Finite Fields

Finite Fields

- Message authentication – Galois Counter Mode
- AES S-boxes
- Elliptic curve cryptography

What is a field

- “In mathematics, a field is a set on which addition, subtraction, multiplication, and division are defined and behave as the corresponding operations on rational and real numbers do.”

  --Wikipedia

- In cryptography, we often want to “undo things” or get the same result two different ways
  - Math!
- On digital computers the math you learned in grade school is not good enough
  - Suppose we want to multiply by a plaintext, and the plaintext is 3. Great!
  - Now the decryption needs the inverse operation. Crap!
  - 1/3 is not easy to deal with (not even in floating point or fixed point)

What kind of field do we want for crypto?

- Should be integers
- Should be finite
- Should be efficient in digital logic
  - Even for software implementations

Good YouTube videos...

- https://www.youtube.com/watch?v=Ct2fyigNgPY
- https://www.youtube.com/watch?v=ColSUxhpn6A

Field

- Commutative
  - $a + b = b + a$
  - $a * b = b * a$
- Associative
  - $(a + b) + c = a + (b + c)$
  - $(a * b) * c = a * (b * c)$
- Identity
  - $0 \neq 1$, $a + 0 = a$, $a * 1 = a$
- Inverse
  - $a + -a = 0$
  - $a * a^{-1} = 1$
- Distributive
  - $a * (b + c) = (a * b) + (a * c)$

Integers mod 100

- Commutative? Associative? Identity?
- Inverse?

Integers mod 100

- Commutative? Associative? Identity?
- ~~Inverse?~~
  - Sometimes there is one, e.g., 3 and 67 (201 % 100 = 1)
  - Sometimes not, e.g., 5
  - Integers mod 100 is not a finite field!

Integers mod 101

- Commutative? Associative? Identity?
- Inverse?
  - Every number 0 < i < 101 has a multiplicative inverse
    - Co-prime to 101, because 101 is prime
  - Integers mod 101 **is** a finite field!
    - True of any prime number
    - In general p^k where p is prime and k is positive integer

GF(2)

- Want to define a field over 2^k possibilities for a k-bit number
- 2 is prime, all other powers of 2 are not
  - Need to use irreducible polynomials

https://jedcrandall.github.io/courses/cse539spring2023/miniaesspec.pdf

Published in Cryptologia, XXVI (4), 2002.

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

Raphael Chung-Wei **Phan**

2.1 The Finite Field GF(2^4)

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

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

Example 1

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

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

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

2.2 Addition in GF(2^4)

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

Example 2

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

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

2.3 Multiplication in GF(2^4)

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

Example 3

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

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

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

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

Example 4

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

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

This is because:

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

(quotient)

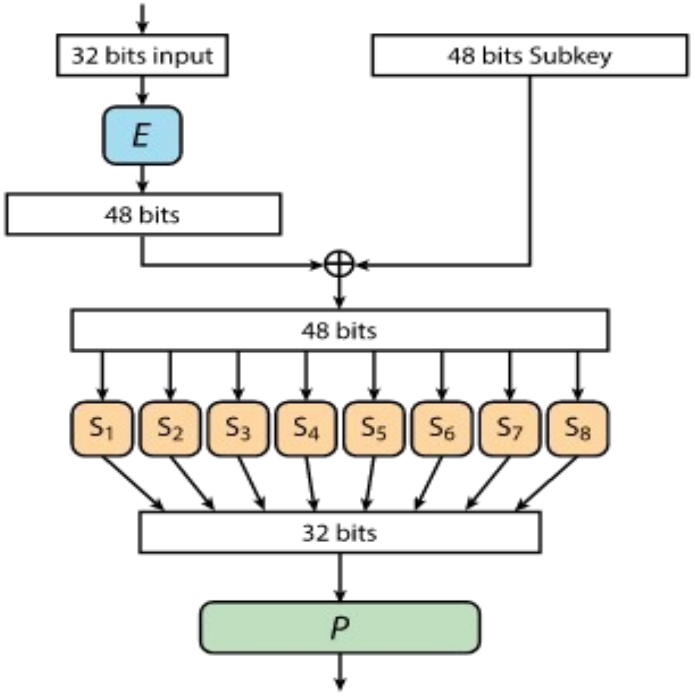
(remainder)

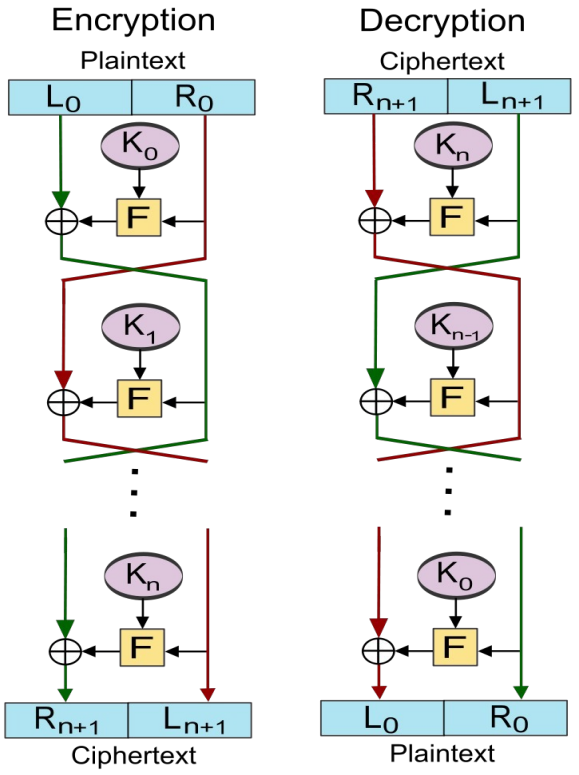
Note that since the coefficients of the polynomials are in {0,1}, then addition is simply exclusive-OR and hence subtraction is also exclusive-OR since exclusive-OR is its own inverse.

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

Why does AES use a finite field?

DES (16 rounds, 64-bit blocks, 56-bit key)





S-boxes

S_1	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0yyyy1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1yyyy0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1yyyy1	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
0yyyy1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
1yyyy0	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
1yyyy1	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3	x0000x	x0001x	x0010x	x0011x	x0100x	x0101x	x0110x	x0111x	x1000x	x1001x	x1010x	x1011x	x1100x	x1101x	x1110x	x1111x
0yyyy0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
0yyyy1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
1yyyy0	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7

How to make S-boxes

- (Have to be invertible if not a Fiestel structure)
- Out of thin air? (DES)
- Randomly? (tricky)
- π? (Blowfish)
- Galois multiplicative inverses? (AES)

Tiny Encryption Algorithm (TEA), Feistel structure with 64 rounds

```c
#include <stdint.h>

void encrypt (uint32_t v[2], const uint32_t k[4]) {
    uint32_t v0=v[0], v1=v[1], sum=0, i;           /* set up */
    uint32_t delta=0x9E3779B9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        sum += delta;
        v0 += ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        v1 += ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}

void decrypt (uint32_t v[2], const uint32_t k[4]) {
    uint32_t v0=v[0], v1=v[1], sum=0xC6EF3720, i;  /* set up; sum is (delta << 5) & 0xFFFFFFFF */
    uint32_t delta=0x9E3779B9;                     /* a key schedule constant */
    uint32_t k0=k[0], k1=k[1], k2=k[2], k3=k[3];   /* cache key */
    for (i=0; i<32; i++) {                         /* basic cycle start */
        v1 -= ((v0<<4) + k2) ^ (v0 + sum) ^ ((v0>>5) + k3);
        v0 -= ((v1<<4) + k0) ^ (v1 + sum) ^ ((v1>>5) + k1);
        sum -= delta;
    }                                              /* end cycle */
    v[0]=v0; v[1]=v1;
}
```

AES is very efficient in both hardware and software

- Gallois multiplication built into hardware
- Different word sizes (8-bit, 16-bit, 32-bit, 64-bit)
- Lots of time-space tradeoffs
  - *E.g.*, rolling operations into the S-boxes
- Lots of parallelism
- Not a Fiestel structure
  - No need to leave half a block untouched every round
- 10, 12 or 14 rounds
  - Corresponds to 128-, 192- or 256-bit keys