Section 2

Cryptography

Chapter 7

Secure Communication Using Cryptography and Covert Channel

Tamer S.A. Fatayer

Abstract

The keys which are generated by cryptography algorithms have still been compromised by attackers. So, they extra efforts to enhance security, time consumption and communication overheads. Encryption can achieve confidentiality but cannot achieve integrity. Authentication is needed beside encryption technique to achieve integrity. The client can send data indirectly to the server through a covert channel. The covert channel needs pre-shared information between parties before using the channel. The main challenges of covert channel are security of pre-agreement information and detectability. In this chapter, merging between encryption, authentication, and covert channel leads to a new covert channel satisfying integrity and confidentiality of sending data. This channel is used for secure communication that enables parties to agree on keys that are used for future communication.

Keywords: encryption, authentication, dynamically, covert channel, confidentiality, algorithm, undetectability, fake key

1. Introduction

Encryption is considered the main key factor of security to achieve confidentiality and to protect data from disclosure [1]. Encryption is not efficient to achieve integrity. It needs another factor called authentication [2]. Covert channel is created to transfer data indirectly between client and server; it was created by Lampson [3].

Before the client and server use the covert channel, they must agree on a pre-agreement knowledge. Also, they agree on how to send that knowledge. A good example is they agree on even word meaning "00" and odd word meaning "11." If the client sends "communication channels," the server will know that the client's message is "1100" [4, 5].

Covert channel cannot be detected if the following two factors exist: plausibility and bit distribution. Plausibility deceives the attacker who thinks that the channel is normal channel and it is not used to send secret information. On the other hand, the bit distribution of normal channel must same distribution of covert channel [5].

The technique is worked as shown in **Figure 1**, where it shows the general idea of the proposed technique. Covert channel needs shared information. In my protocol, it is considered as a table shared between the client and server. This table contains characteristics of the client such as the name which represents the original key. Each

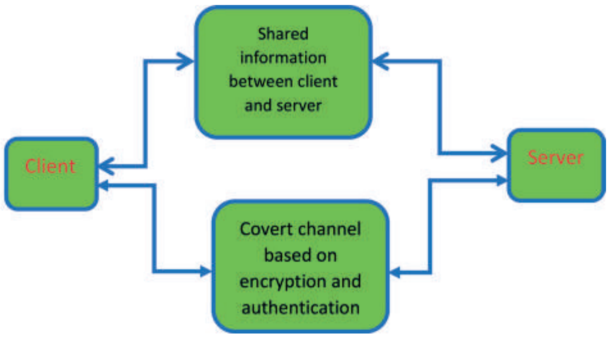

Shared information between client and server
Client
Server
Covert channel based on encryption and authentication

Figure 1.
Secure communication using covert channel, encryption, and authentication.

original key has fake keys. I used encryption algorithm to guarantee the confidentiality. HMAC is used to check integrity. Finally, the time that is needed for the client and server to agree on secret information (e.g., secret keys) is measured.

In this chapter, secure communication channel for transferring data is implemented. The channel between the client and server is considered a covert channel that depends on authentication and encryption.

2. Background

Lampson was the first to introduce the idea of a covert channel [3]. Transferring data between two entities indirectly through a channel is called a covert channel. Before the client and server use the channel to transfer data, they must agree on a pre-agreement knowledge (e.g., shared memory, table). For example, a word containing "mm" means bit "0" other than this means bit "1." So, if the client wants to send "10" to the server indirectly, the client will send "secure communication" to the server. The attacker hardly breaks the covert channel and it is considered to be more secure if it is undetectable [3, 4].

2.1 Covert channel characteristics and properties

Although a covert channel transfers information in a hidden way, it has the same characteristics as other communication channels. These characteristics are:

- Capacity: the amount of data that can be transmitted through the channel. From security viewpoint, increasing channel capacity leads to more information leakage. The covert channel capacity is measured in bits/second. To obtain maximum bandwidth through a covert channel, encoding schemes must be chosen between the sender and receiver.

- Noise: transmitted data through a covert channel are exposed to an amount of perturbations that makes the transmitted and received information between two entities not the same.

- Transmission mode: the transmission of information in covert channels (as in normal channels) can be synchronous or asynchronous. The sender and receiver in synchronous mode should manage their transmission based on a condition or a specific event. On the other hand, in asynchronous mode, the transmission occurs without a prior condition.

2.2 The covert channel is more private and undetectable if it satisfies the following

1. Plausibility: the TCP is usually used for Internet traffic, and it always employs using time stamp option. As a result, TCP using time stamp is a plausible covert channel because the majority of users using TCP will not use it for sending covert data. So, the adversary will believe that TCP time stamps will not be used for sending data covertly.

2. Undetectability: in order for a channel to be more undetectable, the channel must satisfy that the distribution of bits with covert data must be similar to the distribution of the normal channel. If an adversary notices that there are differences (using statistical tests) in bit distribution, then he will detect that the channel is a covert channel. Also, to achieve undetectability, the channel's bits must be random; otherwise, it will be noticed by the adversary.

3. Indispensability: Lampson [3] reports that a communication channel is a covert channel if it is neither designed nor intended to transfer information at all. The channel should introduce several benefits to the users besides sending data covertly; thus, the adversary cannot or will not close off that channel.

2.3 Covert channel classification

Covert channels can be classified as storage or timing channels, noisy or noiseless channels, and program-flow channels.

2.3.1 Storage channels and timing channels

The covert storage channel depends on a shared variable or a storage location, whereby one process (sender) can be allowed to write directly or indirectly to the storage location and the other process (receiver) reads from that storage location. On the other hand, the covert timing channel enables senders to send information to the receiver through signals, whereby the sender manages the time that is needed to perform some operation in such a way that when the receiver observes the time, it will understand a special event or a special piece of information. The main disadvantage of the timing channel is that it is considered very noisy because of the several external factors that affect the execution time of a process. Covert storage channels and timing channels need a synchronization process, which enables the sender and receiver to synchronize with each other to send and receive information. The storage covert channel uses a data variable to enable the sender and receiver to communicate. Therefore, a synchronization variable, called sender-receiver, is needed by the sender to notify the receiver that he has completed reading or writing a data variable. The covert channel uses another synchronization variable, called receiver-sender. To distinguish between storage and timing channels, if a channel uses a storage variable to transfer data between the sender and receiver, it is considered a storage channel. On the other hand, a covert timing channel uses time reference (e.g., a clock) to transfer data between the sender and receiver, whereby the sender and receiver use a common time reference.

2.3.2 Noisy and noiseless channels

I discussed previously that the characteristics of the covert channel are similar to any communication channel. One of these characteristics is that the channel may be noisy. The covert channel can be noiseless if the transmitted data by the sender and received data by the receiver are the same with probability 1; otherwise, the channel is noisy. Usually, data transmitted through a covert channel is represented by bit "0" or "1." Nevertheless, if the receiver decodes every bit transmitted by the sender correctly, then the covert channel is considered noiseless. Thus, to reduce error rate, which is produced by noise, correction codes are used [6].

2.3.3 Program-flow channels

I present a new type of covert channel, which is program-flow. The program-flow covert channel depends on the flow of program execution to convey information. In our proposed covert channel, the sender tries to guess the correct delta_mmap (encoded information) of the vulnerable server program. The server code which executes in case of successful guess differs from which executes in a failed guess. The receiver distinguishes between server code executed in successful and failed guesses.

2.4 Authentication and key exchange

Authentication process identifies entities that are attempting to access some resources. Diffie-Hellman (DH) algorithm is used as method of public key exchange.

2.4.1 Authentication process

Authentication is a process of checking whether someone or something is authorized or not to access some resources. Authentication can be computer to computer or process to process and mutual in both directions [7, 8]. Bob can authenticate Alice's identity depending on four factors [7, 9], which are:

2.4.1.1 Something you know

Alice sends a request to the server to access some resources; Bob authenticates Alice by asking her about a secret thing that she knows, such as password. If Alice issues a correct password, then Bob will accept her request for accessing some resources. Fortunately, a password is needed to login into the system and access its resources. Yet, unfortunately, the user is always asked to reuse the password when he wants to log into the system, which gives attackers opportunities to hack the password and reuse it. The solution for this problem is to use a onetime password (OTP) so that the user each time she logs into a system needs a new password.

2.4.1.2 Something you have

One of the disadvantages of the first authentication factor (something you know) is that the user may forget his password. Thus, the second authentication factor (something you have) overcomes this problem, whereby the user has an object (e.g., automatic teller machine (ATM) cards, OTP cards [7], and smart cards [9]) to access the system. Unfortunately, the objects may get stolen by attackers.

2.4.1.3 Something you are

The third authentication factor is based on the measurements of the user's physical characteristics such as the fingerprints, iris, and voice. The techniques that measure the behavioral characteristics of the user are called biometrics [7, 8]. This factor overcomes the problems of the previous factors because it does not depend on a password or a token.

2.4.1.4 Somebody you know

Brainard et al. [9] proposes a fourth factor of authentication that is dependent on emergency authenticator, and it is used when the primary authenticator is unavailable to a user. A good example for emergency system is email; thus, when a user forgets his password, he often has the option of having password reset instructions. A system called “vouching” is introduced. A voucher system permits swapping of the roles of the token and PIN to deal with the case when the user has forgotten his PIN but still has his token.

2.4.2 Message authentication

When Alice and Bob want to exchange messages, they do not want an attacker to modify the contents of their messages. This can be achieved by using message authentication odes (MACs), where the MAC is a tag, attached to the message by Alice to Bob or vice versa. If Bob validates this tag, the request of Alice will be accepted by Bob; otherwise, it is rejected [7]. MAC that is based on cryptographic hash functions is called HMAC [10]. There are many hash functions, such as message digest 5 (MD5) and Secure Hash Algorithm 1 (SHA1). When HMAC is used with MD5, it is called Hashed Message Authentication Code-Message Digest 5 (HMAC-MD5), and when it is used with SHA1, it is called Hashed Message Authentication Code-Secure Hash Algorithm 1 (HMAC-SHA1) [7, 10, 11]. In our dissertation, we use the secure hash algorithm SHA256 with pre-shared key to form HMAC-SHA256, where the secure hash algorithm SHA256 takes a message of 512-bit blocks as input and returns a digest message with 256 bits as output [7].

2.4.3 Diffie-Hellman (DH) key exchange algorithm

Key exchange algorithms are cryptographic methods that generate cryptographic shared keys that are shared among users. After Alice and Bob agree on a shared key, they can use it in HMAC, and they can also use it in symmetric encryption algorithms to encrypt or to decrypt files. Alice encrypts file using one of the symmetric algorithms and sends it to Bob. Bob in turn uses the same symmetric algorithm to decrypt the file. Note that Alice and Bob must agree on a shared key before using symmetric algorithm. Many key agreement protocols have been proposed. The Diffie-Hellman (DH) algorithm [12] is a very popular example that introduces a key exchange protocol using the discrete logarithm problem [13]. DH algorithm enables Alice and Bob to exchange secure keys over an insecure channel.

Figure 2 shows the mechanism of DH. The values g and p are public parameters known to Alice and Bob, whereby p is a prime number and g (generator of p) is an integer less than p. This means that for all every number n between 1 and p-1, there is a power k of g such that $n = g^k \bmod p$. Both Alice and Bob choose a secret random integer number, a and b, respectively. After that, Alice sends to Bob ($g^b \bmod p$), and Bob sends to Alice ($g^b \bmod p$). Finally, they agree on a secret key by using this formula ($(g^a)^b \bmod p$).

Figure 3 shows that a “man-in-the-middle” (Mallory) can listen and modify the conversation messages between Alice and Bob. In so doing, she can convince Alice and Bob that they are communicating with each other while in fact both are communicating with Mallory [7]. Moreover, **Figure 3** shows that the main vulnerability in the DH protocol is that it does not have an authentication process. Several

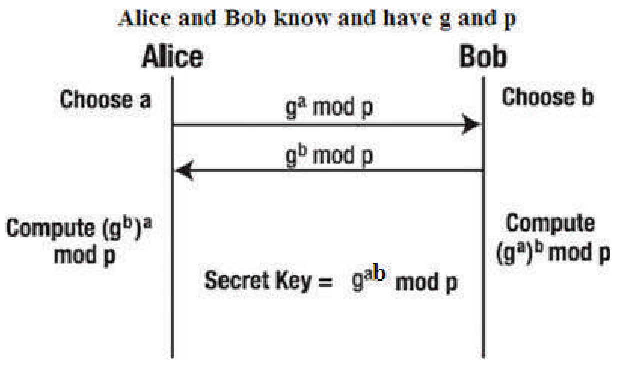


Figure 2.
Diffie-Hellman key exchange algorithm. Alice and Bob agree on a secret key over an insecure channel. The secret key that they agree on is computed as (ga)b mod p.

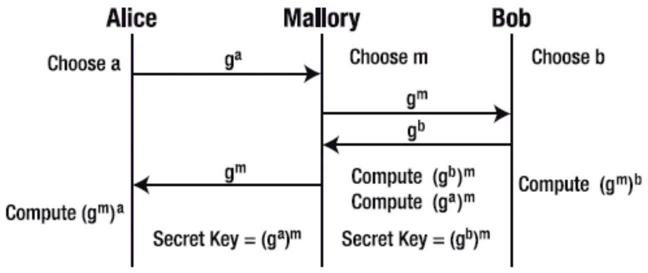


Figure 3.
A Diffie-Hellman weakness. A man-in-the-middle (e.g., Mallory) impersonates Alice to agree on a shared key with Bob. Also, she impersonates Bob to agree on a shared key with Alice.

versions of DH protocol exist to overcome this problem, for example, by using DH with digital signature [11]. Diffie et al. [14] enhance a Diffie-Hellman protocol with an authentication process, whereby Alice and Bob must authenticate themselves using a digital signature. Alice and Bob must have a pair of keys (public key and private key) and a certificate for the public key. So, during execution of DH protocol, Alice and Bob transmit massages with signature; Mallory cannot forge the signature because she needs to share Alice's private key and Bob's private key.

The DH algorithm relies on heavy computation, which may not be suitable to resource-constrained platforms.

3. Related work

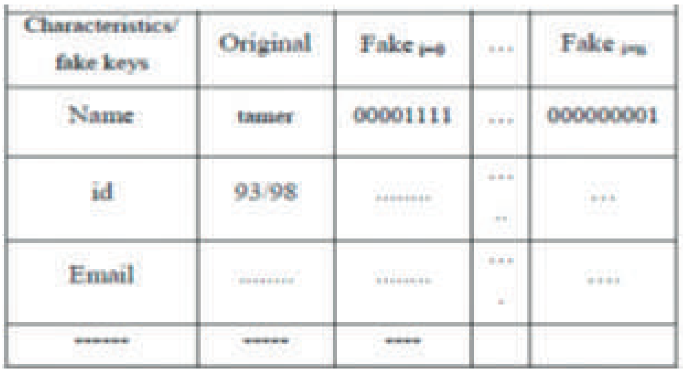
There are many researches that target covert channel undetectability [15–17], but most of the works have drawbacks and lack in channel detectability (Girling [18] in 1987). He creates three covert channels through a local area network (LAN): two of them are storage channels, and the third is a timing channel. The two storage channels depend on "what-is-sent" strategy, whereby one of them depends on the frame size, which is sent by the sender. If the frame size equals to 256, the amount of covert information decoded by receiver, who monitors the sender activity on the LAN, would be 8 bits. On the other hand, the timing channel depends on the time that represents the time interval between successive sends. The time difference between successive sends may be odd or even, and the prior agreement between the sender and receiver (who monitors the time between successive sends) is such that the odd time means bit "0" and even time means bit "1." So, the timing channel obviously depends on "when-is-sent" strategy.

TCP/IP protocol is used to create covert channel that is targeted by many researchers, where they used TCP to hide information [19–22]. Zhang et al. [6] propose covert channel to transfer messages to control (increasing or decreasing) the period of silence in traffic of VoLTE traffic. Create covert channel through hiding information in IP fields [20, 23]. Mead et al. [24] propose timing covert channel for wireless communication; they developed android application to communicate through local area network and mobile network. The results show that the channel is very undetectable in spite of the existence of malware and intrusion detection system.

Some researches, Fatayer et al. [15–17], try to use covert channel as benign channel, and it can be used to send legal information between the client and server. They used gaps in memory to create covert channel. Also, they used the channel to send text and audio files in acceptable time. The proposed technique depends on pre-agreement database which consists of original keys and its corresponding fake keys. Each original key has multiple fake keys. The database consist of the characteristics of clients; each feature represents an original key, and it has multiple fake keys. **Figure 4** figures out the pre-agreement between the client and server before using covert channel.

Customer asks cloud provider to access his resources. The summarization of approach is as follows. First, pre-agreement between the server and client is shown in **Figure 4**. Second, the customer sends a packet which contains "Fakei" attribute belongs to a specific customer (e.g., name) to the cloud provider. Third, the cloud provider will analyze the packet and make sure that the "Fakei" belongs to which customer. If yes, the provider goes to next step. Fourth, the cloud provider will ask for extra information to verify the customer and then he sends a packet that contains another fake key to the customer. Fifth, the customer receives the packet and he verifies the packet. The customer will send the required information to cloud provider such as one of the fake keys of email. Seventh, the cloud provider will analyze the packet to make sure that the "Fakei" (email) belongs to which customer. If yes, the cloud provider will accept the request. Eight, steps from 4 to 7 are considered the first level of security, so if these steps are repeated more than one time, it can achieve multilevel of security.

A new detection approach of covert timing channel is proposed by Fahimeh et al. [25], where this approach enables to detect covet time channel through traffic distribution. They used statistical test to measure the network traffic online.



Characteristics/ fake keys	Original	Fake $_{i=0}$	…	Fake $_{i=n}$
Name	tamer	00001111	…	000000001
id	93/98	....	… ..	…
Email	....	....	… .	....
------	-----	----		

Figure 4.
Database as pre-agreement between the client and server.

4. Covert channel with authentication leads to secure communication

The proposed technique is responsible for creating secure communication channel using covert channel, encryption, and authentication. In the first step before using covert channel, the client and server must agree on pre-agreement table as shown in **Figure 4**, where it depicts out the pre-agreement table consisting of original key (OK) and its corresponding fake keys. The key point here is that the fake key is used in communication channel and the original key is kept secret in both sides (client and server). **Figure 5** illustrates the proposed technique, where the network consists of the client and server. The client agrees on shared information (e.g., table as database) with the server.

The client wants to send to the server a message. Then, he encrypts the message using the original key. After that the encrypted message is attached with the fake key to be a parameter to HMAC function. HMAC depends on shared key between the client and server. Then the client sends (HMAC + encrypted message + fake key) to the server. Where, HMAC is used for integrity and fake key for server to know the

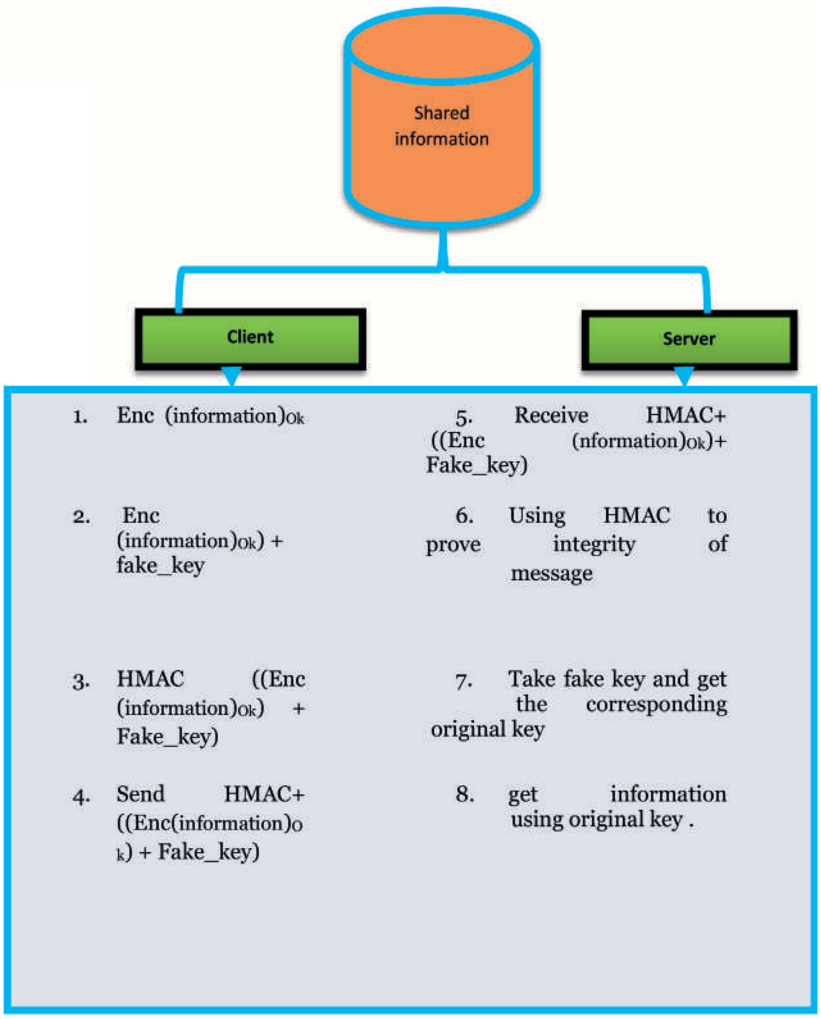

Shared information
Client
Server
1. Enc (information)Ok
2. Enc (information)Ok) + fake_key
3. HMAC ((Enc (information)Ok) + Fake_key)
4. Send HMAC+ ((Enc(information)Ok) + Fake_key)
5. Receive HMAC+ ((Enc (nformation)Ok)+ Fake_key)
6. Using HMAC to prove integrity of message
7. Take fake key and get the corresponding original key
8. get information using original key .

Figure 5.
Technique for transferring secure data through covert channel using covert channel, encryption, and authentication.

original key to decrypt the encrypted message. On the other hand, the server receives the client's message. After that he separates the message to the HMAC, encrypted message, and fake key. Then, he checks the integrity of the message. After that he gets the original key from its corresponding fake key to decrypt the encrypted message.

5. Performance discussion

The implementation of technique is done by: first create a network as client and server with implemented java application. Client and server machines are 32 bit ×86, CPU Core (TM) i5 2.40 GHz, and Ram 4GB. Advanced Encryption Standard (AES) [26] algorithm is used in the implementation. Hashed message authentication code (HMAC) is used to guarantee the integrity [7, 11, 12]. The following issues are satisfying in this technique:

1. Confidentiality: the technique guarantees that the messages are protected from disclosure, which is done by encrypting the messages with original keys that do not send through communication channel. Instead, the original keys are sent encrypted by fake keys.

2. Integrity: the information is protected from being changed by unauthorized parties through using HMAC function which checks if the content of the message is altered or not.

3. Undetectability: undetectability is achieved depending on two conditions: first, plausibility, the messages that are sent through the covert channel are protected from adversary by making the covert channel appear like a normal channel through sending normal encrypted messages, and, second, hiding the fake key inside the message which does not affect bit distribution, especially when the size of the fake key is small.

4. Comparative analysis: My technique is used in two ways, malicious and benign usages. Also, encryption and authentication are used that differ from other techniques.

5. Dynamically: the generating keys between the client and server have a flexible length. Because when you repeat the scenario in **Figure 5** several times, you get new keys with different sizes. If you repeat the technique four times and each time generates key with 16-bit size, then you will get a key with 64-bit size.

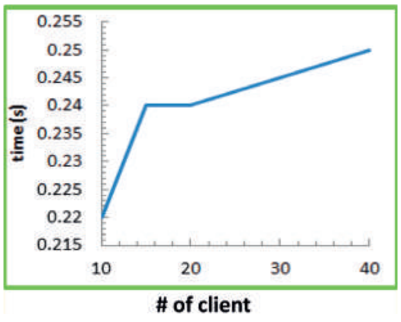


Figure 6.
The technique can handle multiple users in acceptable time.

The time that is needed by the server to serve the client is measured. The technique can handle several clients with acceptable time as shown in **Figure 6**. The server needs less than 0.25 s to deal and handle 30 clients.

6. Conclusion and future work

Encryption is used to achieve confidentiality to protect data from stealing from the third party (e.g., attacker). If users use encryption, they cannot achieve integrity. They need authentication and covert channel besides encryption technique to achieve integrity. In this chapter, we used encryption, authentication, and covert channel to produce a secure communication between eligible parties. This technique satisfies confidentiality through encryption and integrity using authentication algorithm. Finally, this technique generates undetectability covert channel through normal distribution of bits. Secure communication channel between the client and server that enables them to transfer data securely and to agree on keys that are used for future communication. The technique needs pre-shared table that consists of original and fake keys.

Acknowledgements

This is to happily express my sincere thanks and appreciation to the following for their support and guidance throughout the chapter writing. I would like to thank my friends who stood beside me and helped me pursue my work. This chapter is dedicated to my parents, without whom, after the blessings of Allah, all this work would not be possible. They have been a source of endless love, encouragement, and support. They believed in me and in whatever decision I took and are proud of me on whatever achievement I may have.

Thanks

Your support means a lot of thanks.

Author details

Tamer S.A. Fatayer
Computer Science and Information Technology, Al-Aqsa University, Gaza, Palestine

*Address all correspondence to: ts.fatayer@alaqsa.edu.ps

References

[1] Dlaminia EM, Eloffb M. Information security: The moving target. Computers & Security. 2009;**28**:10

[2] Tilborg H, Jajodia S. Encyclopedia of Cryptography and Security. 2nd edition. Boston: Springer; 2011. ISBN: 978-1441959058

[3] Lampson B. A note on the confinement problem. Communications of the ACM. 1973;**16**:613-615

[4] Ray B, Mishra S. A protocol for building secure and reliable covert channel. In: PST'08: Proceedings of the Sixth Annual Conference on Privacy, Security and Trust; Washington, DC, USA; 2008. pp. 246-253

[5] Giffin J, Greenstadt R, Litwack P, Tibbetts R. Covert messaging through TCP timestamps. Presented at the PET'02: The Workshop on Privacy Enhancing Technologies; San Francisco, CA, USA; 2002

[6] Zhang X, Tan Y-A, Liang C, Li U, Li J. A covert channel over VoLTE via adjusting silence periods. IEEE Access. February 2018;**6**:9292-9302

[7] Daswani N, Kern C, Kesavan A. Foundations of Security: What Every Orogrammer Needs to Know. 1st ed. Berkely, USA: Apress; 2007

[8] Sandhu R, Samarati P. Authentication, access control, and audit. ACM Computing Surveys. 1996;**28**:241-243

[9] Brainard J, Juels A, Rivest R. Fourth factor authentication: Somebody you know. In: CCS'06: Proceedings of the 13th ACM conference on Computer and communications security; Virginia, USA; 2006. pp. 168-178

[10] Kumar M. An enhanced remote user authentication scheme with smart card. International Journal of Network Security. 2010;**10**:175-184

[11] Information Technology Laboratory. The keyed-hash message authentication code (HMAC). National Institute of Standards and Technology, Technical Report FIPS PUB 198; 2002

[12] Bellare M, Canetti R, Krawczyk H. Keying hash functions for message authentication. In: CRYPTO'96: Proceedings of the 16th Annual International Cryptology conference on Advances in Cryptology; London, UK; 1996. pp. 1-15

[13] Diffie W, Hellman M. New directions in cryptography. IEEE Transactions on Information Theory. 1976;**IT-22**:644-654

[14] Mahalanobis A. Diffie-Hellman key exchange protocol, its generalization and nilpotent groups [Ph.D. dissertation]. Florida: The Charles E. Schmidt College of Science, Florida Atlantic University; August 2005

[15] Fatayer T, Khattab S, Omara F. A key-exchange protocol based on the stack-overflow software vulnerability. In: ISCC'10: IEEE Symposium on Computers and Communications; Riccione, Italy; June 2010. pp. 411-416

[16] Fatayer T, Khattab S, Omara F. OverCovert: Using stack overflow software vulnerability to create a covert channel. In: NTMS'11: 4th IFIP International Conference on New Technologies, Mobility and Security; Paris, France; February 2011

[17] Fatayer T, Timraz K. MLSCPC: Multi-level security using covert channel to achieve privacy through cloud computing. Presented at the WSCNIS'2015 the 2nd World Symposium on Computer Networks

and Information Security; Hammamet, Tunisia; 2015

[18] Girling CG. Covert channels in LAN's. IEEE Transactions on Software Engineering. 1987;**13**:292-296

[19] Gligor V. A Guide to Understanding Covert Channel Analysis of Trusted System. National Computer Security Center (NCSC) Technical Report, Version 1; 1993

[20] Valentin B, Annessi R, Tanja Z. Decision tree rule induction for detecting covert timing channels in TCP/IP traffic. In: International Cross-Domain Conference for Machine Learning and Knowledge Extraction; 2017

[21] Katzenbeisser S, Petitcolas F. Information Hiding Techniques for Steganography and Digital Watermarking. 1st ed. Norwood, USA: Artech House, Inc; 2000. ISBN: 1580530354

[22] Ahsan K. Covert channel analysis and data hiding in TCP/IP [Master of applied Science thesis]. Electrical and Computer Engineering, Toronto University; 2002

[23] Rowland C. Covert channels in the TCP/IP protocol suite. First Monday Journal. 1997;**2**:5

[24] Mead FC, Zielinski JM, Watkins L, Robinson WH. A mobile two-way wireless covert timing channel suitable for peer-to-peer malware. In: 2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC); Montreal, QC; 2017. pp. 1-6

[25] Rezaei F, Hempel M, Sharif H. Towards a reliable detection of covert timing channels over real-time network traffic. IEEE Transactions on Dependable and Secure Computing. 2017;**4**:249-264

[26] Shao F, Chang Z, Zhang Y. AES encryption algorithm based on the high performance computing of GPU. In: 2010 Second International Conference on Communication Software and Networks; 2010. pp. 588-590

Chapter 8

High-Speed Area-Efficient Implementation of AES Algorithm on Reconfigurable Platform

Altaf O. Mulani and Pradeep B. Mane

Abstract

Nowadays, digital information is very easy to process, but it allows unauthorized users to access to this information. To protect this information from unauthorized access, cryptography is one of the most powerful and commonly used techniques. There are various cryptographic algorithms out of which advanced encryption standard (AES) is one of the most frequently used symmetric key cryptographic algorithms. The main objective of this chapter is to implement fast, secure, and area-efficient AES algorithm on a reconfigurable platform. In this chapter, AES algorithm is designed using Xilinx system generator, implemented on Nexys-4 DDR FPGA development board and simulated using MATLAB Simulink. Synthesis results show that the implementation consumes 121 slice registers, and its maximum operating frequency is 1102.536 MHz. Throughput achieved by this implementation is 14.1125 Gbps.

Keywords: cryptography, AES, FPGA, VLSI, system generator

1. Introduction

NIST has started a development process of FIPS for AES algorithm stating that this is the replacement for data encryption standard (DES) algorithm. Alternatively, this algorithm is also known as Rijndael algorithm. Rijndael algorithm has the advantages like resistance against all recognized attacks, code and speed compactness, and simple design. Cryptography is a process in which the information to be sent is added with secret key so as to transmit the data securely at the destination. There are two types of cryptography based on the type of key applied: symmetric key cryptography and asymmetric key cryptography. In symmetric key cryptography, equal key is utilized for encryption as well as decryption, whereas in asymmetric key cryptography, different keys are required in encryption and decryption. AES algorithm is selected for implementation because it is secure and its components and design principles are completely specified. AES is a symmetric key block cipher. The design of AES algorithm is based on linear transformation. Due to the use of Rijndael algorithm, different block and key sizes can be selected which was not possible in DES algorithm. Block and key size can be selected from 128/160/192/224/256 bits and need not be the same. According to AES standard, this algorithm can only accept 128 bits of block, and key size can be selected from 128/192/256 bits. Based on the key size, the number of rounds will vary. For example, if key size is 128, 192, or 256, then the number of rounds will be 10, 12, and 14, respectively. The structure of AES algorithm is shown in **Figure 1**. In this chapter,

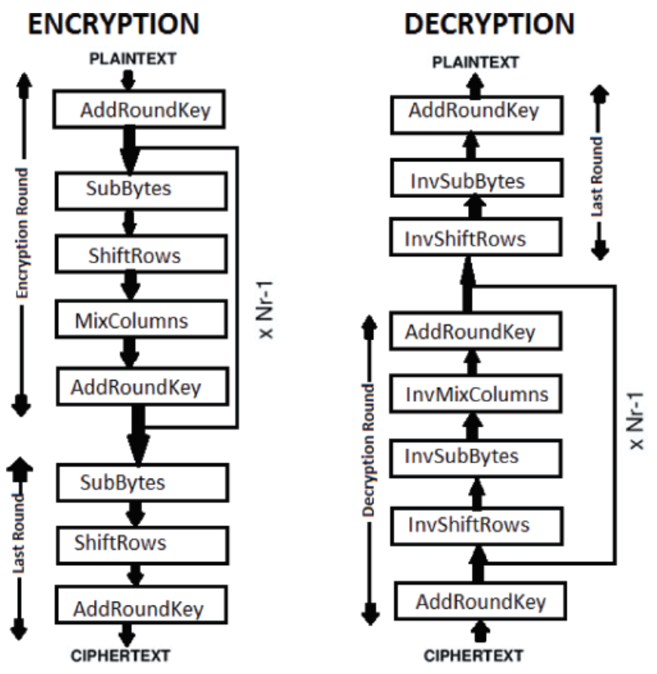


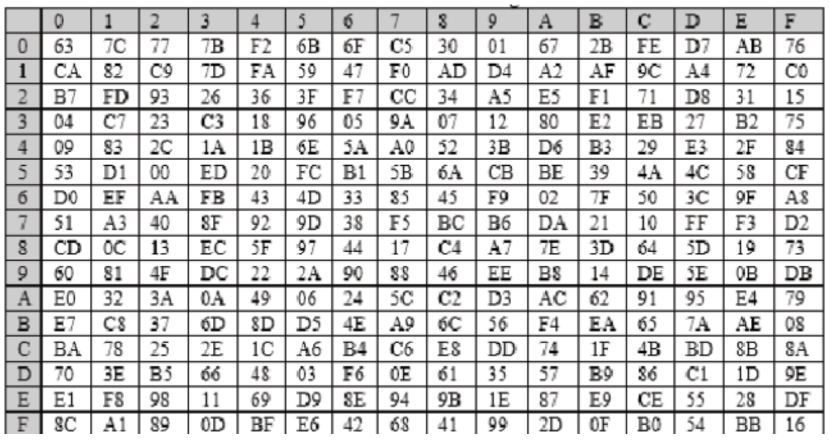
Figure 1.
Structure of AES algorithm.

this algorithm is designed with 128 bits of block size and key size, respectively, that is, AES generates cipher text of 128 bits for 128 bits of plaintext. After the initial round, plaintext processes through ten rounds. Each round contains processes like byte substitution, shift rows, mix columns, and add round key.

1.1 Byte substitution

The 16 input bytes are substituted by using fixed lookup table known as s-box. **Figure 2** shows s-box of AES algorithm. This s-box consists of all possible combinations of 8-bit sequence. The resulting new 16 bytes are organized in a matrix having four rows and four columns.

Figure 3 shows byte substitution stage in AES algorithm.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Figure 2.
S-box of AES algorithm.

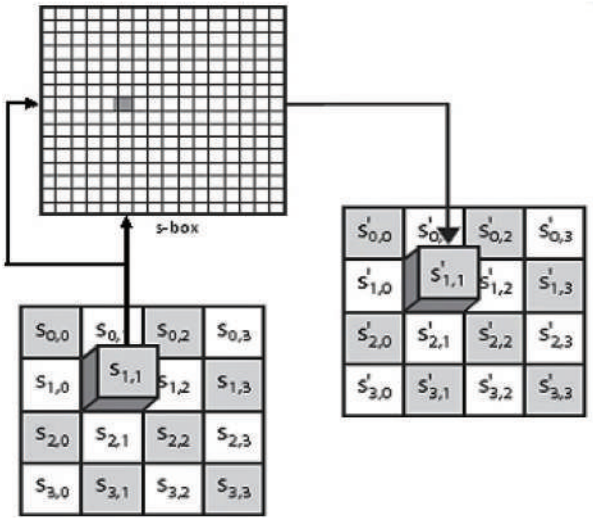

s-box
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure 3.
Byte substitution stage.

1.2 Shift row

Each row from the matrix generated from the byte substitution is cyclically shifted to the left. Any entry that is dropped off is reinserted to the right side. The first row is kept as it is, the second row is shifted by one-byte position to the left, the third row is shifted by two-byte position to the left, and the fourth row is shifted by three-byte position to the left. The resultant matrix consists of same 16 bytes but at different position. **Figure 4** shows shift row stage in AES algorithm.

1.3 Mix column

Each column of four bytes is now transformed using special arithmetical function of Galois field (GF) 28. This function takes four bytes of the column as input and

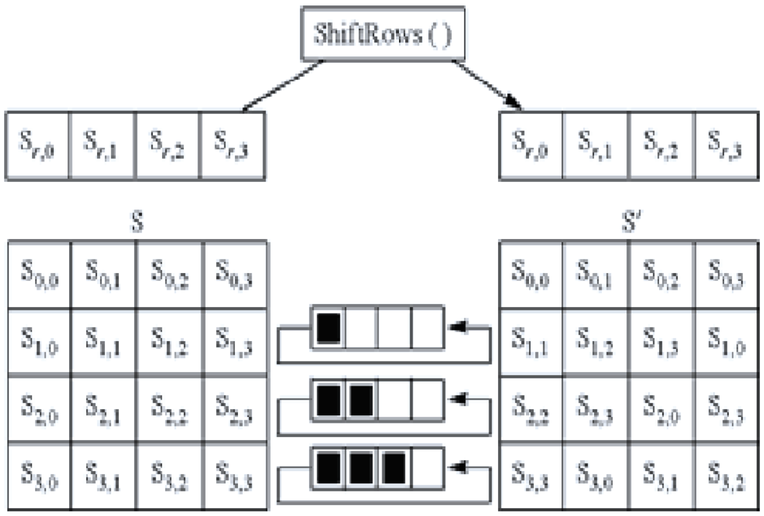

ShiftRows ( )
$S_{r,0}$ $S_{r,1}$ $S_{r,2}$ $S_{r,3}$
$S_{r,0}$ $S_{r,1}$ $S_{r,2}$ $S_{r,3}$
S
S'
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,1}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

Figure 4.
Shift row stage.

outputs completely new four bytes that replaces the original four bytes. **Figure 5** shows mix column stage in AES algorithm.

1.4 Add round key

The 16 bytes of the resultant matrix generated from mix column stage are then considered as 128 bits. In add round key stage, 128 bits of state are bitwise EX-ORed with 128 bits of round key. If this result belongs to the last round, then the output is cipher text else the resulting 128 bits is considered as 16 bytes, and another round is started with new byte substitution process. This is a column-wise operation between four bytes of state column and one word of round key. In the last round, there is no mix column step. **Figure 6** shows add round key stage in AES algorithm.

Decryption of cipher text, generated from AES encryption, contains all the stages in encryption but in reverse order. AES decryption starts with inverse initial round. The remaining nine rounds in decryption consist of processes like add round key, inverse shift rows, inverse byte substitution, and inverse mix columns.

Add round key: Add round key has its own inverse function since XOR functions its own inverse and the round keys should be selected in reverse order.

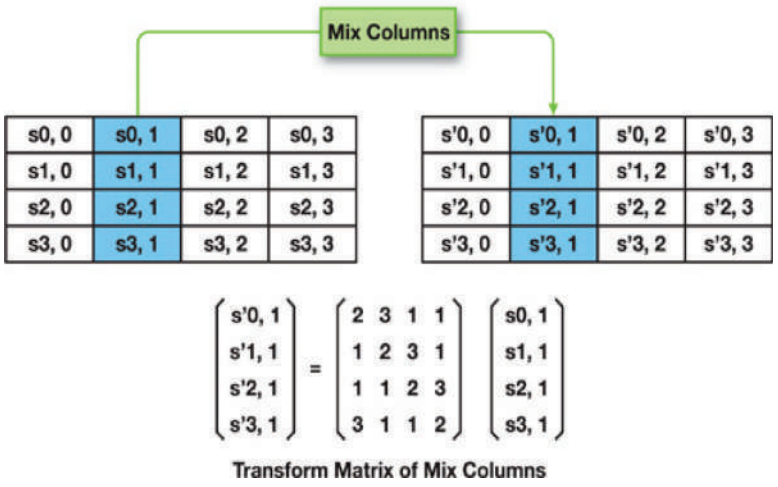
Inverse shift rows: Inverse shift rows functions exactly in the same way as shift row stage but in opposite direction. The first row is kept as it is, the second row is shifted by one-byte position to the right, the third row is shifted by two-byte position to the right, and the fourth row is shifted by three-byte position to the right. The resultant matrix consists of same 16 bytes but at different position. **Figure 7** shows inverse shift row stage in AES algorithm.

Inverse byte substitution: Inverse byte substitution is done using predefined substitution table known as inverse s-box. **Figure 8** shows inverse s-box in AES algorithm.

Inverse mix column: Transformation in inverse mix column is done using polynomials of degree less than 4 over Galois field (GF) 28 in which coefficients are the elements from the column of the state.

The rest of the chapter is organized as follows:

Section 2 presents the survey based on the various kinds of implementation of AES algorithm on reconfigurable platform. In Section 3, implementation of AES algorithm using the proposed approach is discussed. In Section 4, experimental results achieved using the proposed method along with the comparative analysis with existing methods are discussed.



$$\begin{pmatrix} s'0,1 \\ s'1,1 \\ s'2,1 \\ s'3,1 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} s0,1 \\ s1,1 \\ s2,1 \\ s3,1 \end{pmatrix}$$

Figure 5.
Mix column stage.

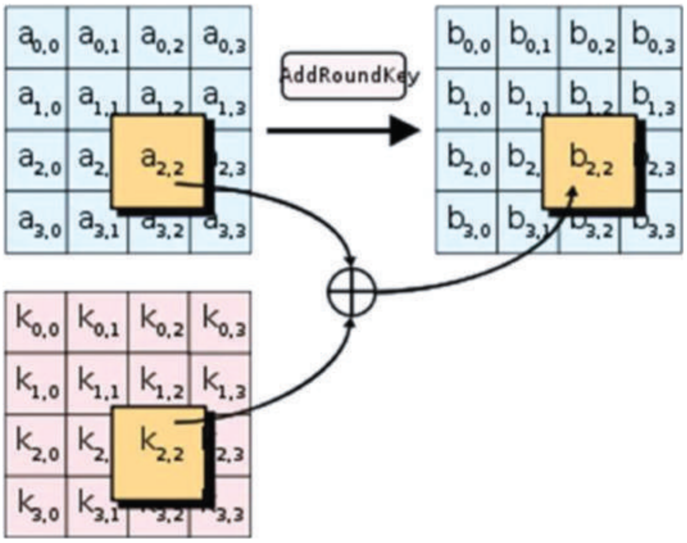

AddRoundKey
$a_{0,0}$ $a_{0,1}$ $a_{0,2}$ $a_{0,3}$
$a_{1,0}$ $a_{1,3}$
$a_{2,0}$ $a_{2,2}$
$a_{3,0}$ $a_{3,1}$
$b_{0,0}$ $b_{0,1}$ $b_{0,2}$ $b_{0,3}$
$b_{1,0}$ $b_{1,3}$
$b_{2,0}$ $b_{2,2}$
$b_{3,0}$
$k_{0,0}$ $k_{0,1}$ $k_{0,2}$ $k_{0,3}$
$k_{1,0}$ $k_{1,1}$ $k_{1,2}$ $k_{1,3}$
$k_{2,0}$ $k_{2,2}$
$k_{3,0}$ $k_{3,1}$

Figure 6.
Add round key stage.

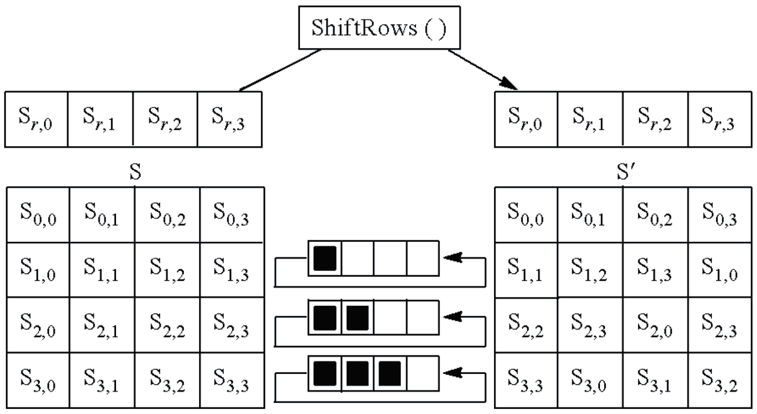
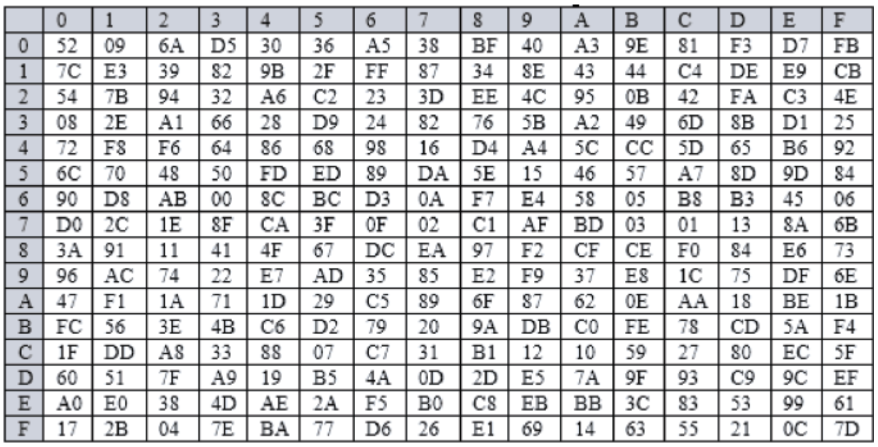

ShiftRows ( )
$S_{r,0}$ $S_{r,1}$ $S_{r,2}$ $S_{r,3}$
S
S'
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,0}$ $S_{1,1}$ $S_{1,2}$ $S_{1,3}$
$S_{2,0}$ $S_{2,1}$ $S_{2,2}$ $S_{2,3}$
$S_{3,0}$ $S_{3,1}$ $S_{3,2}$ $S_{3,3}$
$S_{0,0}$ $S_{0,1}$ $S_{0,2}$ $S_{0,3}$
$S_{1,1}$ $S_{1,2}$ $S_{1,3}$ $S_{1,0}$
$S_{2,2}$ $S_{2,3}$ $S_{2,0}$ $S_{2,3}$
$S_{3,3}$ $S_{3,0}$ $S_{3,1}$ $S_{3,2}$

Figure 7.
Inverse shift row.



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	82	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	89	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	84	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	87	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	38	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Figure 8.
Inverse S-box of AES algorithm.

2. Literature survey

In this section, focus is given on the work done by various researchers on FPGA-based implementation of AES algorithm. There are various researchers which have either concentrated on area optimization or speed optimization. Mulani and Mane [1] discussed integrating of DWT and AES algorithm for implementation of watermarking on FPGA. The design was implemented on xc6vcx75t-2ff484, and it utilizes 2117 slices at maximum operating frequency of 228.064 MHz. Ratheesh and Narayanan [2] proposed implementation of AES algorithm with low-power MUX LUT-based s-box on FPGA. This design achieved total power distribution of 0.55 W. Agarwal et al. [3] suggested implementation of AES algorithm using Verilog on Spartan-3E FPGA. This design utilizes 1464 slices. Farooq and Faisal Aslam [4] discussed implementation of AES algorithm on FPGA device using five different techniques which are suitable for area critical applications and speed critical applications. This design was implemented on Spartan-6 FPGA device, and it utilizes 161 slices at maximum operating frequency which is 886.64 MHz. The throughput of this system is 113.5 Gbps. Sai Srinivas and Akramuddin [5] proposed less complex hardware implementation of AES Rijndael algorithm on Xilinx Virtex-7 XC7VX90T FPGA. In the proposed design, synthesis tool was set to optimize speed, area, and power. Mathur and Bansode [6] proposed a cryptosystem, which is a combination of AES algorithm and ECC. This is a hybrid encryption scheme and the key size is 192 bits and there are 12 numbers of iterations in this system. Kalaiselvi and Mangalam [7] proposed a low-power and high-throughput FPGA implementation of AES algorithm using key expansion technique. This design accepts key size of 256 bits for both encryption and decryption. This design utilizes 5493 slices, and its maximum operating frequency is 277.4 MHz. The throughput of this system is 0.06 Gbps. Deshpande et al. [8] suggested BRAM-based and FPGA-based implementation of AES algorithm. Due to the use of BRAMs for implementing s-box, this design utilizes less number of slices. The design was implemented on XC3S1400AN and it utilizes 3376 slices. Ibrahim [9] presented FPGA implementation of AES encryption core that is suitable for limited resource-limited applications. This design was implemented on Spartan-3, and it utilizes 150 slices at maximum operating frequency of 90 MHz. Khose and Raut [10] proposed implementation of AES algorithm on FPGA in order to achieve high speed of data processing and also to reduce time for generating key. This design utilizes 201 slices and 2 BRAMs at maximum operating frequency of 70 MHz. Mulani and Mane [11] proposed FPGA implementation of DES algorithm. The design was implemented on XC2S200, and it utilizes 2118 slices and 97 IOBs. Yewale Minal and Sayyad [12] proposed implementation of AES encryption using VHSIC hardware description language VHDL) and decryption using Visual Basic. With this approach, 1403 slices are utilized at maximum operating frequency of 160.875 MHz, and it has a throughput of 2.059 Gbps. Deshpande et al. [13] discussed FPGA-based optimized architecture that utilizes less area. This design was intended for plaintext of 128 bits and key of 128 bits. Tonde and Dhande [14] discussed FPGA-based implementation of AES algorithm using iterative looping approach for 128 bits of block and key size. Varhade and Kasat [15] proposed a FPGA-based AES algorithm, which utilizes 1746 logic elements and 32,768 memory bits. This design was synthesized on Cyclone-II using Altera. Wadi and Zainal [16] proposed some modifications like decreasing number of rounds and replacing S-box with new s-box to reduce hardware requirements in order to enhance the performance of AES algorithm in terms of time ciphering and pattern appearance. Wang et al. [17] suggested high-speed implementation of AES algorithm on FPGA to transmit the data securely using pipelining and parallel processing methods. Shylashree et al. [18] focused on various novel FPGA architectures of AES algorithm. Borkar et al. [19] proposed iterative design approach for FPGA implementation of AES algorithm using VHDL. This design utilizes 1853 slices, and its operating frequency is 140.390 MHz. Deshpande et al. [20] presented very low complexity FPGA-based architecture for integrated AES encryptor and decryptor. This design is synthesized on Spartan-3 XC3S400 FPGA. Kaur and Vig [21] suggested an efficient implementation of AES algorithm on FPGA in which multiple rounds are processed simultaneously. Due to this implementation, speed is increased but it increases area. This design utilizes 6279 slices and 5 BRAMs, and its operating frequency is 119.954 MHz. Samanta [22] proposed fast and efficient reconfigurable platform-based implementation of AES algorithm using pipelining. This design utilizes 1051 slices and 11 BRAMs, and its operating frequency is 76.699 MHz. Good and Benaissa [23] discussed hardware implementation of fastest and slowest AES algorithm which utilizes 16,693 slices at maximum operating frequency of 184.8 MHz.

From the literature survey, it is clear that many researchers have either worked on optimizing the area or speed. Few researchers have concentrated on optimizing the speed as well as area. Implementation of AES algorithm, which is optimized in speed as well as area, is discussed in this chapter.

3. Implementation of AES algorithm

The proposed design is implemented with the aim to achieve both area and speed optimization. In the proposed design, keys for each round are initially generated by using MATLAB code, and then those keys are used in the design. Due to this approach, the design occupies less number of slices, and also the speed is faster than the normal approach. The design is implemented using Xilinx system generator. **Figure 9** shows Xilinx system generator-based model for AES algorithm.

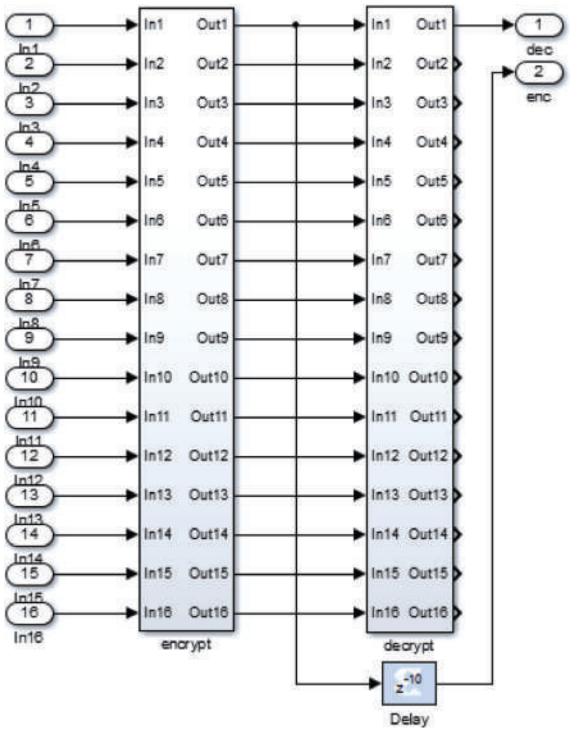


Figure 9.
System generator model for AES algorithm.

3.1 AES encryption

A plaintext of 128-bit is processed through 10 rounds. Each round contains processes like byte substitution, shift rows, mix columns, and add round key. As keys are generated using MATLAB code, only remaining system generator-based models like byte substitution, shift rows, and mix columns are discussed in this section.

Round function is one of the important processes in AES algorithm. **Figure 10** shows system generator-based model for implementing round0 function.

Round function consists of s-box, shift row, and mix column as shown in **Figure 11**.

Figure 12 shows implementation of s-box.

Figure 13 shows implementation of shift row.

Figure 14 shows implementation of mix column.

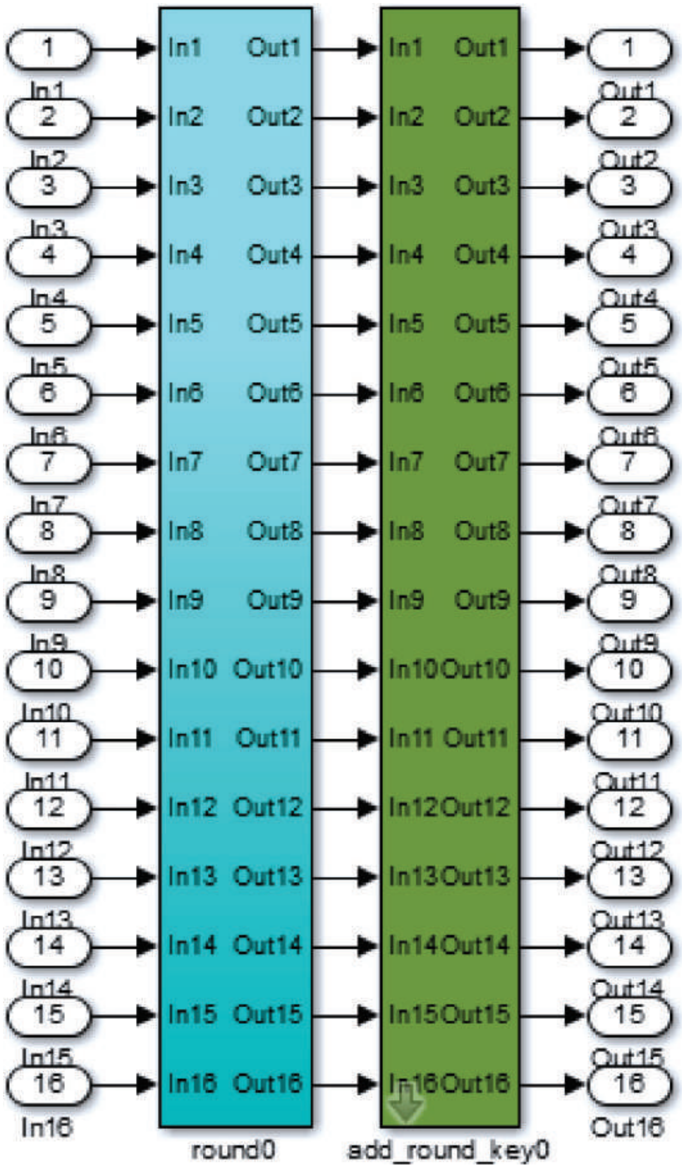


Figure 10.
System generator-based model of round function.

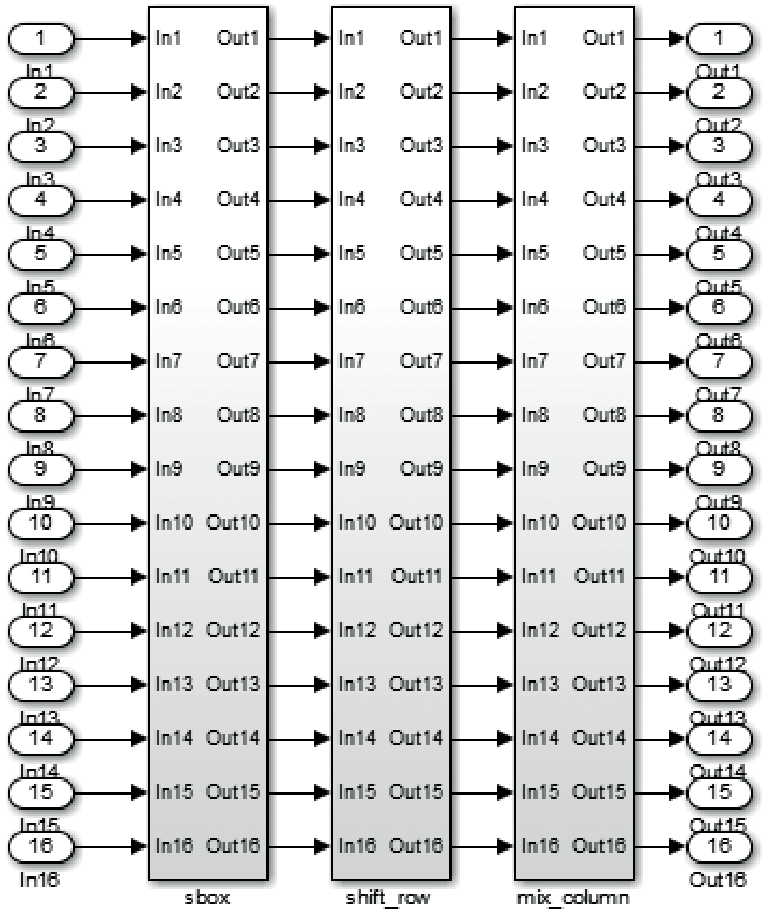


Figure 11.
Roundo.

Mix column consists of group_1, group_2, group_3, and group_4. **Figure 15** shows implementation of group. Further each group consists of four multiplication blocks such as mul_blk, mul_blk1, mul_blk2, and mul_blk3. **Figure 16** shows implementation of multiplication block.

3.2 AES decryption

A cipher text of 128-bits is processed through 10 inverse rounds. Each round contains processes like inverse byte substitution, inverse shift rows, inverse mix columns, and add round key.

Figure 17 shows implementation of inverse round function.

Inverse round function consists of inverse s-box, inverse shift row, and inverse mix column as shown in **Figure 18**.

Figure 19 shows implementation of inverse mix column.

Inverse mix column consists of four groups, i.e., group_1, group_2, group_3, and group_4. **Figure 20** shows implementation of group. Each group consists of multiplication blocks like mul_blk, mul_blk1, mul_blk2, and mul_blk3. **Figure 21** shows implementation of multiplication block.

Each multiplication block consists of three multipliers mul_2, mul_4, and mul_8 and EX-OR operations. **Figure 22** shows implementation of multipliers.

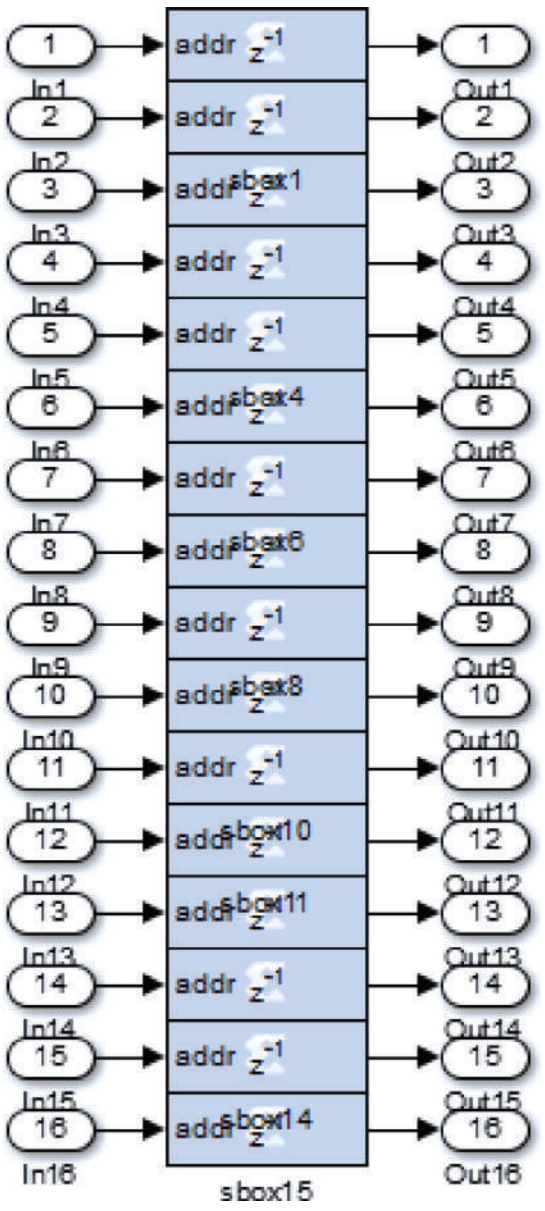


Figure 12.
Implementation of s-box.

Figure 23 shows implementation of inverse shift row.
Figure 24 shows implementation of inverse s-box.

3.3 Tools utilized

3.3.1 Software utilized

For implementing the proposed design, MATLAB 2013a and Xilinx ISE Design Suite are used. MATLAB is used for generating the keys and also to get the results in terms of images, whereas Xilinx ISE Design Suite is used to get the synthesis result, RTL schematic, and throughput of this implementation.

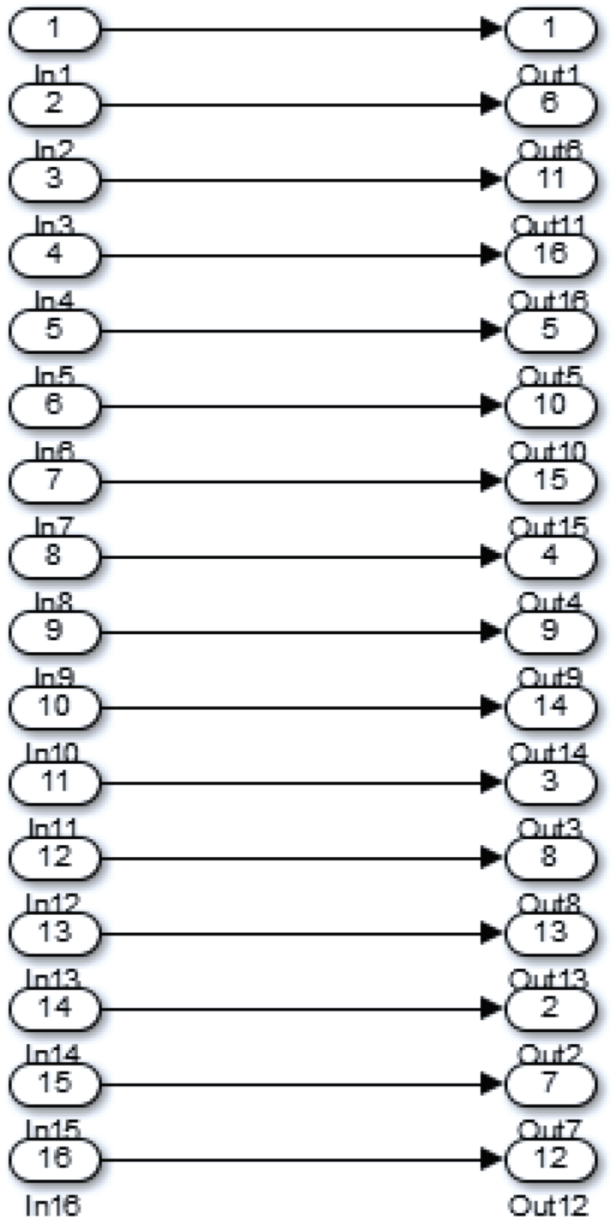


Figure 13.
Implementation of shift row.

3.3.2 Hardware utilized

Nexys-4 DDR development board is used for implementation. This board has the following features:

a. Xilinx Artix-7 FPGA XC7A100T-1CSG324C

b. 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops

c. 4860 Kbits of fast block RAM

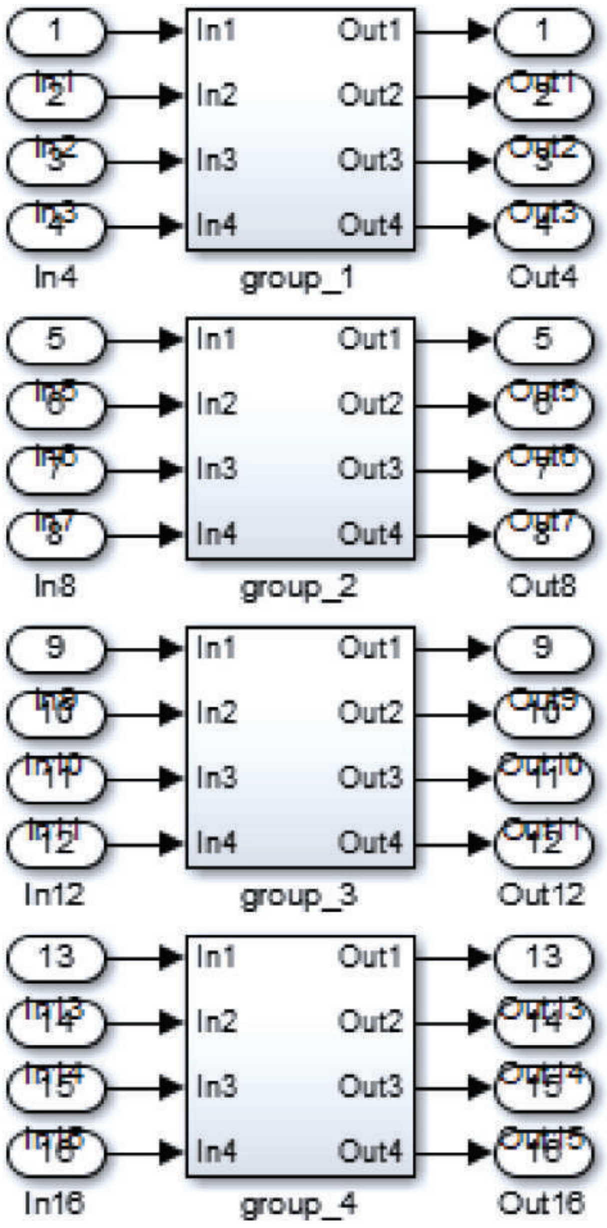


Figure 14.
Implementation of mix column.

d. Six clock management tiles, each with phase-locked loop (PLL)

e. 240 DSP slices

f. Internal clock speeds exceeding 450 MHz

g. On-chip analog-to-digital converter (XADC)

h. 128 MiB DDR2

i. Serial Flash

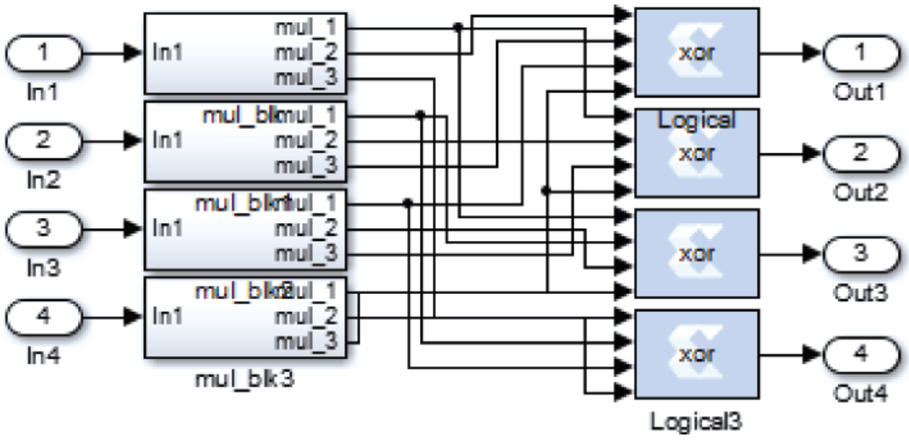


Figure 15.
Implementation of group.

j. Digilent USB-JTAG port for FPGA programming and communication

k. MicroSD card connector

l. Ships with rugged plastic case and USB cable

m. USB-UART Bridge

n. 10/100 Ethernet PHY

o. PWM audio output

p. 3-axis accelerometer

q. 16 user switches

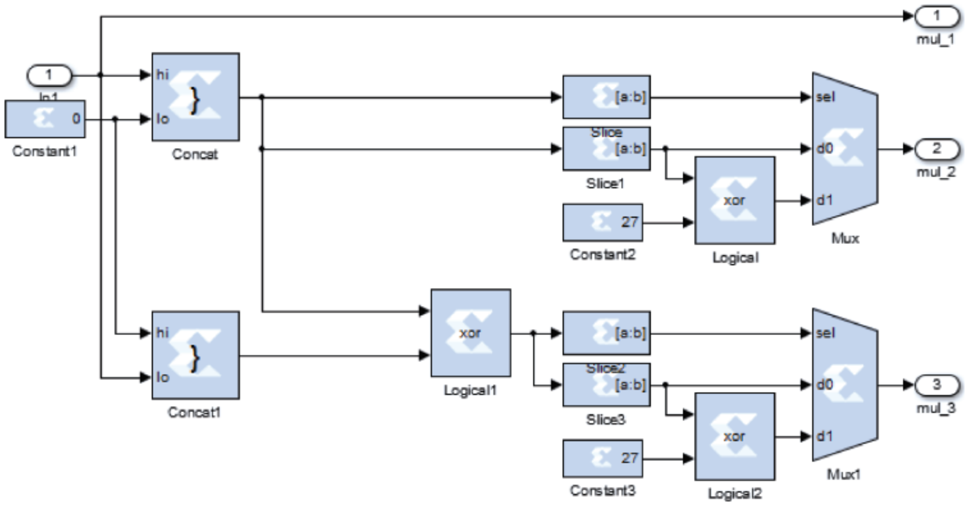


Figure 16.
Implementation of multiplication block.

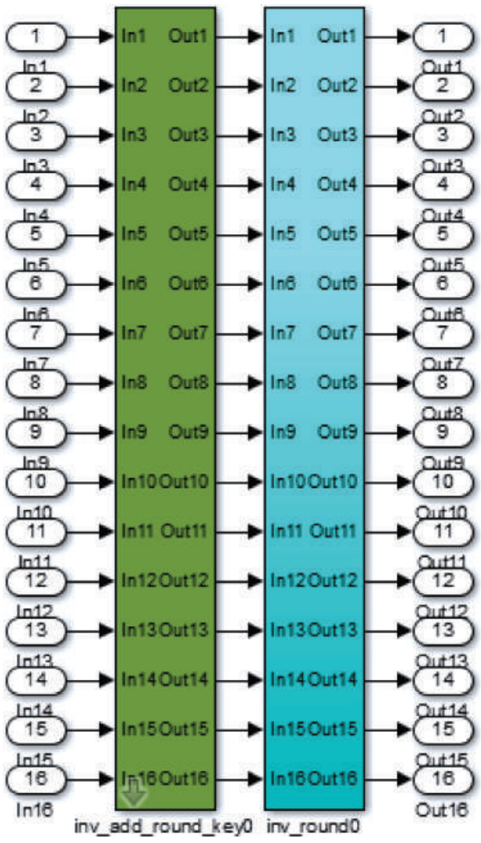


Figure 17.
System generator-based model of inverse round function.

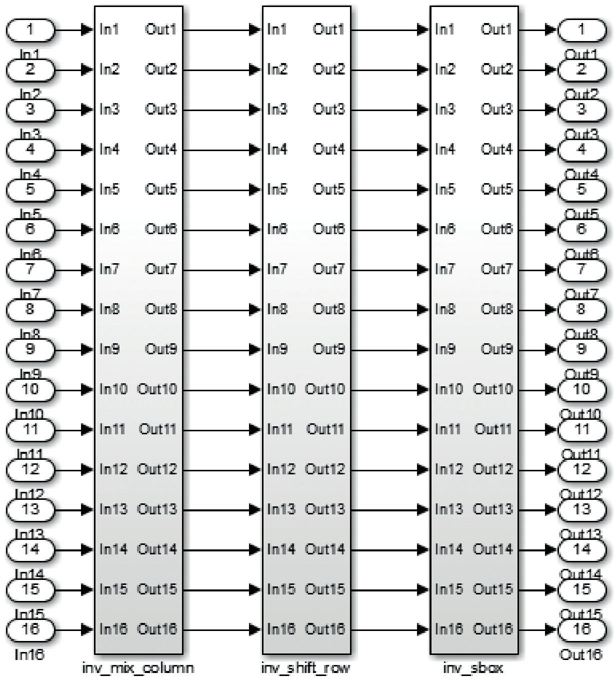


Figure 18.
Inverse roundo.

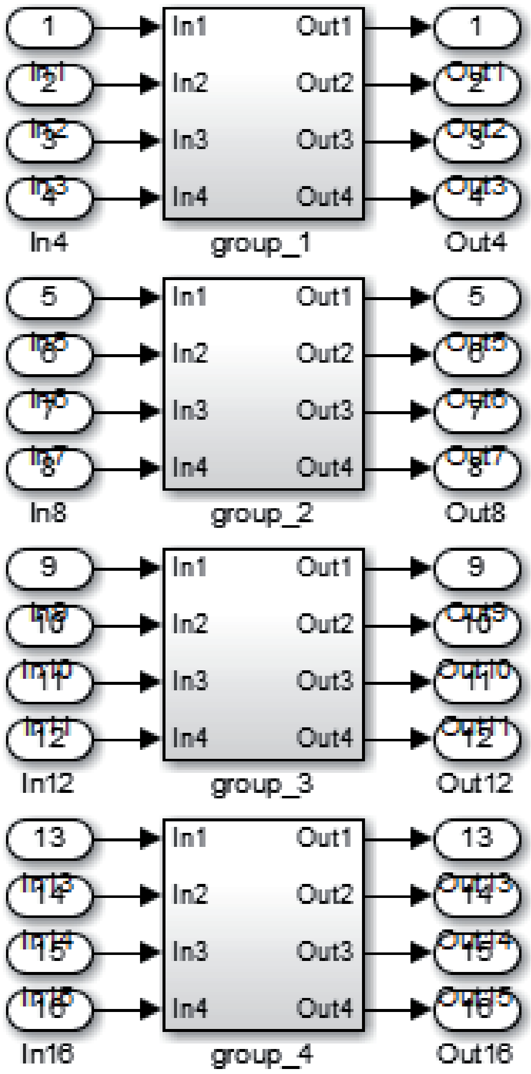


Figure 19.
Inverse mix column.

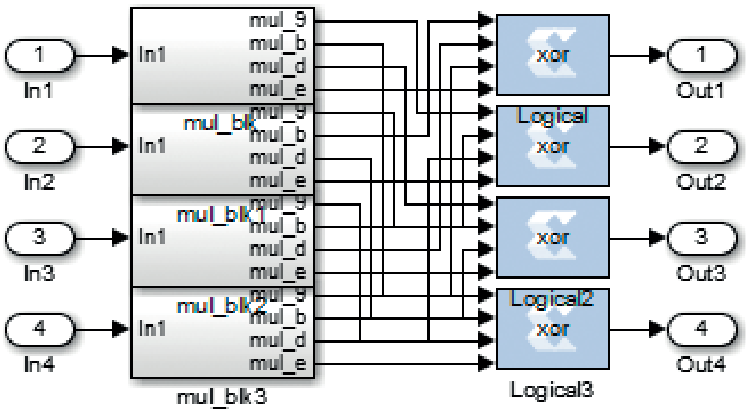


Figure 20.
Implementation of group.

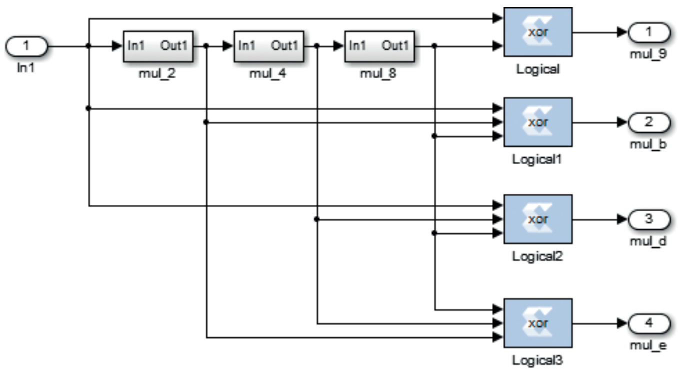


Figure 21.
Implementation of multiplication block.

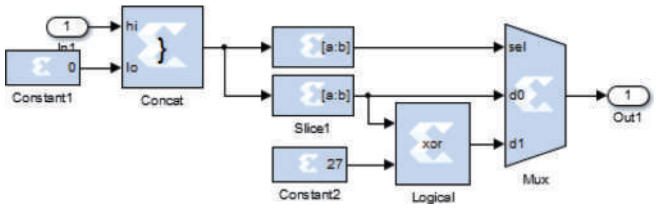


Figure 22.
Implementation of multipliers.

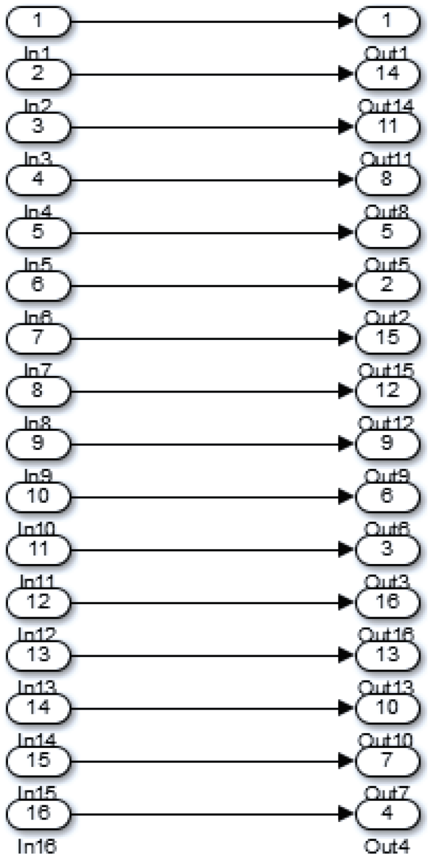


Figure 23.
Implementation of inverse shift row.

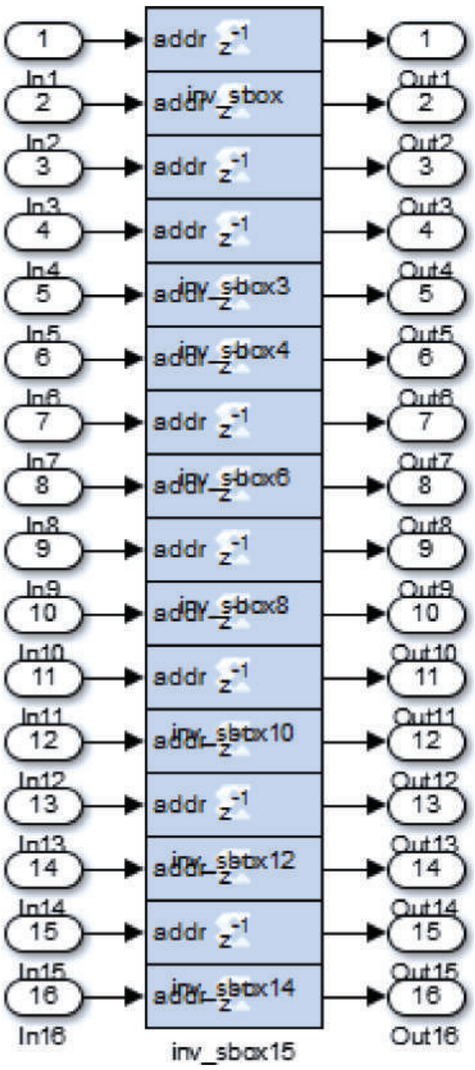


Figure 24.
Implementation of inverse s-box.

r. 16 user LEDs

s. Two tri-color LEDs

t. PDM microphone

u. Temperature sensor

v. Two 4-digit 7-segment displays

w. USB HID Host for mice, keyboards, and memory sticks

x. PMOD for XADC signals

y. 12-bit VGA output

z. Four PMOD ports

4. Experimental results

4.1 RTL schematic

Figure 25 shows detailed RTL schematic of the proposed implementation of AES algorithm.

4.2 Synthesis result

The design is synthesized using Xilinx XST synthesizer. In the proposed design, an optimized and synthesizable very high speed integrated circuit (VHSIC) hardware description language (VHDL) code for the implementation of image as well as 128-bit data encryption is developed so as to utilize less area and increase the speed. **Table 1** shows design utilization summary of the proposed design.

From the synthesis results of the proposed design, it is clear that this system utilizes only 121 slice registers, and its maximum operating frequency is 1102.536 MHz. The throughput of the system is calculated using the following formula:

$$\text{(Throughput) of the system} = \frac{128 \text{ bits} \times \text{Clock frequency}}{\text{Cycles per Encrypted block}} \quad (1)$$

By substituting the values in Eq. (1), throughput of the systems is 14.1125 Gbps.

4.3 Simulation result

Figure 26 shows simulation result when an image is applied as an input.

4.4 Performance analysis

Performance analysis is a must to compare the performance of the proposed implementation with existing methods. The performance is compared on the basis of area and operating frequency. Till date various researchers have worked on FPGA-based implementations of AES algorithm; some of them have optimized speed and

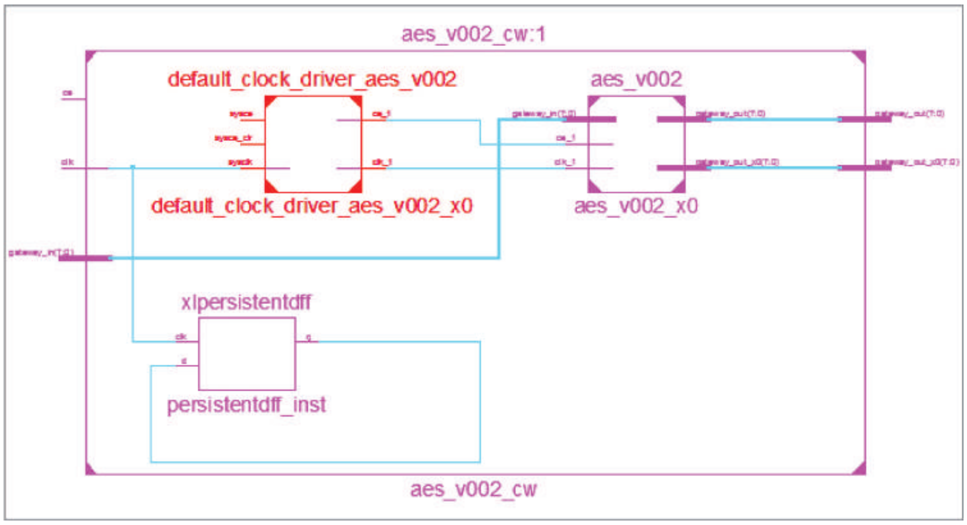


Figure 25.
Detailed RTL schematic of AES algorithm.

Design utilization summary			
Logic utilization	**Used**	**Available**	**% utilization**
Number of slice registers	121	126,800	0.00095
Number of slice LUTs	4782	63,400	7
Number of bonded IOBs	25	210	11

Table 1.
Design utilization summary.

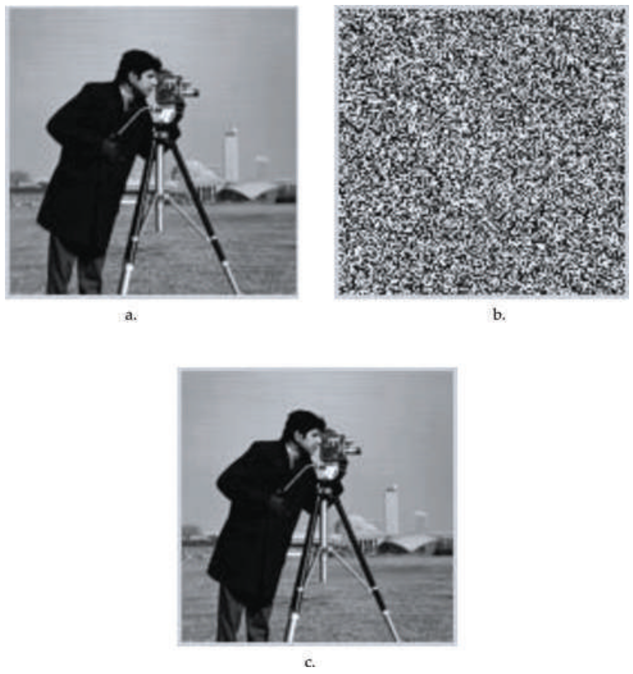


Figure 26.
Simulation result (a) Original image, (b) Encrypted image, and (c) Decrypted image.

Sr. No.	**Authors**	**Slices**	**Operating freq. (MHz)**
1	Proposed work	121	1102.536
2	[3]	1464	—
3	[4]	161	886.64
4	[7]	5493	277.4
5	[8]	3376	—
6	[9]	150	90
7	[10]	201	70
8	[12]	1403	160.875
9	[15]	1746	—
10	[19]	1853	140.390
11	[21]	6279	119.954

Table 2.
Performance comparison of the proposed system with previous work.

some have optimized area. In the proposed system, both area and speed are optimized. **Table 2** shows performance comparison of the proposed system with previous work.

5. Conclusion

In this chapter, fast, area-efficient, and secure implementation of AES algorithm on FPGA is suggested. As per the literature survey, it is clear that Farooq and Faisal Aslam [4] achieved better performance in terms of speed, whereas Ibrahim [9] achieved better performance in terms of area. In this design, due to better Xilinx system generator-based design, the system is optimized, and it utilizes only 121 slice registers at maximum operating frequency of 1102.536 MHz. Also, throughput of the proposed system is 14.1125 Gbps.

Conflict of interest

There is no conflict of interest.

Acronyms and abbreviations

AES	advanced encryption standard
DDR	double data rate
DES	data encryption standard
FPGA	field-programmable gate array
Gbps	gigabits per second
MHz	megahertz
VHDL	VHSIC Hardware Description Language
VHSIC	very high speed integrated circuit

Author details

Altaf O. Mulani* and Pradeep B. Mane
AISSMS Institute of Information Technology, Pune, Maharashtra, India

*Address all correspondence to: aksaltaaf@gmail.com

References

[1] Mulani AO, Mane PB. Watermarking and cryptography based image authentication on reconfigurable platform. Bulletin of Electrical Engineering and Informatics. June 2017;**6**(2):181-187

[2] Ratheesh T, Narayanan S. FPGA based implementation of AES encryption and decryption with low power multiplexer LUT based S-box. IOSR Journal of Electronics and Communication Engineering. April 2017;**12**(2):57-61

[3] Agarwal A, Singh G, Sharma N. Implementation of AES algorithm. International Journal of Engineering Research and Science (IJOER). April 2016;**2**(4):112-116

[4] Farooq U, Faisal Aslam M. Comparative analysis of different AES implementation techniques for efficient resource usage and better performance of an FPGA. Journal of King Saud University-Computer and Information Sciences. March 2016;**29**(3):295-302

[5] Sai Srinivas NS, Akramuddin Md. FPGA based hardware implementation of AES Rijndael algorithm for encryption and decryption. In: IEEE International Conference on Electrical, Electronics and Optimization Techniques; March 2016

[6] Mathur N, Bansode R. AES based text encryption using 12 rounds with dynamic key selection. In: International Conference on Communication, Computing and Virtualization. Elsevier; 2016

[7] Kalaiselvi K, Mangalam H. Power efficient and high performance VLSI architecture for AES algorithm. Journal of Electrical Systems and Information Technology. September 2015;**2**(2):178-183

[8] Deshpande HS, Karande KJ, Mulani AO. Area optimized implementation of AES algorithm on FPGA. In: IEEE International Conference on Communications and Signal Processing (ICCSP); April 2015

[9] Ibrahim A. FPGA based hardware implementation of compact AES encryption hardware core. WSEAS Transactions on Circuits and Systems. 2015;**14**:364-371

[10] Khose PN, Raut VG. Implementation of AES algorithm on FPGA for low area consumption. In: IEEE International Conference on Pervasive Computing (ICPC); January 2015

[11] Mulani AO, Mane PB. Area optimization of cryptographic algorithm on less dense reconfigurable platform. In: IEEE International Conference on Smart Structures and Systems (ICSSS); October 2014

[12] Yewale Minal J, Sayyad MA. Implementation of AES on FPGA. IOSR Journal of VLSI and Signal Processing (IOSR-JVSP). October 2014;**4**(5):65-69

[13] Deshpande HS, Karande KJ, Mulani AO. Efficient implementation of AES algorithm on FPGA. In: IEEE International Conference on Communications and Signal Processing (ICCSP); April 2014

[14] Tonde AR, Dhande AP. Review paper on FPGA based implementation of advanced encryption standard (AES) algorithm. International Journal of Advanced Research in Computer and Communication Engineering. January 2014;**3**(1):4878-4880

[15] Varhade SA, Kasat NN. Implementation of AES algorithm using FPGA and its performance analysis. International Journal of Science and Research. May 2013;**4**(5):2484-2492

[16] Wadi SM, Zainal N. Rapid encryption method based on AES algorithm for Grey scale HD image encryption. In: International Conference on Electrical Engineering and Informatics. Elsevier; 2013

[17] Wang W, Chen J, Xu F. An implementation of AES algorithm based on FPGA. In: IEEE International Conference on Fuzzy Systems and Knowledge Discovery; May 2012

[18] Shylashree N, Bhat N, Shridhar V. FPGA implementations of advanced encryption standard: A survey. International Journal of Advances in Engineering and Technology (IJAET). May 2012;**3**(2):265-285

[19] Borkar AM, Kshirsagar RV, Vyawahare MV. FPGA implementation of AES algorithm. In: IEEE International Conference on Electronics Computer Technology (ICECT);April 2011

[20] Deshpande AM, Deshpande MS, Kayatanavar DN. FPGA implementation of AES encryption and decryption. In: IEEE International Conference on Control, Automation, Communication and Energy Conservation; June 2009

[21] Swinder K, Vig R. Efficient implementation of AES algorithm in FPGA device. In: IEEE International Conference on Computational Intelligence and Multimedia Applications; December 2007

[22] Samanta S. FPGA Implementation of AES Encryption and Decryption. Surat: Sardar Vallabhbhai National Institute of Technology; 2007

[23] Good T, Benaissa M. AES on FPGA from fastest to smallest. In: Proceedings of International Workshop on Cryptographic Hardware and Embedded systems. Springer; 2005