Galois Field GF(2) and Boolean Algebra

A Comprehensive Guide to Cryptography

Mathematical Foundations and Practical Applications

From Theory to Implementation

July 31, 2025

This document provides a comprehensive understanding of GF(2), Boolean Algebra, and their applications in modern cryptography.
Suitable for beginners with step-by-step examples and mathematical rigor.

Contents

Introduction

Cryptography relies heavily on mathematical structures that provide both security and computational efficiency. Among the most fundamental of these structures are Galois Fields, particularly GF(2), and Boolean Algebra. This document provides a comprehensive introduction to these concepts and their applications in modern cryptography.

Galois Field GF(2) is the smallest finite field, containing only two elements: 0 and 1. It forms the foundation for binary arithmetic and is essential in modern cryptography, error correction codes, and digital signal processing.

Boolean Algebra is a mathematical structure that deals with binary variables and logical operations. It provides the theoretical foundation for digital logic design and is closely related to GF(2) arithmetic.

Mathematical Foundations

What is a Field?

A field is a mathematical structure that satisfies the following axioms:

1. **Closure**: Operations on field elements produce field elements
2. **Associativity**: $(a + b) + c = a + (b + c)$ and $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
3. **Commutativity**: $a + b = b + a$ and $a \cdot b = b \cdot a$
4. **Identity Elements**: There exist 0 and 1 such that $a + 0 = a$ and $a \cdot 1 = a$
5. **Inverse Elements**: For every a, there exists $-a$ such that $a + (-a) = 0$
6. **Distributivity**: $a \cdot (b + c) = a \cdot b + a \cdot c$

Galois Field GF(2)

GF(2) is the simplest finite field, containing only two elements: $\{0, 1\}$.

GF(2) **Arithmetic Rules:**

$$\begin{aligned}
0 + 0 &= 0 \quad \text{(XOR)} \\
0 + 1 &= 1 \\
1 + 0 &= 1 \\
1 + 1 &= 0 \\
0 \cdot 0 &= 0 \quad \text{(AND)} \\
0 \cdot 1 &= 0 \\
1 \cdot 0 &= 0 \\
1 \cdot 1 &= 1
\end{aligned}$$

Addition in GF(2)

Addition in GF(2) is equivalent to the XOR operation:

a	b	$a+b$ (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

Multiplication in GF(2)

Multiplication in GF(2) is equivalent to the AND operation:

a	b	$a \cdot b$ (AND)
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Algebra

Basic Operations

Boolean Algebra operates on binary variables and uses three fundamental operations:

1. **AND** ($\wedge$): $a \wedge b = 1$ if and only if both $a = 1$ and $b = 1$
2. **OR** ($\vee$): $a \vee b = 1$ if either $a = 1$ or $b = 1$ (or both)
3. **NOT** ($\neg$): $\neg a = 1$ if $a = 0$, and $\neg a = 0$ if $a = 1$

Truth Tables

a	b	$a \wedge b$	$a \vee b$	$\neg a$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

Boolean Identities

Key Boolean Identities:

$$
\begin{aligned}
a \wedge 0 &= 0 \quad \text{(Annihilation)} \\
a \wedge 1 &= a \quad \text{(Identity)} \\
a \vee 0 &= a \quad \text{(Identity)} \\
a \vee 1 &= 1 \quad \text{(Annihilation)} \\
a \wedge a &= a \quad \text{(Idempotent)} \\
a \vee a &= a \quad \text{(Idempotent)} \\
\neg(\neg a) &= a \quad \text{(Double Negation)} \\
a \wedge \neg a &= 0 \quad \text{(Contradiction)} \\
a \vee \neg a &= 1 \quad \text{(Tautology)}
\end{aligned}
$$

Relationship Between GF(2) and Boolean Algebra

Correspondence

There is a direct correspondence between GF(2) arithmetic and Boolean operations:

GF(2) Operation	**Boolean Operation**
Addition ($+$)	XOR ($\oplus$)
Multiplication ($\cdot$)	AND ($\wedge$)

XOR Properties

XOR has several important properties that make it useful in cryptography:

1. **Commutativity**: $a \oplus b = b \oplus a$
2. **Associativity**: $(a \oplus b) \oplus c = a \oplus (b \oplus c)$
3. **Identity**: $a \oplus 0 = a$
4. **Self-Inverse**: $a \oplus a = 0$
5. **Distributivity over AND**: $a \wedge (b \oplus c) = (a \wedge b) \oplus (a \wedge c)$

Step-by-Step Examples

Example 1: Basic GF(2) Arithmetic

Let's perform arithmetic operations in GF(2):

Problem: Calculate $(1+0)\cdot(1+1)$ in GF(2)
Step-by-step solution:

$$\begin{aligned}(1+0)\cdot(1+1) &= 1\cdot(1+1) \quad (\text{since } 1+0=1)\\ &= 1\cdot 0 \quad (\text{since } 1+1=0)\\ &= 0 \quad (\text{since } 1\cdot 0=0)\end{aligned}$$

Verification: We can verify this using Boolean operations:

$$(1\oplus 0)\wedge(1\oplus 1) = 1\wedge 0 = 0$$

Example 2: Boolean Expression Simplification

Simplify the Boolean expression: $(a\wedge b)\vee(a\wedge\neg b)$

Step-by-step simplification:

$$\begin{aligned}(a\wedge b)\vee(a\wedge\neg b) &= a\wedge(b\vee\neg b) \quad (\text{Distributivity})\\ &= a\wedge 1 \quad (\text{Tautology: } b\vee\neg b=1)\\ &= a \quad (\text{Identity})\end{aligned}$$

Verification with truth table:

a	b	$a\wedge b$	$a\wedge\neg b$	$(a\wedge b)\vee(a\wedge\neg b)$
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	0	1

The result matches a in all cases, confirming our simplification.

Example 3: XOR Chain Properties

Demonstrate the properties of XOR chains:

Problem: Show that $a\oplus b\oplus b = a$
Solution:

$$\begin{aligned}a\oplus b\oplus b &= a\oplus(b\oplus b) \quad (\text{Associativity})\\ &= a\oplus 0 \quad (\text{Self-inverse: } b\oplus b=0)\\ &= a \quad (\text{Identity})\end{aligned}$$

Verification with truth table:

a	b	$a\oplus b\oplus b$	a
0	0	0	0
0	1	0	0
1	0	1	1
1	1	1	1

Applications in Cryptography

One-Time Pad

The one-time pad is a perfect cipher that uses XOR operations:

One-Time Pad Encryption:

$$\begin{aligned} \text{Ciphertext} &= \text{Plaintext} \oplus \text{Key} \\ \text{Plaintext} &= \text{Ciphertext} \oplus \text{Key} \end{aligned}$$

Example: Encrypt the message "HELLO" using one-time pad
Step 1: Convert to binary

$$\begin{aligned} \text{H} &= 01001000 \\ \text{E} &= 01000101 \\ \text{L} &= 01001100 \\ \text{L} &= 01001100 \\ \text{O} &= 01001111 \end{aligned}$$

Step 2: Generate random key (same length)

$$\begin{aligned} \text{Key} &= 10110011 \\ &= 11001010 \\ &= 00110101 \\ &= 11100011 \\ &= 01010100 \end{aligned}$$

Step 3: XOR plaintext with key

$$\begin{aligned} \text{H} \oplus \text{Key}_1 &= 01001000 \oplus 10110011 = 11111011 \\ \text{E} \oplus \text{Key}_2 &= 01000101 \oplus 11001010 = 10001111 \\ \text{L} \oplus \text{Key}_3 &= 01001100 \oplus 00110101 = 01111001 \\ \text{L} \oplus \text{Key}_4 &= 01001100 \oplus 11100011 = 10101111 \\ \text{O} \oplus \text{Key}_5 &= 01001111 \oplus 01010100 = 00011011 \end{aligned}$$

Step 4: Decrypt by XORing ciphertext with same key

$$\begin{aligned} 11111011 \oplus 10110011 &= 01001000 = \text{H} \\ 10001111 \oplus 11001010 &= 01000101 = \text{E} \\ 01111001 \oplus 00110101 &= 01001100 = \text{L} \\ 10101111 \oplus 11100011 &= 01001100 = \text{L} \\ 00011011 \oplus 01010100 &= 01001111 = \text{O} \end{aligned}$$

Linear Feedback Shift Registers (LFSR)

LFSRs are used in stream ciphers and pseudo-random number generation:

LFSR Operation:

$$s_{n+1} = c_1 s_n \oplus c_2 s_{n-1} \oplus \cdots \oplus c_k s_{n-k+1}$$

where c_i are the feedback coefficients and s_i are the state bits.

Example: 4-bit LFSR with polynomial $x^4 + x + 1$
Initial state: $[1, 0, 1, 1]$
Feedback: $s_{n+1} = s_n \oplus s_{n-3}$
State sequence:

$$\begin{aligned}
\text{State 0 :} & [1,0,1,1] \\
\text{State 1 :} & [0,1,1,1] \quad (1 \oplus 1 = 0) \\
\text{State 2 :} & [1,1,1,0] \quad (0 \oplus 1 = 1) \\
\text{State 3 :} & [1,1,0,1] \quad (1 \oplus 0 = 1) \\
\text{State 4 :} & [1,0,1,1] \quad (1 \oplus 1 = 0)
\end{aligned}$$

The sequence repeats every 15 states (period $= 2^4 - 1 = 15$).

S-Boxes in Block Ciphers

S-Boxes use Boolean functions to provide non-linearity:

Example: Simple 2x2 S-Box
Input: 2-bit value $[a, b]$ **Output**: 2-bit value $[x, y]$ where:

$$x = a \oplus b$$
$$y = a \land b$$

Truth table:

a	b	$x = a \oplus b$	$y = a \land b$
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

This S-Box provides both linear (XOR) and non-linear (AND) operations.

Advanced Concepts

GF(2^n) Fields

While GF(2) is the simplest field, cryptography often uses GF(2^n) for larger fields:

GF(2^n) Construction:

$$\mathrm{GF}(2^n) = \frac{\mathrm{GF}(2)[x]}{\langle p(x) \rangle}$$

where $p(x)$ is an irreducible polynomial of degree n.

Example: GF(2^3) with irreducible polynomial $p(x) = x^3 + x + 1$
Elements: $\{0, 1, x, x+1, x^2, x^2+1, x^2+x, x^2+x+1\}$
Addition: Component-wise XOR

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

Multiplication: Polynomial multiplication modulo $p(x)$

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

Boolean Functions and Cryptography

Boolean functions are crucial for S-Box design and cryptographic security:

Important Properties:

1. **Non-linearity**: Distance from linear functions
2. **Balancedness**: Equal number of 0s and 1s in truth table
3. **Correlation immunity**: Resistance to correlation attacks
4. **Algebraic degree**: Highest degree term in algebraic normal form

Practical Implementation

C Implementation

```c
// GF(2) addition (XOR)
uint8_t gf2_add(uint8_t a, uint8_t b) {
    return a ^ b;
}

// GF(2) multiplication (AND)
uint8_t gf2_mul(uint8_t a, uint8_t b) {
    return a & b;
}

// Boolean operations
uint8_t bool_and(uint8_t a, uint8_t b) {
    return a & b;
}

uint8_t bool_or(uint8_t a, uint8_t b) {
    return a | b;
}
```

```c

uint8_t bool_not(uint8_t a) {
    return ~a;
}

uint8_t bool_xor(uint8_t a, uint8_t b) {
    return a ^ b;
}
```

Listing 1: GF(2) Operations in C

Python Implementation

```python
class GF2:
    def __init__(self, value):
        self.value = value & 1

    def __add__(self, other):
        return GF2(self.value ^ other.value)

    def __mul__(self, other):
        return GF2(self.value & other.value)

    def __str__(self):
        return str(self.value)

# Boolean operations
def bool_and(a, b):
    return a & b

def bool_or(a, b):
    return a | b

def bool_not(a):
    return ~a & 1

def bool_xor(a, b):
    return a ^ b

# Example usage
a = GF2(1)
b = GF2(1)
print(f"{a} + {b} = {a + b}")  # Output: 1 + 1 = 0
print(f"{a} * {b} = {a * b}")  # Output: 1 * 1 = 1
```

Listing 2: GF(2) Operations in Python

Security Considerations

Linear Cryptanalysis

Linear cryptanalysis exploits linear relationships in cryptographic functions:

Linear Approximation:

$$P[a \cdot x \oplus b \cdot y = 0] = \frac{1}{2} + \epsilon$$

where ϵ is the bias and a, b are masks.

Differential Cryptanalysis

Differential cryptanalysis studies how input differences propagate:

Differential Probability:

$$P[\Delta y = \beta | \Delta x = \alpha] = \frac{\#\{x : F(x) \oplus F(x \oplus \alpha) = \beta\}}{2^n}$$

Algebraic Attacks

Algebraic attacks solve systems of equations over GF(2):

Example: Simple algebraic attack on a cipher
Given a cipher with equations:

$$\begin{aligned} x_1 \oplus x_2 \oplus x_3 &= y_1 \\ x_2 \oplus x_3 \oplus x_4 &= y_2 \\ x_1 \oplus x_3 \oplus x_4 &= y_3 \end{aligned}$$

Solution using Gaussian elimination:

$$\begin{aligned} x_1 \oplus x_2 \oplus x_3 &= y_1 \\ x_2 \oplus x_3 \oplus x_4 &= y_2 \\ x_1 \oplus x_3 \oplus x_4 &= y_3 \end{aligned}$$

Adding equations 1 and 3:

$$x_2 \oplus x_4 = y_1 \oplus y_3$$

This reduces the system and can help recover the key.

Modern Applications

AES S-Box

The AES S-Box is constructed using GF(2^8) arithmetic:

AES S-Box Construction:

1. Find multiplicative inverse in GF(2^8)
2. Apply affine transformation
3. Result provides high non-linearity and algebraic complexity

Hash Functions

Many hash functions use GF(2) operations extensively:

Example: SHA-256 uses GF(2) operations
Ch function: $Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$
Maj function: $Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z)$
Sigma functions:

$$\Sigma_0(x) = (x \ggg 2) \oplus (x \ggg 13) \oplus (x \ggg 22)$$
$$\Sigma_1(x) = (x \ggg 6) \oplus (x \ggg 11) \oplus (x \ggg 25)$$

All operations are performed in GF(2) arithmetic.

Conclusion

Galois Field GF(2) and Boolean Algebra form the mathematical foundation of modern cryptography. Understanding these concepts is essential for:

- Designing secure cryptographic algorithms
- Analyzing cryptographic security
- Implementing efficient cryptographic systems
- Understanding advanced cryptographic techniques

Key Takeaways:

1. GF(2) arithmetic is equivalent to XOR and AND operations
2. Boolean Algebra provides the theoretical framework for digital logic
3. These concepts are fundamental to symmetric cryptography
4. Understanding enables both design and cryptanalysis

This document provides a comprehensive introduction to GF(2) and Boolean Algebra in cryptography.
For advanced applications, refer to specialized cryptographic literature and standards.