography algorithm AES-256 in the same implementing model; we developed a very difficult hybrid cryptosystem that cannot break and achieved a higher data security in terms of complexity.

REFERENCES

[1] Jing Wang, Xiaoyang Zeng, Jun Chen, "A VLSI implementation of ECC combined with AES" 1-4244-0161 5/06/$20.00 ©2006 IEEE.
[2] William Stallings, "Cryptography and Network Security Principles and Practices, Fifth Edition", Prentice Hall, 2011.
[3] Dr.Geoff Hamilton, "CA642: CRYPTOGRAPHY AND NUMBER THEORY" "
[4] Hans-Petter Halvorsen ," LabVIEW Programming"
[5] FIPS 197, "Advanced Encryption Standard (AES)", November 26, 2001.
[6] Seagate–Technology Paper, "128 Bit Versus 256 Bit AES Encryption", Practical Business Reasons Why 128 bit Solution Provide Comprehensive Security for Every Need.
[7] M. Juliato,. C. Gebotys and R. Elbaz, "Efficient Fault Tolerant SHA-2 Hash Functions for Space Applications,"2009.
[8] M. K. R. Danda, DESIGN AND ANALYSIS OF HASH FUNCTIONS, 2007.
[9] R. P. McEvoy, F. M. Crowe, C. C. Murphy and W. P. Marnane, "Optimisation of the SHA-2 Family of Hash Functions on FPGAs," Emerging VLSI Technologies and Architectures, IEEE Computer Society Annual Symposium,2006.
[10] Ryan Glabb, Laurent Imbert and Graham Jullien, "Multi-mode operator for SHA-2 hash functions," Journal of Systems Architecture ,ELSEVIER, pp. 127-138, 2007.
[11] L. Selvakumar and C. Ganandhas, "The Evaluation Report of SHA-256 Crypt Analysis Hash Function," International Conference on Communication Software and Networks, IEEE Computer Society, pp. 588-592, 2009
[12] H. Mestiri, F. Kahri, B. Bouallegue and M. Machhout, "Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2," I.J. Computer Network and Information Security ,Published Online in MECS (http://www.mecs-press.org/), pp. 9-15, 2015.
[13] FIPS PUB 180-2, USA: NIST, 2002.
[14] FIPS PUB 202, SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," NIST, Information Technology Laboratory National Institute of Standards and Technology , 2002.
[15] P.V.S. Shastry, A. Agnihotri, D. Kachhwaha, J. Singh and M.S.Sutaone, "A Combinational Logic Implementation of S-Box of AES," IEEE 54th Int. Midwest Symp on Circuits and Systems (MWSCAS), Aug. 2011, pp. 1-4.
[16] Dr. Helena Handschuh, Dr. Henri Gilbert, "Security Level of Cryptography -SHA-256", Issy-les-Moulineaux 31 January 2002.