**Thesis Title Page**

Efficient Implementation of IEEE 802.11i Wi-Fi Security (WPA2-PSK) Standard Using

FPGA

A Thesis Submitted to the Graduate Faculty of

GRAND VALLEY STATE UNIVERSITY

In

Partial Fulfillment of the Requirements

For the Degree of

<name of your degree here>

<your academic unit>

<Month of graduation> <Year of graduation>

# **Acknowledgements**

# **Abstract**

The rationale behind the thesis was to design efficient implementations of cryptography algorithms used for Wi-Fi Security as per IEEE 802.11i Wi-Fi Security (WPA2-PSK) standard. The focus was on software implementation of Password-Based Key Derivation Function 2 (PBKDF2) using Keyed-Hash Message Authentication Code (HMAC)-SHA1, which is used for authentication, and , hardware implementation of AES-256 cipher, which is used for data confidentiality.

In this thesis, PBKDF2 based on HMAC-SHA1 was implemented on software using C programming language, and, AES-256 was implemented on hardware using Verilog HDL. The overall implementation was designed and tested on Nexys4 FPGA board. The performance of the implementation was compared with other existing designs. Latency (us) was used as the performance metric for PBKDF2, whereas, throughput (Gb/s), resource utilization (Number of Slices), efficiency (GB/s per slice) and latency (ns) were used as performance metrics for AES-256. MRF24WG0MA PMOD Wi-Fi module was the 2.4 GHz Wi-Fi module which was interfaced with Nexys4 FPGA board for wireless communication.

When the correct security credentials were entered in the implemented system interfaced to the Wi-Fi module, it was successfully authenticated by a 2.4 GHz wireless router (or mobile hotspot) configured to work in WPA2-PSK security mode. Once this system was authenticated, the implemented AES-256 cipher within the system was used to provide a layer of encryption over the data being communicated in the network.

# **Table of Contents**

[**Acknowledgements** 2](#_Toc528185443)

[**Abstract** 3](#_Toc528185444)

[**Table of Contents** 4](#_Toc528185445)

[**List of Figures** 5](#_Toc528185446)

[**List of Tables** 6](#_Toc528185447)

[**Abbreviations** 7](#_Toc528185448)

[**Chapter 1: Introduction** 8](#_Toc528185449)

[**1.1** **Background** 8](#_Toc528185450)

[**1.2** **Wireless Security Principle** 9](#_Toc528185451)

[**1.3** **WPA2-PSK** 10](#_Toc528185452)

[**1.4** **Project Scope** 12](#_Toc528185453)

[**References** 13](#_Toc528185454)

# **List of Figures**

[**Figure 1: Example of Wi-Fi Network** 9](#_Toc528520755)

[**Figure 2: WPA2-PSK Security** 11](#_Toc528520756)

[**Figure 3: WPA2-PSK Authentication** 11](#_Toc528520757)

[**Figure 4: WPA2-PSK Data Confidentiality** 12](#_Toc528520758)

[**Figure 5: Block Diagram for PBKDF2** 16](#_Toc528520759)

[**Figure 6: PBKDF2 with HMAC-SHA1** 17](#_Toc528520760)

[**Figure 7: Block Diagram for HMAC** 19](#_Toc528520761)

[**Figure 8: Block Diagram for SHA1 Processing Function** 23](#_Toc528520762)

[**Figure 9: Rcon( ) Values** 25](#_Toc528520763)

[**Figure 10: Example of ShiftRows()** 25](#_Toc528520764)

[**Figure 11: MixColumns() Calculation** 26](#_Toc528520765)

[**Figure 12: Example if InvShiftRows()** 27](#_Toc528520766)

[**Figure 13: InvMixColumns Calculation** 27](#_Toc528520767)

[**Figure 14: Pseudo Code for Key Expansion for Encryption** 28](#_Toc528520768)

[**Figure 15: Additional Pseudo Code to be Added for Key Expansion for Decryption** 29](#_Toc528520769)

[Figure 16: Input Bytes arranged into state array at beginning of AES operation 29](#_Toc528520770)

[**Figure 17: Pseudo Code for AES Encryption** 30](#_Toc528520771)

[**Figure 18: Pseudo Code for the Equivalent Inverse Cipher** 31](#_Toc528520772)

[**Figure 19: AES-256 Block Diagram** 32](#_Toc528520773)

# **List of Tables**

[**Table 1: S-Box Lookup Table** 33](#_Toc528510515)

[**Table 2: Inverse S-Box Lookup Table** 33](#_Toc528510516)

[**Table 3: Mul2 Lookup Table** 34](#_Toc528510517)

[**Table 4: Mul3 Lookup Table** 34](#_Toc528510518)

[**Table 5: Mul9 Lookup Table** 35](#_Toc528510519)

[**Table 6: Mul11 Lookup Table** 35](#_Toc528510520)

[**Table 7: Mul13 Table** 36](#_Toc528510521)

[**Table 8: Mul14 Table** 36](#_Toc528510522)

# **Abbreviations**

# **Chapter 1: Introduction**

## **Background**

As most technologies have continued to transition from traditional wired systems to wireless ones, the number of wireless devices has grown by leaps and bounds over the last decade. Wireless devices have become a part of our day-to-day lives with its presence seen in household, educational and business institutions, to name a few. These devices are inter-connected with one another and share a variety of data, ranging from mundane to very personal and confidential information. Such interconnected devices that share data among themselves form a network. There can be various types of networks based on topology, size, area, organization, etc. One such type of network based on area is called Local Area Network (LAN). Such network is confined within a localized area such as a room, building or a group of buildings. However, it can be inter-connected to other LANs using wired or wireless media. If wireless medium is used to connect such LANs, then the overall network is called Wireless LAN (WLAN) [1].

The communication between the devices within a network is governed by a set of rules called communication protocols. The devices within a network must adhere to such protocols to successfully share and interpret data among other devices connected to the network. To maintain interoperability between the devices manufactured by various vendors, standardized communication protocols are defined for different type of networks. One such protocol for communication between wireless devices over LAN is the IEEE 802.11 protocol and is commonly known as Wi-Fi [2]. An example of Wi-Fi network is shown in Figure 1.



**Figure 1: Example of Wi-Fi Network**

## **Wireless Security Principle**

Security is paramount in any type of network, but it is more so in the case of wireless networks, as they are far more vulnerable to attack in comparison to wired networks. In a wired network, the communicating devices must be physically connected using a cable. Hence, it is easier to verify the identity of the device to which the data is being communicated, as opposed to in wireless networks, where this is not quite easy. Also, unlike in wired networks, where the data is communicated through copper wires or optical fibers, in wireless networks, the wireless devices use RF signals in open air as their communication medium. So, theoretically any transceiver which is within the range of this RF signal and tuned to its frequency can read and/or meddle with the data being communicated [3]. Hence, for a secure communication, it is necessary to identity whether a device trying to connect to the network has proper security credential or not. This process is called authentication [3]. After a wireless device is authenticated to a network, the data being communicated within that network must be made confidential using a secure cryptography algorithm [3].

## **WPA2-PSK**

The current standardized security protocol for Wi-Fi is IEEE 802.11i standard. This is also commonly known as Wi-Fi Protected Access II (WPA2). WPA2 was launched in September 2004 and supports PSK technology and includes an advanced encryption mechanism using the Counter-Mode/CBC-MAC Protocol (CCMP) called the Advanced Encryption Standard (AES) [4]. The PSK technology (in personal networks) is used to verify the identity of the communicating wireless devices. In PSK, the authentication process is performed by the access point (wireless router, mobile hotspot, etc.). With PSK, we can configure the access point (wireless router or hotspot) with a passphrase of 8 to 63 printable ASCII characters [5]. Using a technology called PBKDF2, that passphrase, along with the network SSID, is used to generate unique encryption keys for wireless clients. In WPA2-PSK security, the same set of SSID and PSK is shared between all Wi-Fi end devices and the access point as shown in Figure 2 [6]. The SSID is analogous to Username and PSK is analogous to Passphrase in Figure 2. The wireless devices are authenticated and granted access to the network, if the password to the particular SSID matches [5]. After authentication, AES cipher is used to maintain the confidentiality of the data being communicated within the network.



**Figure 2: WPA2-PSK Security**

In Figure 3, SSID and Passphrase goes through PBKDF2 to derive the 256-bit PMK which is used as the main key for AES cipher. The validity of this key is confirmed using the 4-way handshake process (Figure 2) between the Wi-Fi device and the access point [6]. If the key matches, then, the Wi-Fi device is successfully authenticated by the access point.



**Figure 3: WPA2-PSK Authentication**

After successful authentication, the data between Wi-Fi end devices and the access point is encrypted using AES cipher with the 256-bit PMK as the main key (Figure 4).



**Figure 4: WPA2-PSK Data Confidentiality**

## **Project Scope**

The purpose of this thesis is to optimize the cryptography algorithms used in device authentication and data confidentiality in Wi-Finetworks configured with WPA2-PSK security. To achieve this, the main key derivation part of the authentication process, as well as, the AES cipher algorithm required for data confidentiality will be optimized. The scope of the implementation will encompass the following areas:

* Efficient software implementation of PBKDF2 based on HMAC-SHA1 which is used for device authentication.
* Efficient hardware implementation of AES-256 cipher which is used for data confidentiality.

The performance of these implementations will be compared with other existing designs. Latency (us) will be used as the performance metric for PBKDF2, whereas, throughput (Gb/s), resource utilization (Number of Slices), efficiency (GB/s per slice) and latency (ns) will be used as performance metrics for AES-256.

# **Chapter 2: Theory**

## **2.1 WPA2-PSK Device Authentication**

Authentication is the process by which you prove that you are eligible to join a network (and that the network is legitimate) [3]. Pre-Shared Key (PSK) is a device authentication method used in WPA2-PSK networks, and it uses a passphrase of 8 to 63 printable ASCII characters to generate unique encryption keys [5]. The general idea of PSK mode is to use the same secret key on an access point and on a Wi-Fi device to authenticate the device and establish an encrypted connection for networking [6]. Hence, both Wi-Fi device and access point must prove to each other that they know the pre-shared key to ensure a secure connection. In WPA2-PSK, the access point (wireless router, hotspot, etc.) with a network SSID is configured with a passphrase. Using PBKDF2, that passphrase along with network SSID is used to generate the 256-bit Pairwise-Master-Key (PMK). The Wi-Fi device must also derive the same PMK using the same passphrase and SSID for the access point to authenticate the device.

PMKs are never transmitted across the network as the channel of communication is not secure before the authentication process has completed. Because, without authentication, sharing of PMK would be done through an unencrypted channel and susceptible to be discovered by outside parties [7]. To overcome this, WPA2-PSK uses 4-way handshake to verify whether the Wi-Fi device and the access point have the same PMK or not (Figure 2). The 4-way handshake is designed so that the access point and Wi-Fi device can independently prove to each other that they know the PMK, without ever disclosing it. In Figure 2, the 4-way handshake is broken down into 4 messages [7]:

* **Message 1 (From Access Point to Wi-Fi Device):** The first step is for the access point to generate a nonce value. The nonce value is a pseudo random value generated by a publicly known and repeatable process. This pseudo random value is generated by the Pseudo Random Function 256 or PRF-256, as defined by WPA2 specifications. The nonce value generated by access point is called A-nonce. The access point sends a message containing this A-nonce value to the Wi-Fi device.
* **Message 2 (From Wi-Fi Device to Access Point):** The Wi-Fi device generates a nonce value using the same process as the access point and it is denoted as S-nonce. When the Wi-Fi device receives Message 1, it will generate Pairwise-Transient-Key (PTK). This key is required to be generated by both parties, and allows each party to verify that the other has the correct PMK. The creation of PTK is performed via another Pseudo Random Function (PRF), which uses a combination of the PMK, Access Point MAC Address, Wi-Fi Device MAC Address, A-nonce and S-nonce [8]. A part of the PTK is known as the message integrity check (MIC). This value, along with the S-Nonce is then transmitted back to the access point.
* **Message 3 (From Access Point to Wi-Fi Device):** When the access point receives Message 2, it has all the values required to generate the PTK. The access point then generates the PTK, and checks whether the MIC value in Message 2 matches the MIC value that it has just generated. If the two MIC values matches, this proves that the Wi-Fi device knows the value of the PMK. If the MIC value is correct, the access point, then sends Message 3 to the Wi-Fi device. Message 3 allows the Wi-Fi device to ensure that the access point is a trusted party. If the access point did not have a matching PMK, the MIC would be different. Message 3 also informs the Wi-Fi device that the communication channel is about to be encrypted.
* **Message 4 (From Wi-Fi Device to Access Point):** The final part of the handshake allows the Wi-Fi device to acknowledge that the access point is now going to use encryption for the communication. After the Wi-Fi device transmits Message 4, it will install the encryption keys on the channel. After the access point receives message 4, it will install the encryption keys as well. All further unicast communication is protected by this encryption, until the client disconnects from the access point [3].

### **2.1.1 PBKDF2 (Password-Based Key Derivation Function 2)**

Using PBKDF2, the passphrase and SSID are hashed 4096 times to produce a 256-bit PMK [9]. Internally, the PBKDF2 key derivation function employed in WPA2-PSK utilizes 4096 iterations of HMAC-SHA1 to obtain 160-bit hash outputs. Since the PMK in WPA2-PSK is of 256-bits, two rounds of PBKDF2 are necessary [10]. Their outputs are concatenated, but for the second iteration the output is truncated to 64 bits to achieve the 256-bit PMK. The PBKDF2 key derivation function is defined as follow:

DK = PBKDF2(PRF, P, S, C, dkLen) ................... (1)

where,

DK: Derived key

PRF: Pseudorandom function of two parameters with output length hLen

P: Password

S: Salt (sequence of bits)

C: Iteration count, a positive integer

dkLen: Length of Derived key

To derive key from PBKDF2, each hLen bit block Ti of derived key DK, is computed as follows:

DK = T1 || T2 || ... || Tdklen/hlen ..................... (2)

Ti = F (P, S, C, i) ...................... (3)

In equation (3), the function F is the Exclusive-OR operations of C iterations of PRFs (as shown in equation (4)). In the first iteration, the PRF uses Password as the key and Salt concatenated with i (encoded as a big-endian 32-bit integer) as the 2 parameters (as shown in equation (5)). For, subsequent iterations, PRF uses Password as the key and the output of the previous PRF computation as the salt (as shown in equations (6) and (7)). The block diagram for PBKDF2 key derivation function is shown is Figure 5.

F (Password, Salt, C, i) = U1 ⊕ U2 ⊕ ... ⊕ Uc .... (4)

where,

U1 = PRF (Password, Salt || INT\_32\_BE(i)) ......... (5)

U2 = PRF (Password, U1) ........................... (6)

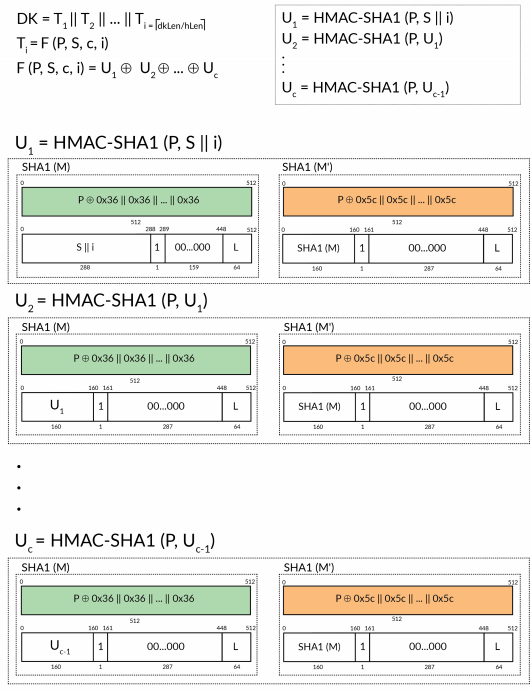
...

Uc = PRF (Password, Uc - 1) ......................... (7)



**Figure 5: Block Diagram for PBKDF2**

The overview of PBKDF2 along with HMAC and SHA1 is shown in Figure 6 [11].



**Figure 6: PBKDF2 with HMAC-SHA1**

In case of WPA2-PSK, the output and parameters in equation (1) are as follows:

PMK = PBKDF2(HMAC−SHA1, pwd, ssid, 4096, 256) ...... (8)

### **2.1.2 HMAC (Keyed-Hashing for Message Authentication)**

HMAC provides a mechanism to calculate a message authentication code (MAC) based around a cryptographic hashing function [7]. A message authentication code (MAC) is a short piece of information used to authenticate a message. MACs are used between two parties that share a secret key to validate information transferred between them [12]. The use of HMAC in PBKDF2 is shown in Figure 6. The definition of HMAC requires a cryptographic hash function denoted by H, the secret key denoted by M and the message to be authenticated denoted by m. The HMAC function is defined as follows:

HMAC (M, m) = H ((M' ⊕ opad) || H ((M' ⊕ ipad)|| m)) ..........(9)

where,

H: a cryptographic hash function

M: the secret key

m: the message to be authenticated

M': another secret key, derived from the original key K

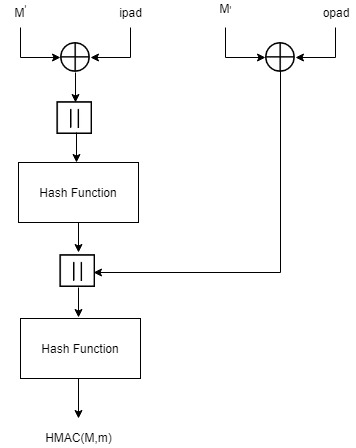
(by padding K to the right with extra zeroes to the

input block size of the hash function, or by hashing K if it is longer than that block size)

opad: the outer padding (0x5c5c5c…5c5c, one-block- long hexadecimal constant

ipad: the inner padding (0x363636…3636, one-block- long hexadecimal constant).

The block diagram for HMAC function is shown in Figure 7.



**Figure 7: Block Diagram for HMAC**

For WPA2-PSK, the parameters in equation (9) are as follows:

HMAC-SHA1 (pwd, ssid) = SHA1 ((pwd ⊕ opad) || SHA1((pwd

⊕ opad ⊕ ipad) ||ssid) ... (10)

### **2.1.3 SHA1**

Hashing algorithms are used to process a message and produce a condensed representation of the message which is called a message digest, and for a perfect hashing function, it should be only one-way and a unique digital signature of the message [7]. The use of SHA1 in PBKDF2 is shown in Figure 6. The SHA1 hashing algorithm is valid for messages with a size less than 264 bits, it operates on blocks of size 512 bits, it uses a word size of 32 bits, and has a resultant message digest of 160 bits [7]. SHA1 algorithm primarily consists of 6 steps [13]:

**Step1: Append Padding Bits**: The original message is padded based on the following rules:

* The original message is first padded with one bit ‘1’.
* Zeros ‘0’ are then padded to bring the length of message to 64 bits less than multiple of 512.

**Step2: Append Length:** A 64-bit value indicating the length of the original message is appended to end the message obtained from Step 1 based on the following rules:

* 64-bit value of the original message is appended at the end of the padded message. If overflow occurs, the lower order of the 64-bit value is appended.
* The lower 32-bit word of the 64-bit value is appended first followed by the upper 32-bit value.

**Step3: Prepare Processing Functions:** SHA1 has 80 processing rounds. There are 4 mathematical operations assigned to each of the 4 sets of 20 rounds. These operations are as follows:

for 0 <= r <= 19,

F (r: B, C, D) = (B & C) | ((! B) & D) .............(11)

for 20 <= r <= 39,

F (r: B, C, D) = B ⊕ C ⊕ D................(12)

for 40 <= r <= 59,

F (r: B, C, D) = (B & C) | (B & D) | (C & D) .......(13)

for 60 <= r <= 79,

F (r: B, C, D) = B ⊕ C ⊕ D .....................(14)

**Step4: Prepare Processing Constants**: SHA1 has 4 different constants assigned to 4 sets of 20 rounds of SHA1. These constants are as follows:

for 0 <= r <= 19,

K(r) = 0x5A827999 ............................... (15)

for 20 <= r <= 39,

K(r) = 0x6ED9EBA1 ............................... (16)

for 40 <= r <= 59,

K(r) = 0x8F1BBCDC ............................... (17)

for, 60 <= r <= 79

K(r) = 0xCA62C1D6 ............................... (18)

**Step5: Initialize Buffer:** SHA1 has five 32-bit buffers which are initialized as follows:

H0 = 0x67452301 ................................. (19)

H1 = 0xEFCDAB89 ................................. (20)

H2 = 0x98BADCFE ................................. (21)

H3 = 0x10325476 ................................. (22)

H4 = 0xC3D2E1F0 ................................. (23)

**Step6: Process 512-bit block messages:** The algorithm to process this 512-bit block of message is as follows:

For loop on k = 1 to N /\* 1st For loop \*/

(W (0), W (1) ..., W (15)) = M[k] /\*Divide M[k] into 16 words\*/

For t = 16 to 79 do: /\* 2nd For loop \*/

     W(t) = (W(t-3) XOR W(t-8) XOR W(t-14) XOR W (t-16)) << 1

End of For loop /\* 2nd For loop \*/

 A = H0, B = H1, C = H2, D = H3, E = H4

  For t = 0 to 79 do: // 3rd for loop

        TEMP = A<<5 + f (t: B, C, D) + E + W(t) + K(t)

        E = D, D = C, C = B<<30, B = A, A = TEMP

   End of For loop // 3rd for loop

  H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D,

H4 = H4 + E

  End of for loop /\* End of 1st For loop \*/

Output = H0 << 128 | H1 << 96 | H2 << 64 | H3 << 32 | H4

The block diagram for SHA1 processing function is given in Figure 8.



**Figure 8: Block Diagram for SHA1 Processing Function**

## **2.2 WPA2-PSK Data Confidentiality**

WPA2-PSK uses Advanced Encryption Standard (AES) cipher for data security. The AES algorithm is a symmetric block cipher that can encrypt and decrypt information. Encryption converts data to an unintelligible form called ciphertext and decryption converts the ciphertext back into its original form, called plaintext [14]. AES has input block size of 128 bits and key size can be of 128, 192 or 256 bits. WPA2-PSK uses key size of 256 bits i.e. AES-256. AES-256 requires 60 rounds for key expansion and the size of the expanded key is 240 bytes. If the decryption is performed using Equivalent Inverse Cipher method, then there are separate set of expanded keys for encryption and decryption processes [14]. Hence, in total, there will be 480 bytes of expanded keys if decryption is performed using Equivalent Inverse Cipher method . AES-256 requires 14 rounds each for the completion of encryption and decryption processes and the size of output block is 128 bits.

AES consists of key expansion, encryption and decryption processes. These processes are realized using the following transformations [14]:

* **RotWord():**This function takes 4 bytes as an argument. It performs circular left shift on the 4 bytes.

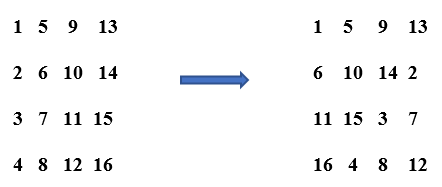
Example: 1,2,3,4 to 2,3,4,1

* **Rcon ():** This function returns a 4-byte value based on the Table 2.



**Figure 9: Rcon( ) Values**

* **AddRoundKey():** In the AddRoundKey() transformation, a Round Key is added to the state by a simple bitwise XOR operation.
* **SubBytes():**In this transformation, each byte of data is substituted with the corresponding value from the S-box lookup table (Table1 in Appendix 1.1).
* **ShiftRows():**This function arranges the bytes of the state in 4x4 matrix and performs byte-wise circular left shift. The order of the shift varies with rows. The shift operation is not performed for the first row. The second row is shifted by 1 byte, the third row is shifted by 2 bytes and the fourth row is shifted by 3 bytes. An example of shift row operation is shown in Figure 10.



**Figure 10: Example of ShiftRows()**

* **MixColumns():**The matrix obtained from the Shift Row () operation goes through the multiplication over Galois Field (Figure 11) . The lookup tables required for multiplication over Galois Field in theMix Column () operation are shown in Table 3 and Table 4 in Appendix 1.1.



**Figure 11: MixColumns() Calculation**

* **InvSubBytes():**In this transformation, each byte of data is substituted with the corresponding value from the inverse S-box table (Table2 in Appendix 1.1).
* **InvShiftRows():**This function arranges the bytes of the state in 4x4 matrix and performs byte-wise circular right shift. The order of the shift varies with rows. The shift operation is not performed for the first row. The second row is shifted by 1 byte, the third row is shifted by 2 bytes and the fourth row is shifted by 3 bytes. An example of shift row is shown in Figure 12.



**Figure 12: Example if InvShiftRows()**

* **InvMixColumns():**The matrix obtained from the Inverse Shift Row () operation goes through the multiplication over Galois Field (Figure 13) . The lookup tables required for multiplication over Galois Field in theInverse Mix Column () operation are shown in Table 5, Table 6, Table 7 and Table 8 in Appendix 1.1

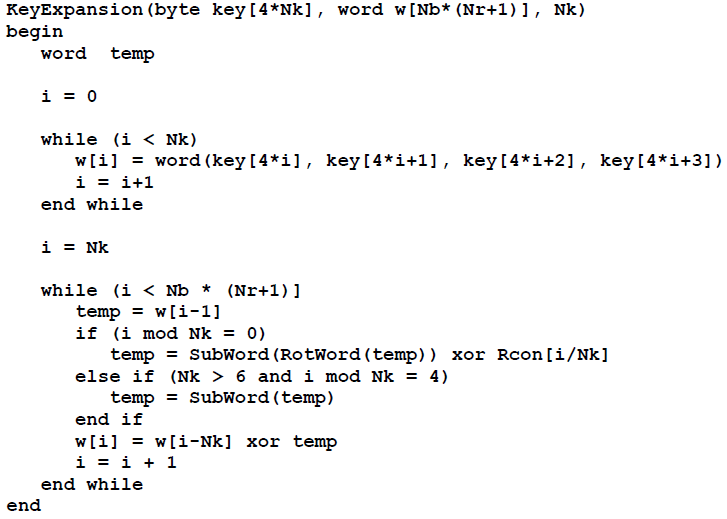


**Figure 13: InvMixColumns Calculation**

### **2.2.1 AES-256 Key Expansion**

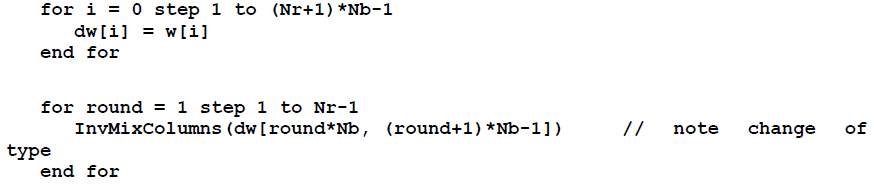
The AES algorithm takes the Cipher Key and performs a Key Expansion routine to generate a key schedule. The Key Expansion generates a total of Nb (Nr + 1) words: the algorithm requires an initial set of Nb words, and each of the Nr rounds requires Nb words of key data. The resulting key schedule consists of a linear array of 4-byte words, denoted [wi ], with i in the range 0 £ i < Nb(Nr + 1) [14].

In AES-256, Nb = 4, and Nr = 14, so the key expansion generates a total of 60 words (240 bytes). The key expansion routine runs for 14 rounds and generates 240 bytes of expanded key. Two different sets of 240 bytes of expanded keys are generated for encryption and decryption, when the decryption is done using equivalent inverse cipher method. The expansion of the input key into the key expansion routine proceeds according to the pseudo code in Figure 14.



**Figure 14: Pseudo Code for Key Expansion for Encryption**

For the Equivalent Inverse Cipher, the following pseudo code shown in Figure 15 must be added to the end of the pseudo code shown in Figure 14.

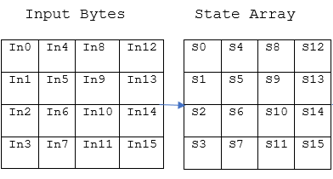


**Figure 15: Additional Pseudo Code to be Added for Key Expansion for Decryption**

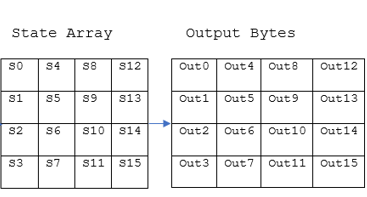
### **2.2.2 AES-256 Encryption**

In each round of AES encryption, the cipher makes four different transformations to the block of data. The four transformations are: AddRoundKey (), SubBytes (), ShiftRows () and MixColumns (). The exception is the final round, which only has three transformations since it does not have the MixColumns()operation. During these rounds each block of data is depicted as 4 x 4 byte matrix and also the key is divided to 4 x 4 byte subkeys. Each round gets these matrices as an input and produces 4 x 4 byte state matrix as an output.

AES algorithm’s operations are performed on a two-dimensional array of bytes called the **State** [14]. For AES-256, the State consists of four rows of bytes, each containing 4 bytes. At the start of the encryption and decryption, the input – the array of bytes In0, In1, … In15 – is copied into the State array as illustrated in Figure 16. The encryption and decryption operations are then conducted on this State array, after which its final value is copied to the output – the array of bytes Out0, Out1, … Out15 as shown in Figure 17. [14].

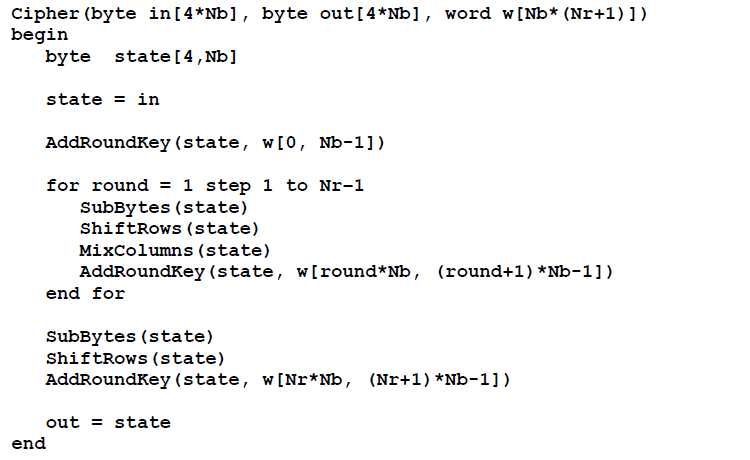


**Figure 16: Input Bytes Arranged in State Array at Beginning of AES Operation**



**Figure 17: Output Bytes Arranged from State Array at End of AES Operation**

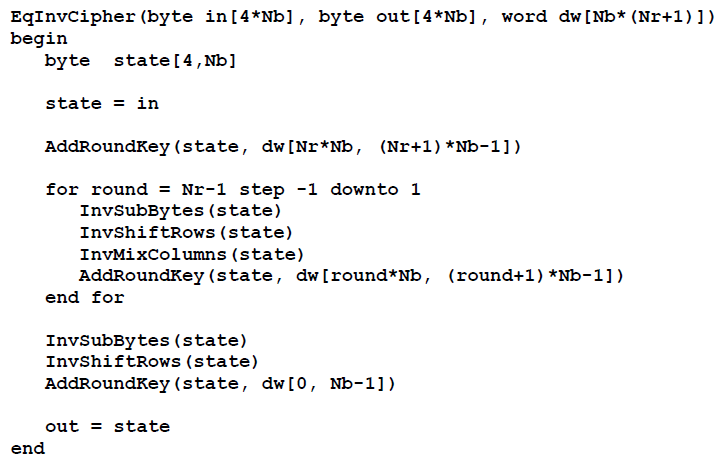
The pseudo code for AES encryption is shown in Figure 18. For AES-256, Nb = 4 and Nr = 14.



**Figure 18: Pseudo Code for AES Encryption**

### **2.2.3 AES-256 Decryption**

Equivalent Inverse Cipher method of decryption follows the same sequence of transformations as the encryption, but with the transformations replaced by their inverses [14]. It also uses a different set of expanded keys than the keys used for encryption. Pseudo code for the Equivalent Inverse Cipher is shown in Figure 19.



**Figure 19: Pseudo Code for the Equivalent Inverse Cipher**

The overall block diagram of AES is shown in Figure 18.



**Figure 20: AES-256 Block Diagram**

# **Appendices**

## **1.1 Lookup Tables**

**Table 1: S-Box Lookup Table**



**Table 2: Inverse S-Box Lookup Table**



**Table 3: Mul2 Lookup Table**



**Table 4: Mul3 Lookup Table**



**Table 5: Mul9 Lookup Table**



**Table 6: Mul11 Lookup Table**



**Table 7: Mul13 Table**



**Table 8: Mul14 Table**



# **References**

|  |  |
| --- | --- |
| [1] | M. Souppaya and K. Scarfone, *Guidelines for Securing Wireless Local Area Networks (WLANs),* NIST Special Publication 800-153, 2012. |
| [2] | S. Banerji and R. S. Chowdhury, *On IEEE 802.11: Wireless LAN Technology,* International Journal of Mobile Network Communications & Telematics (IJMNCT) , 2013. |
| [3] | J. Edney and W. A. Arbaugh, Real 802.11 Security: Wi-Fi Protected Access and 802.11i, Addison-Wesley, 2004. |
| [4] | Wi-Fi Alliance, *Deploying Wi-Fi Protected Access,* 2005. |
| [5] | Juniper Networks, "Understanding PSK Authentication," Juniper Networks, 16 October 2018. [Online]. Available: https://www.juniper.net/documentation/en\_US/junos-space-apps/network-director3.0/topics/concept/wireless-wpa-psk-authentication.html. [Accessed 24 October 2018]. |
| [6] | V. Fadyushin, "Common WLAN Protection Mechanisms and their Flaws," 16 March 2016. [Online]. Available: https://hub.packtpub.com/common-wlan-protection-mechanisms-and-their-flaws/. [Accessed 24 October 2018]. |
| [7] | T. B. Johnson, "An FPGA architecture for the recovery of WPA/WPA2 keys," Graduate Theses and Dissertations, 2014. |
| [8] | "Understanding WPA/WPA2 Pre-Shared-Key Cracking," 27 December 2015. [Online]. Available: https://www.ins1gn1a.com/understanding-wpa-psk-cracking/. [Accessed 26 October 2018]. |
| [9] | O. Nakhila, A. Attiah, Y. Jin and C. Zou, "Parallel Active Dictionary Attack on WPA2-PSK Wi-Fi Networks". |
| [10] | M. Kammerstetter, M. Muellner, D. Burian, C. Kudera and W. Kastner, "Efficient High-Speed WPA2 Brute Force Attacks using Scalable Low-Cost FPGA Clustering," 2016. |
| [11] | A. Visconti, S. Bossi, H. Ragab and A. Calò, *On the weaknesses of PBKDF2,* 2015. |
| [12] | H. Krawczyk, B. M. and R. Canetti, *RFC2104: HMAC: Keyed-Hashing for Message Authentication,* 1997. |
| [13] | H. Yang, "SHA1 Message Digest Algorithm Overview," 2017. [Online]. Available: http://www.herongyang.com/Cryptography/SHA1-Message-Digest-Algorithm-Overview.html. [Accessed 27 October 2018]. |
| [14] | National Institute of Standards and Technology(NIST), *Announcing the ADVANCED ENCRYPTION STANDARD (AES),* Federal Information Processing Standards Publication 197, 2001. |