

Finite Fields (Fields and Modular Arithmetic)

Prepared by Dr. Sumaya Alkubaisi

FNT625
Cybersecurity Concepts

Lecture 6



Lecture

6

Learning Goals

By the end of this lecture, you will be able to:

- Understand modular arithmetic and its properties.
- Explain what fields and finite fields are.
- Perform basic operations in modular arithmetic (addition, subtraction, multiplication, inverse).
- Recognize how these principles apply in encryption algorithms.

Why Study Finite Fields?

- All modern encryption algorithms — AES, RSA, ECC — rely on mathematics of finite fields.
 - These ensure predictable (follows strict mathematical rules), repeatable (same inputs give same results), and reversible (you can undo the operation — important for decryption operations) on numbers.
 - In FinTech, they're used in secure digital transactions, blockchain consensus, and digital signatures.
- ❖ In Lectures 4–5, we studied how DES uses bit-level operations.
In modern cryptography (like AES), we need structured mathematics — that's where finite fields come in.

Lecture

6

What is Modular Arithmetic?

It's math where numbers “wrap around” after reaching a certain value — the modulus (n).

Think of it like a clock: after 12, it goes back to 1.

Example:

$$17 \equiv 2 \pmod{5}$$

because both 17 and 2 leave a remainder of 2 when divided by 5.

Explanation:

$$17 \div 5 = 3 \text{ remainder } 2$$

$$2 \div 5 = 0 \text{ remainder } 2$$

Lecture

6

What is Modular Arithmetic?

Example:

$$17 \equiv 2 \pmod{5}$$

because both 17 and 2 leave a remainder of 2 when divided by 5.

We're talking about mod 5, which means numbers “wrap” after 5.

Think of it like a circle with numbers 0, 1, 2, 3, 4 — and then it goes back to 0 again.

Now, start counting up to 17:

$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow (\text{wrap}) 0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow (\text{wrap}) \dots 0 \rightarrow 1 \rightarrow 2$

When you reach 17, you land on 2 in the circle.

That's why:

$$17 \equiv 2 \pmod{5}$$

Lecture

6

Why It Matters in Cryptography

- Computers work with limited-size numbers, so modular arithmetic keeps results within range.
It allows encryption operations to stay finite and reversible.
- Examples:
- **RSA** → uses mod n :
 - we choose two large prime numbers, called p and q , then we multiply them to get n :
 $n = p \times q$
 - That n becomes the modulus — it's the number where all encryption and decryption operations “wrap around.”
- **AES** → uses mod 2 (binary numbers)

Lecture
6

Basic Operations (mod 7 example)

Operation	Example	Result
Addition	$(5 + 4) \bmod 7$	2
Subtraction	$(3 - 5) \bmod 7$	5
Multiplication	$(3 \times 4) \bmod 7$	5
Division	multiply by modular inverse	$3^{-1} \bmod 7 = 5$

- Those are also randomly chosen examples to show how modular arithmetic works.
 $(5 + 4) \bmod 7 = 2$ just demonstrates addition wrapping around.
 $(3 \times 4) \bmod 7 = 5$ just demonstrates multiplication under mod 7.
- In real cryptography, though —the numbers (like modulus, keys, etc.) are not random examples; they're carefully generated using algorithms to make them large, secure, and mathematically valid

Modular Inverse

- The inverse of a number under mod n is another number that gives 1 when multiplied mod n .
- In other words, the **modular inverse** of a is b if $a \times b \equiv 1 \pmod{n}$

Example:

$$3 \times 5 = 15 \equiv 1 \pmod{7} \rightarrow \text{so } 3^{-1} = 5$$

- Used in RSA, ECC, and key exchange.

You can find it by:

- Testing small numbers manually, or
- Using the Extended Euclidean Algorithm (for big cryptographic numbers).

Properties of Modular Arithmetic

- **Closure Property**

For any integers a and b , the result of $(a + b) \bmod n$ and $(a \times b) \bmod n$ will always be between 0 and $n-1$.

Example: $(4 + 9) \bmod 7 = 13 \bmod 7 = 6$ (still within 0–6).

This means results “wrap around” and never leave the set $\{0, 1, 2, \dots, n-1\}$.

- **Associative Property**

You can group operations in any way — the result won't change.

- $(a + b) + c \equiv a + (b + c) \pmod{n}$

- $(a \times b) \times c \equiv a \times (b \times c) \pmod{n}$

Example: $((2 + 3) + 4) \bmod 5 = (2 + (3 + 4)) \bmod 5 = 4$

Properties of Modular Arithmetic

- **Commutative Property**

The order of addition or multiplication doesn't matter.

- $a + b \equiv b + a \pmod{n}$
- $a \times b \equiv b \times a \pmod{n}$

Example: $(3 \times 4) \bmod 5 = (4 \times 3) \bmod 5 = 2$

- **Distributive Property**

Multiplication distributes over addition.

- $a \times (b + c) \equiv (a \times b + a \times c) \pmod{n}$

Example: $3 \times (2 + 4) \bmod 5 = 3 \times 6 \bmod 5 = 3$

$(3 \times 2 + 3 \times 4) \bmod 5 = (6 + 12) \bmod 5 = 18 \bmod 5 = 3$

Properties of Modular Arithmetic

- **Identity Elements**

- Additive identity: 0 (because $a + 0 \equiv a \pmod{n}$)
- Multiplicative identity: 1 (because $a \times 1 \equiv a \pmod{n}$)

- **Existence of Inverses**

- Additive inverse: for every a , there exists $-a$ such that $a + (-a) \equiv 0 \pmod{n}$.
Example: $3 \pmod{7} \rightarrow$ additive inverse is 4 (since $3 + 4 \equiv 0 \pmod{7}$).
- Multiplicative inverse: for every a that is coprime with n , there exists b such that $a \times b \equiv 1 \pmod{n}$.
Example: if $n = 7 \rightarrow 3 \times 5 \equiv 15 \equiv 1 \pmod{7}$, so inverse of 3 is 5.

Lecture

6

Examples: Mod in Daily Life

- In real-life: A clock is mod 12 math:
 $10 + 5 = 3 \rightarrow (10 + 5) \bmod 12 = 3$
- Cryptography uses the same logic, but with binary numbers.
- In FinTech, Modular arithmetic is used in:
 - Blockchain hashing
 - Public key generation (RSA)
 - Digital wallets (ECC arithmetic)

Introduction to Fields

- What Is a Field?

A field is a mathematical system where you can:

- Add, subtract, multiply, and divide (except by zero),
- And always get results within the same set.

Example: Real numbers $\mathbb{R}$ (7, -3, 0, 2.5, $\sqrt{2}$, π), Rational numbers $\mathbb{Q}$ ($\frac{1}{2}$, -3, 0.75).

Field Properties

To be a field, a set must satisfy:

- Closure (results stay in the set)
- Associativity (grouping doesn't matter)
- Commutativity (order doesn't matter)
- Identity elements (0, 1)
- Inverse elements (additive and multiplicative)
- Distributive property ($a \times (b + c) = ab + ac$)

Why Fields in Cryptography?

- Fields ensure every operation has a valid result and inverse.
- Essential for reversible encryption/decryption.

Example: AES uses finite field $GF(2^8)$ for its byte-level operations.

$GF(2^8)$ = “finite field with 256 possible values (0–255), used in AES encryption.”

Example: Integers Mod 5 can form a Field

- Set: $\{0,1,2,3,4\}$
- Addition and multiplication modulo 5 $\rightarrow$ valid results.

Operations (mod 5):

Addition: $3+4=7\equiv 2 \rightarrow$ result is in the set

Multiplication: $3\times 2=6\equiv 1 \rightarrow$ result is in the set

- Field properties:
 - Closure: All results stay in the set
 - Inverse exists: $3^{-1}=2$ since $3\times 2\equiv 1 \pmod{5} \rightarrow 1$ is in the set.
 - Other properties (associative, commutative, identity) hold

Conclusion: Mod 5 is a finite field

Non-Field Example (Mod 6)

- The set is $\{0, 1, 2, 3, 4, 5\}$.
- When we multiply $2 \times 3 = 6 \equiv 0 \pmod{6}$
 - This means 2 and 3 are zero divisors (two nonzero numbers giving 0).
 - Neither 2 nor 3 has an inverse (no number multiplied by 2 gives 1 mod 6).
- Therefore, mod 6 is not a field.
 - the field rule says: If you multiply two nonzero elements (not 0) in a field, the result should never be 0.
 - Why? Because nonzero elements must have inverses in a field.
 - If a nonzero number could multiply with another nonzero number to get 0, it couldn't have an inverse, so the set wouldn't be a field.

Rule :

Integers mod p form a field only if p is prime.

Prime Fields $GF(p)$

A Galois Field (GF) is a finite field, meaning it has a finite number of elements and satisfies all field properties (closure, associativity, commutativity, identities, inverses, distributivity).

- $GF(p)$ is a finite field based on a prime number, ensuring all nonzero elements have inverses.
- Specifically:
 - Prime Field $GF(p)$
 - “GF” stands for Galois Field.
 - If p is a prime number, the field is written as $GF(p)$.
- The set of elements is: $GF(p)=\{0,1,2,\dots,p-1\}$

Prime Fields $GF(p)$

- All operations — addition, subtraction, multiplication, and division (except by 0) — are done mod p .

Example:

- $GF(7)=\{0,1,2,3,4,5,6\}$
- $3 \times 5 = 15 \equiv 1 \pmod{7} \rightarrow 5$ is the multiplicative inverse of 3.

Differences from field:

- A Galois Field is just a finite field.
- All Galois Fields are fields, but not all fields are finite.

Example:

- $\mathbb{R}$ (real numbers) $\rightarrow$ field, infinite, not a Galois Field
- $GF(5) \rightarrow$ field, finite, is a Galois Field

Lecture

6

Field Size Notation ($\text{GF}(p^n)$)

Symbol	Meaning	Example
$\text{GF}(p)$	Prime field (p is prime)	$\text{GF}(2) \rightarrow \{0,1\}$
$\text{GF}(p^n)$	Extension field — built from $\text{GF}(p)$ but with n -bit elements	$\text{GF}(2^8)$ used in AES

- $\text{GF}(p^n)$ means there are p^n elements.

Example: $\text{GF}(2^8)$ has $2^8=256$ elements (values from 0–255).

- AES uses $\text{GF}(2^8)$ for its byte operations — each byte is treated as an element of this field, allowing special math rules that ensure every byte has an inverse.

What is a Polynomial?

- A polynomial is an expression made of terms that include powers of a variable (like x) with coefficients from a field (like 0 or 1 in $GF(2)$).
- General form: $a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$ (where each a_i is a coefficient)

Example over $GF(2)$: $x^3 + x + 1$

- $GF(2)$ has only two elements: 0 and 1. So the coefficients in the polynomial can only be 0 or 1.

This actually means: $1x^3 + 0x^2 + 1x + 1$

So, the coefficients are:

for x^3 : 1

for x^2 : 0

for x^1 : 1

for x^0 : 1

If we write the coefficients as bits (from x^3 down to x^0): $\rightarrow$ **1011** (a 4-bit binary number representing the polynomial)

Why Polynomials?

- In modern ciphers (like AES), data (bytes) are not treated as numbers but as polynomials over GF(2).
- Each bit represents a coefficient (0 or 1) in a polynomial.

Example:

Byte: 10010101

Polynomial: $x^7 + x^4 + x^2 + 1$

(Bits with 1 become powers of x ; bits with 0 are skipped.)

Polynomial Example 2:

Bit position	Power of x	Bit value	Include term?
7	(x^7)	0	No
6	(x^6)	1	Yes
5	(x^5)	1	Yes
4	(x^4)	0	No
3	(x^3)	1	Yes
2	(x^2)	0	No
1	(x^1)	0	No
0	(x^0)	1	Yes

- Let's take the byte: 01101001
- the polynomial is: $x^6+x^5+x^3+1$
- That's how binary → polynomial conversion works in GF(2).
Only the bits with 1 become terms in the polynomial.

Polynomial Operations

- **Addition:** Done bitwise with XOR (no carries).

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

- **Multiplication:** Multiply normally, but then take mod an irreducible polynomial (just like mod prime in integers).

Addition = XOR

Multiplication = Modulo an irreducible polynomial

Lecture

6

Addition of polynomials over GF(2) = XOR

- In GF(2), addition works exactly like XOR (exclusive OR):

Term 1	Term 2	Result (mod 2)	XOR Logic
0	0	0	$0 \oplus 0 = 0$
0	1	1	$0 \oplus 1 = 1$
1	0	1	$1 \oplus 0 = 1$
1	1	0	$1 \oplus 1 = 0$

- So when you add $(x^3+x+1)+(x^2+x)$
- combine like terms:
 $x^3 + x^2 + (x+x) + 1$
- since $x+x=0$ (because $1 \oplus 1 = 0$)
- Final result = $x^3 + x^2 + 1$

Irreducible Polynomial

- When you multiply polynomials in GF(2), the degree (power) might grow — to keep results inside the field, we take mod (remainder) by a special irreducible polynomial (a polynomial that can't be factored, just like a prime number).
- Used as the “modulus” to keep results inside the field.
- Example (used in AES):
$$x^8+x^4+x^3+x+1$$
- This polynomial is irreducible over GF(2).

Lecture

6

Example: Modular Polynomial Arithmetic

Compute: $(x^3+x+1)(x^2+1) \bmod (x^4+x+1)$

Steps:

1. Multiply normally: $x^3 \times x^2 + x^3 \times 1 + x \times x^2 + x \times 1 + 1 \times x^2 + 1 \times 1 = x^5 + x^3 + x^3 + x + x^2 + 1$

2. Combine like terms (mod 2 $\rightarrow$ XOR): $x^5 + x^2 + x + 1$

3. Reduce mod $x^4 + x + 1$: Final result: $x^3 + x^2 + 1$

Result stays inside the field $\rightarrow$ **closure holds.**

Lecture

6

Galois Field $GF(2^8)$

- AES operates in the finite field $GF(2^8)$.
- Each byte (8 bits) is treated as a polynomial of degree ≤ 7 .
- Arithmetic is done mod an irreducible polynomial of degree 8.
 - During multiplication, results can exceed degree 7 $\rightarrow$ To bring them back to 8 bits, AES reduces modulo an irreducible polynomial.
 - AES uses: $x^8+x^4+x^3+x+1 \rightarrow$ This keeps all operations within 8-bit values (0–255).

$\rightarrow$ Example: 10010101 $\rightarrow x^7+x^4+x^2+1$

Example: Byte Multiplication in AES

To multiply two bytes (e.g., 0x57 and 0x83):

1.Convert each to its polynomial form.

Byte (Hexadecimal)	Binary	Polynomial
0x57	0101 0111	$(x^6 + x^4 + x^2 + x + 1)$
0x83	1000 0011	$(x^7 + x + 1)$

2.Multiply the polynomials (using XOR for addition), Result may exceed degree 7.

3.Reduce the result mod $x^8+x^4+x^3+x+1$

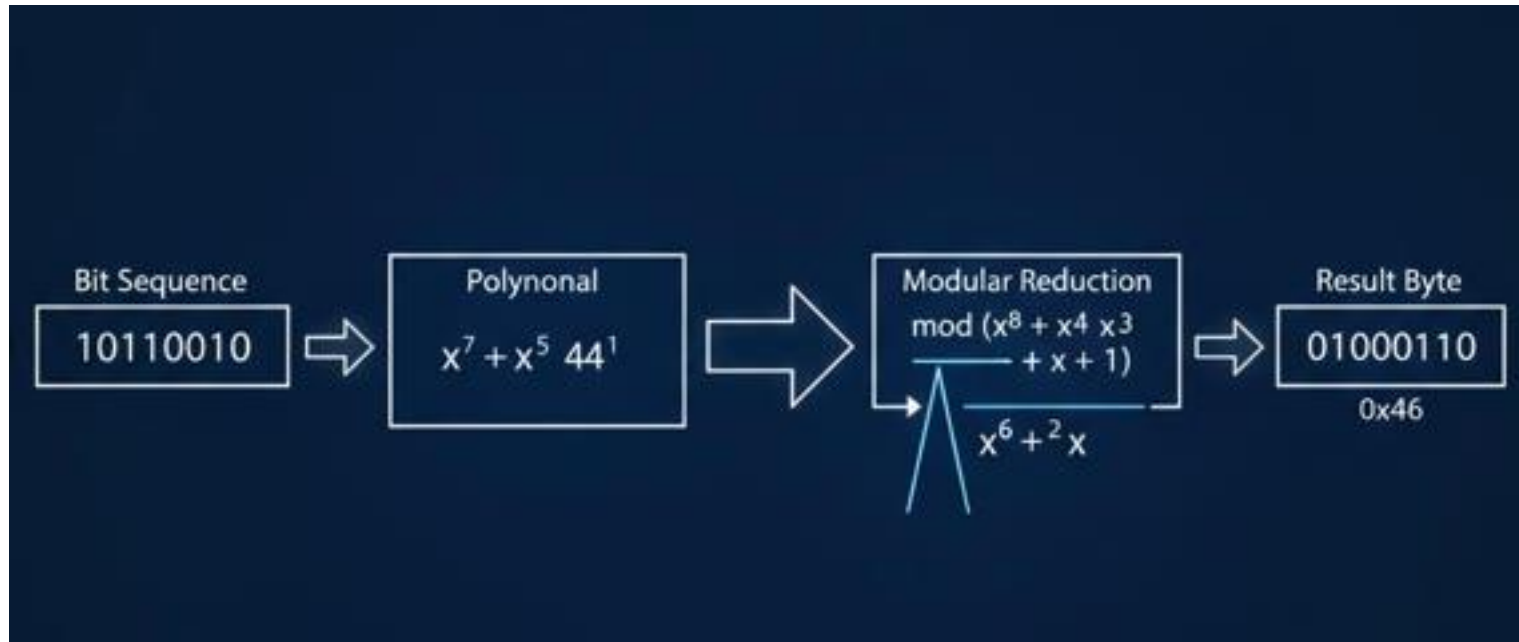
4.The final polynomial fits back into 8 bits.

This ensures closure — every result is still a valid byte in $GF(2^8)$.

Lecture

6

Byte Multiplication in AES (Flow Sequence)



Why $GF(2^8)$?

- Matches byte-sized (8-bit) computer architecture.
- Allows efficient hardware and software implementation.
- Ensures every nonzero byte has an inverse, which is crucial for AES's S-box (substitution step).
- In short: $GF(2^8)$ combines math elegance with practical performance.

Hands-on Practice

Try computing:

$(1011 \times 0110) \bmod (10011)$ over $GF(2)$.

Solution:

Step 1 – Convert to polynomials

- $1011 \rightarrow x^3+x+1$
- $0110 \rightarrow x^2+x$
- $10011 \rightarrow x^4+x+1$

Try computing:

$(1011 \times 0110) \bmod (10011)$ over $GF(2)$.

Solution:

Step 2 – Multiply

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

Step 3 – Reduce modulo $(x^4 + x + 1)$

Divide $x^5+x^4+x^3+x^2$ by x^4+x+1 using XOR subtraction

Result: $x^3+1 \rightarrow \text{Binary} = 1001$

Lecture

6

Why Cryptography Uses Finite Fields

- Predictable & Reversible:
Each operation gives a unique, repeatable result — essential for encryption/decryption symmetry.
- Prevents overflow/underflow:
Mod operation keeps results within a fixed range (like 8 bits in AES).
- Structured substitution & permutation:
Modular rules enable predictable transformations — used in AES S-box and MixColumns.

Security Insight

1. Modular arithmetic ensures predictable results

- Every operation (add, multiply, mod) always gives a defined and repeatable result within the field.

- This predictability is what makes encryption and decryption perfectly reversible — no data loss or randomness in computation.

2. Hard problems (like discrete log, factoring) rely on finite fields

- Certain math problems in finite fields are computationally hard — meaning they take an extremely long time to solve without the key.

- Cryptosystems like RSA, Diffie–Hellman, and Elliptic Curve Cryptography (ECC) base their security on these difficult problems.

Lecture
6

Summary Table: Field Properties in Cryptography

Property	Importance
Closure	Ensures all operations valid
Inverse	Enables decryption
Finite Size	Prevents infinite computation
Prime Modulus	Ensures field structure

Lecture

6

Recap

- Modular arithmetic = arithmetic “with wrap-around.”
- Finite fields = limited, well-defined sets of numbers.
- Fields provide reversible operations → essential for cryptography.
- $GF(p)$ and $GF(2^n)$ are the backbone of modern algorithms like AES and ECC.
- Cryptography = math + implementation.
- Finite field operations guarantee reliable, secure, repeatable encryption across digital systems.

References

- Stallings, Cryptography and Network Security, 7th Ed., Ch. 4
- Cyber Security Tools, Ch. 3
- Handbook of Applied Cryptography, Ch. 2
- NIST SP 800-38A (AES math appendix)