International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Hierarchical Dynamic Key Generation and Selective Transformation Model for Optimized AES Block Cipher Core

Mahesh Chittim MV[1], DN Kuldeep Shamgar S[2]
M. Tech Student[1], Assistant Professor[2]
Department of Embedded Systems and VLSI Design, Sri Krishna Devaraya University College of Engineering, Anantapur, India

Article Info
Volume 9, Issue 5
Page Number : 301-308

Publication Issue
September-October-2022

Article History
Accepted : 01 Oct 2022
Published : 09 Oct 2022

ABSTRACT

Symmetric key cryptography, Hash functions and public key cryptography. Symmetric key algorithms namely Advanced Encryption Standard (AES), and Data Encryption Standard use the same key for encryption and decryption. This research examined a brand-new rapid picture key extraction from image. To create a key stream with outstanding statistical properties from picture have to use image processing. An effective and safe image based key AES S- is presented in this research. First, an image has to be selected after that by using image. The proposed architecture includes 8-bit data path and five main blocks. We design two specified register banks, Key-Register and State-Register, for storing the plain text, keys, and intermediate data. In order to increase security, the proposed Bio-Metric 256-bit AES method is heavily used for key management. A FPGA implementation therefore reduces power as a result of this. The suggested implementation's proposed throughput (Mbps) using Virtex-7 (xc7vx485tffg1157) FPGA improved.

Keywords : AES (Advanced Encryption Standard), FPGA (field programmable gate array)

I. INTRODUCTION

Data and information communication has become very important ingredient of today's technological life and considered as significant assets of an individual or organization. If the confidentiality of information is compromised, then the information can be used for harmful purposes. Current innovations in information technology and their prolific applications in our life have caused in a gigantic growth in the size of the data being transmitted online. The private information being very sensitive assets require protection from attackers

On December 2001, The Advance Encryption Standard was published by the National Institute of Standards and Technology (NIST) (AES), a symmetric cryptographic block. The encryption and decryption of a fixed data block of 128 bits is performed by this non-Feistel block cypher. Three distinct key lengths exist. For 128-bit keys, 12 processing rounds are used, for 192-bit keys, 14 processing rounds are used, and for 256-bit keys, 14 processing rounds are used.

Ordinary plain text is transformed into unintelligible Using cryptography, you can transfer data between text as well as vice versa. Hash functions, public key cryptography, as well as symmetric key cryptography are the three distinct types of cryptographic techniques. Same key is used for encryption and decryption in the Advanced Encryption Standard (AES) and Data Encryption Standard. It uses fewer processing resources, is simpler to use, and is noticeably sharper.

Data encryption standard (DES)

It has been found vulnerable to very powerful attacks and therefore, the popularity of DES has been found slightly on the decline. Since DES is an encryption algorithm, it encrypts in 64 bits each. As a result, DES receives 64 bits of clear text as input and outputs 64 bits of cipher - text. Despite a few minor variations, the same algorithm and key are utilized for encryption as well as decryption. The key length is 56 bits. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key. DES is based on the two fundamental attributes of cryptography: substitution (also called confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

1. In the beginning step, a 64-bit simple text fragment is supplied to an initial permutation (IP) algorithm.
2. Text in plain text is used for the initial permutation.

   3. The two sides of the permuted blocks (RPT) are created by the initial permutations (IP), Left Plain Text (LPT), and Right Plain Text (RPT).

   4. For each LPT and RPT, the encryption process now completes 16 cycles.

   5. The united block is then put in via a Final Permutation when LPT and RPT are properly re-joined (FP).

   6.This procedure generates 64-bit ciphertext as the end result.

Cryptography is the science of secret, or hidden writing.

It has two main Components:

- Encryption
  Practice of hiding messages so that they cannot be read by anyone other than the intended recipient
- Authentication & Integrity
  Ensuring that users of data/resources are the persons they claim to be and that a message has not been surreptitiously altered

Requirements of secure communication

- Secrecy
  Only intended receiver understands the message
- Authentication
  Sender and receiver need to confirm each other's identity
- Message Integrity

Ensure that their communication has not been altered, either maliciously or by accident during transmission

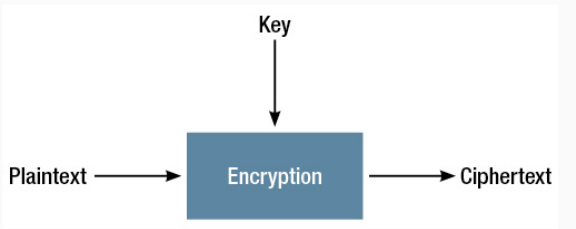


Fig 1: Block diagram of Encryption

Data confidentiality is protected by encryption, one of many security controls. Data that needs to be protected (plaintext) is transformed mathematically into a form that is difficult for machines or unauthorized individuals to understand (ciphertext).

The plaintext is random-looking and contains no information about the content of the original data after being converted into ciphertext. After being encrypted, the data cannot be read by a person (or machine) in a way that would reveal anything about the original data's content.

A bidirectional conversion is encryption. It only serves a purpose when it is possible to convert encrypted data (ciphertext) back to its old, unencrypted state (plaintext). The encrypted data are regarded as unreadable and useless if the encryption is not reversible. Decryption is the procedure of going backwards. Decryption is the method of restoring encrypted data (also known as ciphertext) towards its original state after encryption (plaintext).

An appropriate cryptographic key is necessary for each encryption as well as decryption operation. A series of binary digits is referred to as a cryptographic key and serves as the input for encryption and decryption processes. The encryption as well as decryption functions must employ the exact same cryptographic key in order for the encryption function to convert the plaintext into ciphertext and the decryption method to convert the ciphertext back to its original form. A symmetric key is what this is. The Advanced Encryption Standard's encryption functionalities are extensively supported by modern hardware as well as software.

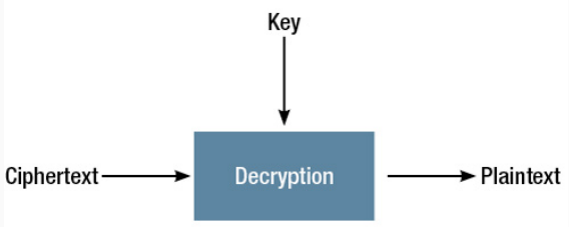


Fig 2: Block Diagram of Decryption

Decryption is the process of reversing the encryption of data. The process of encryption is typically reversed. Since decryption needs a secret key or password, it decodes the encrypted messages ensuring that only a permitted user can do so.

Confidentiality is a factor in the implementation of an encryption-decryption system. It's important to check the access from illegal organisations or people as information goes through the Internet. The data is encrypted as a result to lessen theft and data loss. Text files, photos, emails, user data, and directories are some examples of often encrypted stuff.

The person who receives decryption gets a prompt or window where they can input a code to access the encrypted data. In order to decipher the data, the method extracts and turns it into phrases and visuals that may be easily understood by both a reader and a system. Either manually or automatically decrypting data is feasible. It might also be accomplished by using a password or a set of keys.

The Hill cypher combines encryption and decryption is one of the most widely used methods of conventional cryptography. Most significant and well-liked techniques because it produces a random matrix and fundamentally gives security power. The Hill cipher's matrix must be inverted in order to be decrypted. Inverse of the matrix is not always present, which creates a challenge during decryption. Decrypting the encrypted content is impossible if the matrix is not invertible. That is the main drawback of this. In order to overcome the disadvantages, AES has been invented.

II. EXISTING METHOD

The existing 32-bit AES implementation. We are doing operations per word (32-bits) in each cycle. Number of blocks required for conventional (128 bit) and existing (32-bit) implementation are as follows

1) S box - 16, 4 per clock cycle
2) Mix column block - 4, 1 per clock cycle

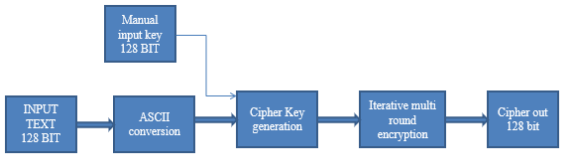


Fig 3: Existing AES functional block diagram

In the existing implementation, we are using the same Sbox hardware with regard to both encryption and decryption. Inverse S-box and S-box (encryption) are identical except for the affine transform (decryption). The Substitution-box and inverse S-box have the same similar encryption and decryption algorithms. Consequently, we are recycling every logic other than the Affine transform for encryption and decryption. Fig.3 shows the mux selection between S-box and inverse S-box. While doing AES encryption S-box path is chosen and while doing AES decryption inverse S-box path is chosen.

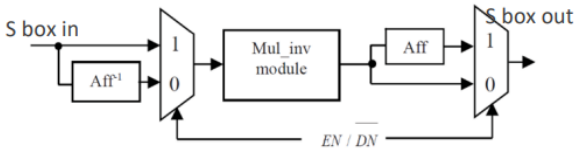


Fig.4 Combined structure of S box and Inverse S box

S box	Shift	Mix	Cycle
		Mix_0	1
Sub_0	-	Mix_1	2
Sub_1	Shift_0	Mix_2	3
Sub_2	Shift_1	Mix_3	4
Sub_3	Shift_2	-	5
Key_2	Shift_3	Mix_0	6
Sub_0	-	Mix_1	7
Sub_1	Shift_0	Mix_2	8
Sub_2	Shift_1	Mix_3	9
Sub_3	Shift_2	-	10
Key_3	Shift_3	Mix_0	11
Sub_0	-	Mix_1	12
Sub_1	Shift_0	Mix_2	13
Sub_2	Shift_1	Mix_3	14
Sub_3	Shift_2	-	15
-	Shift_3	Mix_0	16
.	.	Mix_1	17
Key_14	.	Mix_2	18
Sub_0	-	Mix_3	19
Sub_1	Shift_0	.	.
Sub_2	Shift_1	.	.
Sub_3	Shift_2	-	.
-	Shift_3	Mix_0	71
		Mix_1	72
		Mix_2	73
		Mix_3	74

In existing 32-bit operation method, we are reusing S-box and Mix Column blocks. In existing design Mix Column and Add Round Key We referred to the group as Mix block.

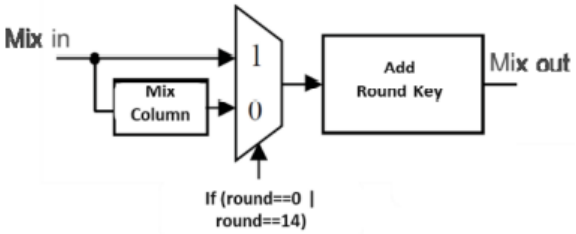


Fig.5 Mix Column as well as Add Round Key Structure Combined - Mix

Pipelined structure of existing method

the pipeline structure of the existing design, where each color represents different round as follows, Mix – round 0

Mix – round 1

Mix – round 2

Mix – round 3 and Mix – round 14

Each word having a size of 32 bits. In cycle 1, we are doing Mix operation of word 0 (mix_0). We can denote this as cycle1 [round0 (mix_0)].

Hence this 32-bit word is available to undergo 32- bit Sub operation. Hence in cycle 2 we are doing sub operation of word 0 (sub_0) and Mix operation of word 1 (mix_1). We have valid input for sub block word 0 in clock cycle 2, and hence we don't need to wait for all the 4 words mix block operation to complete. We can denote this as cycle2[round1(sub_0), round0(mix_1)].

In clock cycle 3, we are doing sub operation for word 1 (sub_1), shift operation of word 0 (shift_0) and mix operation of word 2 (mix_2), and. We can denote this as cycle3[round1 (sub_1, shift_0), round0(mix_2)].

In clock cycle 4, we are doing sub operation for word 2 (sub_2) and shift operation for word 1 (shift_1) and mix operation for word 3 (mix_3). We can denote this as cycle4[round1(sub_2, shift_1), round0(mix_3)].

Since all 128-bit (4 words) round 0 mix operation completed, we don't have mix operation in cycle 5. In clock cycle 5, we are doing sub operation for word 3 (sub_3) and shift operation for word 2 (shift_2). We can denote this as cycle5[round1(sub_3, shift_2)].

Since all 128-bit (4 words) round 0 sub operation completed, we don't have sub operation in cycle 6. In

clock cycle 6, we are doing round 1 shift operation of word 3 (shift_3) and mix operation of word 0 (mix_0) and. Since we already have the last byte value from sub_3, we are using that for mix_0. In this way we don't need to wait extra one cycle of shift operation to start the mix operation. We can denote this as cycle6[round1(shift_3, mix_0)].

In clock cycle 7, we are doing sub operation for word 0 (sub_0) and mix operation for word 1 (mix_1). We can denote this as cycle7[round2(sub_0), round1(mix_1)].

In clock cycle 8, we are doing sub operation for word 1 (sub_1) shift operation for word 0 (shift_0) and mix operation for word 2 (mix_2). We can denote this as cycle8[round2(sub_1, shift_0), round1(mix_2)].

In clock cycle 9, we are doing sub operation for word 2 (sub_2), shift operation for word 1 (shift_1) and mix operation for word 3 (mix_3). We can denote this as cycle9[round2 (sub_2, shift_1), round1 (mix_3)].

In clock cycle 10, we are doing sub operation for word 3 (sub_3) shift operation for word 2 (shift_2). The same order of execution repeats for all 14 rounds. We can denote this as cycle10[round2(sub_3, shift_2)]. The same sequence repeats for all 14 rounds.

In cycle 6 we are using sub bytes S box for key generation block, because of this we don't need extra S box for key generation block. We are generating 128-bit key for every 5 cycles, so that it requires only 4 S box in one cycle. In conventional method, we need 8 S box for key generation block. 128-bit key generated in cycle 6 will be used in cycle 14 mix operation. Similarly, key generated in cycle 11 used in mix operation of cycle 19.

In cycle 9 we have valid output for round 1 (cycle 5 to 9), so we need 5 cycles to perform round 1. Total we need 74 clock cycles to complete AES encryption.

III. PROPOSED METHOD

At a high level, the evaluation of a BKG requires designers to show that two properties hold: correctness and security. Intuitively, a scheme that achieves correctness is one that is usable for a high percentage of the population. That is, the biometric of choice can be reliably extracted to within some threshold of tolerance, and when combined with the template the correct key is output with high probability. As correctness is well understood, and is always presented when discussing the feasibility of a proposed BKG, we do not address it further.

In the context of biometric key generation, security is not as easily defined as correctness. Loosely speaking, a secure BKG outputs a key that "looks random" to any adversary that cannot guess the biometric. In addition, the templates and keys derived by the BKG should not leak any information about the biometric that was used to create them. We enumerate a set of three security requirements for biometric key generators, and examine the components that should be analysed mathematically (i.e., the template and key) and empirically (i.e., the biometric and auxiliary information). While the necessity of the first two requirements has been understood to some degree, we will highlight and analyse how previous evaluations of these requirements are lacking. Additionally, we discuss a requirement that is often overlooked in the practical literature, but one which we believe is necessary for a secure and practical BKG.

We consider a BKG secure if it meets the following three requirements for each enrollable user in a population:

- Key Randomness: The keys output by a BKG appear random to any adversary who has access to auxiliary information and the template used to derive the key. For instance, we might require that the key be statistically or computationally indistinguishable from random.
- Weak Biometric *Privacy*: An adversary learns no useful information about a biometric given

auxiliary information and the template used to derive the key. For instance, no computationally bounded adversary should be able to compute any function of the biometric.

- Strong Biometric Privacy: An adversary learns no useful information about a biometric given auxiliary information, the template used to derive the key, and the key itself. For instance, no computationally bounded adversary should be able to compute any function of the biometric

Even more problematic is that many approaches for demonstrating biometric security merely provide some sort of measure of entropy of a biometric (or key) based on variation across a population. For example, one common approach is to compute biometric features for each user in a population, and compute the entropy over the output of these features. However, such analyses are generally lacking on two counts. For one, if the correlation between features is not accounted for, the reported entropy of the scheme being evaluated could be much higher than what an adversary must overcome in practice. Second, such techniques fail to compute entropy as a function of the biometric templates, which we argue should be assumed to be publicly available. Consequently, such calculations would declare a BKG "secure" even if, say, the template leaked information about the derived key. For example, suppose that a BKG uses only one feature and simply quantizes the feature space, outputting as a key the region of the feature space that contains the majority of the measurements of a specific user's feature. The quantization is likely to vary between users, and so the partitioning information would need to be stored in each user's template. Possession of the template thus reduces the set of possible keys, as it defines how the feature space is partitioned.

In our demonstration, we show how the choice of key size Cryptographic algorithms' safety could be considerably compromised by both and the amount of computing cycles. It is imperative that lengthen the encryption key in order to optimize security and level of resistance. Different transformations can also be utilized to reduce the design complexity without sacrificing security. In order to achieve reduced complexity and enhanced unpredictability over the cypher text conversion process, this work uses AES with 256 key size and incorporates various combinations of transformations for each round. A fully automated key generation system based on digital biometrics is developed to solve the key management issue with AES core. This automatic key creation uses hierarchical phases that contain various AES confusion level metrics.

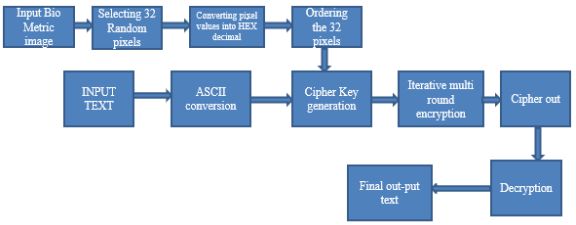


Figure 6: Pipelined structure of proposed method

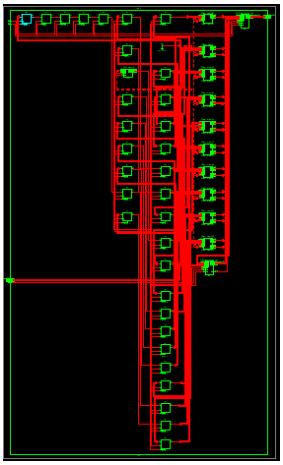
There will be more Pixels accessible in an individual photograph. Therefore, it is quite difficult to determine which pixels we are using as our key. Even if someone can determine the number of pixels and their locations, they still need to understand how the pixels are organized. The KEY's place of genesis is unknown.

IV. RESULTS AND DISCUSSION

The recommended AES algorithm as well as the present AES technique have both undergone functional testing using the Xilinx 14.7 version. The current proposal has a greater area than the current design when compared to it. without sacrificing the other characteristics. While utilizing dynamic key, the proposed solution will reduce area while enhancing architecture performance.

Biometric Image

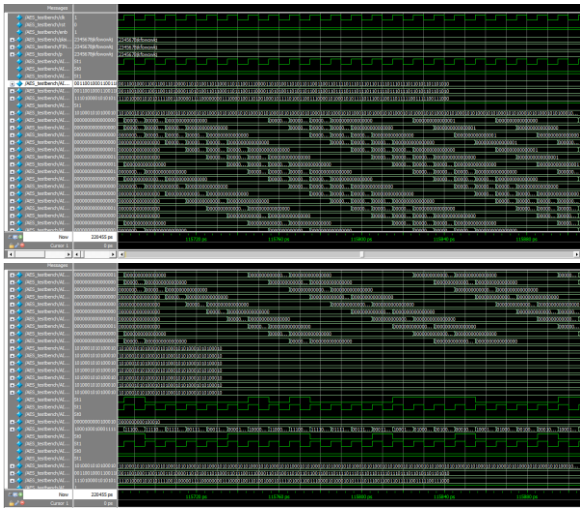
From an above image, the MATLAB software will read the all pixels in the form of digital data and it will be stored in one text file. Among all data we have to select random data for key generation. Once the data is collected, we have to do concatenation of random data in different order.



RTL Schematic

Technology Schematic

Simulation Results:



Evaluation of Area, Delay report:

	Area	Delay
Existing	5716	32.346ns
Proposed	5286	14.377ns

V. CONCLUSION

The AES algorithm with a key derived from a image is presented in this study as a generalized design and practical implementation. The suggested strategy outperformed the static key approach about safety. Utilizing Xilinx 14.7, the functionality of the new AES algorithm and the present AES algorithm were both tested. The aforementioned results show that, in contrast to the current design, the proposed design requires less space and moves more quickly.

VI. REFERENCES

[1]. M. Rajeswara Rao, Dr.R.K.Sharma, SVE Department, NIT Kurushetra FPGA Implementation of combined S box and Inv S box of AES 2017 4th International conference on signal processing and integrated networks (SPIN).

[2]. Nalini C. Iyer ; Deepa ; P.V. Anandmohan ; D.V. Poornaiah Mix/InvMixColumn decomposition and resource sharing in AES.

[3]. Xinmiao Zhang, Student Member, IEEE, and Keshab K. Parhi, Fellow, High Speed VLSI architectures for the AES Algorithm, IEEE. VOL.12. No.9. September 2004

[4]. Shrivathsa Bhargav, larry Chen, abhinandan Majumdar, Shiva Ramudit 128 bit AES Decryption, CSEE 4840 – Embedded system Design spring 2008, Columbia University.

[5]. Atul M. Borkar ; R. V. Kshirsagar ; M. V. Vyawahare FPGA implementation of AES algorithm.

[6]. Announcing the ADVANCED ENCRYPTION STANDARD (AES), November 26 2001.

[7]. Yulin Zhang ; Xinggang Wang; Pipelined implementation of AES encryption based on FPGA 2010 IEEE International Conference on Information Theory and Information Security.

[8]. Yuwen Zhu ; Hongqi Zhang ; Yibao Bao ; Study of the AES Realization Method on the Reconfigurable Hardware 2013 International Conference on Computer Sciences and Applications.

[9]. Tsung-Fu Lin ; Chih-Pin Su ; Chih-Tsun Huang ; Cheng-Wen Wu; A high-throughput low-cost AES cipher chip Proceedings. IEEE AsiaPacific Conference on ASIC.

Cite this article as :

Mahesh Chittim MV, DN Kuldeep Shamgar S, "Hierarchical Dynamic Key Generation and Selective Transformation Model for Optimized AES Block Cipher Core", International Journal of Scientific Research in Science and Technology (IJSRST), Online ISSN : 2395-602X, Print ISSN : 2395-6011, Volume 9 Issue 5, pp. 301-308, September-October 2022.
Journal URL : https://ijsrst.com/IJSRST229558