Understanding the Advanced Encryption Standard through Matrix Operations

Matthew O'Connell

Background

Cryptology is the study of writing secret messages or codes that have been used since the inception of communication to conceal data from unwanted audiences. The ultimate goal of cryptography is to layer sensitive information with strong security boundaries for its transportation (Kahn 1967). However it serves the additional role of saving information via an encoded format. This means that even if a thief were to acquire stolen data, he or she would not be able to readily interpret the information. Mathematics has progressed cryptology through the implementation of numbers which allow for more complex algorithms to be created and lessen the reliance on hardware to encrypt data. Mathematics has brought cryptology's applications into many fields that deal with sensitive information including communications in economic, military, and computer science spheres.

Purpose

I am working on a project for the International Pharmaceutical Federation to relay medication instructions from prescribers to their patients. The project consists of the development of a computer system wherein prescribers transfer and store data of their patients. The protection of prescriber and patient information is very important and because this will be a standalone system (one that runs local to the accessing computer); I need to store data and make it readily available. In this investigation, I will evaluate Advanced Encryption Standard (AES) encryption as a means to encode information that is readily available into a format that is not readily readable. Ideally, this will allow users of my program to view information by opening files; however, the information will be saved in an encrypted format that they will not be able to understand or interpret unless they log into a password protected account that will decrypt the information for them. Through this method, I can save data securely in plain sight and still meet ergonomic requirements to facilitate user-driven data transfer (such as copying a single file).

Analysis of Matrix Multiplication: A Cipher

AES encryption implements extensive algorithms, known as ciphers, to manipulate matrices (Bogdanov & Khovratovich 2011). Matrices are a notation method used to represent numbers via multidimensional vectors. The individual sub-algorithms that compose AES encryption can be referenced as distinct ciphers Matrix encryption relies on conveying characters or letters into numbers and then rearranging them so that they can be stored securely. Understanding the basic processes of matrix encryption via a matrix multiplication cipher example will be the first step in understanding matrices' role in the greater AES encryption scheme.

As an example: I need to save the name of a patient. However, I must ensure if the information is intercepted, no one would be able to understand it. For this example, I will use the last name "Smith" as the information to be relayed.

I. The first step is to convert the characters into numbers. For this example, every letter will be replaced with its position in the alphabet. Therefore, "Smith" can be represented as 19-13-9-20-8 (where "S" is represented as "19" because it is the 19th letter in the alphabet). This is a very simple conversion from letters to numbers. However, it should be understood that information can be hidden under many more processes in this step alone that extend past the scope of this example. Case-sensitive delineation, variance in font, and multi-byte encoding all represent alternative methodologies that could complex the initial "character-to-number" process.

II. Once the characters are represented as numbers, an encoding matrix must be selected. This is a very important part of the process because the matrix must be known by both parties (the encoder and decoder of the information). This matrix ultimately holds the secret to how the data is encoded; it is known as the "key" because it will both lock and unlock the information. The matrix's selection is arbitrary; however, the selected matrix must have an inverse because its inverse will be used to decode the information. Below on the left is K: a matrix I have selected to use as a key for our example. Its inverse: K^{-1} is on below on the right and will be used later in step 5.

$$\mathrm{K} = \begin{bmatrix} -3 & -3 & -4 \\ 0 & 1 & 1 \\ 4 & 3 & 4 \end{bmatrix} \qquad \mathrm{K}^{-1} = \begin{bmatrix} 1 & 0 & 1 \\ 4 & 4 & 3 \\ -4 & -3 & -3 \end{bmatrix}$$

III. Because K is a three by three matrix, the information will need to be represented in a three row matrix (So that I can multiply them in the next step). Zeroes can be used as a place holder because five (how many characters I will be encoding) is not perfectly divisible by three.

$$19 - 13 - 9 - 20 - 8 \rightarrow \begin{bmatrix} 19 & 20 \\ 13 & 8 \\ 9 & 0 \end{bmatrix}$$

IV. The key, K, is then dot multiplied with the information to encode. This creates the following encoded matrix. This step completes the process and encoded information can now be saved. If someone were to read -132, 22, 151, -84, 8, 104, it is unlikely they would know this information represented "Smith". But now that it is saved as a series of numbers how is the data usable?

$$\begin{bmatrix}-3 & -3 & -4\\ 0 & 1 & 1\\ 4 & 3 & 4\end{bmatrix} \bullet \begin{bmatrix}19 & 20\\ 13 & 8\\ 9 & 0\end{bmatrix} = \begin{bmatrix}-132 & -84\\ 22 & 8\\ 151 & 104\end{bmatrix} \rightarrow -132, 22, 151, -84, 8, 104,$$

V. To access the information in a decoded format, it must be put back into the matrix it was encoded into. Once it is in the encoded matrix form, it is dot multiplied with K^{-1}. The product is the original matrix used to represent the information.

$$\begin{bmatrix}1 & 0 & 1\\ 4 & 4 & 3\\ -4 & -3 & -3\end{bmatrix} \bullet \begin{bmatrix}-132 & -84\\ 22 & 8\\ 151 & 104\end{bmatrix} = \begin{bmatrix}19 & 20\\ 13 & 8\\ 9 & 0\end{bmatrix}$$

VI. The matrix can then be represented as a line of numbers where each corresponds to one of the original characters. The numbers are then converted to their character counterparts. The information is now in its original format.

19-13-9-20-8 → "Smith"

This method of encoding data manually takes a lot of time. Inputting each datum into my calculator is also prone to human error. However, through the use of computer software, this process can be automated. Computers make the use of encryption strategies that incorporate matrix multiplication feasible. Unfortunately, computers are also what make matrix encryption rather easy to hack.

Skilled cryptographers with access to lots of encrypted information can use computers to map out the frequency of characters or phrases against equivalent frequency charts that utilize standard characters or phrases. This form of linear cryptanalysis is supplemented with computers' ability to analyze relationships between characters or phrases. Because of the effectiveness of these hacking methods, theoretical mathematicians continually research into finding ways through which data can be misconstrued to confuse computers. Their work has led to the development of several cryptosystems.

Matrix multiplication is an example of a cipher that encrypts and decrypts information. A cryptosystem implements many different algorithms such as matrix multiplication to encrypt and decrypt information as well as develop keys. Secure cryptosystems implement elements of both confusion and diffusion (Trappe 2006).

Confusion involves the variability of information representation. Information is changed into a novel representation that is no easily readable. This process is exemplified through Step I of the matrix multiplication example wherein characters, which people can read easily as words, were replaced with numbers which people do not understand as words. Within diffusion, information is dissipated and restructured. This process is exemplified in Step III which converts the numbers into a matrix structure.

Analysis of AES: A Cryptosystem

AES stands for the Advanced Encryption Standard and is used worldwide as a viable cryptosystem. AES implements elements of both confusion and diffusion to encrypt data. As we explore the processes that constitute AES, it is important to remember that they were specifically engineered to confuse people and computers. Because of this, they implement recursive functions and many arbitrary values which are reflective of anti-computer tactics (a series of theoretical mathematics stemming from game theory on designing ways to confuse computers). However AES is simple to understand when thought of as four separate steps:

Step 1. Key Expansion

Step 2. The Initial Round

Step 3. The Intermediary Rounds

Step 4. The Final Round

I will explain each function to convey its purpose within the AES cryptosystem. I will specifically highlight the use of matricies throughout the processes to highlight their role in the greater AES scheme.

The initial processes of AES are reflective of those in the matrix multiplication cipher. A matrix called “the state” will be created to hold the information to encode; and the keys will be represented in the form of matrices. We will first explore the keys used in AES and then analyze their role in encrypting the state.

Step 1: Key Expansion, Part A “Creating the Cipher Key”

Similar to as in the matrix cipher, a matrix will be chosen by the encoder to be used as a key. The encoder-specified key is known as the cipher key. Unlike the matrix cipher, AES requires several keys that will be “expanded” from the cipher key. Key expansion is an integral part of the AES cryptosystem and will be the first algorithm we explore. As in the matrix cipher, keys need to meet certain requirements. Keys are composed of 4x4 matrices where each index holds exactly one byte of information.

Because the numbers can become very large in the AES process, they are represented in hexadecimal. While an in-depth analysis of format systems is beyond the scope of this investigation, it should be understood that hexadecimal is simply another way of writing numbers that allows for greater values to be represented in the same amount of space. For example, in the decimal system three digits afford a number as great a value as “999”. However in hexadecimal the value represented in three digits could be 4095 as in the case of the number “FFF”.

Hexadecimal values take up four bits each. A bit is a binary digit (1 or 0). Two hexadecimal values (or eight bits) constitute a byte, or more specifically an octet. Therefore, each index of the 4x4 matrix can hold two hexadecimal values as displayed below.

$$\begin{bmatrix} \text{2B} & 28 & \text{AB} & 09 \\ \text{7E} & \text{AE} & \text{F7} & \text{CF} \\ 15 & \text{D2} & 15 & \text{4F} \\ 16 & \text{A6} & 88 & \text{3C} \end{bmatrix} = \text{the cypher key}$$

Now that I have a cypher key that meets the specifications of a 4x4 matrix of 16 singular byte indices I can begin the second part of key expansion.

Step 1: Key Expansion, Part B “Expanding the Cipher Key”

AES will process the state through several “rounds” of encryption. This simply means that the same processes will need to be repeated several times. Each round will require a new key. These keys, which I will now derive, are collectively referred to as round keys. Round keys are derived from the cipher key via the Rijndael key schedule (Fragneto, et al., 1967). This is a process which is conducted on the cipher key in order to make the first round key. After the first round key is created, it is used in place of the cipher key to make the second round key. The second round key is then used to make the third and so on until all keys are created. The process is completed as follows:

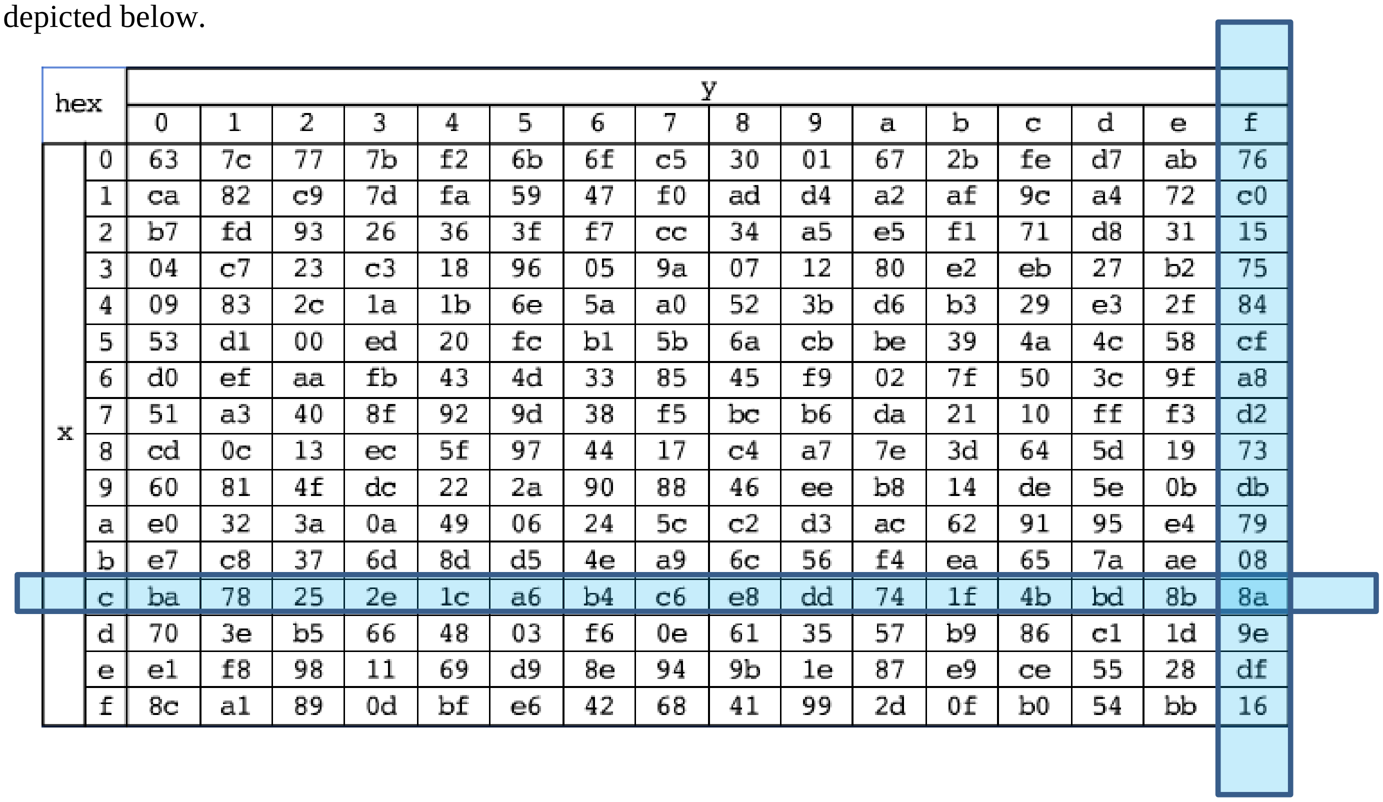
1. The fourth column from the *i*th round key (where *i*+1 is the round of the current key being created and the 0th round key is the cipher key) is selected for expansion and placed into its own matrix.

$$\begin{bmatrix} \text{2B} & 28 & \text{AB} & 09 \\ \text{7E} & \text{AE} & \text{F7} & \text{CF} \\ 15 & \text{D2} & 15 & \text{4F} \\ 16 & \text{A6} & 88 & \text{3C} \end{bmatrix} \rightarrow \begin{bmatrix} 09 \\ CF \\ 4F \\ 3C \end{bmatrix}$$

2. The top row of this column is then placed at the bottom of the column and all indices are shifted up a row.

$$\begin{bmatrix} 09 \\ cf \\ 4f \\ 3c \end{bmatrix} \rightarrow \begin{bmatrix} 09 \\ cf \\ 4f \\ 3c \end{bmatrix} \rightarrow \begin{bmatrix} CF \\ 4F \\ 3C \\ 09 \end{bmatrix}$$

3. Each byte is then substituted for another value. This process holds the same function as the “character-to-number” process of Step I in the matrix cipher. However AES uses a mapping process that is one of the most complicated options previously alluded to. AES uses a 17X17 matrix known as the S-Box to replace values. The S-Box can be visualized as a large grid as depicted below.



hex		y															
		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
x	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

Using the first byte of an index as the “x value” and the second as the “y value”, the S-Box maps the values of each index to a new value. I have highlighted the “c row” and “f column” to demonstrate how the first row index of “CF” maps to “8A”. This process is completed for all rows within the column.

$$\begin{bmatrix} CF \\ 4F \\ 3C \\ 09 \end{bmatrix} \rightarrow \begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix}$$

4. The column now undergoes an exclusive-or (Xor) comparison with a round constant. Xor is a simple operator that compares values to produce a new value. One reason Xor is used is because it is reversible. Because of this, to decrypt data the same steps will only need to be applied in reverse order. Xor compares values to see if one or the other is true but not both, hence it is called the exclusive-or. This true-false comparison corresponds with the binary values of a single bit (which contains a 1 or 0 where 1=true and 0=false). Because this is a bit-level operator, each byte of information must be decomposed into its bits, then compared via the Xor operator, and then recomposed into a novel byte. Using the first row index as an example again, 8A can be

converted into a bit (binary) representation of 1000 1010. This number is now ready for the Xor comparison with the round constant.

The round constant, much like the round key, will be a unique value for each round. The round constant is also created via its own algorithm. The round constants are created using the value X. X can be assigned any single-digit integral value for variability in the processes of AES. I use its most commonly assigned value of X = 2 for this example. X can be represented in decimal or binary:

$$X = 2_{(decimal)} = 00000010_{(binary)}$$

Additionally, X can be presented as a polynomial where each term corresponds to a digit in its binary representation.

$$\begin{tikzcd}
0 \arrow[d] & 0 \arrow[d] & 0 \arrow[d] & 0 \arrow[d] & 0 \arrow[d] & 0 \arrow[d] & 1 \arrow[d] & 0_{(binary)} \arrow[d] \\
= 0x^7 & + 0x^6 & + 0x^5 & + 0x^4 & + 0x^3 & + 0x^2 & + 1x & + 0
\end{tikzcd}$$

Round constants are created with X via the following formula.

$$rcon(i) = x^{(i-1)} \, mod \; x^8 + x^4 + x^3 + x + 1$$

This formula finds the rcon or round constant for *i*, the current round, by raising X to the power of *i-1*. The resulting value is then replaced by the mod or modulo of $\frac{x^{(i-1)}}{x^8+x^4+x^3+x+1}$. The "modulo operator" simply returns the remainder of the two numbers' division.

Below, I use this formula to find the first round constant so that it can be used in the Xor comparison with the first key:

$$rcon(1) = x^{(1-1)} \, mod \; x^8 + x^4 + x^3 + x + 1$$

$$= 1 \, mod \; x^8 + x^4 + x^3 + x + 1 = 1$$

The first round constant, 1, is obtained and can be converted into a bit (binary) representation of 0000 0001. I can now compare this value and the first round key via the Xor operator.

$$\begin{array}{ccccccccc}
 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\
\oplus & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\hline
 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}$$

The result is a bit sequence that can be represented as the byte 8B. This value is then used to replace the index and the first index is replaced. Likewise, all other indices are replaced via the same algorithm.

$$\begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix} \rightarrow \begin{bmatrix} 8B \\ 84 \\ EB \\ 01 \end{bmatrix} \rightarrow \begin{bmatrix} 8B \\ 85 \\ EA \\ 00 \end{bmatrix}$$

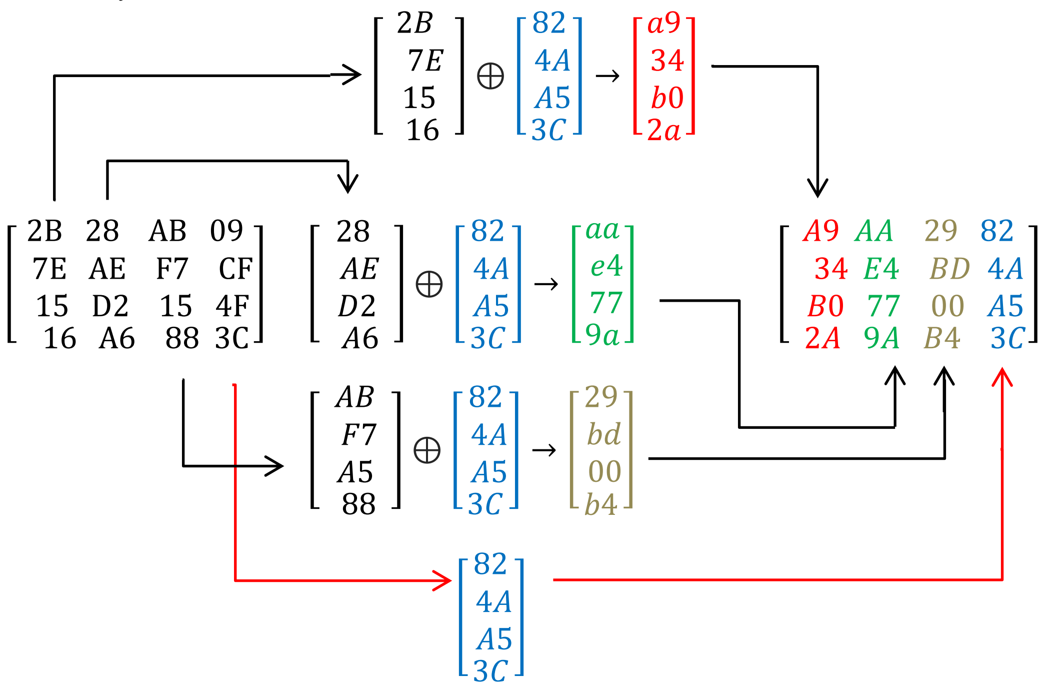
5. The resulting column is yet again compared via the Xor operator but this time with the previous round's key. Because this is the first round key it is compared with the 0th round key, which as stated in Step 1, is the cipher key's fourth column.

$$\begin{bmatrix} 8B \\ 85 \\ EA \\ 00 \end{bmatrix} \oplus \begin{bmatrix} 09 \\ CF \\ 4F \\ 3C \end{bmatrix} = \begin{bmatrix} 82 \\ 4A \\ A5 \\ 3C \end{bmatrix}$$

6. To complete the round key, the resulting column (in blue) is now compared, via the Xor operator, with the previous round's key's remaining columns. Once again the cipher key will be used in place of a previous round key because I am working with the first round. Since I used the 4th column in Step 5, columns 3, 2, and 1 will now be compared via the Xor.

   Other variations use the 1st column instead of the 4th in Step 5 and then iterate through columns in the order of 2, 3, and 4. While this may seem like a subtle difference it is yet another example of how the process can be varied only slightly but completely alter the outcome.

   In my diagram I utilized the red line to highlight that the derived blue matrix is used as the 1st round key'fourth column.



$$\begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix}$$

$$\begin{bmatrix} 2B \\ 7E \\ 15 \\ 16 \end{bmatrix} \oplus \begin{bmatrix} 82 \\ 4A \\ A5 \\ 3C \end{bmatrix} \rightarrow \begin{bmatrix} a9 \\ 34 \\ b0 \\ 2a \end{bmatrix}$$

$$\begin{bmatrix} 28 \\ AE \\ D2 \\ A6 \end{bmatrix} \oplus \begin{bmatrix} 82 \\ 4A \\ A5 \\ 3C \end{bmatrix} \rightarrow \begin{bmatrix} aa \\ e4 \\ 77 \\ 9a \end{bmatrix}$$

$$\begin{bmatrix} AB \\ F7 \\ A5 \\ 88 \end{bmatrix} \oplus \begin{bmatrix} 82 \\ 4A \\ A5 \\ 3C \end{bmatrix} \rightarrow \begin{bmatrix} 29 \\ bd \\ 00 \\ b4 \end{bmatrix}$$

$$\begin{bmatrix} 82 \\ 4A \\ A5 \\ 3C \end{bmatrix}$$

$$\begin{bmatrix} A9 & AA & 29 & 82 \\ 34 & E4 & BD & 4A \\ B0 & 77 & 00 & A5 \\ 2A & 9A & B4 & 3C \end{bmatrix}$$

The resulting matrix is the 1st round key. At this point, the key expansion algorithm is re-invoked to create the 2nd round matrix key using the 1st round key in replacement of the cipher key (or 0th key). Once the 2nd round key is made, it is used to make the 3rd and the process continues until all round keys are made.

Step 2: The Initial Round, "Add Round Key"

Now that all the keys are created, data can be entered into the system. Data to encrypt must be stored in the format of a 4x4 matrix with one byte in each index (this is the same specification that the key matrices needed to meet). Once the data to encode is entered into a matrix it is referred to as the state. As an example, I have created a state matrix of data below.

$$\begin{bmatrix} FE & ED & 8A & 81 \\ 35 & 62 & CD & 42 \\ EA & 3f & C4 & 11 \\ 45 & 14 & 40 & 1E \end{bmatrix} = state\ (information\ to\ encode)$$

Within the initial round the state will now undergo a process referred to as Add Round Key. This step consists of a simple Xor operation with the state and the current round's key. The initial round is the 0th round so the cipher key will be used.

$$\begin{bmatrix} FE & ED & 8A & 81 \\ 35 & 62 & CD & 42 \\ EA & 3f & C4 & 11 \\ 45 & 14 & 40 & 1E \end{bmatrix} \oplus \begin{bmatrix} 2B & 28 & AB & 09 \\ 7E & AE & F7 & CF \\ 15 & D2 & 15 & 4F \\ 16 & A6 & 88 & 3C \end{bmatrix} = \begin{bmatrix} D5 & C5 & 21 & 88 \\ 4B & CC & 3A & 8D \\ FF & ED & D1 & 5E \\ 53 & B2 & B2 & 22 \end{bmatrix}$$

This completes the initial round. The resulting matrix will now be passed to the intermediary rounds.

Step 3: The Intermediary Rounds

In the intermediary rounds the state undergoes several rounds of algorithms. Typically, an AES cryptosystem will implement 9 intermediary rounds. Each round consists of the following four steps:

- S-Box Mapping (this process is the same as Step 3 from key expansion)
- Shift Rows
- Mix Columns
- Add Round Key (this process is the same as the initial round)

These steps contain two processes (S-Box Mapping & Add Round Key) that have already been covered as well as two novel diffusion processes (Shift Rows & Mix Columns) which we will now analyze.

Shift Rows:

Within the Shift Rows stage the indices of a row are simply shifted to different columns from right to left. When an index is shifted past the first column it is transferred to the fourth (Moser 2009). The first row is shifted zero indices (no change is made). The second row is shifted one index. The third row is shifted two indices. Finally the fourth row is shifted three indices. These shifts are demonstrated on an example state matrix below.

$$\begin{bmatrix} A2 & B8 & 8A & 7F \\ 32 & CA & B1 & 3B \\ 1B & 42 & 4C & 21 \\ EC & 1A & 82 & D1 \end{bmatrix} \rightarrow$$

$$\begin{tikzcd} A2 & B8 & 8A & 7F \\ 32 & CA & B1 & 3B \arrow[l, bend right, "1"'] \\ 1B & 42 & 4C & 21 \arrow[ll, bend right, "2"'] \\ EC & 1A & 82 & D1 \arrow[lll, bend left, "3"] \end{tikzcd}$$

$$\rightarrow \begin{bmatrix} A2 & B8 & 8A & 7F \\ CA & B1 & 3B & 32 \\ 4C & 21 & 1B & 42 \\ D1 & EC & 1A & 82 \end{bmatrix}$$

Mix Columns:

Within the Mix Columns algorithm each column is processed separately. The following matrix will represent a column from an example state.

$$\begin{bmatrix} A2 \\ CA \\ 4C \\ D1 \end{bmatrix}$$

The column is first multiplied with the following circulant maximum distance separable matrix (below on left) where all values are integral hexadecimal values. This matrix is special because it can be quickly manipulated by a special computer algorithm known as the Fast Fourier Transform (Vaudenay 1994).

$$\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} A2 \\ CA \\ 4C \\ D1 \end{bmatrix}$$

The product's rows are then converted into polynomials similar to the one that was used to represent X in Step 4 of key expansion.

$$\begin{bmatrix} 2(A2) + 3(CA) + 1(4C) + 1(D1) \\ 1(A2) + 2(CA) + 3(4C) + 1(D1) \\ 1(A2) + 1(CA) + 2(4C) + 3(D1) \\ 3(A2) + 1(CA) + 1(4C) + 2(D1) \end{bmatrix}$$

The next step is one of the most interesting of all AES processes. Because each part of these polynomials represents a byte or eight bits (as demonstrated with X in Step 2), the addition operator can be replaced by the Xor operator to combine the data in a simple and easily reversible process (Moser 2009).

$$\begin{bmatrix} 2(A2) \oplus 3(CA) \oplus 1(4C) \oplus 1(D1) \\ 1(A2) \oplus \ 2(CA) \oplus \ 3(4C) \oplus \ 1(D1) \\ 1(A2) \oplus \ 1(CA) \oplus \ 2(4C) \oplus \ 3(D1) \\ 3(A2) \oplus \ 1(CA) \oplus \ 1(4C) \oplus \ 2(D1) \end{bmatrix}$$

The resulting 4x1 matrix is now ready to be simplified to complete the process; but how can I multiply the (red) integral coefficients? Because only two digits can be used to represent the information (a condition known as a finite field), a special type of math is used to complete this operation known as Galois Multiplication (Plank n.d.).

I will demonstrate how the first term “ $2(A2)$ ” is converted to binary representation via Galois Multiplication for comparison via the Xor operator.

I first convert the two hexadecimal coefficients to binary and then polynomial representation.

$$2(A2)$$

$$0010(1010\ 0010)$$

$$0x^3 + \ 0x^2 + \ 1x^1 + 0(1x^7 + 0x^6 + 1x^5 + 0x^4 + 0x^3 + 0x^2 + 1x^1 + 0)$$

I then simplify and multiply the polynomials

$$1x^1(1x^7 + 1x^5 + 1x^1)$$

$$(1x^8 + 1x^6 + 1x^2)$$

The polynomial is then converted to an eight bit binary representation

$$(1x^8 + 0x^7 + 1x^6 + 0x^5 + 0x^4 + 0x^3 + 1x^2 + 0x^1 + 0\ )$$

$$(1\ 0100\ 0100)$$

The resulting binary number has nine bits however after it is compared with the other values in its row via the Xor operator, the resulting value will always be eight bits (one byte). AES is valued for this crisp and time effective process. The resulting matrix after each row is put through this algorithm will be once again a 4x1 matrix with one byte in each row.

$$\begin{bmatrix} 2E \\ B1 \\ 7D \\ 34 \end{bmatrix}$$

This column is now replaced into its position in the state. The Mix Columns algorithm is then completed for the other columns. What I find most rewarding about using this algorithm is that it expands the terms into a very complex expression but is then able to simplify all of the terms back into a single byte.

Step 4: The Final Round

Once the state has completed the intermediary rounds is must undergo the following processes once more. Note that these algorithms are the same as those within Step 3 with the omission of mix columns. Mix Columns has been omitted because its sole use is diffusion. Because this is the last step, diffusion at this stage would not further the security of the data; therefore, the Mix columns process is skipped.

- S-box Mapping
- Shift Rows
- Add Round Key

Once the state has been processed through the final round it is fully encrypted. This concludes the encryption process. In order to decrypt the information the same steps are simply completed in reverse order.

Discussions/Conclusions

Though matrix multiplication alone may be only a simple cipher, understanding its concepts and processes is imperative to understanding modern cryptosystems such as AES. Matrices are used in all stages of the AES encryption process and provide a means to implement both confusion and diffusion tactics. The matrix structure allows for easy manipulation of bytes of data and allows whole columns to undergo encryption algorithms at once (Bogdanov & Khovratovich 2011). Additionally, entire 4x4 matrices of data can be used as operands to functions such as the Xor and modulo operations.

Cryptosystems provide people with a secure way to relay information between each other. They form the virtual protection of the digital age and have continued to advance cryptography in becoming increasingly difficult to decipher. Researchers continue to find better ways to protect people’s information.

I will use AES as the encryption method for my software program because it provides a concrete protection system that will serve my users well. I am currently working on a section of my program that will be used to both encrypt and decrypt user information via AES. I look forward to implementing the many algorithms I learned about in this analysis of matrix use in AES encryption so that my users will be confident in their data’s security.

References

Bogdanov, Andrey, and Dmitry Khovratovich. "Biclique Cryptanalysis of teh Full AES." *Microsoft Research Redmond*. (2011): n. page. Print. <http://research.microsoft.com/enus/projects/cryptanalysis/ aesbc.pdf>.

Fragneto, Pasquilina, Guido Bertoni, Marco Maccheti, and Stefano Marchesin. "Efficient Software Implementation of AES on 32-Bit Platforms*." (1967): n. page. Web. 29 May. 2014. <http://link.springer.com/chapter/10.1007/3-540-36400-5_13?LI=true

Kahn, David. *The Codebreakers – The Story of Secret Writing*. 2. Macmillan and Sons, 1967, 1996. Print.

Moser, Jeff. *A Stick Figure Guide to the Advanced Encryption Standard (AES)*. 2009. Photograph. MoserwareWeb. 29 May 2014.

Plank, James. "Fast Galois Field Arithmetic Library in C/C++." . University of Tennessee, n.d. Web. . <http://web.eecs.utk.edu/~plank/plank/papers/CS-07-593/>.

Trappe, Wade. *Introduction to Cryptography with Coding Theory*. Second edition ed. : Pearson Prentice Hall, 2006. Print.

Vaudenay, Serge. *On the Need for Multipermutations: Cryptanalysis of MD4 and SAFER*. 2nd International Workshop on Fast Software Encryption: 1994. 99-111. Web.