Galois Fields for Cryptography

Tom Elliott

April 6, 2019

Contents

1	Introduction	2
2	Modular arithmetic	7
3	Polynomial arithmetic	14
4	Finite fields	20
5	GF(2e3)	25
6	Extended Euclidean algorithm	41
7	GF(2e4)	48
8	Irreducible polynomials	53
9	GF(2e8)	67
10	Code	74
11	More on GF(2e8)	79
12	Matrices	86

Chapter 1

Introduction

Finite fields are called Galois fields, honoring Galois, and often given a name such as GF(7) or GF(2^8). The wikipedia article on finite fields is pretty dense, as is typical for math topics there.

https://en.wikipedia.org/wiki/Finite_field

This short set of chapters is an exploration of Galois fields as they relate to cryptography in AES, the advanced encryption standard. Galois field theory is a famously complex subject, and I don't pretend to understand it.

However, I have worked carefully through the bits of it that relate to AES, and written it here.

Wikipedia:

> As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules. The most common examples of finite fields are given by the integers mod p when p is a prime number.

One of my better sources is

https://engineering.purdue.edu/kak/compsec/Lectures.html

I'll start with some notes on early chapters from Prof. Kak's course on cryptography.

Definitions

Group

A **group** is a set of objects plus a binary operation. The binary operator may be given a generic symbol like o, but a common notation is to use {G,+}, (even if the operation is not really like addition).

Properties: if a,b $\in$ G, then the operations exhibit:

- Closure: $a + b = c \Rightarrow$ c $\in$ G
- Associativity: $(a + b) + c = a + (b + c)$
- Identity element: $a + i = a$
- Inverse element: $a + b = i$

Groups are about *addition*, for some suitable definition of addition.

Typically 0 is used for the identity element $(a + 0 = a)$.

Commutativity is not necessarily a property of a group.

- Commutative: $a + b = b + a$

If a group is commutative it is called an **Abelian group**.

Ring

A **Ring** is a group which also has the multiplication operator $\times$ (even if the operation is not really like multiplication). It may be designated as {R,+,$\times$} and has the following properties:

- Closure: $a \times b \in R$
- Associativity: $(a \times b) \times c = a \times (b \times c)$
- Distributivity: $a \times (b + c) = (a \times c) + (a \times b)$

Repeating from the group definition, for rings we see closure and associativity *under* multiplication. Since there are two operations we can order them in different ways, hence the example for distributivity.

Notation: often the $\times$ is dropped: $a(b + c) = ac + ab$.

As with groups, a ring *may* be

- Commutative: $ab = ba$

An **integral domain** {R,+,$\times$} is a commutative ring that also has a

- Multiplicative identity element: $a \times 1 = a$

If $ab = 0$, then either $a = 0$ or $b = 0$.

Field

A field has all of the above properties. It is an integral domain.

And for every a, there exists a b such that

- Multiplicative inverse: $ab = 1$

1 is its own multiplicative inverse.

According to wikipedia

https://en.wikipedia.org/wiki/Finite_field

> In mathematics, a finite field or Galois field ... is a field that contains a finite number of elements. As with any field, a finite field is a set on which the operations of multiplication, addition, subtraction and division are defined and satisfy certain basic rules.

You can read all about it there. I *think* this conveys the general idea.

Operations

At this point we talk (briefly) about the special operations used for the Galois fields in cryptography.

"Addition" is defined as "exclusive or", XOR, often symbolized $\oplus$.

For binary numbers

$$0 \oplus 0 = 1 \oplus 1 = 0, \qquad 1 \oplus 0 = 0 \oplus 1 = 1$$

$$0011 \oplus 0101 = 0110$$

It is perhaps easier to see if they are lined up properly

```
0011
0101
----
0110
```

Two points:

- binary numbers > 1 can be used
- $x \oplus y = z \rightarrow x \oplus z = y$ and $z \oplus y = x$

Python has an XOR for decimal numbers:

```
>>> 23 ^ 17
6
>>>
```

Explanation:

```
10111 = 23
10001 = 17
------
00110 = 6
```

This *not* addition, there is no carry.

We'll talk about multiplication in a later part. The next chapter is about modular arithmetic.

Chapter 2

Modular arithmetic

Modular arithmetic is all of

- addition
- multiplication
- subtraction
- division

with integers, all carried out modulo (or mod) some integer n.

The rule is to keep the remainder after dividing by the modulus. This called the "mod" operation and is often symbolized % as well as mod.

Addition

Here is an addition table for $n = 4$:

```
    0  1  2  3
   ------------
0 | 0  1  2  3
1 | 1  2  3  0
2 | 2  3  0  1
3 | 3  0  1  2
```

Example: $3 + 2 = 5 \bmod 4 = 1$

Since $36 \bmod 7 = 1$ and $15 \bmod 7 = 1$ as well, it follows that

$$36 = 15 \bmod 7 = 1$$

These two numbers are also equal to $8, 22, 29$ and so on.

Alternatively, we may write

$$36 = 1(\bmod 7)$$

Multiplication

Consider multiplication mod 6 (first table) and mod 7 (second table)

```
  6
  1  |  0  1  2  3  4  5
  2  |  0  2  4  0  2  4
  3  |  0  3  0  3  0  3
  4  |  0  4  2  0  4  2
  5  |  0  5  4  3  2  1

  7
  1  |  0  1  2  3  4  5  6
  2  |  0  2  4  6  1  3  5
  3  |  0  3  6  2  5  1  4
```

```
4  |  0  4  1  5  2  6  3
5  |  0  5  3  1  6  4  2
6  |  0  6  5  4  3  2  1
```

The tables were constructed by standard multiplication, followed by the indicated mod operation.

Looking at the contents of the tables, notice that for $n = 6$, only the rows with multiplications of 1 and 5 generated all the integers less than 6. This is a common pattern, it's always true that 1 and $n-1$ generate all the values $< n$.

That's because

$$a \times 1 + a \times (n-1) = 0 \pmod{n}$$

and that's because

$$a \times 1 + a \times (n-1)) = a + na - a = na = 0 \text{ (mod n)}$$

For $n = 7$, every number generated all the other ones.

Since all integers smaller than n are generated from each starting integer in the special case, a unique result must be produced for each multiplication.

That statement is worth parsing carefully.

If a number, multiplied by all the elements of the field generates all the elements of the field, then at least one of those results must be equal to 1.

The two multiplicands with this property are called multiplicative inverses.

Each row contains all possible values $< n$ if n is prime.

An interesting case is one where n is not prime, yet certain integers other than 1 and $n - 1$ generate all the integers smaller than n.

It turns out that this happens when the smaller number is co-prime with n (they have no common factors other than 1).

So for example, here is the multiplication table for $n = 8$ where all integers smaller than 8 are generated by 3 and 5:

```
8
  1  |  0  1  2  3  4  5  6  7
  2  |  0  2  4  6  0  2  4  6
  3  |  0  3  6  1  4  7  2  5
  4  |  0  4  0  4  0  4  0  4
  5  |  0  5  2  7  4  1  6  3
  6  |  0  6  4  2  0  6  4  2
  7  |  0  7  6  5  4  3  2  1
```

6 is not co-prime to 8 because they share the common factor 2, whereas 3 and 5 are co-prime. So both 3 and 5 are their own multiplicative inverses.

Powers

With $n = 7$, consider the powers of each $i < 7$:

```
    1  2  3  4  5  6
   --------------------
1 | 1  2  3  4  5  6
2 | 2  4  1  2  4  1
3 | 3  2  6  4  5  1
4 | 4  2  1  4  2  1
```

```
5 | 5  4  6  2  3  1
6 | 6  1  6  1  6  1
```

This table can be a challenge to construct. I wrote a script to help with the calculations the first time through.

We discover that the powers of 3 and 5 (but not $2, 4, 6$) give all the integers < 7.

These two (but not only these two) have the property that

- $3^{n-1} = 3^6 = 1 \quad 3^n = 3$

- $5^{n-1} = 5^6 = 1 \quad 5^n = 5$

This is what wikipedia means when they talk about the "q - 1 power of unity".

Take the modulus at each step

> ...in any expression involving $+$, $-$, $\times$ and positive integer exponents (that is, any "polynomial"), if individual terms are replaced by other terms that are congruent to them modulo n, the resulting expression is congruent to the original."

Because of distributivity, a mod can be taken any time. Example:

$$5 + 5 + 5 + 5 + 5 + 5 = 30 \bmod 7 \ = 2$$

Taking the mod at every other step:

$$5 + 5 = 10 \bmod 7 \ = 3$$

$$5 + 5 + 5 + 5 + 5 + 5 = 3 + 3 + 3 = 9 \bmod 7 \ = 2$$

This keeps the numbers small and it works because if $z = x \times y$ then

$$z \bmod n = (x \bmod n) * (y \bmod n)$$

Every integer is equal to an integer q times n, which is divided evenly by n, plus a remainder $r < n$. So

$$x = qn + r$$
$$y = q'n + r'$$

and then

$$x \times y = (qn \times q'n) + (qn \times r') + (r \times q'n) + (r \times r')$$

The only term on the right which is not evenly divisible by n is the last one. Thus

$$(x \times y) \bmod n = (r \times r') \bmod n$$

So, for example

$$5^2 = 25 = 4 \bmod 7$$

and

$$5^4 = (5^2)^2 = 4^2 = 2(\bmod 7)$$

Another example: in the row

```
5 | 5  4  6  2  3  1
```

The value at position 5 is equal to the product of the values at positions 2 and 3: $4 \times 6 = 24 = 3$ (mod 7).

Division

Consider multiplication mod n = 7 again:

7							
1	|	0	1	2	3	4	5	6
2	|	0	2	4	6	1	3	5
3	|	0	3	6	2	5	1	4
4	|	0	4	1	5	2	6	3
5	|	0	5	3	1	6	4	2
6	|	0	6	5	4	3	2	1

Since $2 \times 4 = 1 \pmod 7$, 4 and 2 are multiplicative inverses, similarly $(3, 5)$ and $(6, 6)$ as well as $(1, 1)$. This allows us to say that division by q is the same as multiplication by the multiplicative inverse of q. So

$$5/4 = 2 * 5 = 3(\text{ mod } 7)$$

Chapter 3

Polynomial arithmetic

As you know, a polynomial is an expression of the form:

$$\sum_{0}^{n} a_n x^n$$

where the coefficients come from some set S, for example, the integers:

$$x^5 + 9x^3 + 2x^2 + 1$$

This is a polynomial *of degree* 5.

Polynomial arithmetic deals with addition, multiplication, etc. of polynomials.

Consider this example of division for polynomials with cofactors from the set of real numbers:

$$\frac{8x^2 + 3x + 2}{2x + 1}$$

The first term of the quotient is $4x$ (because $4x \times 2x = 8x^2$) and

$$4x \times (2x + 1) = 8x^2 + 4x$$

so we subtract that from the numerator and the remainder is $-x + 2$ and dividing again

$$\frac{-x+2}{2x+1}$$

The second term of the quotient is -0.5 (because $-0.5 \times 2 = -1$ and

$$-0.5 \times (2x+1) = -x - 0.5$$

Subtracting -0.5 from 2 leaves a remainder of 2.5.

cofactors 0 or 1

The polynomials that we use for cryptography have cofactors a_n equal to either 0 or 1. Thus

$$1 \cdot x^5 + 0 \cdot x^4 + 1 \cdot x^3 + 1 \cdot x^2 + 1 \cdot x^1 + 1 \cdot x^0$$

The terms with zero for the coefficient are not written, the coefficients 1 are suppressed, as so are the exponents for x^1 and x^0, thus:

$$x^5 + x^3 + x^2 + x + 1$$

There are two other representations of polynomials that we'll use quite a bit. The first is to write the coefficients as a binary number:

```
101111 = x^5 + x^3 + x^2 + x + 1
```

The other is to write a digit for the exponent, if the cofactor is not zero, with appropriate spacing:

```
5 . 3 2 1
```

but without the dot

```
5   3 2 1
```

addition and multiplication

Polynomial addition is XOR ($\oplus$):

$$(x^2 + x) + (x^2 + 1) = x + 1$$

In the extended Euclidean algorithm, we will see numbers with a negative cofactor. Since

$$1 + 1 = 0 = 1 - 1$$

$$1 = -1$$

All minus signs can just be dropped. Subtraction is the same as addition.

Polynomial multiplication is as you would expect:

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

except that we also do XOR:

$$= x^5 + x^4 + x^3 + x$$

Using the binary notation:

```
  01011 = x^3 + x + 1
  00110 = x^2 + x
  -----
  10110
 10110
 ------
 111010 = x^5 +x^4 + x^3 + x
```

The same calculation in reverse.

```
 00110 = x^2 + x
 01011 = x^3 + x + 1
 -----
   110
  110
110
------
111010 = x^5 +x^4 + x^3 + x
```

Multiplication by x is the same as multiplication by binary 10. It is a left-shift of one place. Multiplication by 100 is a left-shift of two places. Multiplication by n consists of multiplication by each binary digit of n, followed by XOR.

The results are not the same as from decimal multiplication: 110 is 6 and 1011 is 11, but 111010 is *not* 66. It is $32 + 16 + 8 + 2 = 58$.

There is an additional operation that we are not explaining yet, but don't forget it when the time comes: division by an "irreducible polynomial".

Pseudocode for my routine to multiply $a \times b$ in Python:

```
if a < b:
    switch a,b
reverse the digits of b
r = 0
for each digit c in b (reversed):
    if c == '1':
        r = r ^ a    # addition
    a = a << 1    # left-shift
return r
```

Division

In decimal, multiplication is repeated addition. This is not true in the new system.

$$110 * 1011 = 10 * 1011 + 100 * 1011$$
$$\neq (1011 + 1011) + (1011 + 1011 + 1011 + 1011)$$

Instead, what we've written is just equal to zero.

Multiplication is repeated left-shift by the number of binary digits in the multiplicand, followed by XOR.

In the same way, division is not repeated subtraction (addition). But it has actually the same definition as multiplication. Example:

```
111010 = x^5 +x^4 + x^3 + x
1011     = (x^3 + x + 1) * (x^2)
 ----
 10110
 1011   = (x^3 + x + 1) * (x)
 ----
 00000
```

As we saw, $x^5 + x^4 + x^3 + x$ is evenly divided by $(x^3 + x + 1$. The quotient is $x^2 + x$ (we moved over two units above, and then one unit), and the remainder is 0.

When we do division for the fields to be constructed, division will be by an "irreducible" polynomial. Such a polynomial has no factors in the field, and so the result will never be zero. Justification for this statement will be coming shortly.

Recall that in the definition of a field, the crucial new component was

possession of a multiplicative inverse for each element of the field. For every a there exists a b such that: $ab = 1$.

Then, we can define division by c as multiplication by the inverse of c:

$$\frac{a}{c} = ab, \quad \text{if} \quad bc = 1$$

Chapter 4

Finite fields

Suppose we start doing arithmetic with polynomials whose coefficients belong to a finite field. Example: Z_7 which can also be called $GF(7)$.

We construct such a field simply by doing all our arithmetic modulo 7. If a value is greater than or equal to 7, we divide by 7 and set the value equal to the remainder.

For reference here is the multiplication table mod 7 from earlier

```
7
1  |  0  1  2  3  4  5  6
2  |  0  2  4  6  1  3  5
3  |  0  3  6  2  5  1  4
4  |  0  4  1  5  2  6  3
5  |  0  5  3  1  6  4  2
6  |  0  6  5  4  3  2  1
```

and the power table

```
      1  2  3  4  5  6
----------------------
1  | 1  2  3  4  5  6
```

```
2  | 2  4  1  2  4  1
3  | 3  2  6  4  5  1
4  | 4  2  1  4  2  1
5  | 5  4  6  2  3  1
6  | 6  1  6  1  6  1
```

We will be doing all arithmetic mod 7.

For subtraction we can find the additive inverse of the second term and *add* that to the first term.

Additive inverses

$$1 + 6 = 0 \bmod 7$$

$$2 + 5 = 0 \bmod 7$$

$$3 + 4 = 0 \bmod 7$$

So, for example, subtracting 3 is the same as adding 4

For division find a multiplicative inverse for the denominator and *multiply* the numerator by that. Here is the multiplication table again:

```
7
1  |  0  1  2  3  4  5  6
2  |  0  2  4  6  1  3  5
3  |  0  3  6  2  5  1  4
4  |  0  4  1  5  2  6  3
5  |  0  5  3  1  6  4  2
6  |  0  6  5  4  3  2  1
```

In the table, you can see that 2×4 is equal to 1, so 2 and 4 are multiplicative inverses.

Multiplicative inverses.

1 is its own inverse

$$2 \times 4 = 8 = 1 \bmod 7$$

$$3 \times 5 = 15 = 1 \bmod 7$$

$$6 \times 6 = 35 = 1 \bmod 7$$

so 6 is also its own multiplicative inverse.

The integers mod some number, say $n = 6$ or $n = 7$, with addition and multiplication defined in the standard way, form a finite ring, with additive and multiplicative identities.

However, to be a **field**, for each element, $\{0, 1, 2, 3, 4, 5, 6\}$, there must be a multiplicative inverse — which may be the element itself. If we lay out the multiplication table for arithmetic mod a prime number, then the row for each element contains all of the elements, including 1. There are pairs of elements that when multiplied together give the multiplicative identity, 1. Sometimes, the pair consists of an element which is its own inverse, say $6 \times 6 = 1 \pmod 7$.

We can also see multiplicative inverses in the power table:

```
      1  2  3  4  5  6
----------------------
1   | 1  2  3  4  5  6
2   | 2  4  1  2  4  1
3   | 3  2  6  4  5  1
4   | 4  2  1  4  2  1
5   | 5  4  6  2  3  1
6   | 6  1  6  1  6  1
```

However, there is something unusual about the power table. If you look carefully you will see that only some of the numbers generate all the members of the field as a power of themselves. 3 and 5 do, but 2, 4 and 6 do not.

Kak:

> The number of elements of a finite field is called its order. A finite field of order q exists if and only if the order q is a prime power p^k (where p is a prime number and k is a positive integer).

So not only prime numbers but prime powers p^k can be used to construct fields. If you look back at the table for arithmetic modulo 9, you'll see that rows for integers which are co-prime to 9 have all the elements, but the rows for 3 and 6 do not.

> In a field of order p^k, adding p copies of any element always results in zero; that is, the characteristic of the field is p... The elements of the prime field of order p may be represented by integers in the range $0, ..., p - 1$.

.

So for a finite field of order q $(= 2^k)$, the characteristic of the field is 2.

> ... the polynomial $X^q - X$ has all q elements of the finite field as roots. The non-zero elements of a finite field form a multiplicative group.

So the business about polynomials apparently arises from the fact that we want to construct a Galois field for $2^8 = 256$, which is not prime, but is expressible as a prime power (of 2).

> This group is cyclic, so all non-zero elements can be expressed as powers of a single element called a primitive element of the field. (In general there will be several primitive elements for a given field.)

We will see much more about primitive elements.

To explain more about $X^q - X$: this is an alternative way of writing

X^{q-1} and saying that if an element is the $q-1$ root of unity (i.e. $X^{q-1}=1$) then it can generate all the elements of the field among its powers.

Chapter 5

GF(2e3)

A group is about addition, a ring is about addition plus multiplication, and both satisfy various algebraic properties like closure, associativity and so on. A field is a ring with the important distinction that each element of the field has a multiplicative inverse.

As you might guess, a finite field is a field with a finite number of elements. The order of a finite field is the number of elements in the field.

A familiar type of finite field can be constructed with order equal to a chosen prime number. Z_7 is such a field, constructed using standard addition and multiplication, modulo 7.

prime powers

Rather than being a prime number, alternatively the order can be a prime power, that is, an integer power of a prime. Wikipedia says that there are no other examples of finite fields beyond these two types.

In cryptography we are most interested in the field GF(2^8), but to make the arithmetic easier, we start by exploring GF(2^3).

The usual way to explain the construction is to talk about polynomials of the form

$$ax^2 + bx + c$$

where the cofactors are either 0 or 1 (that's the 2 part of the field's designation).

These are the seven field elements:

$$1$$

$$x$$

$$x + 1$$

$$x^2$$

$$x^2 + 1$$

$$x^2 + x$$

$$x^2 + x + 1$$

Unlike with Z, for GF(2^3), 0 is not an element.

GF(2^n) has, not 2^n elements, but $2^n - 1$.

The rules for arithmetic are those we've already practiced. Addition is XOR with no carry and multiplication is polynomial multiplication:

$$(x + 1)(x + 1) = x^2 + x + x + 1$$

followed by the XOR addition rule:

$$= x^2 + 1$$

The third and final rule is that if the result of multiplication is a polynomial of degree 3 (or higher), carry out "division" by an **irreducible**

polynomial until the result is of degree 2 or less. There are two possible choices for the special polynomial:

$$x^3 + x^2 + 1$$

$$x^3 + x + 1$$

We'll see how these are arrived at in the next section.

For example

$$x(x^2) = x^3$$

$$x^3 \bmod (x^3 + x + 1) = x + 1$$

A nice trick for doing this is to use the binary equivalent:

```
1000 = x^3
1011 = x^3 + x + 1
----
0011
```

The result is $x + 1$, as we will repeatedly verify.

Notice that even though 1000 **is smaller than** 1011 **in binary, we still do the XOR operation, because** 1000 **is of degree 3.**

Continue doing XOR until the result is of degree 2 or less. For example, with x^4 we must promote m by multiplying by $10 \times 1011 = 10110$:

```
10000 = x^4
10110 = x^4 + x^2 + x
-----
00011 = x^2 + x
```

$$x^4 = x^2 + x$$

Multiplication in the binary format is a left-shift. $2x$ is just left-shift one place. $4x$ is left-shift two places..

$3x$ is first $2x$, then XOR with $1x$, the original number.

Now consider

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

One approach is

```
00101 = x^2 + 1
00111 = x^2 + x + 1
------
00101
01010
10100
-----
11011
1011  (mod) x^4 + x^2 + x
----
01101
 1011 (mod) x^3 + x + 1
 ----
 0110
```

It bears repeating that in these fields *multiplication is not repeated addition* and *division is not repeated subtraction (addition).*

We could be tempted to do repeated XOR with the polynomial $x^3 + x + 1$. This doesn't work.

Here, in a different example, it's obvious:

```
  101
```

```
  100
-----
10100
 1011 (mod) x^3 + x + 1
-----
11111
```

And now what? Repeating the XOR just gives 10100 again.

Instead we must start by promoting $x^3 + x + 1$ to the degree required. In the original problem, multiply by $x = 10$.

```
11011
1011  (mod) x^4 + x^2 + x
----
01101
 1011 (mod) x^3 + x + 1
 ----
 0110
```

The answer is

$$\begin{aligned}(x^2 + x + 1)(x^2 + 1) \\ = x^4 + x^3 + x + 1 \\ = x^2 + x\end{aligned}$$

There's another approach which is worth mentioning. We can take the intermediate result

$$= x^4 + x^3 + x + 1$$

and substitute for $x^4 = x^2 + x$ as well as $x^3 = x + 1$ from above

$$\begin{aligned}= x^2 + x + x + 1 + x + 1 \\ = x^2 + x\end{aligned}$$

irreducible polynomials

We will want to find polynomials that do not have factors, that cannot be arrived at by multiplying together two polynomials.

For this multiplication we are not using the special rule mod rule yet, we are not working in the field. We are just looking. However, we do use XOR addition.

The degree 2 or smaller polynomials are

$$1$$

$$x + 1$$

$$x$$

Multiplying by 1 doesn't give anything new.

To go to degree 2 multiply

$$x \cdot x = x^2$$

$$x(x + 1) = x^2 + x$$

$$(x + 1)(x + 1) = x^2 + 1$$

There is one more degree 2 polynomial that is not generated by this procedure:

$$x^2 + x + 1$$

Since $x^2 + x + 1$ has no "factors" in the field (it was not produced by exhaustive multiplication of the field elements), it is irreducible, which is (somewhat) analogous to being prime. It has no factors.

It can be used to form the field GF(2^2), which has three elements. We are going to skip that for now.

Next, by multiplying all pairwise combinations of degree 1 with degree 2 to form degree 3 polynomials, we find eight results and six which are unique.

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

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

Two polynomials of degree 3 are not produced by this method. It is easier to see which they are if we write the results in binary:

```
1000 1010 1001
1100 1110 1111
```

The two missing polynomials, 1011 and 1101, are the irreducible ones.

$$x^3 + x + 1$$

$$x^3 + x^2 + 1$$

We might choose either of these to form the field GF(2^3).

We chose the first one, above. Once we've made the choice, we must stick with it, to produce consistent results.

multiplication table

Constructing a multiplication table at this scale (7 x 7) is a bit of a pain, but it's a useful exercise. Start by re-computing the modulus results for x^3 and x^4:

```
1000 x^3
1011
----
0011 x + 1
```

x^3 is equal to $x + 1$ in this field, and $x^4 = x^2 + x$ because

$$x^4 \bmod x^3 + x + 1 = x^2 + x$$

```
10000 x^4
10110
-----
00110 x^2 + x
```

That was just a left-shift of the irreducible polynomial by one place (corresponding to multiplication by 010).

When we say: "x^3 is equal to $x+1$ in this field" we mean that anywhere x^3 arises it can be replaced by $x + 1$. They are the same.

If there additional terms, say we had $x^3 + x$, the result is $x + 1$ from x^3 then XOR the other term(s): here the result is 1. Thus

$$x^3 + x \bmod x^3 + x + 1 = 1$$

And therefore, factoring, we find that $x^2 + 1$ and x are multiplicative inverses.

Let's start the table

1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x	x^2	.	.	.	.	.
$x+1$	x^2+x	.	.	.	.	.
x^2	$x+1$	.	.	.	.	.
x^2+1	1	.	.	.	.	.
x^2+x	x^2+x+1	.	.	.	.	.
x^2+x+1	x^2+1	.	.	.	.	.

The second column contains each of the elements multiplied by x. For $x(x^2)$ we used the fact that $x^3 = x + 1$ as described above.

Also, $x(x^2 + x) = x^3 + x^2 = (x + 1)$ XOR $x^2 = x^2 + x + 1$.

Finally $x(x^2 + x + 1) = x^3 + x^2 + x = (x + 1)$ XOR $x^2 + x = x^2 + 1$.

Each column and each row will turn out to contain all 7 elements of the field.

1	x	$x+1$	x^2	x^2+1	x^2+x	x^2+x+1
x	x^2	x^2+x	$x+1$	1	x^2+x+1	x^2+1
$x+1$	x^2+x	x^2+1	x^2+x+1	x^2	1	x
x^2	$x+1$	x^2+x+1	x^2+x	x	x^2+1	1
x^2+1	1	x^2	x	x^2+x+1	$x+1$	x^2+x
x^2+x	x^2+x+1	1	x^2+1	$x+1$	x	x^2
x^2+x+1	x^2+1	x	1	x^2+x	x^2	$x+1$

The same 7 elements fill each row and each column. Every polynomial has a unique inverse, and the inverses match (of course).

Having filled out and laborious re-checked the table, it's pretty clear that polynomials are a painful way to do this sort of thing.

Instead, we can try writing binary numbers.

```
001   010   011   100   101   110   111
010   100   110
011   110   101
100   011
101
110
111
```

In filling out the second column, we have reached the first division operation.

$$100 * 010 = 1000 \rightarrow 011$$

$$101 * 010 = 1010 \rightarrow 001$$

It is very fast with pen and paper and not at all error-prone.

Two points: first, this is one way to find multiplicative inverses for a field like GF(2^3): exhaustive calculation.

However, it is not practical for GF(2^8) with 65,000+ multiplications.

Second, the reason for choosing an irreducible polynomial becomes clear: multiplication *must not* produce 0, because 0 is not in the field.

Other methods

What about other approaches? There are at least three.

The one we use extensively elsewhere (in code) is based on knowledge that **0x03** is a primitive element or generator of the field GF(2^8), which is to say that $(\mathbf{0x03})^8 = \mathbf{0x03}$. When wikipedia talks about the $q - 1$ root of unity, this is what they refer to $(\mathbf{0x03})^7 = \mathbf{0x01}$).

We would need to find a generator of the field GF(2^3).

By self-multiplying, we produce a table of powers, and then by inverting the table produce discrete logarithms. Subsequently, find logs which sum to the right number (i.e. 7). We'll see that below.

The second method is the extended Euclidean algorithm. And the third is the one we take up next.

Suppose we want to find the multiplicative inverse of x^2+x in GF(2^3). Write a general polynomial

$$(x^2+x)(ax^2+bx+c)$$

$$= ax^4+bx^3+cx^2+ax^3+bx^2+cx$$

group like terms:

$$= ax^4+(b+a)x^3+(c+b)x^2+cx$$

Now, just substitute for the values of x^4 and x^3 mod the irreducible polynomial x^3+x+1, namely

$$= a(x^2+x)+(b+a)(x+1)+(c+b)x^2+cx$$

so again re-group to obtain

$$= (a+b+c)x^2+(a+b+a+c)x+(a+b)$$

$$= (a+b+c)x^2+(b+c)x+(a+b)$$

The source

https://math.stackexchange.com/questions/2357753/multiplicative-inverse-in-gf23-without-euclidiean-algorithm

has

$$= (b+c-a)x^2+(c-b)x+(a+b)$$

but subtraction is the same as addition, so these are equivalent.

Our goal is that the product should be equal to 1, that is to the polynomial with coefficients $(0, 0, 1)$ thus:

$$a + b + c = 0$$

$$b + c = 0$$

$$a + b = 1$$

Subtracting the second from the first, it is clear that $a = 0$.

If we supposed that $a = 1$ then we would have $b = 0, c = 0$ and $1 + 0 + 0 = 0$, a contradiction. Hence $a = 0$.

Then, $b = 1$ and $c = 1$ also, since $1 + 1 = 0$. Therefore, the inverse should be $x + 1$. Check it:

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

Divide by $x^3 + x + 1$ (or just add $x + 1$) and obtain 1 as the answer. It checks.

generators

Some elements are generators, some are not. I tried 011 because it works with GF(2^8):

```
0011
0011
----
0011
0110
----
0101  =>  0101
==========
```

```
0101
0011
----
0101
1010
----
1111
1011
----
0100  =>  101, 100
==========
0100
0011
----
0100
100
----
1100
1011
----
0111  =>  101, 100, 111
...
```

Rather than go all the way around with this method, let's try polynomials. The last element we got was $x^2 + x + 1$. The generator is $x + 1$ so the next round is

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

Canceling and substituting for x^3:

$$= (x + 1) + 1 = x$$

Then

$$(x+1)(x) = x^2 + x$$

That's five.

$$\begin{aligned}(x+1)(x^2+x) &= x^3 + x^2 + x^2 + x \\ &= (x+1) + x = 1\end{aligned}$$

Finally

$$(x+1)(1) = x+1$$

That's all seven, that's all we need. Here they are, along with their logarithms.

```
011, 101, 100, 111, 010, 110, 001, 011
  1,   2,   3,   4,   5,   6,   7,   8
```

Since the generator to the 0 power is equal to the generator to the 7 power, we need the logs to sum to 7.

As an example, the log of $x^2+x = 110$ is 6. The log of its multiplicative inverse must therefore be 1, which corresponds to 011. We showed already that $011 = x + 1$ is the correct value.

Kak note

According to Kak, the generator g is that element that symbolically satisfies

$$g^3 + g + 1 = 0$$

which implies that

$$g^3 = -g - 1 = g + 1$$

If the irreducible polynomial for GF(2^3) has been chosen as $g^3 + g + 1$, we see the parallel.

The elements are powers of the generator

```
g^0 = 1
g^1 = g
g^2 = g^2
g^3 = g + 1
g^4 = g(g^3) = g^2 + g
g^5 = g(g^4) = g^3 + g^2 = g^2 + g + 1
g^6 = g(g^5) = g^3 + g^2 + g = g^2 + 1
g^7 = g(g^6) = g^3 + g = 1
```

I am not quite sure yet of the difference between this "symbolic" manipulation of the generator and the arithmetic mod m that we do with an actual generator.

We found that binary 11 = 3 is a generator for GF(2^3) with 1011 as the irreducible polynomial.

```
011 * 001 = 0011
011 * 011 = 0101
011 * 101 = 1111 = 100 (mod 1011)
011 * 100 = 1100 = 111 (mod 1011)
011 * 111 = 1001 = 010 (mod 1011)
011 * 010 = 0110
011 * 110 = 1010 = 001 (mod 1011)
```

So that's all 7:

```
001 011 101 100 111 010 110 001
```

The table of logs is

```
001 011 101 100 111 010 110 001
```

```
  0   1   2   3   4   5   6   7
```

Note that with the other possible choice of polynomial

```
011 * 001 = 0011
011 * 011 = 0101
011 * 101 = 1111 = 010 (mod 1101)
011 * 010 = 0110
011 * 110 = 1010 = 111 (mod 1101)
011 * 111 = 1001 = 100 (mod 1101)
011 * 100 = 1100 = 001 (mod 1101)
```

That is

```
001 011 101 010 110 111 100 001
```

Same elements, different order.

So now if we generate a table of logs:

```
001 011 101 010 110 111 100 001
  0   1   2   3   4   5   6   7
```

We find that 010 and 110 are inverses, whereas in the first scheme

```
010 * 101 = 1
110 * 011 = 1
```

So the elements are the same, but the math including multiplicative inverses is all different.

Chapter 6

Extended Euclidean algorithm

We break from the development of GF(2^n) to review the Euclidean algorithm for finding the gcd of two numbers, and its "extended" version.

Let us review the method in decimal first.

Construct two co-prime integers:

$$3 \cdot 7 \cdot 11 = 231$$

$$2 \cdot 5 \cdot 13 = 130$$

and then carry out Euclid's algorithm, keeping the quotient as well as the remainder from each step.

a	b	r	q
231	130	101	1
130	101	29	1
101	29	14	3
29	14	1	2
14	1	0	2

1 is the $GCD(130, 231)$, as we already knew from listing the factors of the two numbers.

Rewrite the results as subtractions

```
  r =   a - (q)  b
101 = 231 - (1)130
 29 = 130 - (1)101
 14 = 101 - (3) 29
  1 =  29 - (2) 14
  0 =  14 - (1) 14
```

One can go forward or backward. I prefer to start from the equation with 1 on the left-hand side:

$$1 = 29 - (2)14$$

Substitute for 14 from the previous equation:

$$1 = 29 - (2)[101 - (3)29]$$

$$= (7)29 - (2)101$$

Then substitute for 29:

$$1 = (7)[130 - 101] - (2)101$$

$$= (7)130 - (9)101$$

And finally, substitute for 101:

$$1 = (7)130 - (9)[231 - 130]$$

$$= (16)130 - (9)231$$

We have thus written 1 as a *linear combination* of a and b.

Finally, do mod 231

$$1 = (16)130 \bmod 231$$

Our result is that 16 and 130 are multiplicative inverses mod 231

$$9 \cdot 231 = 2079 = 16 \cdot 130 - 1$$

There are other inverses revealed as well.

Perhaps it's obvious but still worth pointing out that every stage, we have a valid equation. If you're in trouble, you can check that.

Finite field EEA

So then the question is: how to carry this out for a finite field like $\mathrm{GF}(2^3)$?

If we have two polynomials $a(x)$ and $b(x)$ we can find their gcd, and then find two more polynomials $s(x)$ and $t(x)$ such that

$$gcd(a, b) = s(x)a(x) + t(x)b(x)$$

and if the gcd is 1 then they are co-prime and

$$1 = s(x)a(x) + t(x)b(x)$$

So if $a(x) = m(x)$ is the irreducible polynomial 1011, then certainly the gcd with any polynomial in the field is 1 so we should be able to find

$$1 = s(x)m(x) + t(x)b(x)$$

and then do modulo $m = 1011$ we have

$$1 = t(x)b(x)$$

I have been unable to find a source on the web for this, so I am just trying things. In decimal, the choice of the factor (that we call the quotient q of a below, where $qb = a$) is unambiguous. This is not so in our system. Instead, all we require is that the degree of the resulting qb be the same as a. Sometimes one choice leads to a result more rapidly than another.

So far as I am aware, all choices for q that give the right degree lead to the correct result.

example 1: $x^2 + x$

Recall that $x^2 + x$ is written as 110 I'm going to adopt the convention that elements of the field (degree 2 or less) will be written with as few digits as possible.

We seek gcd(1011,110). The smallest q that works is 10:

```
   a   b   q   qb   r
1011 110  10 1100 111
 110 111  01  111   1
------
111 = 1011 - 10 * 110
  1 =  110 - 01 * 111
    =  110 - 01[1011 - 10 * 110]
    =  11 * 110
    = (x + 1)(x^2 + x)
```

Note $q \cdot b > a$ is still OK.

Suppose we try $q = 11$:

```
   a   b   q   qb   r
1011 110  11 1010   1
------
  1 =  1011 - 11 * 110
    =  11 * 110
    = (x + 1)(x^2 + x)
```

We have that the multiplicative inverse of $x^2 + x$ is $x + 1$, which should not come as a surprise.

$$\begin{aligned}(x^2 + x)(x + 1) &= x^3 + x^2 + x^2 + x \\ &= x^3 + 1 \\ &= (x + 1) + x = 1\end{aligned}$$

example 2: x

We want $\gcd(1101, 010)$.

We might pick $q = 100$ but notice $q = 101$ gives

```
   a   b   q   qb   r
1011 010 101 1010   1
------
  1 = 1011 - 101 * 010
    =  101 * 010
    =  (x^2 + 1)(x)
```

Refer back to above to the multiplication table or the logarithms section to confirm that x and $x^2 + 1$ are inverses. Does $q = 100$ work?

```
   a   b   q   qb   r
1011 010 100 1000  11
  10  11   1   11   1
------
 11 = 1011 - 100 * 010
  1 =   10 - 1 * 11
    =   10 - 1 * [1011 - 100 * 010]
    =  101 * 010
    =  (x^2 + 1)(x)
```

The answer is the same, $x^2 + 1$ and x are inverses.

example 3: $x^2 + x + 1$

```
gcd(m, x^2 + x + 1)
```

```
   a   b   q   qb   r
1011 111  10 1110 101
 111 101   1  101  10
 101  10  10  100   1

101 = 1011 - (10)111
 10 =  111 - (01)101
  1 =  101 - (10)10
    =  101 - (10)[111 - (01)101]
    =  (11)101 - (10)111
    =  (11)[1011 - (10)111] - (10)111
    = -(110)111 - (10)111
    =  (100)111
    = (x^2)(x^2 + x + 1)
```

Suppose we made a different choice for that first quotient:

```
gcd(m, x^2 + x + 1)

   a   b   q   qb   r
1011 111  11 1001  10
 111  10  11  110   1

  10 = 1011 - (11)111
   1 = 111  - (11)10
     = 111 - (11)[1011 - (11)111]
     = 111 + (101)111
     = 100(111)
     = (x^2)(x^2 + x + 1)
```

Provisionally, it seems that one can pick any multiplier that will get the degree right, and we obtain the correct answer.

Chapter 7

GF(2e4)

Let's take a quick look at GF(2^4) before moving on to the main event (GF(2^8)).

These are the polynomial equivalents of the binary numbers with 4 digits, all except 0000. There are 15 elements.

Degree 0:

$$1$$

Degree 1:

$$x \quad x+1$$

Degree 2:

$$x^2 \quad x^2+1 \quad x^2+x \quad x^2+x+1$$

The new ones are degree 3. Write them as binary equivalents:

$$1000 \quad 1001 \quad 1010 \quad 1011$$

$$1100 \quad 1101 \quad 1110 \quad 1111$$

irreducible polynomial

We need an irreducible polynomial. In GF(2^3) we could choose from $x^3 + x + 1$ and $x^3 + x^3 + 1$.

Now we need a polynomial of degree 4. Our method is to multiply together all polynomials of lesser degree and see what's missing.

A degree 4 is obtained either by multiplying a degree 3 times a degree 1, or by multiplying two of degree 2. We'll see the details of this in the next chapter.

I did this and then, to see the patterns, wrote them all without the leading 1 (and grouping by the next digit):

```
10 + 000 001 010 011 100 101 110 111
11 + 000 --- 010 011 100 --- 110 ---
```

I find only 13, so 3 are missing. These are

```
11001 11101 11111
```

We choose $25 = 11001 = x^4 + x^3 + 1$ as the irreducible polynomial for GF(2^4).

generator

I first tried **0x03** and found that it is not a generator for this field.

Rather than guess again, I wrote a short script that calls the gmultiply routine. It's modified to use the irreducible polynomial from above (11001 = decimal 25) and do the mod operation if $n > 15$.

I'll put that code in a later chapter. The output is

```
 2  4  8  9 11 15  7 14  5 10 13  3  6 12  1  2  4
```

```
 3  5 15  8  1  3  5 15  8  1  3  5 15  8  1  3  5
 4  9 15 14 10  3 12  2  8 11  7  5 13  6  1  4  9
 5  8  3 15  1  5  8  3 15  1  5  8  3 15  1  5  8
 ..
 15  3  8  5  1 15  3  8  5  1 15  3  8  5  1 15  3
```

Unexpectedly, 2 and 4 are generators but 3 and 5 are not. Although one can look for repeated values like 5 in the row starting with 3, an easier way is to scan for 1 in the middle of the table.

So

```
 0010
 0010
 ----
 0100 => 0100
 0010
 ----
 1000 => 1000
 0010
 ----
10000
11001 mod
-----
 1001 => 1001
 0010
 ----
10010
11001 mod
-----
 1011 => 1011
..
```

I won't show the whole thing.

I obtained

```
0100 1000 1001 1011 1111 0111 1110 0101
1010 1101 0011 0110 1100 0001 0010
```

and compare that with what the Python code produced:

```
 2  4  8  9 11 15  7 14  5 10 13  3  6 12  1  2
```

That's a match. With the powers, we can compute a table of logarithms.

Here they are: the elements are in the first row and the logarithms below.

```
 2  4  8  9 11 15  7 14  5 10 13  3  6 12  1  2
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
```

According to this, 7 and 14 should be multiplicative inverses because their logarithms add to 15, whose anti-logarithm is 1.

Try it:

```
    0111 =  7
    1110 = 14
    ----
    111
   111
  111
  ----
 101010
 11001 mod
 -----
  11000
  11001 mod
```

```
-----
    1
```

This is pretty tedious. I wrote some code to carry out the procedure, and check the above multiplicative inverses. They are all correct.

The last thing is to try an example of the extended Euclidean algorithm. Find the multiplicative inverse of 5.

```
    a     b     q      bq     r
11001   101   110   11110   111
  101   111     1     111    10
  111    10    11     110     1
-----
111 = 11001 - (110)101
 10 = 101 - (1)111
  1 = 111 - (11)10
    = 111 - (11)[101 - (1)111]
    = (10)111 - (11)101
    = (10)[11001 - (110)0101] - (11)101
    = (1100)101 + (11)101
    = (1111)101
    = (15)(5)
```

which is correct. The log of 5 is 9 and the log of 15 is 6 and the sum of the logs is 15, which is the anti-log of 1.

Chapter 8

Irreducible polynomials

We will usually represent polynomials as binary numbers. The irreducible polynomials of degree 3 are

$$x^3 + x + 1$$

$$x^3 + x^2 + 1$$

For all degrees up to 4, they are

```
11     10
111
1011   1101
10011 11001 11101 11111
```

and 100011011 for degree 8. The question is: how to prove that?

Irreducibility

How to prove that

$$x^8 + x^4 + x^3 + x + 1 = 100011011$$

is irreducible?

The contrary is that two polynomials multiplied together give this as the result If that were so, then one at least must be of degree 4 or less since $x^4 * x^4 = x^8$.

We could generate all the polynomials of degree 4 or less and check them by polynomial division. That's $2^5 + 2^4 + 2^3 + 2^2 + 2 = 63$ multiplications. (This is the NCAA basketball tournament problem — how many games are played?).

There's another way. If the number above *is* reducible, then its factors must themselves be irreducible polynomials.

All 9 of them are written above. We've seen all of them except those of degree 4 before. Let's show find those, and then test the "irreducible polynomial".

As a warmup, determine all the irreducible polynomials of degree 3 or less:

degree 1

```
10 = x
11 = x + 1
```

degree 2

Pairwise multiplication of degree 1 gives degree 2

```
100 = x^2
110 = x * (x + 1) = x^2 + x
101 = (x + 1) * (x + 1) = x^2 + 1
```

These 3 are reducible. What's left? There must be 4 total.

```
111 = x^2 + x + 1
```

must be irreducible.

Check by trial division:

```
111
10
--
 11
 10
 --
  1 r

111
11
--
  1 r
```

I put the r there at the end to show that there was a remainder.

degree 3

Multiply all degree 2's by all degree 1's.

```
1000 = x^3
1100 = x^3 + x^2
1010 = x^3 + x
1110 = x^3 + x^2 + x

1100 = (x + 1) * x^2 = x^3 + x^2
1010 = (x + 1) * (x^2 + x) = x^3 + x
```

```
1111 = (x + 1) * (x^2 + 1) = x^3 + x^2 + x + 1
1001 = (x + 1) * (x^2 + x + 1) = x^3 + 1
```

There are 6 unique ones. It's easier to see if we write them as binary numbers.

```
1000 1100 1010 1110 1111 1001
```

Rearrange:

```
1000 1001 1010
1100 1110 1111
```

What's missing? 1011 and 1101.

Therefore the two irreducibles at degree 3 are:

```
1011 = x^3 + x + 1
1101 = x^3 + x^2 + 1
```

Check by trial division.

We could test against all 4 degree 2's. Because it is degree 3, one degree 2 must be a factor (along with a degree 1). Alternatively, at least one degree 1 must be a factor.

First: $x^3 + x + 1 = 1011$

```
#1
1011
100
 ---
 011 r

#2
1011
101
```

```
 ---
 001 r

#3
1011
110
 ---
 111
 110
 ---
   1 r

#4
1011
111
 ---
 101
 111
 ---
  10 r
```

Next: $x^3 + x^2 + 1 = 1101$

```
#1
1101
100
 ---
 101
 100
 ---
   1 r
```

```
#2
1101
101
 ---
 111
 101
 ---
  10 r

#3
1101
110
 ---
   1 r

#4
1101
111
 ---
  11 r
```

We've confirmed the two postulated irreducibles at degree 3.

degree 4

Self-multiply all four degree 2.

```
             100    101    110    111
100 x ->  10000
101 x     10100  10001
110 x     11000  11110  10100
111 x     11100  11011  10010  10101
```

One duplicate (10100).

Also multiply all eight degree 3 by all degree 1:

```
               10     11
1000 x ->  10000  11000
1001 x     10010  11011
1010 x     10100  10001
1011 x     10110  11011
1100 x ->  11000  10100
1101 x     11010  10111
1110 x     11100  10010
1111 x     11110  10001
```

Figure out what's missing...

```
10000 10001 10010 10011 10100 10101 10110 10111
x     x     x           x     x     o     o

11000 11001 11010 11011 11100 11101 11110 11111
x           o     x     x           x
```

So I seem to have four:

```
10011 11001 11101 11111
```

Trial division

Take all irreducible polynomials of any degree and try to divide the "irreducible" polynomial

Start with the degree 4 irreducibles, and use the trick approach to division:

```
#1
```

```
100011011
10011
-----
   101011
   10011
   -----
     1101 r

#2
100011011
11001
-----
 10001011
 11001
 -----
  1000011
  11001
  -----
   100111
   11001
   -----
    10101
    11001
    -----
     1100 r

#3
100011011
11101
-----
 11001011
```

```
 11101
   -----
   100011
   11101
   -----
    11001
    11101
    -----
      100 r
```

```
#4
100011011
11111
-----
 11101011
 11111
   -----
    10011
    11111
    -----
     1100 r
```

Degree 3 and smaller are five:

```
1101
1011
111
10
11
```

Try them all

```
#1
```

```
100011011
1101
----
 10111011
 1101
 ----
  1101011
  1101
  ----
      011 r

#2
100011011
1011
----
  1111011
  1011
  ----
   100011
   1011
   ----
     1111
     1011
    -----
      100 r

#3
100011011
111
----
 11011011
```

```
 111
 ---
   111011
   111
   ---
      011 r

#4
100011011
10
----
    11011
    10
    ----
     1011
     10
     ----
       11
       10
       --
        1 r

#5
100011011
11
--
 10011011
 11
 --
  1011011
  11
```

```
  --
   111011
   11
   --
     1011
     11
     --
      111
      11
      --
        1 r
```

Checking the last one again, we're asking if "x + 1" divides

```
x^8 + x^4 + x^3 + x + 1
(x+1) * x^7 = x^8 + x^7 XOR above -> x^7 + x^4 + x^3 + x + 1
(x+1) * x^6 = x^7 + x^6 XOR above -> x^6 + x^4 + x^3 + x + 1
(x+1) * x^5 = x^6 + x^5 XOR above -> x^5 + x^4 + x^3 + x + 1
(x+1) * x^4 = x^5 + x^4 XOR above -> x^3 + x + 1
(x+1) * x^2 = x^3 + x^2 XOR above ->x^2 + x + 1
(x+1) * x = x^2 + x XOR above -> 1
```

So, no, it doesn't.

There's got to be a better way. We can do the graphical method:

```
100011011
11
--
 10011011
 11
 --
  1011011
  11
```

```
  --
   111011
   11
   --
     1011
     11
     --
      111
      11
      --
        1 r
```

A better way is to code it. I wrote a script with a function which does "string" division.

Given the irreducible polynomial as ′100011011′‘ and any of the irreducibles up to degree 4 as

```
L = ['10','11',
     '111',
     '1011','1101',
     '10011', '11001', '11101', '11111']
```

it prints something like

```
string division
 100011011
 11001
  10001011
  11001
   1000011
   11001
    100111
```

```
    11001
     10101
     11001
      1100
result:  1100
```

Comparing the output with the calculations I did by hand above, I found four errors, and one bug!

I also found 30 irreducible polynomials of degree 8. Here are the first four:

```
100011011
100011101
100101011
100101101
..
```

You will recognize our old friend as the first one, with the smallest binary representation.

Chapter 9

GF(2e8)

For cryptography we process bytes (integers in the interval $[0, 255]$. The appropriate Galois field is called GF(2^8). We will use cofactors for the polynomials drawn from the set $\{0, 1\}$ as before, but we go up to degree 7.

We will use an irreducible polynomial of degree 8 as the modulus.

We closed the last chapter by showing polynomials defined over $GF(2^3)$ modulo the irreducible polynomial x^3+x+1, which consist of the finite set:

$$0$$

$$1$$

$$x$$

$$x+1$$

$$x^2$$

$$x^2+1$$

$$x^2+x$$

$$x^2+x+1$$

Kak says:

> Our conceptualization of $GF(2^3)$ is analogous to our conceptualization of the set Z_8. The eight elements of Z_8 are to be thought of as integers modulo 8. So, basically, Z_8 maps all integers to the eight numbers in the set Z_8. Similarly, $GF(2^3)$maps all of the polynomials over $GF(2)$ to the eight polynomials shown above.

He continues:

> $GF(2^3)$ contains a unique multiplicative inverse for every non- zero element for the same reason that Z_7 contains a unique multiplicative inverse for every non-zero integer in the set. (For a counterexample, recall that Z_8 does not possess multiplicative inverses for 2, 4, and 6.)
>
> Stated formally, we say that for every non-zero element $a \in GF(2^3)$ there is always a unique element $b \in GF(2^3)$ such that $a \times b = 1$.
>
> The above conclusion follows from the fact if you multiply a non-zero element a with each of the eight elements of $GF(2^3)$, the result will be the eight distinct elements of $GF(2^3)$.
>
> Obviously, the results of such multiplications must equal 1 for exactly one of the non-zero elements of $GF(2^3)$. So if $a \times b = 1$, then b must be the multiplicative inverse for a.
>
> The same thing happens in Z_7. If you multiply a non-zero element a of this set with each of the seven elements of Z_7, you will get seven distinct answers. The answer must therefore equal 1 for at least one such multiplication. When the answer is 1, you have your multiplicative inverse for a.

.

In fact

> For a more formal proof (by contradiction) of the fact that if you multiply a non-zero element a of $GF(2^3)$ with every element of the same set, no two answers will be the same, let's assume that this assertion is false. That is, we assume the existence of two distinct values b and c in the set such that

$$a \times b \equiv a \times c \mod (x^3 + x + 1)$$

which implies

$$a \times (b - c) \equiv 0 \mod (x^3 + x + 1)$$

which implies either $a = 0$ or $b = c$, in either case this is a contradiction.

In exploring multiplication in $GF(2^8)$ constructed with the irreducible polynomial

$$x^8 + x^4 + x^3 + x + 1$$

My approach was to show by exhaustive search that every number has a unique multiplicative inverse, and furthermore for every product other than 1 and every factor a there is a unique b such that $a \times b = p$. We'll get to this later.

addition

The fundamental definition of addition for our Galois field $GF(2^8)$ is that it is the same as the XOR operation:

$$a \oplus b$$

Since $a \oplus a = 0$, it follows that each number is its own additive inverse, with a + a = 0. Addition is the same as subtraction.

And from this a clear implication is that multiplication is *not* repeated addition.

The irreducible polynomial that is used for $GF(2^8)$ is

$$x^8 + x^4 + x^3 + x + 1$$

It is claimed that the finite field $GF(2^8)$ contains 256 distinct polynomials over $GF(2)$.

representations

A common way to write numbers in $GF(2^8)$ is like this

$$= x^7 + x^6 + x^5 + x^2 + x^1 + x^0$$

the exponents on the last two terms are typically suppressed:

$$= x^7 + x^6 + x^5 + x^2 + x + 1$$

If a term is present, then a 1 is present in the place corresponding to the exponent, counting from right to left and starting with index 0. So

$$x^7 + x^6 + x^5 + x^2 + x + 1$$

is the same as binary 1110 0111 or **0xe7**. In fact, exactly the integers 0-255 or hex **00** to **ff** are in this field.

multiplication

Multiplication by 1 is the easiest: $b * 1 = b$, which is a great relief.

Multiplication by any number ≥ 2 consists of two steps: the multiplication itself, possibly followed by a mod operation.

multiplication by 2

For multiplication by 2 there are two cases: if the most significant bit (MSB) of b is not set ($b \leq 127$) then we simply do a left-shift: throw away the left-hand 0 bit from b and insert a 0 bit on the right

$$0111 \;\; 1111 * 2 = 1111 \;\; 1110 = b'$$

If b *does* have its most-significant bit set ($b > 127$), first do the shift (and throw away the bit in the x^8 place). Then do the mod operation, which can be achieved by XOR with 27 (= 0001 1011) :

$$b' \oplus 27$$

We can carry out higher multiplication as successive multiplications and XOR by powers of two. The procedure above carries out XOR every time the product exceeds 256 (and since we're multiplying by 2 it can't exceed 512), So, we never need to worry about products larger than 512.

multiplication by powers of 2

To multiply by 4 simply carry out two successive multiplications by 2.

$$b * 4 = (b * 2) * 2$$

For higher powers, lather, rinse and repeat:

$$b * 8 = (b * 4) * 2 = ((b * 2) * 2) * 2$$

multiplication by any number in GF(2^8)

Suppose we want to multiply $1000\ \ 0011 * b$. First use the distributive law to write

$$\begin{aligned} & 1000\ 0011 * b \\ = & \ [\ 1000\ 0000 + 0000\ 0010 + 0000\ 0001\ ]\ * b \\ = & \ 1000\ 0000 * b + 0000\ 0010 * b + 0000\ 0001 * b \end{aligned}$$

Break the multiplier into its constituent powers of two, multiply as before, and then carry out a final addition step (which is just $\oplus$).

And that is really all we need.

When you get into the mechanics of AES, you 'll find that multiplying a 4-byte sequence like **db 13 53 45** by the matrix below to yield **8e 4d a1 bc**

$$fM = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} * \begin{bmatrix} db \\ 13 \\ 53 \\ 45 \end{bmatrix} = \begin{bmatrix} 8e \\ 4d \\ a1 \\ bc \end{bmatrix}$$

is reversed by multiplying the latter by

$$rM = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix}$$

The reason is that

$$fM \times fM \times fM = rM$$

and

$$fM \times fM \times fM \times fM$$
$$= fM \times rM = I$$

Which means that

$$rM \times (fM \times \mathbf{A})$$
$$= (rM \times fM) \times \mathbf{A}$$
$$= I\mathbf{A} = \mathbf{A}$$

Chapter 10

Code

Once we get to2^5, let alone GF(2^8), it becomes pretty tedious to calculate by hand.

Here is my version of the mod function, set up to work on decimals. The variable mod is the irreducible polynomial. For 2^8 that would be 100011011 which is decimal $256 + 27 = 283$. The exp would be 8 in this case.

```
def nspaces(n):
    b = len(bin(mod))
    return len(bin(n)) - b

def modulus(n,exp,mod):
    while True:
        if n < 2**exp:
            return n
        n = n ^ (mod << nspaces(n))
```

What we're doing is taking bin(283) which is **0b100011011** as a Python string, and then getting the length of that string (which includes the "0b" part, but we leave both here and below on bin(n)).

In the loop, the default value of base is 8 and $2^8 = 256$. If we've already cleared all the high bits, or there were none, just return n. Otherwise $bin(n)$ is at least as large as b, so we promote mod by left-shifting it out the correct amount, and then do XOR with that result and n.

The other part is the multiplication.

```
def multiply(a,b,exp,mod):
    # binary string, reversed
    s = bin(b)[2:][::-1]
    n = 0
    for c in s:
        if c == '1':
            n = n ^ a
        a = a << 1
    return modulus(n,exp,mod)
```

In this function we get the binary digits of the second multiplicand (removing the extra "0b"), and then consume those binary digits one by one. At each stage, if the digit is "1", then XOR a with the accumulator. And at the end, promote a by left-shift. It takes some time to work something like this out.

note

Actually, if you know some Python, you might take a look at the copy of math.py that is in this project. To make calling the functions either, I wrote a wrapper around multiply that returns the multiply function. Thus it "knows" the exp and mod values which are passed into the wrapper.

multiply can be called without supplying exp and mod, but these values can be changed to try different exponents.

text

I test it by reproducing the powers of **0x03** from a reference. My version of the table looks like this in hex. I actually prefer to work with ints, but the reference has hex, so here it is.

```
01 03 05 0f 11 33 55 ff 1a 2e 72 96 a1 f8 13 35
5f e1 38 48 d8 73 95 a4 f7 02 06 0a 1e 22 66 aa
e5 34 5c e4 37 59 eb 26 6a be d9 70 90 ab e6 31
53 f5 04 0c 14 3c 44 cc 4f d1 68 b8 d3 6e b2 cd
4c d4 67 a9 e0 3b 4d d7 62 a6 f1 08 18 28 78 88
83 9e b9 d0 6b bd dc 7f 81 98 b3 ce 49 db 76 9a
b5 c4 57 f9 10 30 50 f0 0b 1d 27 69 bb d6 61 a3
fe 19 2b 7d 87 92 ad ec 2f 71 93 ae e9 20 60 a0
fb 16 3a 4e d2 6d b7 c2 5d e7 32 56 fa 15 3f 41
c3 5e e2 3d 47 c9 40 c0 5b ed 2c 74 9c bf da 75
9f ba d5 64 ac ef 2a 7e 82 9d bc df 7a 8e 89 80
9b b6 c1 58 e8 23 65 af ea 25 6f b1 c8 43 c5 54
fc 1f 21 63 a5 f4 07 09 1b 2d 77 99 b0 cb 46 ca
45 cf 4a de 79 8b 86 91 a8 e3 3e 42 c6 51 f3 0e
12 36 5a ee 29 7b 8d 8c 8f 8a 85 94 a7 f2 0d 17
39 4b dd 7c 84 97 a2 fd 1c 24 6c b4 c7 52 f6 01
```

We won't exercise this a great deal here (see the markdown pages for that).

But it's nice to have it set up so that we can specify the base and the irreducible polynomial. Recall that we generated the powers of **0x02** for GF(2^4):

```
> python gmath.py 3
01 03 05 04 07 02 06 01
> python gmath.py 4
```

```
01 02 04 08 09 0b 0f 07
0e 05 0a 0d 03 06 0c 01
> python gmath.py 5
01 02 04 08 10 05 0a 14
0d 1a 11 07 0e 1c 1d 1f
1b 13 03 06 0c 18 15 0f
1e 19 17 0b 16 09 12 01
>>>>
```

which matches what we had before in decimal.

Here is code that checks all the multiplicative inverses in GF(2^4), which we also saw previously.

```
>>> from gmath import multiplier
>>> f = multiplier(exp=4,mod=25)
>>> for i in range(1,16):
...     for j in range(1,16):
...         if f(i,j) == 1:
...             print i,j
...
1 1
2 12
3 8
4 6
5 15
6 4
7 14
8 3
9 13
10 11
11 10
12 2
```

```
13 9
14 7
15 5
>>>
```

With a few quick changes, it runs just as fast for GF(2^8).

GF(2^5)

I obtained an irreducible polynomial for this field from

http://mathworld.wolfram.com/IrreduciblePolynomial.html

$$x^5 + x^2 + 1 = 100101 = 37$$

```
> python gmath.py 5
  1   2   4   8  16   5  10  20
 13  26  17   7  14  28  29  31
 27  19   3   6  12  24  21  15
 30  25  23  11  22   9  18   1
>
```

I just guessed that **0x02** would work as a generator, and it appears that it does.

Chapter 11

More on GF(2e8)

I found some other write-ups that provide a different perspective and a little more insight.

http://www.cs.utsa.edu/~wagner/laws/FFM.html)

Consider this multiplication: **0xb6 * 0x53**. Write it in binary as:

$$1011\ 0110\ *\ 0101\ 0011$$

We adopt a new notation. Rewrite this as

```
(7 5 4 2 1) * (6 4 1 0)
```

This is a shorthand version of the Galois field notation:

$$(x^7 + x^5 + x^4 + x^2 + x) * (x^6 + x^4 + x + 1)$$

The first line below restates the problem, then we do the multiplication one digit at a time. It turns out that we can multiply by each of the digits, XOR the result, and not worry about the mod until the end.

```
(7 5 4 2 1) * (6 4 1 0)

(7 5 4 2 1) * 6 =      13     11 10   8 7
(7 5 4 2 1) * 4 =           11     9 8   6  5
(7 5 4 2 1) * 1 =                    8   6  5   3 2
(7 5 4 2 1) * 0 =                      7    5 4   2 1
XOR                    13        10 9 8      5 4 3   1
```

Recall above where we said that multiplying by decimal 2, which is the same as multiplying by x in the field notation, is also the same as a left-shift of 1 in the binary representation. In the line $(7\ 5\ 4\ 2\ 1) * 6$, we simply add 6 to each of the digits in the first number to give the result (13,11,10,8,7). The spacing is used to simplify the XOR on the last line.

Now, we do the mod operation.

```
                       13        10 9 8      5 4 3   1
(8 4 3 1 0) * 5        13           9 8    6 5
XOR                                 10        6   4 3   1

(8 4 3 1 0) * 2                     10        6 5   3 2
XOR                                             5 4   2 1
```

The special divisor is $(8, 4, 3, 1, 0)$.

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

We multiply by 5 to get it to align with the 13 in the interim result. This is the same as multiplying the irreducible polynomial by x^5 or multiplying its binary equivalent (1 0001 1011) by 2^5.

After the XOR, the 13 is gone. We repeat the process, lining up with the 10 and zeroing it out. If there had been an 8 we would go that far, but no farther.

Our answer, then, is 0011 0110 which is **0x36**.

The complete multiplication is: **0xb6 * 0x53 = 0x36**.

Effectively what we've done is to do repeated divisions by powers of two of the super duper special number (1 0001 1011). We guarantee that every place higher than 8 will be turned to zero by this operation.

wikipedia example

Another view of the modulus or division operation comes from the wikipedia example: **0x53 * 0xca**. We write this in binary: 0101 0011 $*$ 1100 1010.

In our new notation, this would be

```
(6 4 1 0) * (7 6 3 1)

{6 4 1 0) * 7 =   13    11      8 7
{6 4 1 0) * 6 =      12    10     7 6
{6 4 1 0) * 3 =               9   7     4 3
{6 4 1 0) * 1 =                   7   5     2 1

XOR              13 12 11 10 9 8   6 5 4 3 2 1
```

which looks like it's pretty special. And it is.

Here is the mod step:

```
                    13 12 11 10 9 8   6 5 4 3 2 1
(8 4 3 1 0) * 5 = 13          9 8   6 5

XOR                    12 11 10          4 3 2 1
(8 4 3 1 0) * 4 =      12          8 7   5 4

XOR                       11 10   8 7   5   3 2 1
(8 4 3 1 0) * 3 =         11        7 6   4 3

XOR                          10   8   6 5 4   2 1
(8 4 3 1 0) * 2 =            10       6 5   3 2

XOR                                8       4 3   1
(8 4 3 1 0) * 0 =                  8       4 3   1 0

XOR                                                0
```

That 0 at the end is not really zero. It is a power of 2, or 2^0 = **0x01**. What we've shown is that **0x53 * 0xca = 0x01**. In other words, **0x53** is the *multiplicative inverse* of **0xca**.

Let's take a look at doing the same mod operation as long division, which should hammer home the point we've been absorbing.

```
0011 1111 0111 1110
 ^10 0011 011
  ----------
  01 1100 0001 1110
  ^1 0001 1011
   ----------
   0 1101 1010 1110
    ^1000 1101 1
     ----------
     0101 0111 0110
     ^100 0110 11
      ----------
        1 0001 1010
        1 0001 1011
        ----------
                  1
```

Note, I should not have put those caret marks in this figure, they are just for alignment, and don't mean anything else.

extended Euclidean algorithm

It is possible to apply the "extended" Euclidean algorithm to the problem of finding multiplicative inverses. We developed this topic in connection with GF(2^3).

The calculations for GF(2^8) are a bit hairier, but it seems to work as it should . Here is one example:

```
100011011 is the irreducible polynomial

consider
x^6 + x^3 + x + 1 = 1001011
we need 2 places thus q = 100
ignore the rest
```

```
a          b        q    qb
100011011  1001011  100  100101100

100011011
100101100
---------
   110111

a        b       q   qb
1001011  110111  11  1011001

1001011
1011001
-------
  10010

a       b      q   qb
110111  10010  11  110110
110110
------
     1

Backtrack:

110111 = 100011011 - 100 * 1001011
 10010 =   1001011 -  11 *  110111
     1 =    110111 -  11 *   10010

     1 =    110111 -  11 * [1001011 -  11 *  110111]
       = 100 * 110111 - 11 * 1001011
```

```
       = 100 [100011011 - 100 * 1001011] - 11 * 1001011
       = -10000 * 1001011 - 11 * 1001011
       = (10011) * 1001011
       = (x^4 + x + 1)(x^6 + x^3 + x + 1)

Check

(x^4 + x + 1)(x^6 + x^3 + x + 1)
= x^10 + x^7 + x^7 + x^6 + x^5 + x^4 + x^4 + x^3 + x^2 + x + x + 1
= x^10 + x^6 + x^5 + x^3 + x^2 + 1
= 100 0110 1001

mod
100 0110 1101
100 0110 11
-------------
            1
```

We calculate that $(x^4 + x + 1)(x^6 + x^3 + x + 1) = 1$

In hex, that's **13** * **4b** = **01**. If you check the table of multiplicative inverses, you'll see that's a match.

I don't care to do any more!

Chapter 12

Matrices

So far we've only done one multiplication at a time. Now we graduate to computing what are often called **dot products**.

Here's an example:

```
[2,3,1,1] * [2,1,1,3]
```

The way this is done is to compute the product for each corresponding pair (i.e. $(2*2), (3*1), (1*1), (1*3)$) and then add them together by XOR.

$$(2*2) \oplus (3*1) \oplus (1*1) \oplus (1*3)$$

In binary notation:

```
10 * 10 = 100
11 * 01 = 011
01 * 01 = 001
01 * 11 = 011
          ---
          101
```

The answer is $101 = 5$. It isn't hard to code something like:

```
def xor_reduce(L):
    r = 0
    for n in L:
        r = r ^ n
    return r

def dot(L1,L2):
    rL = [gm(x,y) for x,y in zip(L1,L2)]
    return xor_reduce(rL)
```

The next idea is to put numbers into rows and columns (matrix format), like:

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix}$$

What can we do with this? Try multiplying the matrix from above times a column (often called a vector):

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} \times \begin{matrix} 2 \\ 1 \\ 1 \\ 3 \end{matrix}$$

The result is also a vector, or a column, where the first (top) value is the dot product of the top row of the matrix with the elements of the column. This is the same calculation as we carried out above, and the result is 5. Here is the whole thing:

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} \times \begin{matrix} 2 \\ 1 \\ 1 \\ 3 \end{matrix} = \begin{matrix} 5 \\ 0 \\ 4 \\ 0 \end{matrix}$$

How did we get those zeros? The computation for that zero (just below the 5) is the *second* row dotted with the column:

$$1, 2, 3, 1 \times 2, 1, 1, 3 = 1 * 2 \oplus 2 * 1 \oplus 1 * 3 \oplus 3 * 1$$

We don't even have to calculate. We obtain:

$$1 * 2 \oplus 2 * 1 = 0$$

$$1 * 3 \oplus 3 * 1 = 0$$

Duplicate computations always cancel.

It can be hard to keep track of which row we need at any one time. The way I like to do this is to put the second multiplicand above the space where we're going to write the answer.

$$\begin{matrix} & & & & 2 \\ & & & & 1 \\ & & & & 1 \\ & & & & 3 \end{matrix}$$

times

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} = \begin{matrix} a_{11} \\ a_{21} \\ a_{31} \\ a_{41} \end{matrix}$$

The subscript on a_{21} means that value is the product of row 2 and column 1 (here there is only one column).

We can also multiply two matrices together. If we label this matrix as **fM** (f for forward), try multiplying it times itself.

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix}$$

times

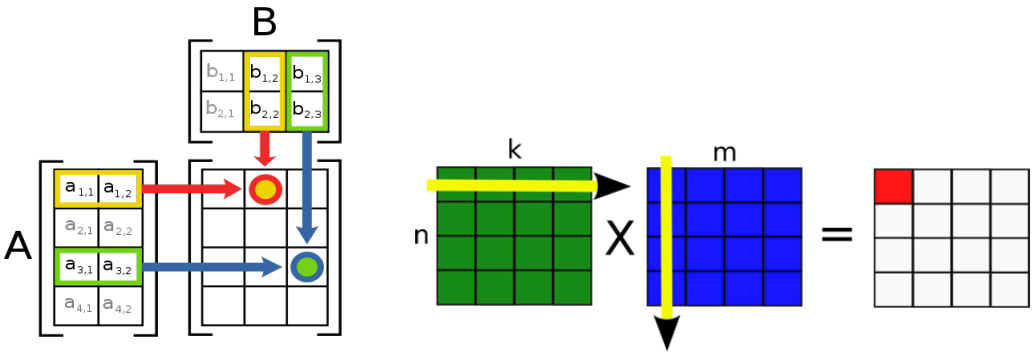
$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} = \begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{matrix}$$

For example, a_{21} is the dot product of the second row and the first column.

If this explanation doesn't make sense see wikipedia.

https://en.wikipedia.org/wiki/Matrix_multiplication

Here are two figures I found on the web, one from that page:



The vertical alignment is a pain to typeset, so I will just do this:

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} \times \begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} = \begin{matrix} 5 & 0 & 4 & 0 \\ 0 & 5 & 0 & 4 \\ 4 & 0 & 5 & 0 \\ 0 & 4 & 0 & 5 \end{matrix}$$

This is fairly tedious, since there are four multiplications and three XORs for each place in the table.

I won't show the code but I wrote a routine to do matrix multiplication with these matrices and format the results. See

https://github.com/telliott99/Crypto/blob/master/AES-math/README.md

The matrix we obtained is sort of interesting. Half the entries are zero, and the rows are shifted right as we go down. But what is very interesting is what happens with one more multiplication:

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} \times \begin{matrix} 5 & 0 & 4 & 0 \\ 0 & 5 & 0 & 4 \\ 4 & 0 & 5 & 0 \\ 0 & 4 & 0 & 5 \end{matrix} = \begin{matrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{matrix}$$

This matrix will be familiar if you've been looking at AES. When encrypting by AES we use what I called the forward matrix **fM**. The above matrix I called the reverse matrix **rM**. We multiply by **rM** to reverse the **fM** step.

The reason it works is that

$$\begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} \times \begin{matrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Commutativity is not generally true for matrices, but in *this case* it is so:

$$\begin{matrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{matrix} \times \begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix} = \begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$$

The reason for commutativity is the repeated structure in the matrices.

Let's just do two of those dot products. There is no irreducible polynomial needed because the numbers are so small.

$$14, 11, 13, 9 \times 2, 1, 1, 3 = 28, 11, 13, 27$$

The XOR is

```
11100
01011
01101
11011
-----
00001
```

and

$$9, 14, 11, 13 \times 2, 1, 1, 3 = 18, 14, 11, 39$$

[This has an error in the last figure]

These are a bit tricky. All results so far are equal to those from decimal multiplication, but the fourth is not.

```
 1101 = 13
   11 = 3
 ----
 1101
11010
-----
10111 = 23
```

The crucial thing is that 10111 = 23 is smaller than 32 so then the XOR is

```
10010 = 18
01110 = 14
01011 = 11
10111 = 23
------
00000
```

If you look at the computations for each coefficient in the matrix, you'll see that the same computations are done for $\mathbf{fM} \times \mathbf{rM}$ as for $\mathbf{rM} \times \mathbf{fM}$.

So finally, we have that:

$[\mathbf{fM}]^3 = \mathbf{fM} \times \mathbf{fM} \times \mathbf{fM} = \mathbf{rM}$

and

$[\mathbf{fM}]^4 = \mathbf{rM} \times \mathbf{fM} = \mathbf{I}$

$\mathbf{I}$ is the identity matrix. $\mathbf{I} \times$ any other matrix is equal to that matrix. So if $\mathbf{P}$ is some matrix we're working on during encryption and we do

$\mathbf{fM} \times \mathbf{P} = \mathbf{C}$

then

$\mathbf{rM} \times (\mathbf{fM} \times \mathbf{P})$

$= (\mathbf{rM} \times \mathbf{fM}) \times \mathbf{P} = \mathbf{I} \times \mathbf{P} = \mathbf{P}$