Algebra Fields

최승주

https://youtu.be/sX3FXujOMkk

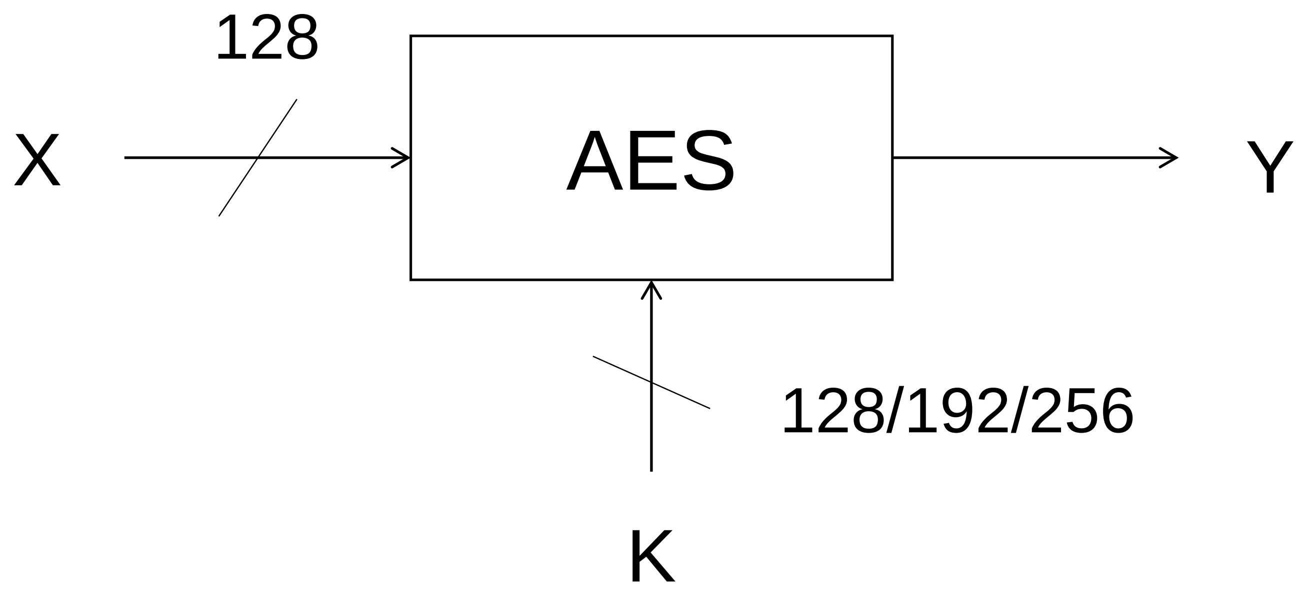
HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

Field in AES

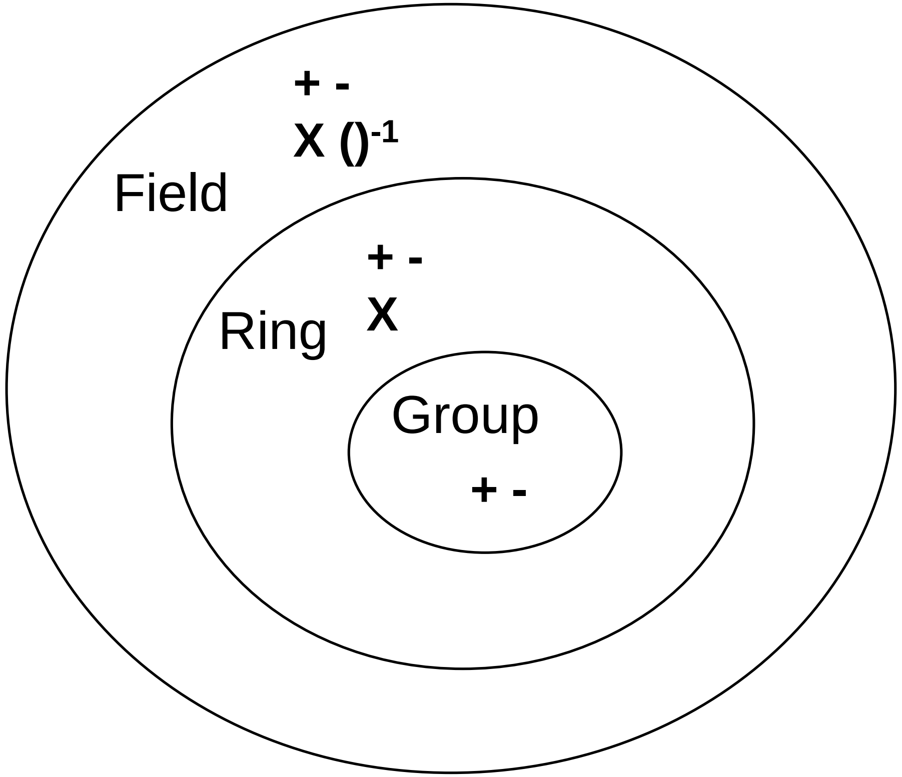


All internal AES operations of AES are based on finite fields

Finite Fields

- Finite Field = Galois Field

- Basic algebraic structures



Finite Fields

- Elements of the Field(F)
  - All elements of F form an additive group with the group operation "+" and the neutral element 0

  - All elements of F except 0 form a multiplicative group with the group operation "x" and the neutral element 1

  - When the two group operations are mixed, the distributivity law holds, i.e., for all $a,b,c \in F$: $a(b+c) = (ab) + (ac)$

Finite Fields

- Field:

  Set of numbers in which we can add, subtract, multiply and divide

- FF only exist if they P^m elements

  P: prime m: integers

  ex)

  (a) There is a FF with 77 → GF(77)

  (b) There is a FF with 81 → GF(81) = GF(3^4)

  (c) There is a FF with 256 → GF(256) = GF(2^8) AES Field

Finite Fields

- Types of FF

$GF(P^m)$

$m = 1$ $m > 1$

$GF(p)$ Prime Fields

$GF(p^m)$ Extension fields

$GF(2^m)$

Prime Fields

- The elements of a prime field GF(p) are the integers {0,1, …,p-1}
- a) add, subtract, multiply

  Let $a,b \in$ GF(p) = {0,1 …, p-1}

  $a + b \equiv c \bmod p$

  $a - b \equiv d \bmod p$

  $a \times b \equiv e \bmod p$
- b) inversion

  $a \in$ GF(p)

  The inversion a^{-1} must satisfy $a \times a^{-1} \equiv 1 \bmod p$

  - Extended Euclidean algorithm

Extension Fields $GF(2^m)$

- a) Element representation
  - The elements of $GF(2^m)$ are polynomials

    $$a_{m-1}X^{m-1} + \ \dots + a_1X + a_0 = A(x) \in GF(2^m)$$

  - $a_i \in GF(2) = \{0,1\}$

    → Prime Field

Extension Fields GF(2^m)

- a) Element representation

  - $a_i \in GF(2) = \{0,1\}$

    → Prime Field

ex) $GF(2^3) = GF(8)$

$A(x) = a_2X^2 + a_1X + a_0$

$GF(2^3) = \{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$

$(a_2,$	$a_1,$	$a_0)$
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Extension Fields GF(2^m)

- b) Addition and Subtraction

  - in GF(2^m)

  → use regular polynomial add or subtraction, where the coefficients are computed in GF(2)

Extension Fields GF(2^m)

- b) Addition and Subtraction

  - in GF(2^m)

  → use regular polynomial add or subtraction, where the coefficients are computed in GF(2)

ex) GF(2^3)

$$
\begin{aligned}
A(x) &= x^2 + x + 1 \\
B(x) &= x^2 + 1 \\
\hline
A+B &= (1+1)x^2 + x + (1+1) \\
&= 0x^2 + x + 0 \\
&= x
\end{aligned}
$$

*Add and Subtraction in GF(2^m) are the same operations

Extension Fields GF(2^m)

- c) Multiplication in GF(2^m)

  - Just do regular polynomial multiplication

  Ex) GF(2^3)

  $$A \times B = (x^2 + x + 1)(x^2 + 1)$$
  $$= x^4 + x^3 + x^2 + x^2 + x + 1$$
  $$= x^4 + x^3 + (1+1)x^2 + x + 1$$
  $$= x^4 + x^3 + x + 1$$

Extension Fields GF(2^m)

- c) Multiplication in GF(2^m)

  recall prime fields

  Ex: GF(7)

  $3 \times 4 = 12$

  GF(7)'s element: 3,4
  *12 is not element of the GF(7)
  *GF(7) = {0, 1, …,6}

Extension Fields GF(2^m)

- c) Multiplication in GF(2^m)

  recall prime fields

  Ex: GF(7)

  $$3 \times 4 = 12 \equiv 5 \bmod 7$$

  GF(7)'s element: 3,4
  *12 is not element of the GF(7)
  *GF(7) = {0, 1, …,6}

Extension Fields $GF(2^m)$

- c) Multiplication in $GF(2^m)$

  - Just do regular polynomial multiplication

  Ex) $GF(2^3)$

  $$A \times B = (x^2 + x + 1)(x^2 + 1)$$
  $$= x^4 + x^3 + x^2 + x^2 + x + 1$$
  $$= x^4 + x^3 + (1+1)x^2 + x + 1$$
  $$= x^4 + x^3 + x + 1 = C'(x)$$

Solution: Reduce C'(x) modulo a polynomial that "behaves like a prime".
These are called irreducible polynomials.

Extension Fields GF(2^m)

- Extension field multiplication

  Let A(x), B(x) ∈ GF(2^m) and let

$$P(x) = \sum_{i=0}^{m} p_i x^i \ , pi \in GF(2)$$

  be an irreducible polynomial. Multiplication of the two elements A(x), B(x) is performed as

$$C(x) \equiv A(x) \cdot B(x) \bmod P(x)$$

Extension Fields $GF(2^m)$

- Irreducible polynomial for $GF(2^3)$

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

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

$$\begin{array}{rl} + & (x^4 + + x^2 + 1) \\ \hline & x^3 + x^2 + 1 \\ + & (x^3 + x + 1) \\ \hline & x^2 + x \equiv A \cdot B \bmod P(x) \end{array}$$

Extension Fields GF(2^m)

- For every field GF(2^m), there are **several** irreducible polynomials

  - In case of the GF(7), there is only one prime number → 7

  - $P(x) = x^3 + x + 1$

  - 해당 P(x)가 어떻게 주어지냐에 따라 modular 연산의 결과값이 달라짐

  The "AES irreducible polynomial"

  → $P(x) = x^8 + x^4 + x^3 + x + 1$

Extension Fields $GF(2^m)$

- d) Inversion in $GF(2^m)$
  - The inverse $A^{-1}(x)$ of an elt, $A(x) \in GF(2^m)$ must satisfy

  - $A(x) \cdot A^{-1}(x) \equiv 1 \bmod P(x)$

Extended Euclidean Algorithm

NTS-KEM

- ff.h, ff.c
- Implementation of Finite Fields

NTS-KEM(ff.h)

```
typedef uint16_t ff_unit;
typedef struct FF2m {
  int m;

  ff_unit (*ff_add)(const struct FF2m* ff2m, ff_unit a, ff_unit b);
  ff_unit (*ff_mul)(const struct FF2m* ff2m, ff_unit a, ff_unit b);
  ff_unit (*ff_sqr)(const struct FF2m* ff2m, ff_unit a);
  ff_unit (*ff_inv)(const struct FF2m* ff2m, ff_unit a);
  ff_unit* basis;

#if defined(INTERMEDIATE_VALUES)

ff_unit *log2poly;
ff_unit *poly2log;
#endif
} FF2m;
FF2m* ff_create();
void ff_release(FF2m* ff2m);
#endif
```

NTS-KEM(ff.h)

```c
typedef uint16_t ff_unit;
```

```c
typedef signed char        int8_t;
typedef short              int16_t;
typedef int                int32_t;
typedef long long          int64_t;
typedef unsigned char      uint8_t;
typedef unsigned short     uint16_t;
typedef unsigned int       uint32_t;
typedef unsigned long long uint64_t;
```

stdint.h

NTS-KEM(ff.c)

```c
typedef struct FF2m {
  int m;

  ff_unit (*ff_add)(const struct FF2m* ff2m, ff_unit a, ff_unit b);
  ff_unit (*ff_mul)(const struct FF2m* ff2m, ff_unit a, ff_unit b);
  ff_unit (*ff_sqr)(const struct FF2m* ff2m, ff_unit a);
  ff_unit (*ff_inv)(const struct FF2m* ff2m, ff_unit a);
  ff_unit* basis;

#if defined(INTERMEDIATE_VALUES)

ff_unit *log2poly;
ff_unit *poly2log;
#endif
} FF2m;
#endif
```

NTS-KEM(ff.c)

- `ff_unit (*ff_add)(const struct FF2m* ff2m, ff_unit a, ff_unit b);`
- `ff_unit (*ff_mul)(const struct FF2m* ff2m, ff_unit a, ff_unit b);`

NTS-KEM(ff.c)

```
ff_unit (*ff_add)(const struct FF2m* ff2m, ff_unit a, ff_unit b)
{
        return a ^ b;
}
```

- $GF(2^m)$
- $X + X = 2X$ ($!X^2$)
- $1 + 1 = 0$
  $1 + 0 = 1$
  $0 + 1 = 1$
  $0 + 0 = 0$

NTS-KEM(ff.c)

```
ff_unit (*ff_mul)(const struct FF2m* ff2m, ff_unit a, ff_unit b){
uint32_t t;
  t = a * (b & 1);
  t ^= (a * (b & 0x0002));
  t ^= (a * (b & 0x0004));
  t ^= (a * (b & 0x0008));
  t ^= (a * (b & 0x0010));
  t ^= (a * (b & 0x0020));
  t ^= (a * (b & 0x0040));
  t ^= (a * (b & 0x0080));
  t ^= (a * (b & 0x0100));
  t ^= (a * (b & 0x0200));
  t ^= (a * (b & 0x0400));
  t ^= (a * (b & 0x0800));
/* Return the modulo reduction of t */
return ff_reduce_12(t);
}
```

NTS-KEM(ff.c)

```
ff_unit (*ff_mul)(const struct FF2m* ff2m, ff_unit a, ff_unit b);
```

- Add Shift Operation

M(11)	C	A	Q(13)	Operation
1011	0	0000	1101	Init
	0	1011	1101	1st A = A + M
	0	0101	1110	Shift Right CAQ
	0	0010	1111	Shift Right CAQ
	0	1101	1111	3th A = A + M
	0	0110	1111	Shift-Right CAQ
	1	0001	1111	4th A = A + M
	0	1000	1111	Shift-Right CAQ

NTS-KEM(ff.c)

```c
ff_unit (*ff_mul)(const struct FF2m* ff2m, ff_unit a, ff_unit b){
uint32_t t;
  t = a * (b & 1);
  t ^= (a * (b & 0x0002));
  t ^= (a * (b & 0x0004));
  t ^= (a * (b & 0x0008));
  t ^= (a * (b & 0x0010));
  t ^= (a * (b & 0x0020));
  t ^= (a * (b & 0x0040));
  t ^= (a * (b & 0x0080));
  t ^= (a * (b & 0x0100));
  t ^= (a * (b & 0x0200));
  t ^= (a * (b & 0x0400));
  t ^= (a * (b & 0x0800));
/* Return the modulo reduction of t */
return ff_reduce_12(t);
}
```

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

NTS-KEM(ff.c)

```
#if defined(INTERMEDIATE_VALUES)
```

KATs and Intermediate Values

- KATs: known answers test
  "A publicly available set of parameters and values that allow you to check the correctness of an implementation.“

  - 구현한 암호 모듈이 알맞게 구현이 된 것인지 확인해 볼 수 있는 입력 값 세트

- NTS-KEM → NIST에서 제공한 코드를 갖고 KATs를 진행
  - PQCgenKAT_kem.c, AES-CTR-DRBG 난수 생성기
  - KAT, intermediate value: set as 100

감사합니다