Published in Cryptologia, XXVI (4), 2002.

Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students

Raphael Chung-Wei **Phan**

ADDRESS: Swinburne Sarawak Institute of Technology, 1st Floor, State Complex, 93576 Kuching, Sarawak, Malaysia. rphan@swinburne.edu.my

ABSTRACT: In this paper, we present a mini version of Rijndael, the symmetric-key block cipher selected as the Advanced Encryption Standard (AES) recently. Mini-AES has all the parameters significantly reduced while at the same time preserving its original structure. It is meant to be a purely educational cipher and is not considered secure for actual applications. The purpose is such that once undergraduate students and amateur cryptanalysts have grasped the basic principles behind how Mini-AES works, it will be easy for them to move on to the real AES. At the same time, an illustration of how the Square attack can be applied to Mini-AES is presented in the hope that Mini-AES would also serve as a testbed for students to begin their cryptanalysis efforts.

KEYWORDS: Advanced Encryption Standard, Rijndael, Block cipher, Cryptanalysis, Square attack

1 Introduction

The National Institute of Standards and Technology (NIST) issued in 1997 a call for proposals for the Advanced Encryption Standard (AES) [7]. Twenty one proposals were submitted, out of which 15 were accepted. Two years later, after undergoing public review and analysis, the list was narrowed down to 5 finalists, and more extensive analysis ensued. In October 2000, Rijndael emerged as the winner and was selected as the Advanced Encryption Standard [8]. The specifications of the AES are now available as a Federal Information Processing Standard (FIPS) [9].

The AES has a block size of 128 bits, and supports key sizes of 128, 192 and 256 bits. The number of rounds is 10, 12 or 14 for the three different key sizes respectively. Just like the DES, the AES is expected to draw much attention from cryptographers and cryptanalysts alike within the space of time from now until the next few decades. In order to aid undergraduate cryptography students and aspiring cryptanalysts in better understanding the internal workings of the AES, we present a mini version of the AES, with all the parameters significantly reduced while preserving its original structure. This mini version is purely educational and hence it is hoped to aid students in grasping the underlying concepts in the design of Rijndael-like ciphers and also to serve as a testbed for aspiring cryptanalysts to try out various cryptanalytic attacks.

In section 2, we present the mathematical background to help the student in understanding the components of Mini-AES. We then proceed to describe Mini-AES in Section 3. In Section 4, we relate Mini-AES to the real AES. The Square attack, a fairly new cryptanalytic attack popularised by Rijndael is presented in detail in Section 5. We conclude in Section 6.

2 Mathematical Background

Mini-AES has a component, NibbleSub, which operates on a nibble (4 bits) at a time. In addition, another component, MixColumn operates on words of 4 nibbles. In this section, we

present the mathematical background needed for the reader to have a clearer understanding of the components of Mini-AES.

2.1 The Finite Field GF(2^4)

The nibbles of Mini-AES can be thought of as elements in the finite field GF(2^4). Finite fields have the special property that operations (+,–, × and ÷) on the field elements always cause the result to be also in the field. Consider a nibble n = (n_3, n_2, n_1, n_0) where $n_i \in \{0,1\}$. Then, this nibble can be represented as a polynomial with binary coefficients i.e having values in the set {0,1}:

$$n = n_3 x^3 + n_2 x^2 + n_1 x + n_0$$

Example 1
Given a nibble, n = 1011, then this can be represented as

$$n = 1 x^3 + 0 x^2 + 1 x + 1 = x^3 + x + 1$$

Note that when an element of GF(2^4) is represented in polynomial form, the resulting polynomial would have a degree of at most 3.

2.2 Addition in GF(2^4)

When we represent elements of GF(2^4) as polynomials with coefficients in {0,1}, then addition of two such elements is simply addition of the coefficients of the two polynomials. Since the coefficients have values in {0,1}, then the addition of the coefficients is just modulo 2 addition or exclusive-OR denoted by the symbol ⊕. Hence, for the rest of this paper, the symbols + and ⊕ are used interchangeably to denote addition of two elements in GF(2^4).

Example 2
Given two nibbles, n = 1011 and m = 0111, then the sum, n + m = 1011 + 0111 = 1100 or in polynomial notation:

$$n + m = (x^3 + x + 1) + (x^2 + x + 1) = x^3 + x^2$$

2.3 Multiplication in GF(2^4)

Multiplication of two elements of GF(2^4) can be done by simply multiplying the two polynomials. However, the product would be a polynomial with a degree possibly higher than 3.

Example 3
Given two nibbles, n = 1011 and m = 0111, then the product is:

$$(x^3 + x + 1)(x^2 + x + 1) = x^5 + x^4 + x^3 + x^3 + x^2 + x + x^2 + x + 1$$
$$= x^5 + x^4 + 1$$

In order to ensure that the result of the multiplication is still within the field GF(2^4), it must be reduced by division with an irreducible polynomial of degree 4, the remainder of which will be taken as the final result. An irreducible polynomial is analogous to a prime number in arithmetic, and as such a polynomial is irreducible if it has no divisors other than 1 and itself. There are many such irreducible polynomials, but for Mini-AES, it is chosen to be:

$$m(x) = x^4 + x + 1$$

Example 4
Given two nibbles, n = 1011 and m = 0111, then the final result after multiplication in GF(2^4), called the 'product of n × m modulo m(x)' and denoted as ⊗, is:

$$(x^3 + x + 1) \otimes (x^2 + x + 1) = x^5 + x^4 + 1 \text{ modulo } x^4 + x + 1$$
$$= x^2$$

This is because:

$$\begin{array}{r} x + 1 \\ x^4 + x + 1 \overline{)\, x^5 + x^4 + 1} \\ + \; x^5 + x^2 + x \\ \hline x^4 + x^2 + x + 1 \\ + \quad x^4 + \quad x + 1 \\ \hline x^2 \end{array}$$

(quotient)

(remainder)

Note that since the coefficients of the polynomials are in {0,1}, then addition is simply exclusive-OR and hence subtraction is also exclusive-OR since exclusive-OR is its own inverse.

Refer to the Appendix for the table describing the multiplication of two elements in GF(2^4) modulo $x^4 + x + 1$.

3 Mini-AES

In order to encrypt messages with Mini-AES, the original input message, called the plaintext is broken up into blocks of 16 bits each. At any one time, only one plaintext block is encrypted with Mini-AES into ciphertext, after which the next plaintext block is encrypted and the process repeats until all of the plaintext blocks have been encrypted. Mini-AES encryption is done with a secret key of 16 bits. Figure 1 illustrates the process of encrypting the plaintext message with Mini-AES.

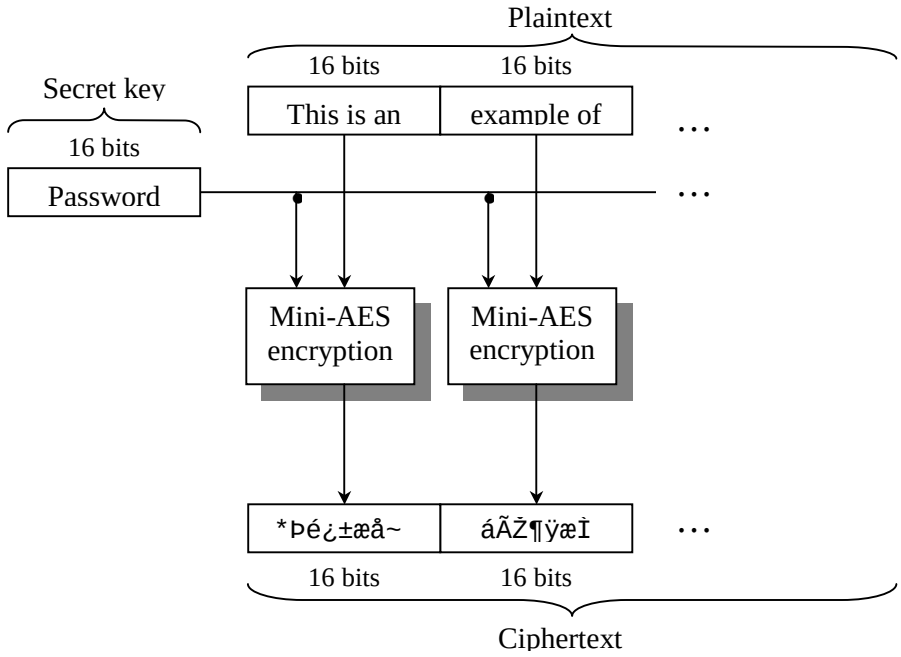

Figure 1: Encrypting A Plaintext Message with Mini-AES

3.1 Mini-AES Components

To make it easier to describe the internal process of the Mini-AES encryption, the input plaintext block of 16 bits, $P = (p_0, p_1, p_2, p_3)$ is represented as a matrix of 2 rows and 2 columns of 4 bits (a nibble), as given in Figure 2.

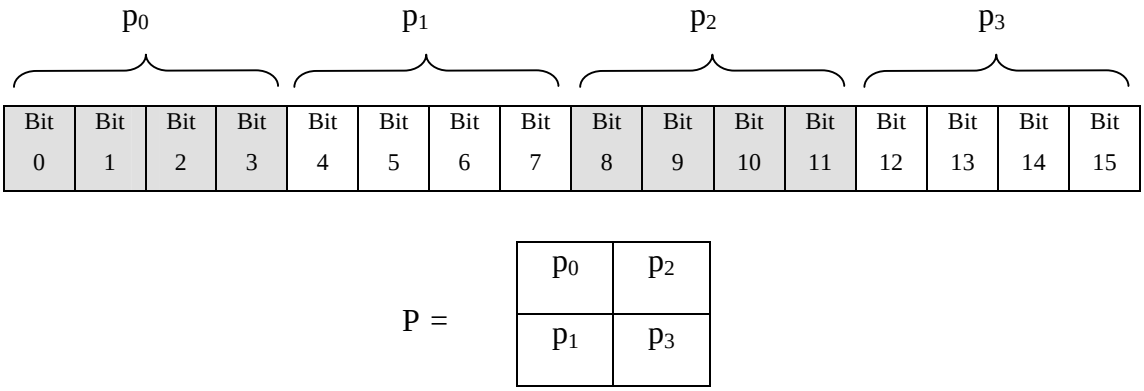

Figure 2: 2×2 Matrix Representation of the 16-bit Block

Within the Mini-AES encryption process, there are 4 main components, namely *NibbleSub*, *ShiftRow*, *MixColumn* and *KeyAddition*. The application of these 4 components in sequence constitutes a *round* of Mini-AES.

3.2 NibbleSub, γ

NibbleSub is a simple operation that substitutes each input nibble with an output nibble according to a 4×4 substitution table (S-box), as given in Table 1. The values in Table 1 are in fact taken from the 1st row of the first S-box in DES.

Input	Output		Input	Output
0000	1110		1000	0011
0001	0100		1001	1010
0010	1101		1010	0110
0011	0001		1011	1100
0100	0010		1100	0101
0101	1111		1101	1001
0110	1011		1110	0000
0111	1000		1111	0111

Table 1: S-box of Mini-AES

The NibbleSub operation is illustrated in Figure 3, where $A = (a_0, a_1, a_2, a_3)$ is the input block and $B = (b_0, b_1, b_2, b_3)$ is the output.

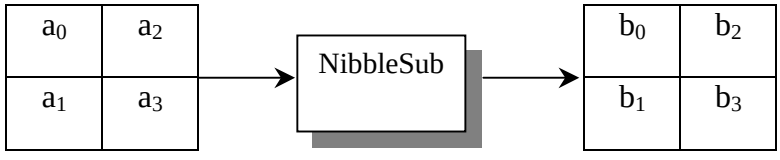

Figure 3: The NibbleSub Operation

Example 5
For an input nibble, $a_0 = 1111$, then based on Table 1, the output nibble is $b_0 = 0111$.

3.3 ShiftRow, π

ShiftRow rotates each row of the input block to the left by different nibble amounts. The first row is unchanged while the second row is rotated left by one nibble. This is illustrated in Figure 4, where $B = (b_0, b_1, b_2, b_3)$ and $C = (c_0, c_1, c_2, c_3)$ are the input and output respectively.

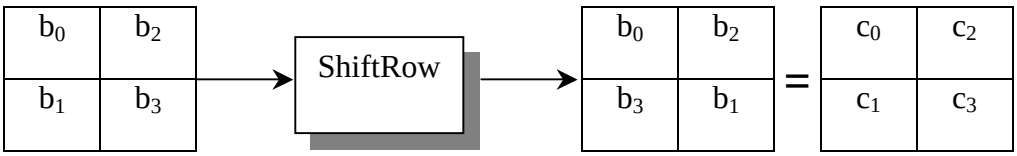

Figure 4: The ShiftRow Operation

Example 6
With an input block $B = (b_0, b_1, b_2, b_3)$, the output block obtained after ShiftRow is $C = (c_0, c_1, c_2, c_3) = (b_0, b_3, b_2, b_1)$.

3.4 MixColumn, θ

MixColumn takes each column of the input block and multiplies it with a constant matrix to obtain a new output column, as given in Figure 5. $C = (c_0, c_1, c_2, c_3)$ and $D = (d_0, d_1, d_2, d_3)$ denote the input and output respectively.

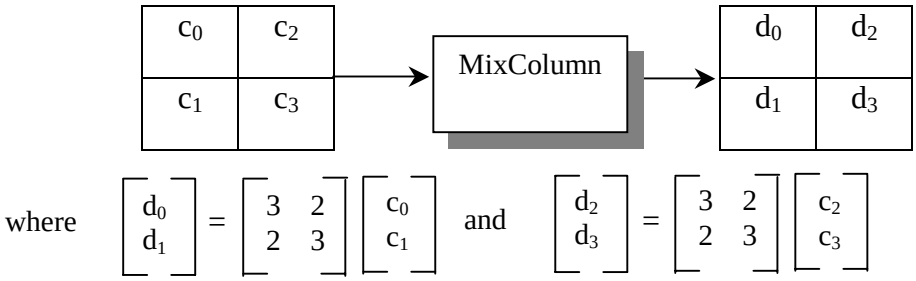

Figure 5: The MixColumn Operation

Example 7
Let an input block be $C = (c_0, c_1, c_2, c_3)$, and the block at the output of MixColumn be $D = (d_0, d_1, d_2, d_3)$. We rearrange the input block as a 2×2 matrix as shown in Figure 5. Then, taking the first column and multiplying it with the constant matrix in Figure 5, we have the output column:

$$\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \end{bmatrix}$$

Hence, $d_0 = (0011 \otimes c_0) + (0010 \otimes c_1)$ and $d_1 = (0010 \otimes c_0) + (0011 \otimes c_1)$

Similarly, taking the second column and multiplying it with the constant matrix, we obtain:

$$\begin{bmatrix} d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} c_2 \\ c_3 \end{bmatrix}$$

Hence, $d_2 = (0011 \otimes c_2) + (0010 \otimes c_3)$ and $d_3 = (0010 \otimes c_2) + (0011 \otimes c_3)$. We see that each output nibble is a function of the two input nibbles in that same column.

3.5 KeyAddition, σ_{Ki}

KeyAddition causes each bit of the input block, D = (d_0, d_1, d_2, d_3) to be exclusived-ORed with the corresponding bit of the i^{th} round key, K_i = (k_0, k_1, k_2, k_3) to obtain the 16-bit output block E = (e_0, e_1, e_2, e_3) as shown in Figure 6. The round key is derived from the secret key, K by using the key schedule, which will be described in Section 3.6. For each bit, the exclusive-OR operation causes the output bit to be '1' if the corresponding bits of the input block and round key are different. Otherwise, the output bit is '0'.

d_0	d_2		k_0	k_2		e_0	e_2
d_1	d_3	$\oplus$	k_1	k_3	=	e_1	e_3

Figure 6: The KeyAddition Operation

Example 8

Given an input block be D = (d_0 , d_1 , d_2 , d_3) = 1111 0000 1010 1100, and the round key K_i = (k_0 , k_1 , k_2 , k_3) = 0101 0011 1111 0000, then the output block, E = (e_0 , e_1 , e_2 , e_3) is:

$$E = D \oplus K_i = 1111\ 0000\ 1010\ 1100 \oplus 0101\ 0011\ 1111\ 0000$$
$$= 1010\ 0011\ 0101\ 1100$$

3.6 The Mini-AES Key-schedule

In Mini-AES, the 16-bit secret key is passed through a key-schedule to produce one 16-bit round key, K_0 to be used prior to the first round, and a 16-bit round key, K_i for use in each round of Mini-AES. Mini-AES encryption is defined to have 2 rounds, hence three round keys, K_0, K_1 and K_2 are generated.

Round	Round Key Values
0	$w_0 = k_0$ $w_1 = k_1$ $w_2 = k_2$ $w_3 = k_3$
1	$w_4 = w_0 \oplus \text{NibbleSub}(w_3) \oplus \text{rcon}(1)$ $w_5 = w_1 \oplus w_4$ $w_6 = w_2 \oplus w_5$ $w_7 = w_3 \oplus w_6$
2	$w_8 = w_4 \oplus \text{NibbleSub}(w_7) \oplus \text{rcon}(2)$ $w_9 = w_5 \oplus w_8$ $w_{10} = w_6 \oplus w_9$ $w_{11} = w_7 \oplus w_{10}$

Table 2: Generation of the Round Keys of Mini-AES

Denote the 16-bit secret key, K as 4 nibbles, K = (k_0, k_1, k_2, k_3), and likewise, K_0 = (w_0, w_1, w_2, w_3), K_1 = (w_4, w_5, w_6, w_7) and K_2 = (w_8, w_9, w_{10}, w_{11}). Then, the round key values are obtained from the secret key as in Table 2. Note that in each round, round constants rcon(i) are used, where rcon(1) = 0001 and rcon(2) = 0010.

3.7 Example Mini-AES Encryption

The application of the four components *NibbleSub*, *ShiftRow*, *MixColumn* and *KeyAddition* in sequence constitutes one round. The full Mini-AES encryption consists of two such rounds, with the exclusion of MixColumn from the last round and the inclusion of an extra KeyAddition prior to the first round. Hence, Mini-AES encryption can be denoted by:

$$\text{Mini-AES}_{\text{Encrypt}} = \sigma_{K2} \circ \pi \circ \gamma \circ \sigma_{K1} \circ \theta \circ \pi \circ \gamma \circ \sigma_{K0}$$

Note that the symbol $\circ$ refers to the composition of functions and the order of execution is from right to left, which means that σ_{K0} is executed first. Figure 7 provides a pictorial view of Mini-AES encryption.

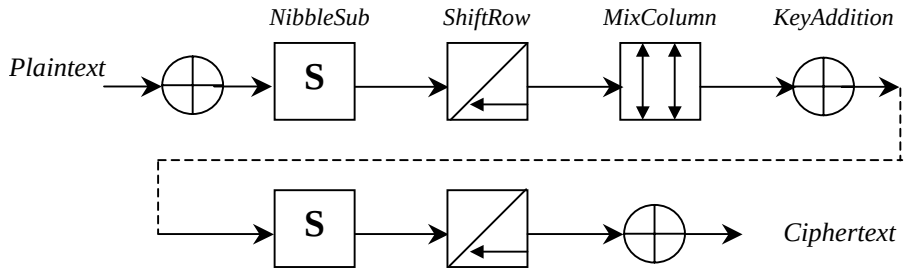

Figure 7: The Mini-AES Encryption Process

Example 9

To provide an illustrative example for the student, suppose that the input 16-bit plaintext block is P = 1001 1100 0110 0011. Expressed as a matrix of 2×2 of nibbles, it is then

$$P = \begin{array}{|c|c|} \hline 1001 & 0110 \\ \hline 1100 & 0011 \\ \hline \end{array}$$

Suppose also that the secret key, K = 1100 0011 1111 0000. This is also expressed as a 2×2 matrix.

$$K = \begin{array}{|c|c|} \hline 1100 & 1111 \\ \hline 0011 & 0000 \\ \hline \end{array}$$

Firstly, the round keys are derived from the secret key, K by way of the key schedule.

Derivation of the Round Keys

$K_0 = (w_0, w_1, w_2, w_3) = (k_0, k_1, k_2, k_3) = K = 1100\ 0011\ 1111\ 0000$

$K_1 = (w_4, w_5, w_6, w_7)$ is calculated as follows:

$$\begin{aligned} w_4 &= w_0 \oplus \text{NibbleSub}(w_3) \oplus 0001 \\ &= 1100 \oplus \text{NibbleSub}(0000) \oplus 0001 \\ &= 1100 \oplus 1110 \oplus 0001 \\ &= 0011 \\ w_5 &= w_1 \oplus w_4 \end{aligned}$$

$= 0011 \oplus 0011$
$= 0000$

$w_6 = w_2 \oplus w_5$
$= 1111 \oplus 0000$
$= 1111$

$w_7 = w_3 \oplus w_6$
$= 0000 \oplus 1111$
$= 1111$

$K_2 = (w_8, w_9, w_{10}, w_{11})$ is calculated as follows:

$w_8 = w_4 \oplus \text{NibbleSub}(w_7) \oplus 0010$
$= 0011 \oplus \text{NibbleSub}(1111) \oplus 0010$
$= 0011 \oplus 0111 \oplus 0010$
$= 0110$

$w_9 = w_5 \oplus w_8$
$= 0000 \oplus 0110$
$= 0110$

$w_{10} = w_6 \oplus w_9$
$= 1111 \oplus 0110$
$= 1001$

$w_{11} = w_7 \oplus w_{10}$
$= 1111 \oplus 1001$
$= 0110$

Encryption of the Plaintext

Next, we proceed with encryption. Given plaintext, P = 1001 1100 0110 0011, then after the first KeyAddition with the 0^{th} round key, K_0, the output is

$A = P \oplus K_0 = 1001\ 1100\ 0110\ 0011 \oplus 1100\ 0011\ 1111\ 0000$
$= 0101\ 1111\ 1001\ 0011$

Round 1

After NibbleSub, the output is

B = NibbleSub(0101), NibbleSub(1111), NibbleSub(1001), NibbleSub(0011)
= 1111 0111 1010 0001

After ShiftRow, the output is

C = ShiftRow (1111 0111 1010 0001)
= 1111 0001 1010 0111

MixColumn is a bit more involved. The input block is rearranged as a 2×2 matrix, hence:

$C =$

1111	1010
0001	0111

Taking the first column and multiplying it with the constant matrix, we get:

$$\begin{bmatrix} d_0 \\ d_1 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} 1111 \\ 0001 \end{bmatrix}$$

$$= \begin{bmatrix} (0011 \otimes 1111) \oplus (0010 \otimes 0001) \\ (0010 \otimes 1111) \oplus (0011 \otimes 0001) \end{bmatrix}$$

$$= \begin{bmatrix} 0010 \oplus 0010 \\ 1101 \oplus 0011 \end{bmatrix}$$

$$= \begin{bmatrix} 0000 \\ 1110 \end{bmatrix}$$

Taking the second column and multiplying it with the constant matrix, we get:

$$\begin{bmatrix} d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} 0011 & 0010 \\ 0010 & 0011 \end{bmatrix} \begin{bmatrix} 1010 \\ 0111 \end{bmatrix}$$

$$= \begin{bmatrix} (0011 \otimes 1010) \oplus (0010 \otimes 0111) \\ (0010 \otimes 1010) \oplus (0011 \otimes 0111) \end{bmatrix}$$

$$= \begin{bmatrix} 1101 \oplus 1110 \\ 0111 \oplus 1001 \end{bmatrix}$$

$$= \begin{bmatrix} 0011 \\ 1110 \end{bmatrix}$$

Therefore, the output after MixColumn is

$$D = \begin{array}{|c|c|} \hline 0000 & 0011 \\ \hline 1110 & 1110 \\ \hline \end{array}$$

or $D = 0000\ 1110\ 0011\ 1110$

After KeyAddition with K_1, the output is

$E = D \oplus K_1 = 0000\ 1110\ 0011\ 1110 \oplus 0011\ 0000\ 1111\ 1111$
$= 0011\ 1110\ 1100\ 0001$

Round 2

After NibbleSub, the output is

$F = \text{NibbleSub}(0011), \text{NibbleSub}(1110), \text{NibbleSub}(1100), \text{NibbleSub}(0001)$
$= 0001\ 0000\ 0101\ 0100$

After ShiftRow, the output is

$G = \text{ShiftRow}(0001\ 0000\ 0101\ 0100)$
$= 0001\ 0100\ 0101\ 0000$

Note that in the last round, there is no MixColumn. After KeyAddition, the output is

$H = G \oplus K_2 = 0001\ 0100\ 0101\ 0000 \oplus 0110\ 0110\ 1001\ 0110$
$= 0111\ 0010\ 1100\ 0110$

Therefore, the final ciphertext is $H = 0111\ 0010\ 1100\ 0110$

3.8 Mini-AES Decryption

In order to get back the original plaintext, the reverse process of encryption must be performed on the ciphertext. This is called decryption. Noting that decryption is the inverse of encryption, then

$$\text{Mini-AES}_{\text{Decrypt}} = (\sigma_{K2} \circ \pi \circ \gamma \circ \sigma_{K1} \circ \theta \circ \pi \circ \gamma \circ \sigma_{K0})^{-1}$$

$$= \sigma_{K0}^{-1} \circ \gamma^{-1} \circ \pi^{-1} \circ \theta^{-1} \circ \sigma_{K1}^{-1} \circ \gamma^{-1} \circ \pi^{-1} \circ \sigma_{K2}^{-1}$$

$$= \sigma_{K0} \circ \gamma^{-1} \circ \pi \circ \theta \circ \sigma_{K1} \circ \gamma^{-1} \circ \pi \circ \sigma_{K2}$$

We arrive at this expression since σ_{Ki} is an XOR operation, which is its own inverse. We have also specially chosen the constant matrix in MixColumn, θ such that the inverse of MixColumn, θ^{-1} is the same as MixColumn itself. Since ShiftRow simply causes the second row to be rotated left by one nibble amount, then the inverse of ShiftRow, π^{-1} causes the second row to be rotated right by one nibble. Rotating the nibble left or right are one and the same operation because one row only has two nibbles, therefore inverse ShiftRow is the same as ShiftRow. NibbleSub is a nibble substitution operation based on Table 1. The inverse of Table 1 is easily computed by interchanging the input nibble with the output nibble, and then resorting it based on the new input nibble, as given in Table 3 below.

Input	Output		Input	Output
0000	1110		1000	0111
0001	0011		1001	1101
0010	0100		1010	1001
0011	1000		1011	0110
0100	0001		1100	1011
0101	1100		1101	0010
0110	1010		1110	0000
0111	1111		1111	0101

Table 3: Inverse S-box of Mini-AES

Example 10
For an input nibble, $a_0 = 1111$, then based on Table 1, the output nibble is $b_0 = 0111$. To get back the original nibble a_0 given the value of b_0, we use the inverse of Table 1, which is Table 3. Referring to Table 3, then given the input $b_0 = 0111$, the output is $a_0 = 1111$.

Observe also that the order in which inverse NibbleSub, γ^{-1} and ShiftRow, π is carried out is not important, since inverse NibbleSub operates on each individual nibble whereas ShiftRow simply shifts the nibbles around. Hence, the final expression for Mini-AES decryption is:

$$\text{Mini-AES}_{\text{Decrypt}} = \sigma_{K0} \circ \pi \circ \gamma^{-1} \circ \theta \circ \sigma_{K1} \circ \pi \circ \gamma^{-1} \circ \sigma_{K2}$$

Comparing the expression for decryption and encryption, we see that they are similar in structure, except that the round keys and the round constants are applied in reverse order, and the NibbleSub component is replaced by its inverse.

As an exercise, the student can try by pen and paper to decrypt the ciphertext, H obtained in Example 9 to get back the original plaintext, P.

4 The Advanced Encryption Standard (AES)

In this section, we relate Mini-AES to the actual Advanced Encryption Standard (AES). Instead of having a block of 16 bits, the AES is a 128-bit block cipher, and supports secret key sizes of 128, 192 or 256 bits. We will describe the details of the AES with reference to a 128-bit key. The other variants are similar in nature.

The 128-bit block of the AES is expressed as a matrix of 4×4 bytes, in contrast to Mini-AES being expressed as a matrix of 2×2 nibbles.

AES consists of 10 rounds, where each round is similar to the round of Mini-AES, with the last round having no MixColumn. There is also a KeyAddition prior to the first

round. The purpose of the extra KeyAddition and the omission of MixColumn is so that encryption and decryption of the AES would be similar in structure, and this simplifies implementation. The same basic hardware can then be used for both encryption and decryption.

The round components of the AES are *ByteSub, ShiftRow, MixColumn and KeyAddition.* ByteSub is similar to NibbleSub, but operates on one byte instead of one nibble. Likewise, ShiftRow rotates each row of the input block to the left by different byte amounts. The first row is unchanged, the second rotated left by 1 byte, the third by 2 and the fourth by 3. MixColumn takes each column of the input block and multiplies it with a constant 4×4 matrix. KeyAddition is similar to that of Mini-AES.

The usage of the various round components of the AES follow the Wide Trail Strategy [3], where every component has its own purpose. ByteSub provides the non-linearity that is vital for the security of any block cipher. ShiftRow and MixColumn provide the linear mixing component that ensures very high diffusion over multiple rounds. KeyAddition allows the secret key bits to influence the encryption process.

The Mini-AES key schedule takes the 16-bit secret key and expresses it as a group of four nibbles. Meanwhile, the AES key schedule takes the 128-bit secret key and expresses it as a group of four 32-bit words. The 0^{th} round key, K_0 equals the secret key itself while each subsequent round key is derived from the secret key in almost the same way as Mini-AES. Further details of the AES can be found online at [9] and in a book [3].

5 The Square Attack

The structure of the AES is derived from its predecessor, the block cipher Square [2]. It is susceptible to a dedicated attack that was first developed on Square, also called the Square attack. In order to demonstrate to the student how the attack works on the AES, we will apply it on Mini-AES.

First, we extend the number of rounds of Mini-AES to 4 so that it is not too trivial to apply the Square attack on it. Let's suppose we have a set of 16 plaintexts, P_i (i = 0 to 15) such that they are equal in all nibbles except in one nibble where they have all the 16 possible different values. We will call this a delta set. The nibbles that have the same values for all 16 plaintexts in the set are called the passive nibbles whereas the nibble with all 16 possible values is called the active nibble.

Example 11

Let P_0 = 0000 0101 1010 1111
P_1 = 0001 0101 1010 1111
P_2 = 0010 0101 1010 1111
P_3 = 0011 0101 1010 1111
P_4 = 0100 0101 1010 1111
P_5 = 0101 0101 1010 1111
P_6 = 0110 0101 1010 1111
P_7 = 0111 0101 1010 1111
P_8 = 1000 0101 1010 1111
P_9 = 1001 0101 1010 1111
P_{10} = 1010 0101 1010 1111
P_{11} = 1011 0101 1010 1111
P_{12} = 1100 0101 1010 1111
P_{13} = 1101 0101 1010 1111
P_{14} = 1110 0101 1010 1111
P_{15} = 1111 0101 1010 1111

Hence, the left-most nibble is the active nibble while the remaining are passive nibbles.

If we confine ourselves within the limits of a nibble, and for each nibble position, compute the XOR of all the 16 plaintexts, we will find that the XOR of the active nibbles of the 16 plaintexts is 0000. This is because

XOR_{active} = 0000 ⊕ 0001⊕ 0010 ⊕ 0011 ⊕ 0100 ⊕ 0101 ⊕ 0110 ⊕ 0111 ⊕
1000 ⊕ 1001⊕ 1010 ⊕ 1011 ⊕ 1100 ⊕ 1101 ⊕ 1110 ⊕ 1111
= 0000

Also, the XORs of the passive nibbles are also 0000. As an example, taking one of the passive nibbles,

$XOR_{passive}$ = 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕
1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕ 1111⊕
= 0000

We say that the set of 16 plaintexts is balanced. Now, let's see what happens as these plaintexts are encrypted through one round of Mini-AES.

NibbleSub:

After NibbleSub, the outputs are

B_0 = NibbleSub(0000), NibbleSub(0101), NibbleSub(1010), NibbleSub(1111)
= 1110 1111 0110 0111

B_1 = NibbleSub(0001), NibbleSub(0101), NibbleSub(1010), NibbleSub(1111)
= 0100 1111 0110 0111

B_2 = NibbleSub(0010), NibbleSub(0101), NibbleSub(1010), NibbleSub(1111)
= 1101 1111 0110 0111

B_3 = NibbleSub(0011), NibbleSub(0101), NibbleSub(1010), NibbleSub(1111)
= 0001 1111 0110 0111

B_4 = NibbleSub(0100), 1111 0110 0111
= 0010 1111 0110 0111

B_5 = NibbleSub(0101), 1111 0110 0111
= 1111 1111 0110 0111

B_6 = NibbleSub(0110), 1111 0110 0111
= 1011 1111 0110 0111

B_7 = NibbleSub(0111), 1111 0110 0111
= 1000 1111 0110 0111

B_8 = NibbleSub(1000), 1111 0110 0111
= 0011 1111 0110 0111

B_9 = NibbleSub(1001), 1111 0110 0111
= 1010 1111 0110 0111

B_{10} = NibbleSub(1010), 1111 0110 0111
= 0110 1111 0110 0111

B_{11} = NibbleSub(1011), 1111 0110 0111
= 1100 1111 0110 0111

B_{12} = NibbleSub(1100), 1111 0110 0111
= 0101 1111 0110 0111

B_{13} = NibbleSub(1101), 1111 0110 0111
= 1001 1111 0110 0111

B_{14} = NibbleSub(1110), 1111 0110 0111
= 0000 1111 0110 0111

B_{15} = NibbleSub(1111), 1111 0110 0111
= 0111 1111 0110 0111

Notice that there is still only one active nibble and
XOR_{active} = 0000
$XOR_{passive}$ = 0000

Hence, the set is still balanced. We see that NibbleSub does not affect the balancedness of the delta set, nor does it spread an active nibble over to other passive nibbles.

ShiftRow:

After ShiftRow, the outputs are
C_0 = 1110 0111 0110 1111
C_1 = 0100 0111 0110 1111
C_2 = 1101 0111 0110 1111
C_3 = 0001 0111 0110 1111
C_4 = 0010 0111 0110 1111
C_5 = 1111 0111 0110 1111
C_6 = 1011 0111 0110 1111
C_7 = 1000 0111 0110 1111
C_8 = 0011 0111 0110 1111
C_9 = 1010 0111 0110 1111
C_{10} = 0110 0111 0110 1111
C_{11} = 1100 0111 0110 1111
C_{12} = 0101 0111 0110 1111
C_{13} = 1001 0111 0110 1111
C_{14} = 0000 0111 0110 1111
C_{15} = 0111 0111 0110 1111

XOR_{active} = 0000
$XOR_{passive}$ = 0000

Hence, the set remains balanced. ShiftRow does not affect the balancedness either. There is still only one active nibble, but notice that two nibbles have been interchanged.

MixColumn:

After MixColumn, the outputs are
D_0 = 1111 0110 0111 1110
D_1 = 0010 0001 0111 1110
D_2 = 1010 0000 0111 1110
D_3 = 1101 1011 0111 1110
D_4 = 1000 1101 0111 1110
D_5 = 1100 0100 0111 1110
D_6 = 0000 1100 0111 1110

D_7 = 0101 1010 0111 1110
D_8 = 1011 1111 0111 1110
D_9 = 0011 1110 0111 1110
D_{10} = 0100 0101 0111 1110
D_{11} = 1001 0010 0111 1110
D_{12} = 0001 0011 0111 1110
D_{13} = 0110 1000 0111 1110
D_{14} = 1110 1001 0111 1110
D_{15} = 0111 0111 0111 1110

Notice that after MixColumn, we have two active nibbles in the same column instead of just one. Therefore, MixColumn spreads one active nibble to two active nibbles in the same column.

Computing the XOR of the active and passive nibbles, we get
XOR_{active} = 0000
$XOR_{passive}$ = 0000

Hence, the set still remains balanced.

KeyAddition:

Let's suppose the round key, K_i is 1010 0011 1111 0100.
Then after KeyAddition, the output is
E_0 = 0101 0101 1000 1010
E_1 = 1000 0010 1000 1010
E_2 = 0000 0011 1000 1010
E_3 = 0111 1000 1000 1010
E_4 = 0010 1110 1000 1010
E_5 = 0110 0111 1000 1010
E_6 = 1010 1111 1000 1010
E_7 = 1111 1001 1000 1010
E_8 = 0001 1100 1000 1010
E_9 = 1001 1101 1000 1010
E_{10} = 1110 0110 1000 1010
E_{11} = 0011 0001 1000 1010
E_{12} = 1011 0000 1000 1010
E_{13} = 1100 1011 1000 1010
E_{14} = 0100 1010 1000 1010
E_{15} = 1101 0100 1000 1010

The number of active nibbles remain at 2, and computing the XORs, we see that
XOR_{active} = 0000
$XOR_{passive}$ = 0000

So the delta set is still balanced. This result is the same regardless of the value of the round key.

We have just seen through an illustrative example that NibbleSub does not spread the active nibbles, nor does it affect the balancedness of a delta set. ShiftRow does not affect the balancedness either, but just interchanges the position of two nibbles. MixColumn spreads one active nibble to two active nibbles in the same column. KeyAddition does not affect the balancedness nor spreads the active nibbles. Here we see that the balancedness and number of active nibbles in a delta set are influenced by only the MixColumn and ShiftRow.

Consider a 4-round Mini-AES. If we have a set of 16 plaintexts such that they are equal in all nibbles except in the first nibble where they have all the 16 possible different values, then after the first round, we will have a delta set that is still balanced and where there are two active nibbles in the first column.

After NibbleSub of the second round, the set is still balanced and the number of active nibbles is two. This remains the same after ShiftRow, but one of the active nibbles has been interchanged with a passive nibble in column 2. Due to this, then passing through MixColumn causes a still balanced delta set, but all nibbles are now active.

At the input of Round 3, we have a balanced delta set with all active nibbles. Passing through NibbleSub causes the same situation to exist. Likewise, ShiftRow does not change the situation either, so at the input of MixColumn, a balanced delta set exists that contains all active nibbles.

Recall again that a balanced delta set means that the XOR of each nibble position, either active or passive is zero. Now, at the input of MixColumn, the delta set is balanced and it contains all active nibbles. This means that each nibble contains all 16 possible values. We see how MixColumn influences this delta set by using an example.

Example 12

Let the delta set at the input to MixColumn be

C_0 = 0101 0101 0000 1010
C_1 = 1000 0010 0001 0101
C_2 = 0000 0011 0010 1111
C_3 = 0111 1000 0011 1110
C_4 = 0010 1110 0100 0000
C_5 = 0110 0111 0101 1000
C_6 = 1010 1111 0110 0111
C_7 = 1111 1001 0111 1011
C_8 = 0001 1100 1000 1101
C_9 = 1001 1101 1001 0010
C_{10} = 1110 0110 1010 0100
C_{11} = 0011 0001 1011 1100
C_{12} = 1011 0000 1100 0110
C_{13} = 1100 1011 1101 1001
C_{14} = 0100 1010 1110 0001
C_{15} = 1101 0100 1111 0011

We see that all nibbles are active and the delta set is balanced at the input to MixColumn. At the output of MixColumn, we have the outputs

D_0 = 0101 0101 0111 1101
D_1 = 1111 0101 1001 1101
D_2 = 0110 0101 1011 0110
D_3 = 1010 0101 1010 0111
D_4 = 1001 0101 1100 1000
D_5 = 0100 0101 1100 0001
D_6 = 0000 0101 0100 0101
D_7 = 0011 0101 1100 0000
D_8 = 1000 0101 0010 0111
D_9 = 0001 0101 1100 0111
D_{10} = 1101 0101 0101 1011
D_{11} = 0111 0101 0101 0010
D_{12} = 1110 0101 1011 0001
D_{13} = 0010 0101 0101 0001
D_{14} = 1011 0101 0011 1100
D_{15} = 1100 0101 0100 1000

Notice that the nibbles do not necessarily contain all 16 possible values any more. Coincidentally, the leftmost nibble still contains all 16 possible values, whereas the second has become a passive nibble while the remaining two are neither active nor passive but are a mixed combination of values. Nevertheless, computing the XOR of each nibble, we find that the XORs are all zero. This means that the set is still balanced even though the individual nibbles are not necessarily active or passive.

The balanced delta set proceeds through KeyAddition and since KeyAddition does not affect its balancedness, it remains that at the output of Round 3, we have a balanced delta set. This balancedness would be destroyed by the next NibbleSub in round 4, hence this is as far as we can go. The propagation of a delta set through the first 3 rounds of Mini-AES is given in Figure 8. Active nibbles are shaded in grey, while the passive nibbles are in white. Diagonally striped boxes denote nibbles that are not necessarily active nor passive, but which are still balanced. The checked boxes in Round 4 denote that the nibbles are no longer balanced.

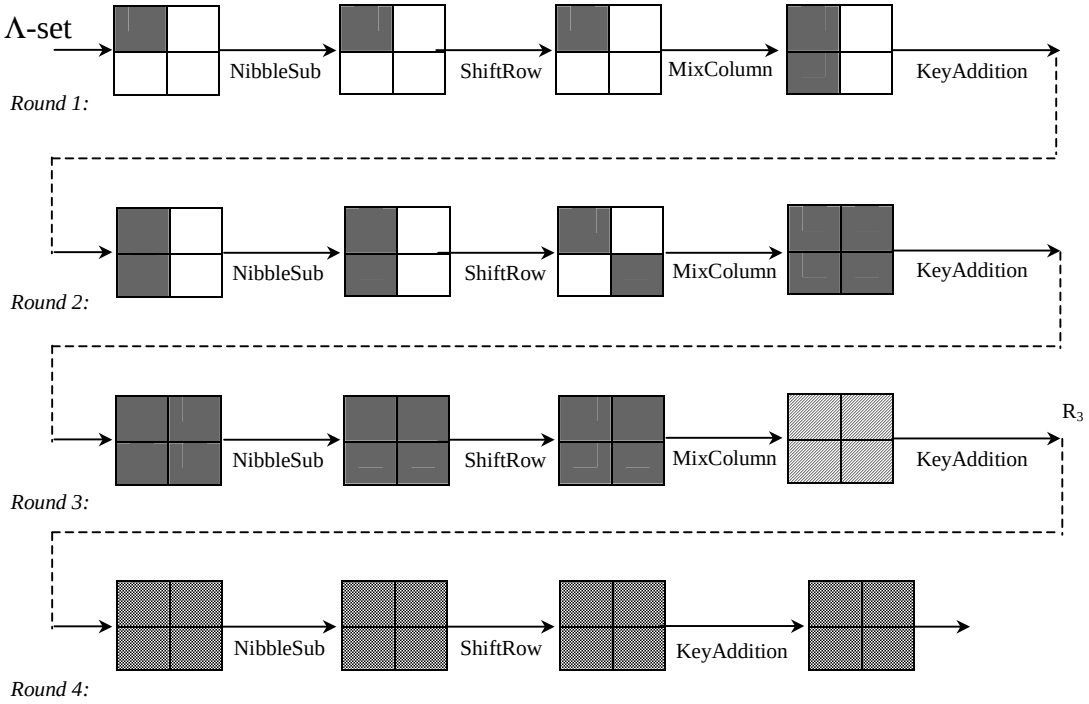

Figure 8: Propagation of the Delta-set through 4-round Mini-AES

The Square attack on 4-round Mini-AES proceeds as follows:

1. Consider a set of 16 plaintexts, P_i (i = 0 to 15) that differ in only one nibble in all 16 possible values, while the remaining nibbles contain constant values. This is similar to Example 11.
2. Obtain the ciphertexts, T_i corresponding to these plaintexts.
3. For each nibble position (nibbles 1 to 4), do
   i. For all possible values (0000 to 1111) of that nibble of the last round key, K_4, do
      a. Calculate the value $R_3 = \gamma^{-1} \circ \pi^{-1} \circ \sigma_{K4} (T_i)$ for all 16 T_i's. This corresponds to the 16 outputs of Round 3.
      b. Compute the XOR of all 16 values of R_3.
      c. If XOR = 0, then the guessed nibble of K_4 is correct. Else, it is incorrect, so cancel it out of the list of possible values for that nibble of K_4.

ii. After going through all possible values of the nibble, one or more nibbles will remain, with one being the correct nibble.
iii. Repeat steps 1, 2 and 3 with a different set of plaintexts to confirm the correct key nibble.

We have just demonstrated how the Square attack can be applied on Mini-AES. To get a feel of the attack, the interested cryptanalysis student is encouraged to try out the attack by hand, with pen and paper.

After becoming familiar with how the Square attack works on Mini-AES, the novice cryptanalyst could refer to [1, 4, 5, 6] for further details on applying the Square attack to the actual AES.

6 Conclusion

We have presented a mini version of the Advanced Encryption Standard (AES) that is well-suited for undergraduate cryptography and cryptanalysis courses. Once the student feels comfortable with Mini-AES, then he will have no problem in understanding the inner workings of the real Advanced Encryption Standard. Mini-AES is also intended to be a testbed for students and aspiring cryptanalysts to experiment with the various cryptanalysis methods that are currently available in academic literature. As an example, we demonstrated in detail how the Square attack can be used on Mini-AES. With this, we hope to have provided the vital stepping stone for the student to advance into the fascinating world of cryptanalysis research.

Acknowledgements

We wish to thank the anonymous referees for their comments which have greatly enhanced this paper.

References

1. Biham, Eli, and Keller, Nathan. 2000. *Cryptanalysis of Reduced Variants of Rijndael*. Available at http://csrc.nist.gov/encryption/aes/round2/conf3/papers/35-ebiham.pdf.

2. Daemen, Joan, and Knudsen, Lars Ramkilde and Rijmen, Vincent. 1997. *The Block Cipher Square*. Proceedings of Fast Software Encryption 1997 (Lecture Notes in Computer Science no. 1267). Springer-Verlag. 149-165.

3. Daemen, Joan, and Rijmen, Vincent. 2002. *The Design of Rijndael: AES – The Advanced Encryption Standard*. Information Security and Cryptography series. Springer-Verlag.

4. Ferguson, Niels, and Kelsey, John, and Lucks, Stefan, and Schneier, Bruce, and Stay, Mike, and Wagner, David, and Whiting, Doug. 2000. *Improved Cryptanalysis of Rijndael*. Proceedings of Fast Software Encryption 2000. Available at http://www.counterpane.com/rijndael.html.

5. Gilbert, Henri, and Minier, Marine. 2000. *A Collision Attack on 7 Rounds of Rijndael*. Proceedings of 3rd AES Conference. Available at http://csrc.nist.gov/encryption/aes/round2/conf3/papers/11-hgilbert.pdf.

6. Lucks, Stefan. 2000. *Attacking Seven Rounds of Rijndael under 192-bit and 256-bit Keys*. Proceedings of 3rd AES Conference. Available at http://csrc.nist.gov/encryption/aes/round2/conf3/papers/04-slucks.pdf.

7. NIST. 2000. *AES Development Effort*. Available at http://csrc.nist.gov/encryption/aes/index2.html.

8. NIST. 2000. *Commerce Department Announces Winner of Global Information Security Competition*. October. Available at http://www.nist.gov/public_affairs/releases/g00-176.htm.

9. NIST. 2002. *AES Homepage*. Available at http://www.nist.gov/aes.

Biographical Sketch

Raphael Phan Chung Wei received his B.Eng (Hons) degree in Computer Engineering from the Multimedia University (MMU), Cyberjaya, Malaysia in 1999. He was a tutor with the Faculty of Engineering, MMU and researcher at the Center for Smart Systems and Innovation, MMU from June 1999 to June 2001 where he also pursued his M.EngSc. degree by research in the 'Cryptanalysis of the Advanced Encryption Standard & Skipjack'. He is currently an Engineering lecturer and researcher with the Swinburne Sarawak Institute of Technology, Kuching, Malaysia. His research interests include cryptanalysis, block ciphers, and antivirus techniques.

Published in Cryptologia, XXVI (4), 2002.

APPENDIX

Here we provide the table of multiplication of two elements in GF(2^4) modulo $x^4 + x + 1$. The entries in the table are represented in hexadecimal notation for compactness. The row and column indices represent the two elements to be multiplied modulo $x^4 + x + 1$, and the product is the corresponding entry in the intersection of that row and column.

Example 13
To find the product of 1011 (B in hexadecimal) and 0111 (7 in hexadecimal), refer to the row index B and column index 7. The intersection contains the entry 4 in hexadecimal. Hence, $1011 \otimes 0111 = 0100$.

⊗	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2	0	2	4	6	8	A	C	E	3	1	7	5	B	9	F	D
3	0	3	6	5	C	F	A	9	B	8	D	E	7	4	1	2
4	0	4	8	C	3	7	B	F	6	2	E	A	5	1	D	9
5	0	5	A	F	7	2	D	8	E	B	4	1	9	C	3	6
6	0	6	C	A	B	D	7	1	5	3	9	F	E	8	2	4
7	0	7	E	9	F	8	1	6	D	A	3	4	2	5	C	B
8	0	8	3	B	6	E	5	D	C	4	F	7	A	2	9	1
9	0	9	1	8	2	B	3	A	4	D	5	C	6	F	7	E
A	0	A	7	D	E	4	9	3	F	5	8	2	1	B	6	C
B	0	B	5	E	A	1	F	4	7	C	2	9	D	6	8	3
C	0	C	B	7	5	9	E	2	A	6	1	D	F	3	4	8
D	0	D	9	4	1	C	8	5	2	F	B	6	3	E	A	7
E	0	E	F	1	D	3	2	C	9	7	6	8	4	A	B	5
F	0	F	D	2	9	6	4	8	1	E	C	3	8	7	5	A