

Global Information Assurance Certification Paper

Copyright SANS Institute Author Retains Full Rights

This paper is taken from the GIAC directory of certified professionals. Reposting is not permited without express written permission.

Interested in learning more?

Check out the list of upcoming events offering "Security Essentials Bootcamp Style (Security 401)" at http://www.giac.org/registration/gsec

A Look at Some of the Mathematics Behind Rijndael

Brett Carpenter January 24, 2001

Introduction

As a layman, I have often been frustrated by the way in which the mechanics of ciphers are passed off as a black box into which plaintext is inserted and from which, with the help of magic, ciphertext is retrieved. The branch of mathematics behind this magic is known as *cryptology*. The purpose of this paper is to shed a tiny ray of light on the concepts at work in this field. Specific attention will be paid to Rijndael (pronounced Rhine-dahl), the National Institute of Standards and Technology's recent choice for the Advanced Encryption Standard (AES).

I apologize in advance to any mathematicians who might happen to read this paper.

Objectives

The objectives of this paper are as follows:

- To introduce, at a very high level, some of the concepts in mathematics underlying *cryptology* and the Rijndael block cipher
- To describe the Rijndael block cipher in light of these concepts

Mathematical Background

The mathematical concepts mentioned in the following sections are taken loosely from the fields of algebra and analysis. This section describes the model that the designers of Rijndael used to represent binary data.

Fields

A field is a set – called F, for example – along with two operations, "addition" (\oplus) and "multiplication" (\otimes). F is *closed* under these operations; that is, the sum or product of any two elements of F is also an element of F. A mathematician might express this property as follows:

$$a, b \in F \Rightarrow a \oplus b \in F$$
$$a, b \in F \Rightarrow a \otimes b \in F$$

It is important to note that these operations need not be what we think of as standard addition (+) and multiplication (*); thus the use of the alternate symbols.

The properties of a field include the following, among others:

- Addition is *commutative*: $a \oplus b = b \oplus a$
- Multiplication is *distributive*: $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$

The real number system, \Re , is an example of a field.

$GF(2^8)$

A finite field – that is, a field containing a finite number of elements – is used as the basis for Rijndael: $GF(2^8)$. This is the *Galois Field* (GF) containing 2^8 , or 256, elements. Note that any byte value can be mapped to exactly one element of $GF(2^8)$. A common representation of the elements of $GF(2^8)$ is a polynomial of degree seven with coefficients in $\{0,1\}$. Go with me on this one! A byte, *b*, consisting of bits $b_7b_6b_5b_4b_3b_2b_1b_0$ is the mapped to $GF(2^8)$ as the polynomial

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$
.

Example 1:

```
The byte with hex value '92' (binary 01011100) is mapped to x^6 + x^4 + x^3 + x^2 \, . \label{eq:xb}
```

Sounds like this might come in handy when dealing with binary data, right?

Addition and Multiplication in $GF(2^8)$ Real numbers can be added and multiplied. All of us do this every day. For example,

2 + 2 = 4.

Well, there is an analogous operation in $GF(2^8)$. The "addition" (Å) of two elements results in the polynomial with coefficients that are given by the sum *modulo* 2.

```
Example 2:

x^5 \oplus x^6 + x^5 + x^4 + x^2 = x^6 + x^4 + x^2

Written in hex, we have:

`32' \oplus `D4' = `98'.

Or, in binary, we have:

00100000 \oplus 01110100 = 01010100.
```

Thus, "addition" (\oplus) in GF(2⁸) is the standard bitwise XOR operation. Pretty straightforward so far!

"Multiplication" (\ddot{A}) is a little trickier. It corresponds with multiplication of the polynomials *modulo* m(x), where

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

or '11B' in hex. Well, I haven't *modulo*'d a polynomial recently, but this is done to ensure that the product is in fact an element of $GF(2^8)$, among other things. Sounds reasonable, though.

Example 2: $(x^{4} + x^{3} + 1) \otimes (x^{3} + x^{2} + x) = x(^{7} + x^{6} + x^{5}) + (x^{6} + x^{5} + x^{4}) + (x^{3} + x^{2} + x) = x^{7} + x^{4} + x^{3} + x^{2} + x$ Then, calculate the previous result modulo m(x): $(x^{7} + x^{4} + x^{3} + x^{2} + x) \mod(x^{8} + x^{4} + x^{3} + x + 1) = x^{7} + x^{4} + x^{3} + x^{2} + x.$ This is equivalent to '25' \otimes '14' = '9E' in hex.

Like the "addition" operation in $GF(2^8)$, the "multiplication" operation satisfies the requisite properties of a field, as described above.

Result: We now have an abstract representation of our binary data that includes some basic mathematical operations.

Why Does Any of This Matter?

The steps described above have resulted in the following: digital information, represented at the lowest logical level as bits and bytes, can be mapped to a mathematical "model" that has certain "nice" qualities. In the case of Rijndael, that model is the finite field $GF(2^8)$. These qualities, and their implications, are then ultimately used to encipher and decipher the data.

For example, *polynomials with coefficients in* $GF(2^8)$ can be used to represent arrays of bytes or multi-byte words. If a_3 , a_2 , a_1 , and a_0 are elements of $GF(2^8)$, then

$$a_3x^3 + a_2x^2 + a_1x + a_0$$

is used to represent a 4-byte vector, or 4-element array of bytes, or a 4-byte word. Imagine it as an array of arrays. Thus, this model lends itself well to operations at both the byte and word level. These byte- and word-level representations are also convenient for a cipher that is to be implemented on a modern computer.

As another example, multiplication of polynomials with coefficients in $GF(2^8)$ is done modulo M(x), where

$$M(x) = x^4 + 1,$$

and can be conveniently represented as a matrix operation:

$$\begin{bmatrix} c_{0} \\ c_{1} \\ c_{2} \\ c_{3} \end{bmatrix} = \begin{bmatrix} a_{0} & a_{3} & a_{2} & a_{1} \\ a_{1} & a_{0} & a_{3} & a_{2} \\ a_{2} & a_{1} & a_{0} & a_{3} \\ a_{3} & a_{2} & a_{1} & a_{0} \end{bmatrix} \begin{bmatrix} b_{0} \\ b_{1} \\ b_{2} \\ b_{3} \end{bmatrix},$$

where a_n and b_n are two polynomials of degree 3 and c_n is their product:

 $a_n \otimes b_n = c_n$.

Again, this lends itself well to being implemented on a computer.

Finally, *multiplication by the polynomial x* corresponds with a bit-level shift left and an XOR with the hex value '1B'. This can also be represented as a matrix operation:

$\begin{bmatrix} c_0 \end{bmatrix}$		00	00	00	01	$\begin{bmatrix} b_0 \end{bmatrix}$
<i>c</i> ₁		01	00	00	00	b_1
<i>c</i> ₂	=	00	01	00	00	b_2
$\begin{bmatrix} c_3 \end{bmatrix}$		00	00	01	00	b_3

where c_n is the product of x and b_n :

 $x \otimes b_n = c_n$.

The Rijndael Block Cipher

Overview

As you might expect from the background given above, the Rijndael block cipher is designed to use simple whole-byte operations. Its supports independent key and block sizes of 128, 192, or 256 bits. The description of the algorithm given here is for the case where key and block sizes are both 128 bits.

The Rounds

Rijndael is composed of an initial XOR step, nine round transformations (or rounds), and an additional round performed at the end with one step omitted. The input to each round is called the State. Each of the first nine rounds is in tum composed of four transformations:

- ByteSub
- ShiftRow
- MixColumn
- AddRoundKey

The MixColumn transformation is omitted from the tenth round.

The Inputs

Since 128 bits is 16 bytes, our State $(a_{m,n})$ and Cipher Key $(k_{m,n})$ can be represented by 4*4 matrices. Each column contains four consecutive bytes, so each successive row is a word. The order of the bytes in the input block is preserved in this manner.

The State					Th	e Cip	her K	Key
$[a_{3,0}]$	$a_{3,1}$	<i>a</i> _{3,2}	$a_{3,3}$	L k	k _{3,0}	<i>k</i> _{3,1}	<i>k</i> _{3,2}	$k_{3,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	<i>a</i> _{2,3}	k	$k_{2,0}$	$k_{2,1}$	<i>k</i> _{2,2}	k 2,3
$a_{1,0}$	$a_{1,1}$	<i>a</i> _{1,2}	$a_{1,3}$	1	$k_{1,0}$	$k_{1,1}$	<i>k</i> _{1,2}	<i>k</i> _{1,3}
$\int a_{0.0}$	$a_{0.1}$	$a_{0.2}$	$a_{0.3}$	[/	k _{0.0}	$k_{0.1}$	$k_{0.2}$	<i>k</i> _{0.3}

The initial step is to XOR the State with a Round Key. See AddRoundKey, below.

Transformation 1 - ByteSub

In this step, the individual bytes of the input block are substituted according to values given in an S-Box, or Substitution Table. The Rijndael specification includes a formula for creating this S-Box. In brief, a given byte value is replaced with its reciprocal in $GF(2^8)$, multiplied by a bitwise modulo 2 matrix, and XORed with hex '63'. Some sample input and corresponding ByteSub values are:

In put	ByteSub
' 00'	'99'
' 01 '	' 48'
'20'	' 124 '
'FF'	'22'

Transformation 2 - ShiftRow

Next, the individual rows of the State are shifted left as follows:

Row	Offset
0	0
1	1
2	2
3	3

[1	5	9	13]		[1]	5	9	13]
2	6	10	14	ShiftRow	6	10	14	2
3	7	11	15	<u>`</u> →	11	15	3	7
4	8	12	16		16	4	8	12

Transformation 3 - MixColumn

Next, each column of the State is multiplied by the polynomial

$$c(x) = '03'x^3 + '01'x^2 + '01'x + '02',$$

which is equivalent to multiplication by the matrix

 $\begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}.$

Transformation 4 - AddRoundKey

Finally, the Round Key is XORed with the State. An Expanded Key is generated from the Cipher Key by a process called Key Expansion, which can be performed before or during the cipher process. The result is a key whose length is 11 times the length of the original Cipher Key, or 1408 bits in our case. The contents consists of the original Cipher Key, followed by 128-bit blocks consisting of four-byte words such that each word is the XOR of the preceding four-byte word and either the corresponding word in the previous block or a function of it. Each Round Key is a 128-bit block of the Expanded Key.

The Big Picture The steps of Rijndael are as follows:

Initial AddRoundKey Round 1 ByteSub ShiftRow

> MixColumn AddRoundKey

Round 9

Byte Sub ShiftRow MixColumn AddRoundKey Round 10 Byte Sub ShiftRow AddRoundKey

The following is a nice illustration of Rijndeal round:



Figure 1: A Rijndael Round

The Inverse Cipher

The inverse of a round is as follows:

- AddRoundKey
- InverseMixColumn
- InverseShiftRow
- InverseByteSub

The AddRoundKey transformation is a simple XOR, and so is its own inverse. By design, the other transformations are invertible, so decryption is fairly straightforward. This is one of those instances where the nice qualities of $GF(2^8)$ come in handy!

Conclusion

The mathematics of *cryptology* is extremely complex and algorithm described above was designed to thwart the efforts of *cryptanalysts*, or those who attempt to break ciphers. For example, they introduce confusion and diffusion to foil statistical analysis. The true brilliance at work here is of course beyond the scope of this paper. It is, however, possible for us non-cryptologists to at least visualize what might occur to data as it passes through a cipher.

References

- 1. Baltimore Technologies. "Technical Overview of RIJNDAEL The AES." URL: <u>http://dev.baltimore.com/aes/tech_overview.html</u> (24 Jan. 2001).
- 2. Rijmen, Vincent. "Rijndæl." 4 Dec. 2001. URL: http://csrc.nist.gov/encryption/aes/rijndael/ (24 Jan 2001).

- 3. RSA Security. "RSA Laboratories' Frequently Asked Questions about Today's Cryptography, Version 4.1." 2000. URL: http://www.rsasecurity.com/rsalabs/faq/index.html (24 Jan. 2001).
- 4. Savard, John J.G. "The Advanced Encryption Standard (Rijndael)." 2000. URL: http://home.ecn.ab.ca/~jsavard/crypto/co040801.htm (24 Jan. 2001). 5. Schneier, Bruce. *Applied Cryptography*. 2nd Edition, John Wiley & Sons, Inc, 1996.

Upcoming Training

Click Here to {Get CERTIFIED!}



SANS Sydney 2016	Sydney, Australia	Nov 03, 2016 - Nov 19, 2016	Live Event
SANS Gulf Region 2016	Dubai, United Arab Emirates	Nov 05, 2016 - Nov 17, 2016	Live Event
Community SANS Detroit SEC401	Detroit, MI	Nov 07, 2016 - Nov 12, 2016	Community SANS
SANS Miami 2016	Miami, FL	Nov 07, 2016 - Nov 12, 2016	Live Event
SANS London 2016	London, United Kingdom	Nov 12, 2016 - Nov 21, 2016	Live Event
Healthcare CyberSecurity Summit & Training	Houston, TX	Nov 14, 2016 - Nov 21, 2016	Live Event
SANS San Francisco 2016	San Francisco, CA	Nov 27, 2016 - Dec 02, 2016	Live Event
Community SANS Richmond SEC401	Richmond, VA	Dec 05, 2016 - Dec 10, 2016	Community SANS
SANS Dublin	Dublin, Ireland	Dec 05, 2016 - Dec 10, 2016	Live Event
Community SANS San Diego SEC401	San Diego, CA	Dec 05, 2016 - Dec 10, 2016	Community SANS
Community SANS St Louis SEC401	St Louis, MO	Dec 05, 2016 - Dec 10, 2016	Community SANS
SANS Cyber Defense Initiative 2016	Washington, DC	Dec 10, 2016 - Dec 17, 2016	Live Event
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201612,	Dec 13, 2016 - Feb 02, 2017	vLive
SANS Security East 2017	New Orleans, LA	Jan 09, 2017 - Jan 14, 2017	Live Event
Community SANS Virginia Beach SEC401	Virginia Beach, VA	Jan 09, 2017 - Jan 14, 2017	Community SANS
Community SANS Marina del Rey SEC401	Marina del Rey, CA	Jan 09, 2017 - Jan 14, 2017	Community SANS
Mentor Session - SEC401	Philadelphia, PA	Jan 10, 2017 - Feb 21, 2017	Mentor
Community SANS New York SEC401	New York, NY	Jan 16, 2017 - Jan 21, 2017	Community SANS
Community SANS Chantilly SEC401	Chantilly, VA	Jan 23, 2017 - Jan 28, 2017	Community SANS
SANS Las Vegas 2017	Las Vegas, NV	Jan 23, 2017 - Jan 30, 2017	Live Event
SANS vLive - SEC401: Security Essentials Bootcamp Style	SEC401 - 201701,	Jan 30, 2017 - Mar 08, 2017	vLive
Community SANS Albany SEC401	Albany, NY	Feb 06, 2017 - Feb 11, 2017	Community SANS
SANS Southern California - Anaheim 2017	Anaheim, CA	Feb 06, 2017 - Feb 11, 2017	Live Event
Community SANS Seattle SEC401	Seattle, WA	Feb 13, 2017 - Feb 18, 2017	Community SANS
SANS Munich Winter 2017	Munich, Germany	Feb 13, 2017 - Feb 18, 2017	Live Event
SANS Scottsdale 2017	Scottsdale, AZ	Feb 20, 2017 - Feb 25, 2017	Live Event
SANS Scottsdale 2017 - SEC401: Security Essentials Bootcamp	Scottsdale, AZ	Feb 20, 2017 - Feb 25, 2017	vLive
Community SANS Philadelphia SEC401	Philadelphia, PA	Feb 20, 2017 - Feb 25, 2017	Community SANS
Community SANS Minneapolis SEC401	Minneapolis, MN	Feb 27, 2017 - Mar 04, 2017	Community SANS
SANS Dallas 2017	Dallas, TX	Feb 27, 2017 - Mar 04, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Mar 06, 2017 - Mar 11, 2017	Community SANS