B.Sc. (Hons) Computer Applications Cyber Security Course number CABX)6003

Unit 2

Syllabus

1. **Symmetric Ciphers**
   - Classical Encryption Techniques
   - Block Ciphers and the Data Encryption Standard (DES)
   - Advanced Encryption Standard (AES)
   - Block Cipher Operation Modes
   - Basic Concepts in Number Theory and Finite Fields,
2. **Asymmetric Ciphers**
   - Introduction to Public-Key Cryptography
   - RSA Algorithm
   - Diffie-Hellman Key Exchange Protocol
   - ElGamal Cryptosystem
   - Elliptic Curve Cryptography (ECC)

Syllabus

3. Quantum Cryptography

- Introduction to Quantum Cryptography
- Quantum Key Distribution (QKD
- Quantum Physics and Photon Reception
- Cryptography with Photons
- Implementation of Quantum Cryptographic Systems

4. Cryptographic Hash Functions

- Introduction to Hash Functions
- Common Hash Algorithms
- Applications of Hash Functions

Syllabus

5. Message Authentication Codes (MACs)

- Concept of Message Authentication
- Types of MACs
- Working of HMAC and CMAC

6. Digital Signatures

- Introduction to Digital Signatures
- Working of Digital Signature Algorithms
- Common Digital Signature Algorithms
- Applications of Digital Signatures

Symmetric Ciphers

Symmetric encryption is a cryptographic method where the **same key** is used for both the **encryption** of plaintext and the **decryption** of ciphertext. This key must be kept secret between the sender and the receiver to maintain the confidentiality of the data.

- **Key Features:**
  - **Same Key for Encryption and Decryption:** The primary characteristic is the use of a single secret key.
  - **Fast and Efficient:** Ideal for encrypting large amounts of data due to lower computational overhead compared to asymmetric encryption.
  - **Key Distribution Challenge:** Securely sharing the secret key is a significant challenge.
- **Real-Life Examples:**
  - **Wi-Fi Security:** WPA2(Wi-Fi Protected Access 2) encryption in routers.
  - **Banking Systems:** Encrypting transaction data.
  - **File Encryption:** Tools like BitLocker use symmetric encryption to secure files.

Symmetric Ciphers

Advantages:

1. **Speed:** Faster encryption and decryption processes.
2. **Efficiency:** Requires less computational power.
3. **Suitable for Large Data:** Effective for bulk data encryption.

Disadvantages:

1. **Key Distribution Problem:** The key must be shared securely, posing a risk if intercepted.
2. **Scalability Issues:** Managing keys becomes complex in large networks.
3. **Lack of Non-Repudiation:** Since both parties share the same key, it’s difficult to prove who encrypted the message.

Classical Encryption Techniques

Classical encryption techniques are the foundational methods that paved the way for modern cryptography. They mainly fall into two categories:

A. Substitution Ciphers: Substitution ciphers replace elements of the plaintext with corresponding elements of ciphertext.

1. **Caesar Cipher:** A shift cipher where each letter in the plaintext is shifted a fixed number of positions down the alphabet. **Example:** With a shift of 3:
    1. Plaintext: **HELLO**
    2. Ciphertext: **KHOOR**
2. **Vigenère Cipher:** Uses a keyword to determine the shift for each letter, making it a polyalphabetic cipher. **Example:**
    1. Plaintext: **ATTACKATDAWN**
    2. Key: **LEMON**
    3. Ciphertext: **LXFOPVEFRNHR**

        Encryption

        Ei = (Pi + Ki) mod 26

        Decryption

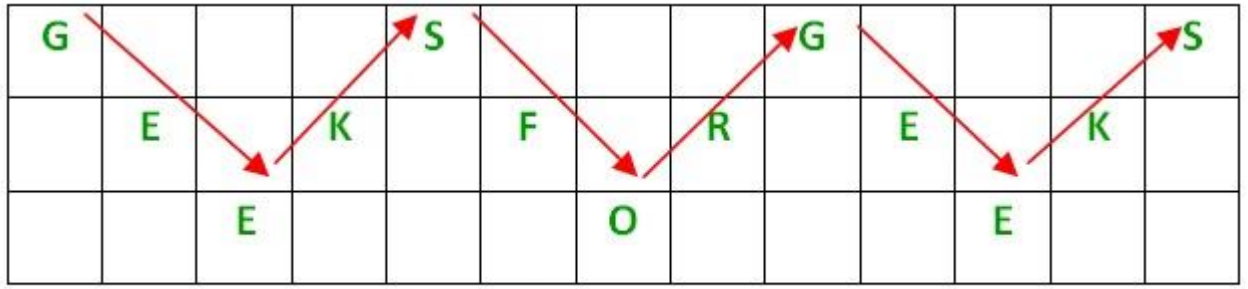
        Di = (Ei - Ki) mod 26

Classical Encryption Techniques

B. Transposition Ciphers: Instead of replacing letters, transposition ciphers rearrange the letters in the plaintext.

Rail Fence Cipher: The plaintext is written diagonally over multiple "rails" and then read row by row.

For example, if the message is “GeeksforGeeks” and the number of rails = 3 then cipher is prepared as:



cipher-text = “GsGsekfrek eoe”

Classical Encryption Techniques

Columnar Transposition Cipher:

Encryption

Given text = Geeks for Geeks

Keyword = HACK **Length of Keyword** = 4 (no of rows) **Order of Alphabets in HACK** = 3124

H	A	C	K
3	1	2	4
G	e	e	k
s	_	f	o
r	_	G	e
e	k	s	_

Print Characters of column 1,2,3,4

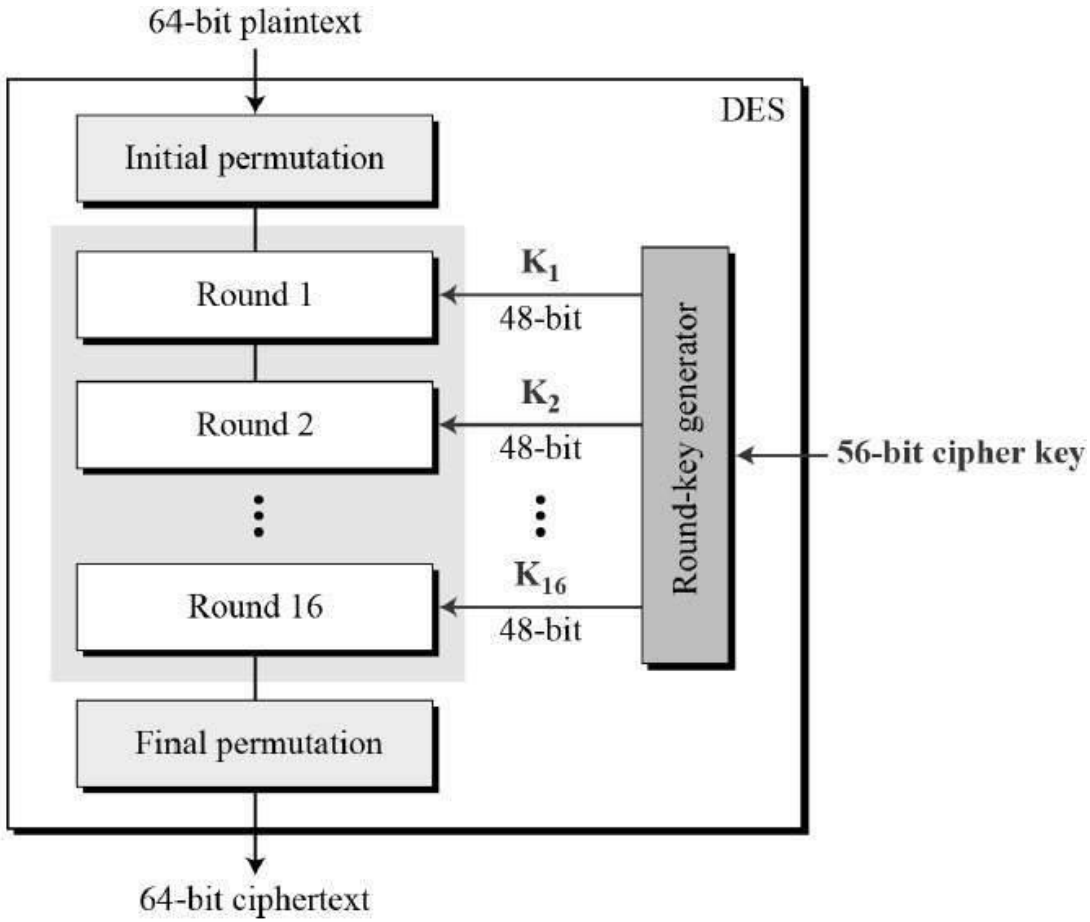
Encrypted Text = e kefGsGsrekoe_

Block Ciphers and the Data Encryption Standard (DES)

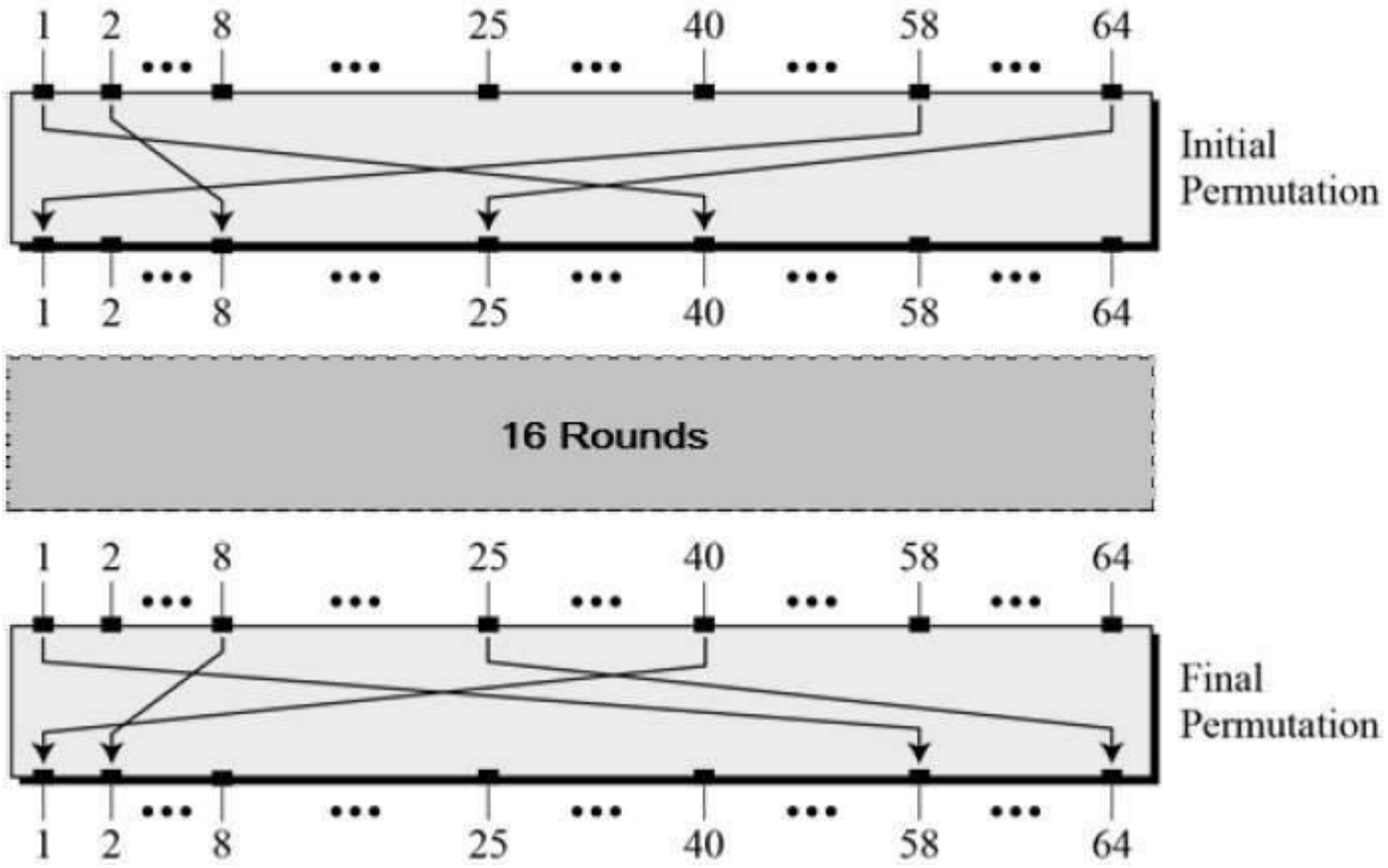
3. Block Ciphers and DES: Block ciphers encrypt data in fixed-size blocks (e.g., 64-bit or 128-bit blocks) using a symmetric key.

- **Data Encryption Standard (DES):**
  - Developed in the 1970s by IBM and adopted as a standard by the U.S. government.
  - Encrypts data in 64-bit blocks using a 56-bit key.
- **DES Structure:**
  - **Feistel Network:** The core structure, where data is split into two halves and processed through multiple rounds of permutation and substitution.
  - **16 Rounds of Processing:** Each round applies key mixing, substitution, and permutation.
- **Steps of DES:**
  1. **Initial Permutation (IP):** Rearranges the bits of the plaintext.
  2. **16 Rounds of Encryption:** Each round uses a unique sub-key derived from the main key.
  3. **Final Permutation (IP-1):** The inverse of the initial permutation.
- **Weaknesses of DES:**
  - **Key Length Vulnerability:** A 56-bit key is too short by today's standards, making DES susceptible to brute-force attacks.
  - **Solution: Triple DES (3DES)** applies DES three times with different keys for enhanced security.

Data Encryption Standard (DES)



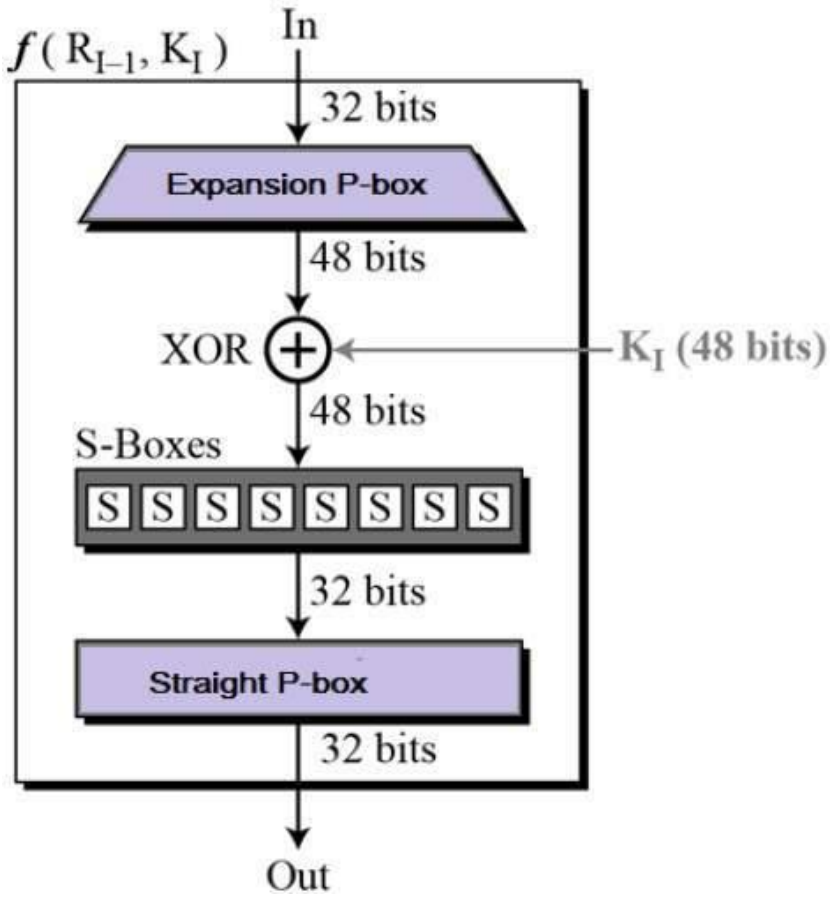
Data Encryption Standard (DES)



Data Encryption Standard (DES)

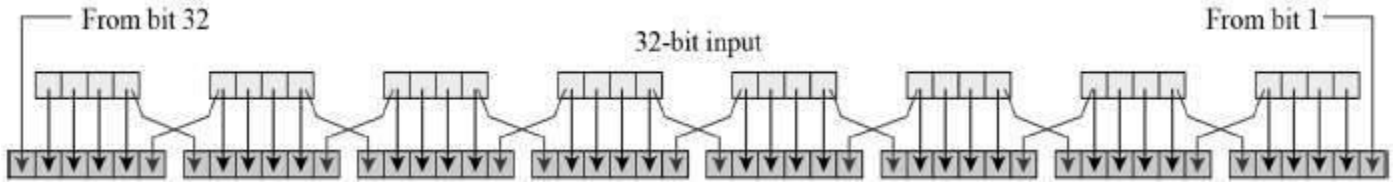
Round Function:

The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.



Data Encryption Standard (DES)

- **Expansion Permutation Box** – Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration –

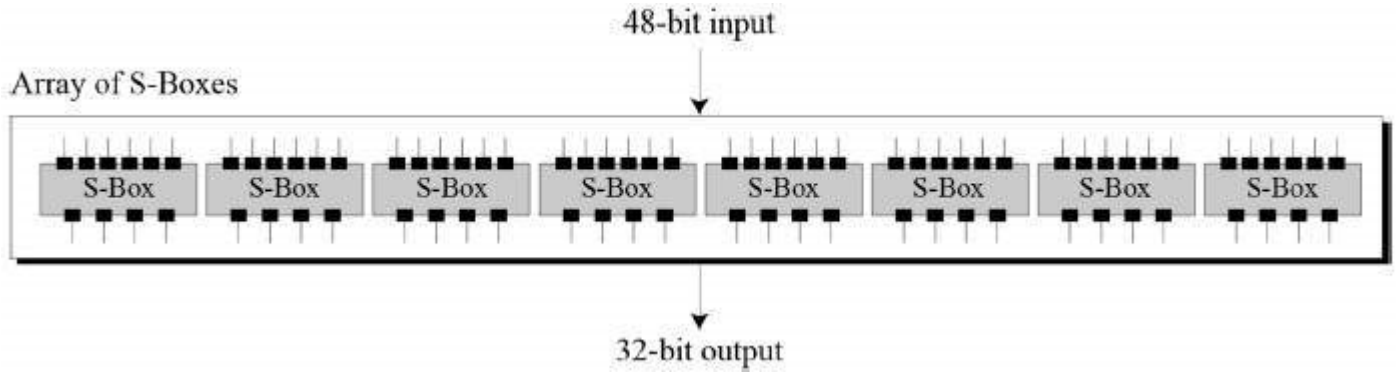


Data Encryption Standard (DES)

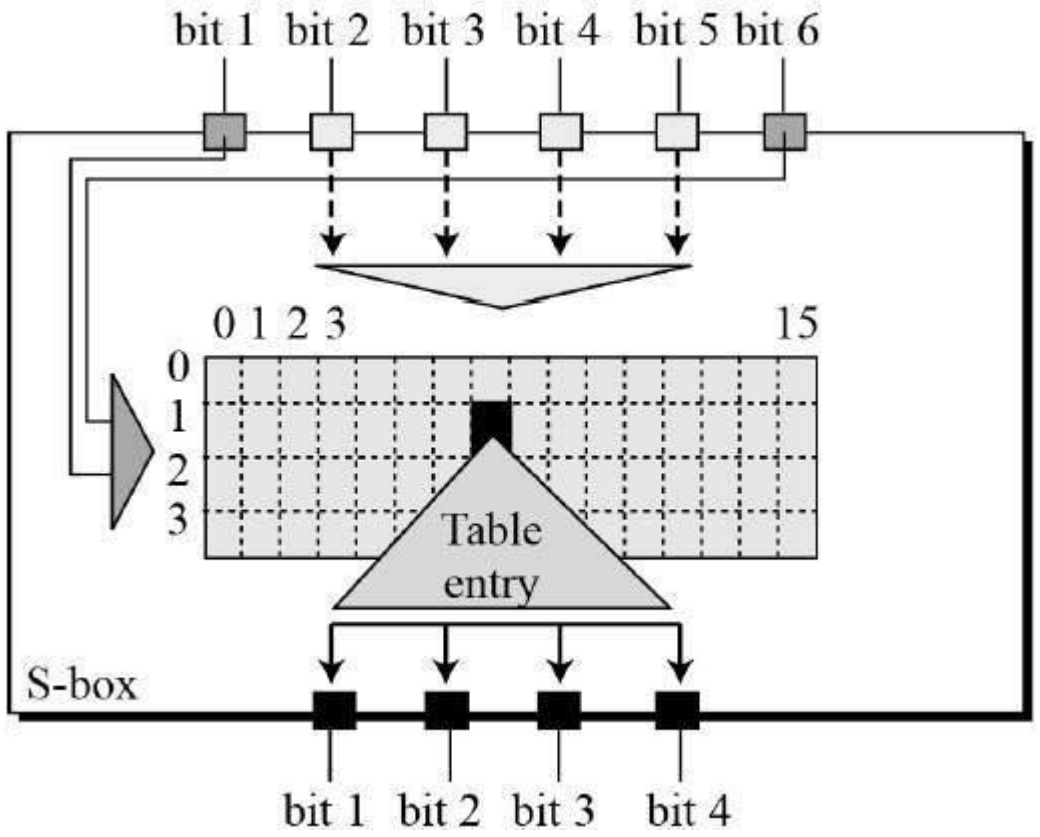
XOR (Whitener). – After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

Data Encryption Standard (DES)

Substitution Boxes. – The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration –



Data Encryption Standard (DES)



Data Encryption Standard (DES)

- **Straight Permutation** – The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

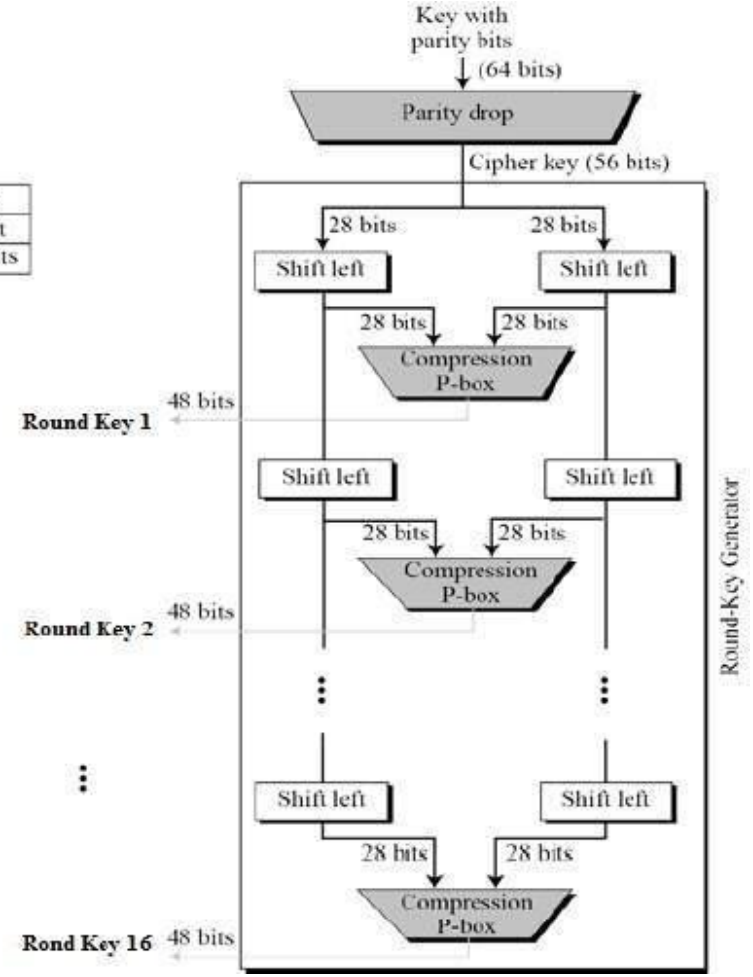
Data Encryption Standard (DES)

Key Generation:

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration –

Shifting

Rounds	Shift
1, 2, 9, 16	one bit
Others	two bits



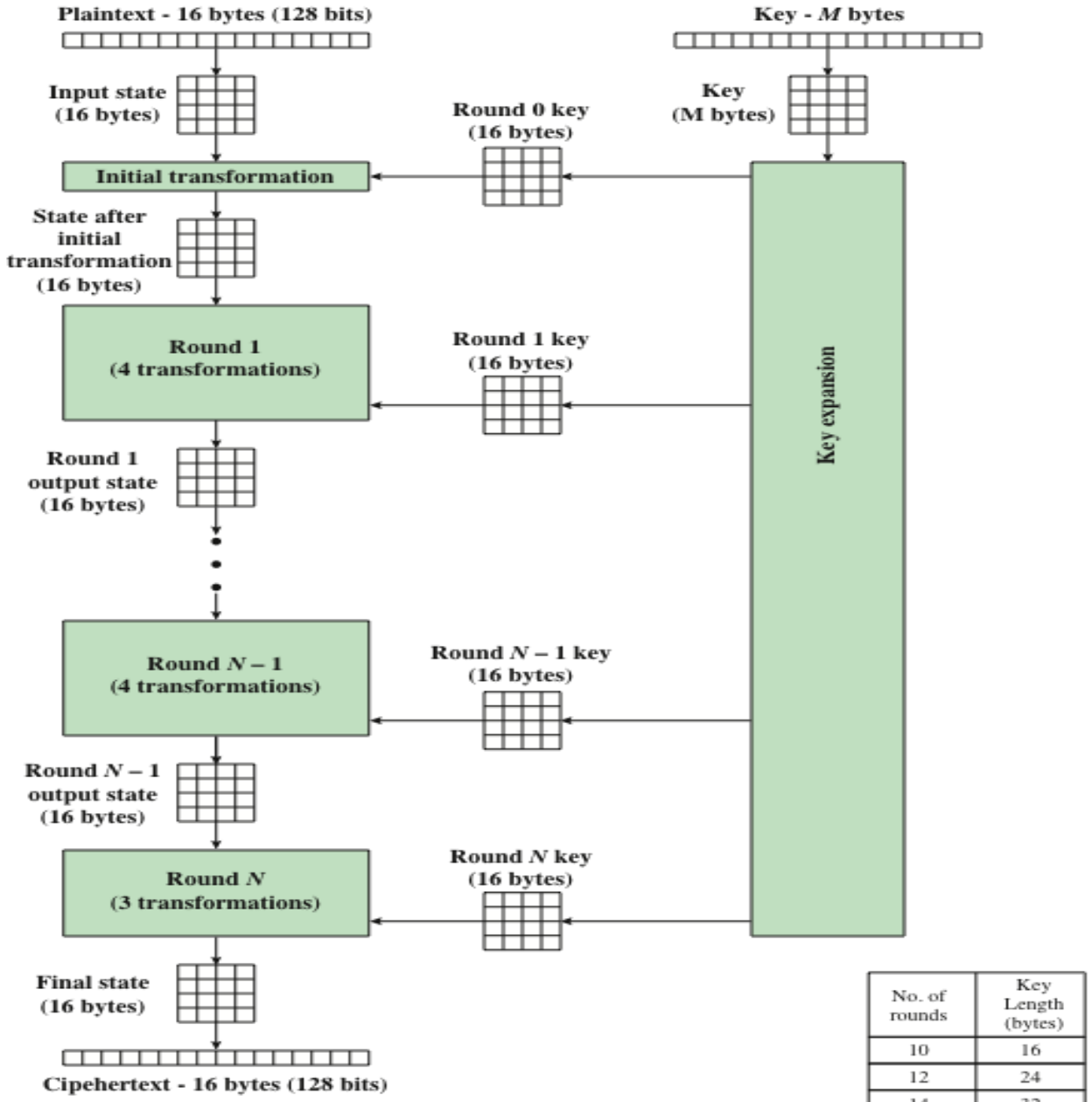
Advanced Encryption Standard (AES)

Advanced Encryption Standard
(AES)

Ref: Cryptography and Network Security by William Stallings

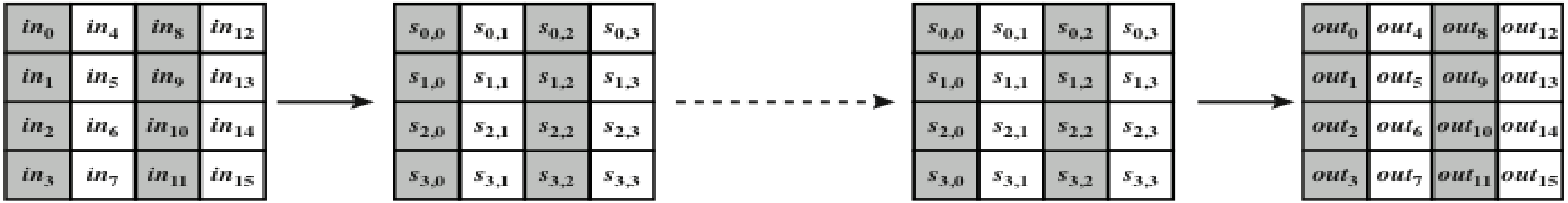
Finite Field Arithmetic

- AES is developed by two Belgium cryptographers, Vinicent Rijmen and Joan Daemen.
- In the Advanced Encryption Standard (AES) all operations are performed on 8-bit bytes
- The arithmetic operations of addition, multiplication, and division are performed over the finite field $GF(2^8)$
- A field is a set in which we can do addition, subtraction, multiplication, and division without leaving the set
- Division is defined with the following rule:
  - $a/b = a(b^{-1})$
- An example of a finite field (one with a finite number of elements) is the set Z_p consisting of all the integers $\{0, 1, \ldots, p - 1\}$, where p is a prime number and in which arithmetic is carried out modulo p

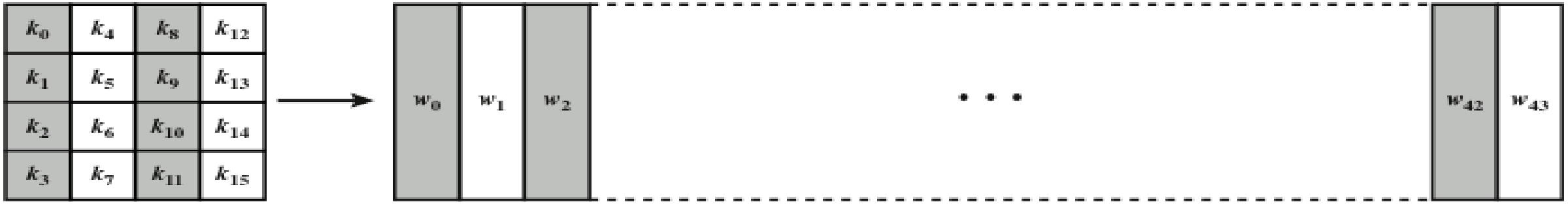


No. of rounds	Key Length (bytes)
10	16
12	24
14	32

Figure 6.1 AES Encryption Process



(a) Input, state array, and output



(b) Key and expanded key

Figure 6.2 AES Data Structures

Table 6.1 AES Parameters

Key Size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext Block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of Rounds	10	12	14
Round Key Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded Key Size (words/bytes)	44/176	52/208	60/240

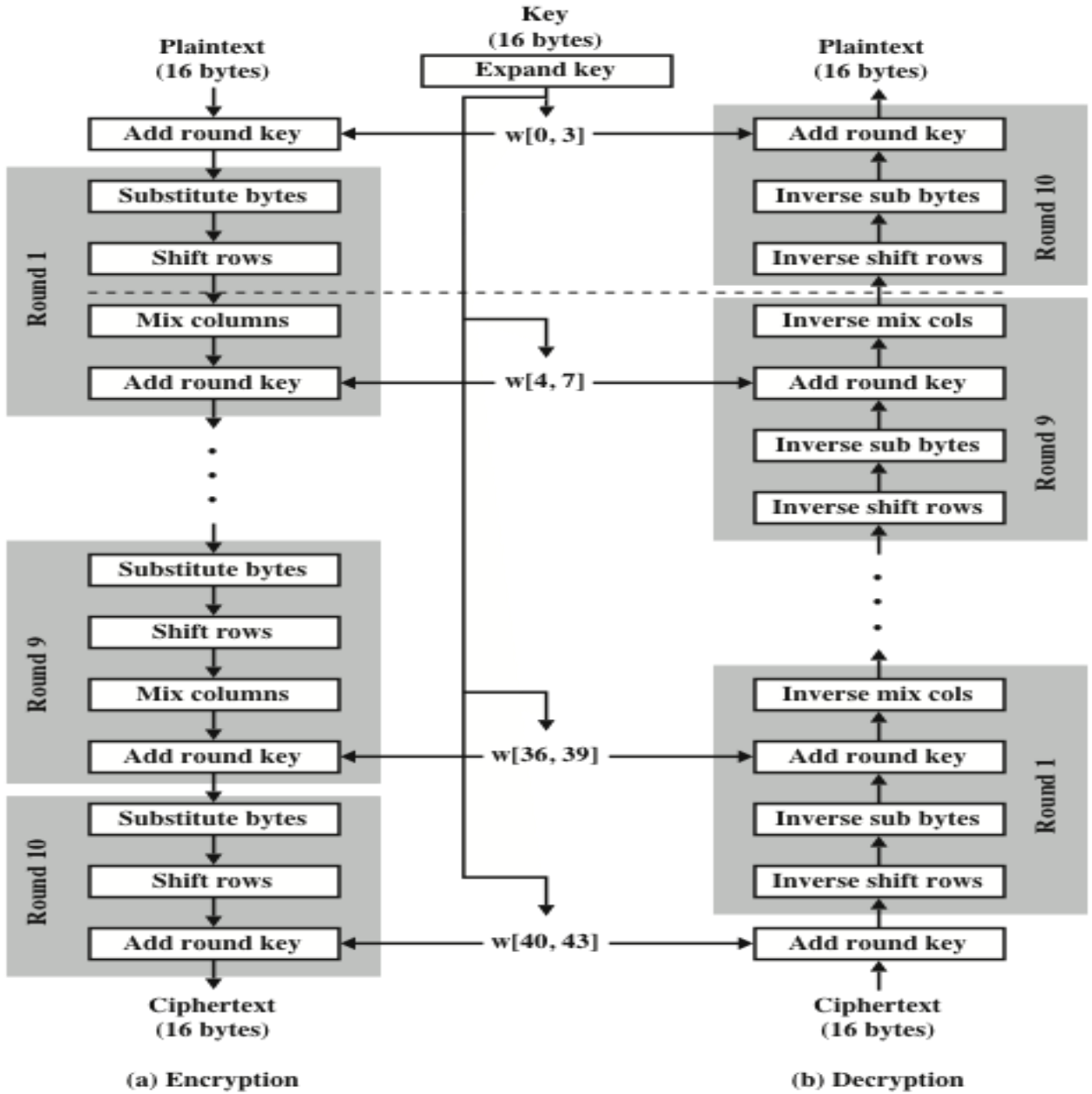


Figure 6.3 AES Encryption and Decryption

Detailed Structure

- Processes the entire data block as a single matrix during each round using substitutions and permutation
- The key that is provided as input is expanded into an array of forty-four 32-bit words, *w[i]*

Four different stages are used:

- Substitute bytes – uses an S-box to perform a byte-by-byte substitution of the block
- ShiftRows – a simple permutation
- MixColumns – a substitution that makes use of arithmetic over GF(2^8)
- AddRoundKey – a simple bitwise XOR of the current block with a portion of the expanded key

- The cipher begins and ends with an AddRoundKey stage
- Can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on
- The decryption algorithm makes use of the expanded key in reverse order, however the decryption algorithm is not identical to the encryption algorithm
- Final round of both encryption and decryption consists of only three stages

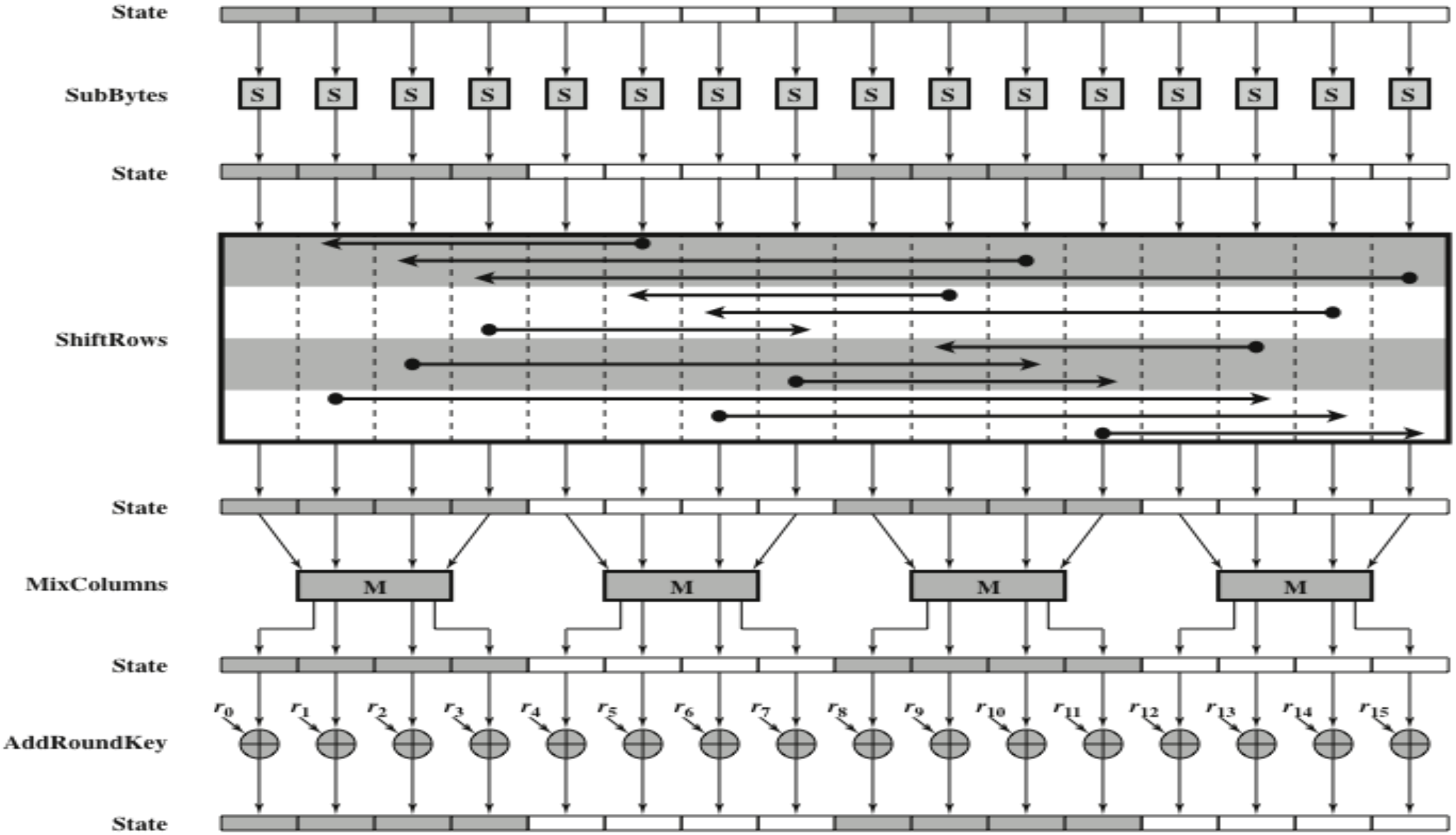
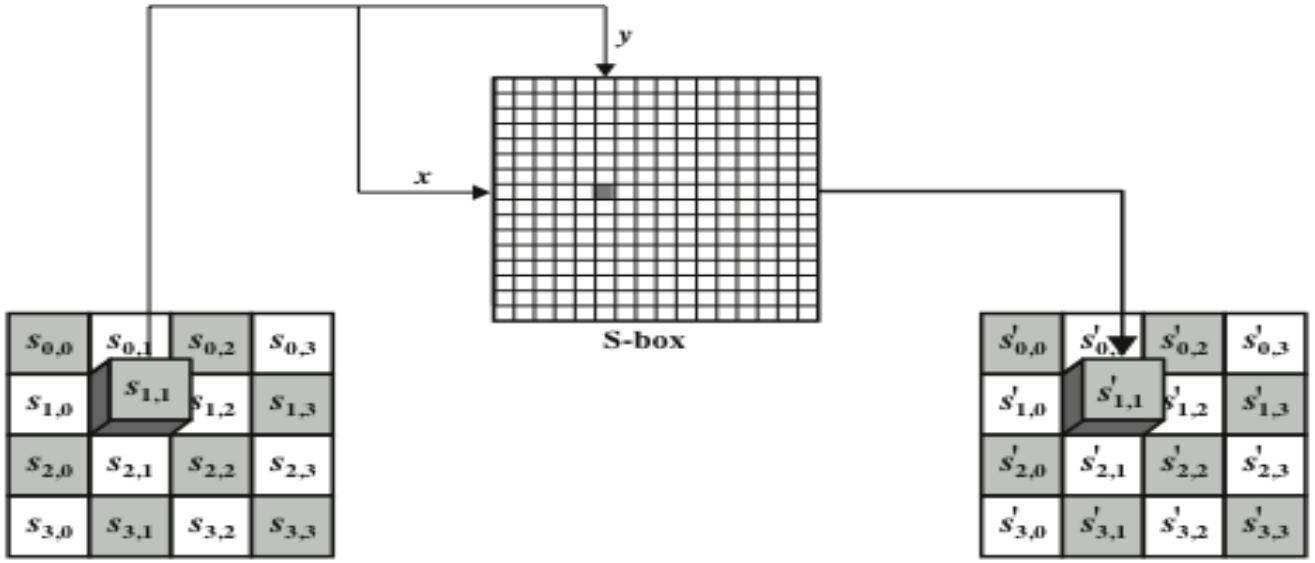
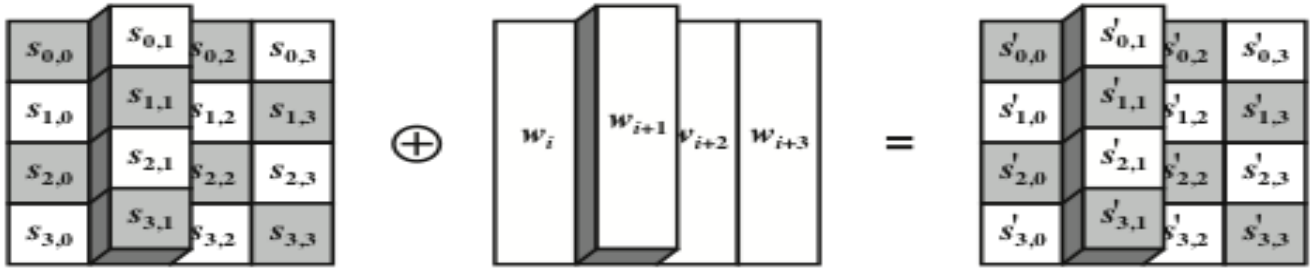


Figure 6.4 AES Encryption Round



(a) Substitute byte transformation



(b) Add round key Transformation

Figure 6.5 AES Byte-Level Operations

Table 6.2

		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
	1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
	2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
	3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
	4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
	5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
	6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
	7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
	8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
	9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
	A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
	B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
	C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
	D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
	E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
	F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

(a) S-box

(Table can be found on page 163 in textbook)

Table 6.2

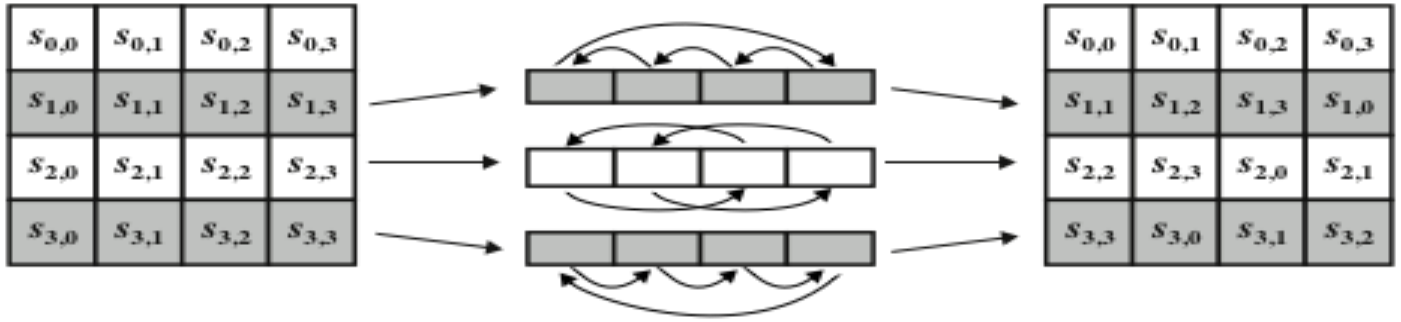
		y															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
x	0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
	1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
	2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
	3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
	4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
	5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
	6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
	7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
	8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
	9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
	A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
	B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
	C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
	D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
	E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
	F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

(b) Inverse S-box

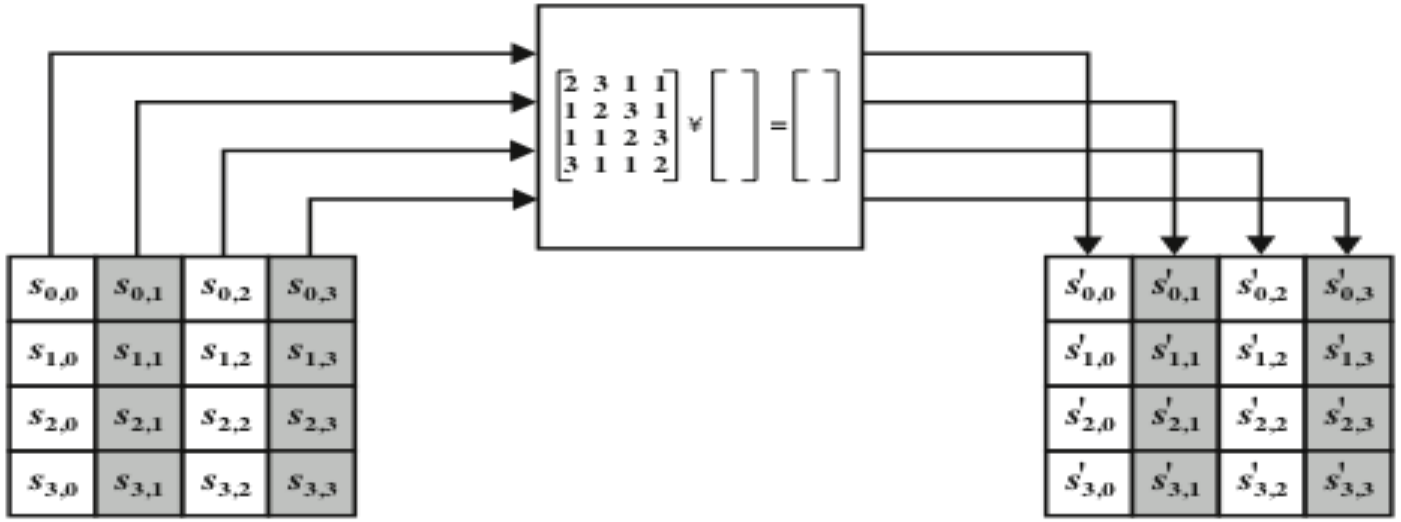
(Table can be found on page 163 in textbook)

S-Box Rationale

- The S-box is designed to be resistant to known cryptanalytic attacks
- The Rijndael developers sought a design that has a low correlation between input bits and output bits and the property that the output is not a linear mathematical function of the input
- The nonlinearity is due to the use of the multiplicative inverse



(a) Shift row transformation



(b) Mix column transformation

Figure 6.7 AES Row and Column Operations

Shift Row Rationale

- The State, as well as the cipher input and output, is treated as an array of four 4-byte columns
- On encryption, the first 4 bytes of the plaintext are copied to the first column of State, and so on
- A row shift moves an individual byte from one column to another.
- Transformation ensures that the 4 bytes of one column are spread out to four different columns

Mix Columns Rationale

- Coefficients of a matrix based on a linear code ensures a good mixing among the bytes of each column
- The mix column transformation combined with the shift row transformation ensures that after a few rounds all output bits depend on all input bits

AddRoundKey Transformation

- The 128 bits of State are bitwise XORed with the 128 bits of the round key
- Operation is viewed as a columnwise operation between the 4 bytes of a State column and one word of the round key
    - Can also be viewed as a byte-level operation

Rationale:

Is as simple as possible and affects every bit of State

The complexity of the round key expansion plus the complexity of the other stages of AES ensure security

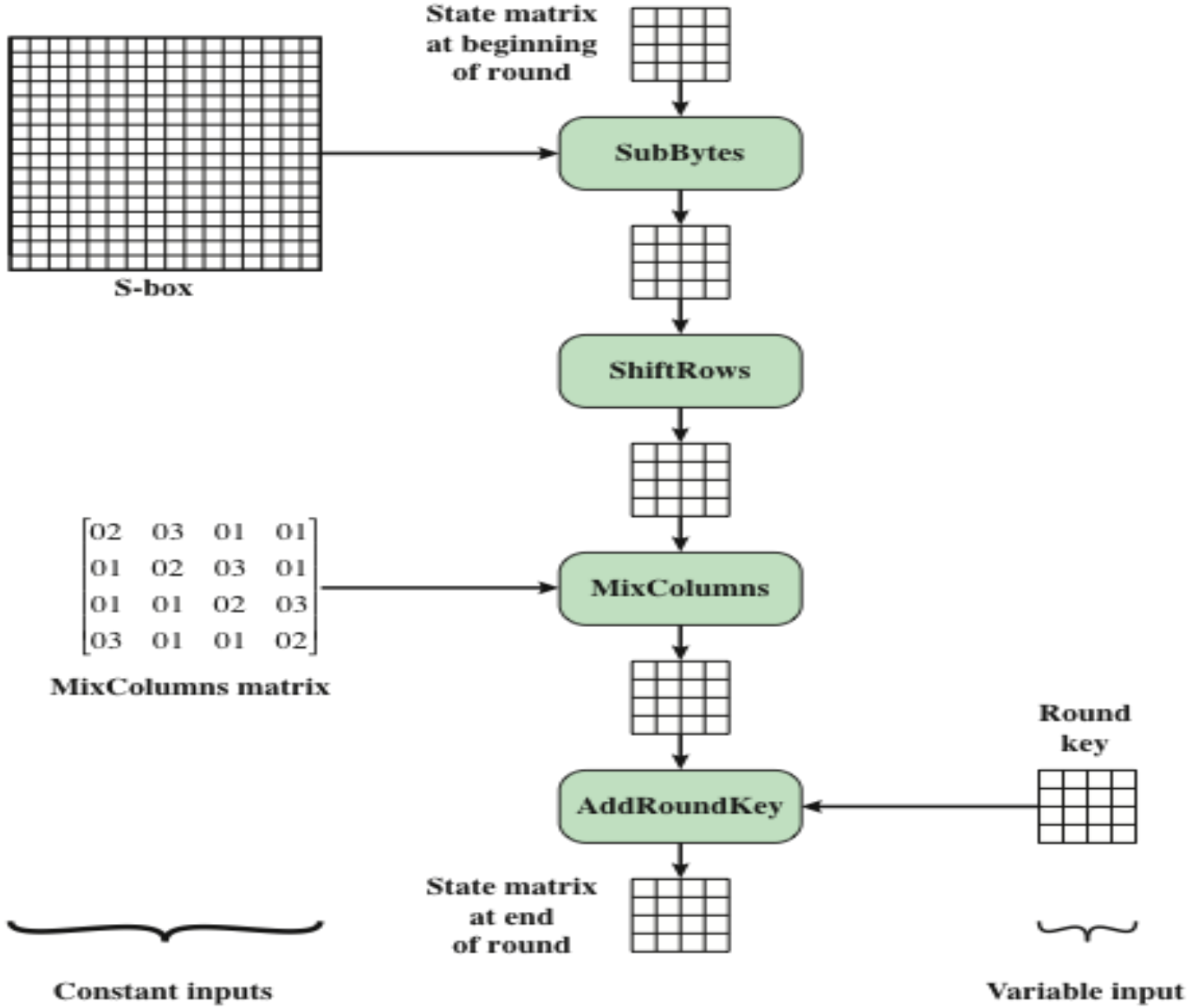
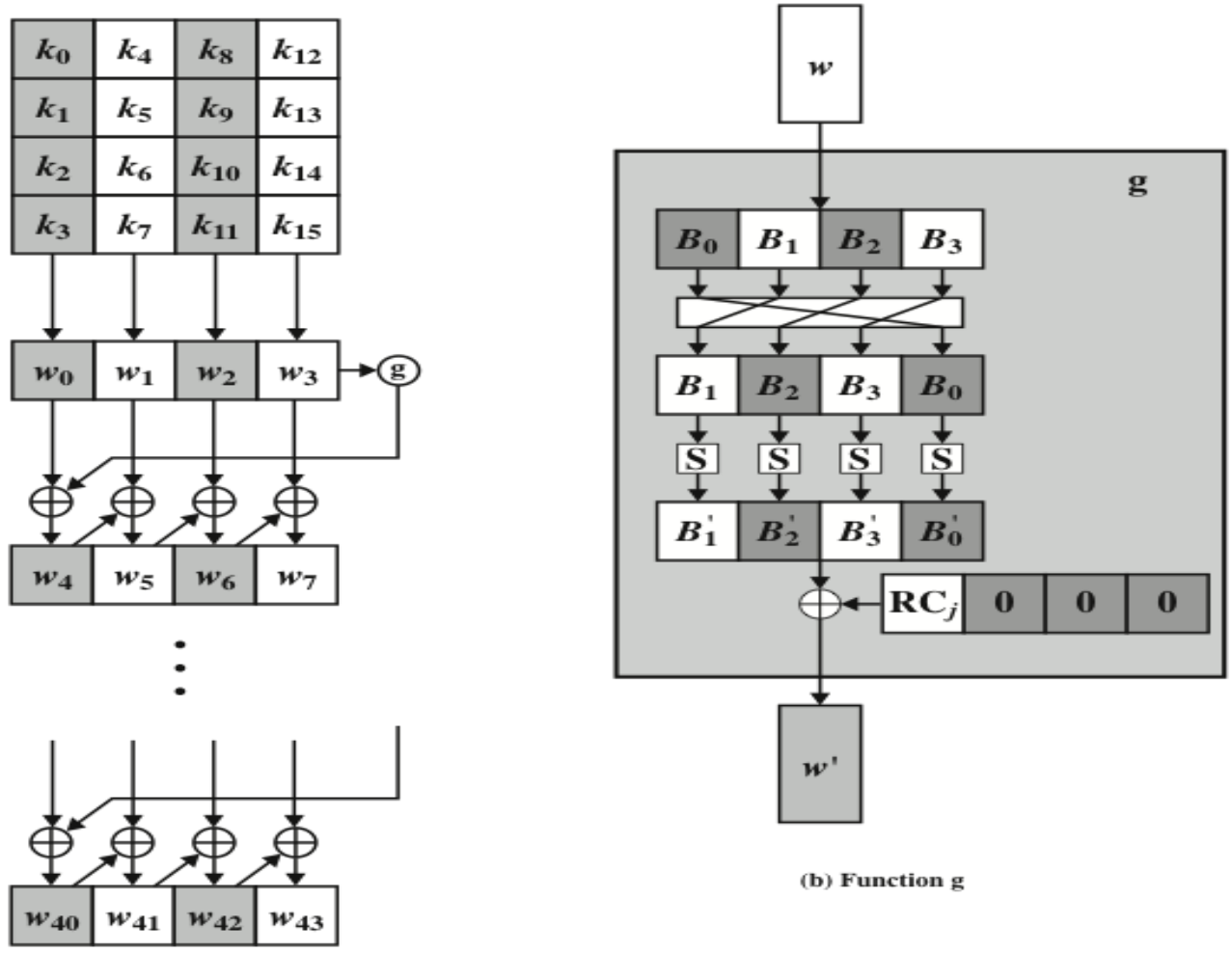


Figure 6.8 Inputs for Single AES Round

AES Key Expansion

- Takes as input a four-word (16 byte) key and produces a linear array of 44 words (176) bytes
  - This is sufficient to provide a four-word round key for the initial AddRoundKey stage and each of the 10 rounds of the cipher
- Key is copied into the first four words of the expanded key
  - The remainder of the expanded key is filled in four words at a time
- Each added word *w*[i] depends on the immediately preceding word, *w[i – 1]*, and the word four positions back, w[i – 4]
  - In three out of four cases a simple XOR is used
  - For a word whose position in the *w* array is a multiple of 4, a more complex function is used



(a) Overall algorithm

(b) Function g

Figure 6.9 AES Key Expansion

Basic Concepts in Number Theory and Finite Fields

- **Number Theory in Cryptography:**
  1. **Prime Numbers:** Fundamental in key generation for asymmetric algorithms like RSA.
  2. **Modular Arithmetic:** Operations performed modulo a number,
  3. **Greatest Common Divisor (GCD):** Important for finding relatively prime numbers.
- **Finite Fields (Galois Fields):** A set of finite numbers where you can perform addition, subtraction, multiplication, and division (except by zero). Used in AES for mixing columns and key scheduling.
- **Important Theorems:**
  - **Fermat's Little Theorem:** Basis for RSA encryption.
  - **Euler's Theorem:** Used in public key cryptography algorithms.

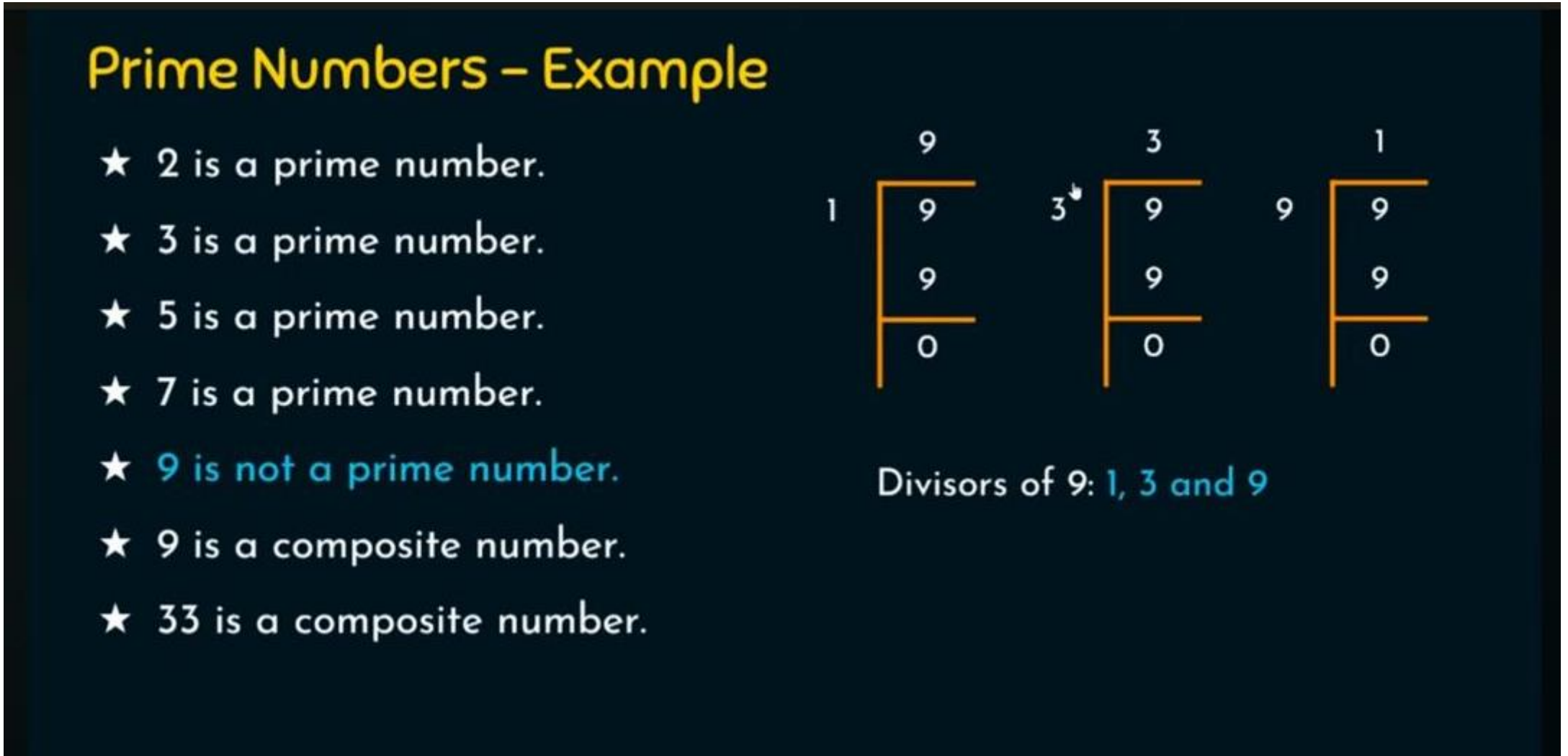
Basic Concepts in Number Theory and Finite Fields

Prime Numbers

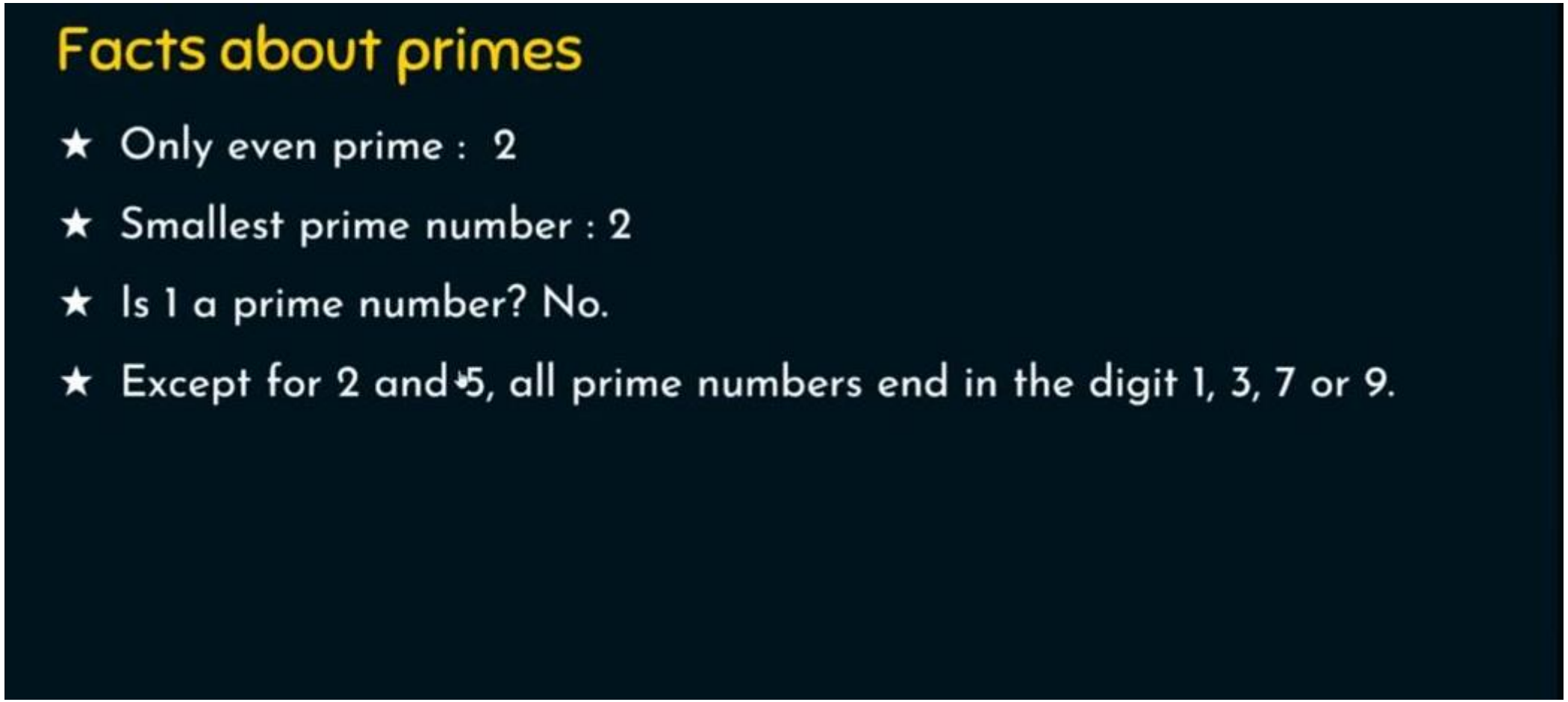
- ★ Prime Numbers: Has exactly two divisors.
- ★ If 'N' is prime, then the divisors are 1 and N.
- ★ All numbers have prime factors.

Numbers	10	11	100	37	308	14688
Prime Factorization	$2^1 \times 5^1$	$1^1 \times 11^1$	$2^2 \times 5^2$	$1^1 \times 37^1$	$2^2 \times 7^1 \times 11^1$	$2^5 \times 3^3 \times 17^1$
Prime Numbers	2, 5	1, 11	2, 5	1, 37	2, 7, 11	2, 3, 17

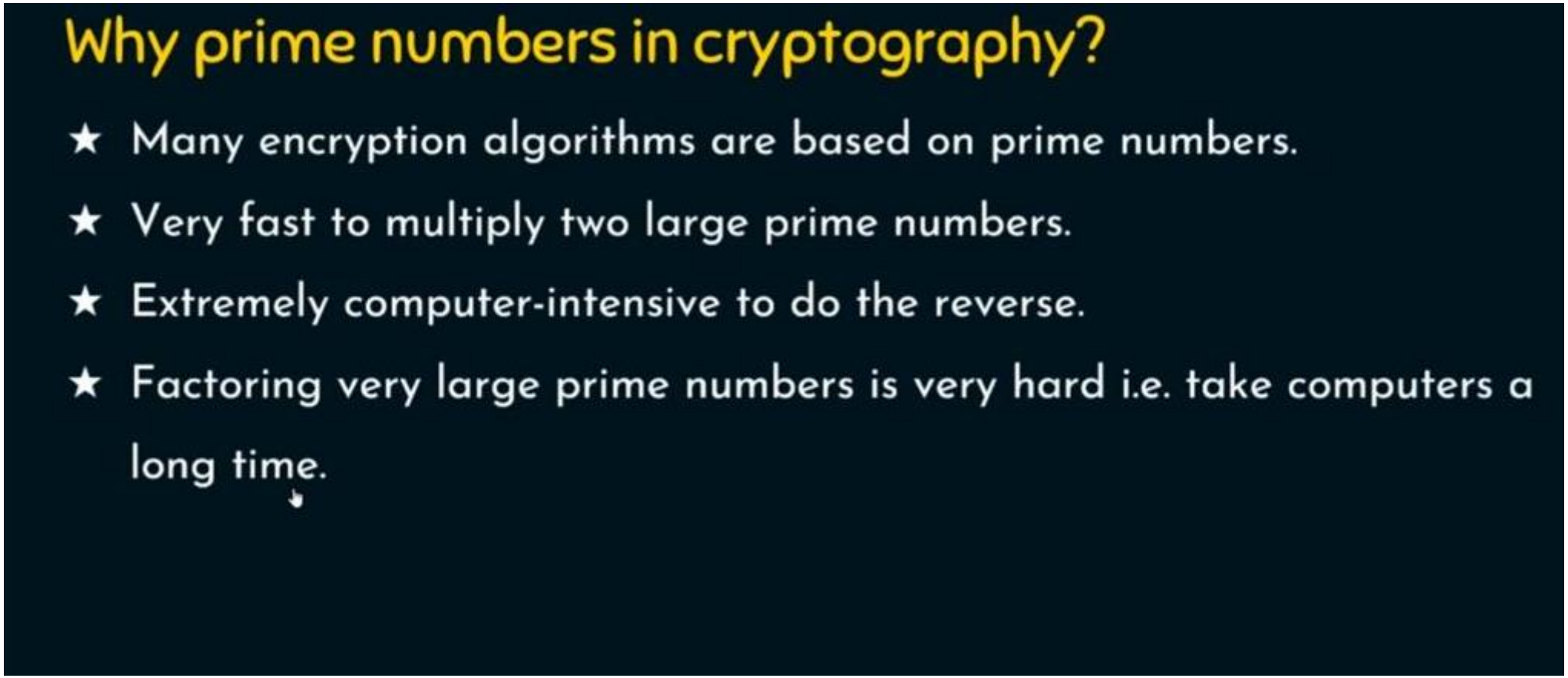
Basic Concepts in Number Theory and Finite Fields



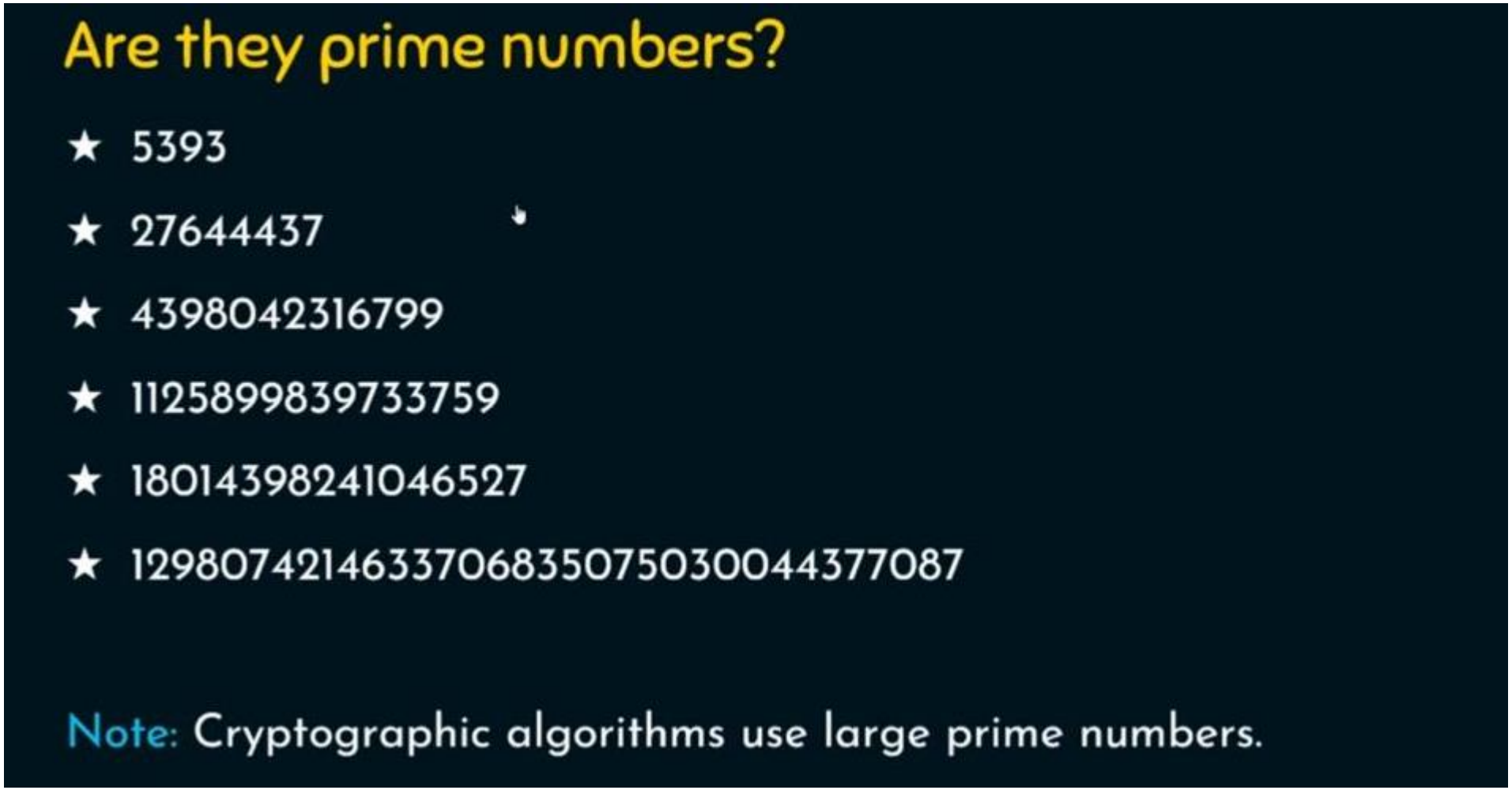
Basic Concepts in Number Theory and Finite Fields



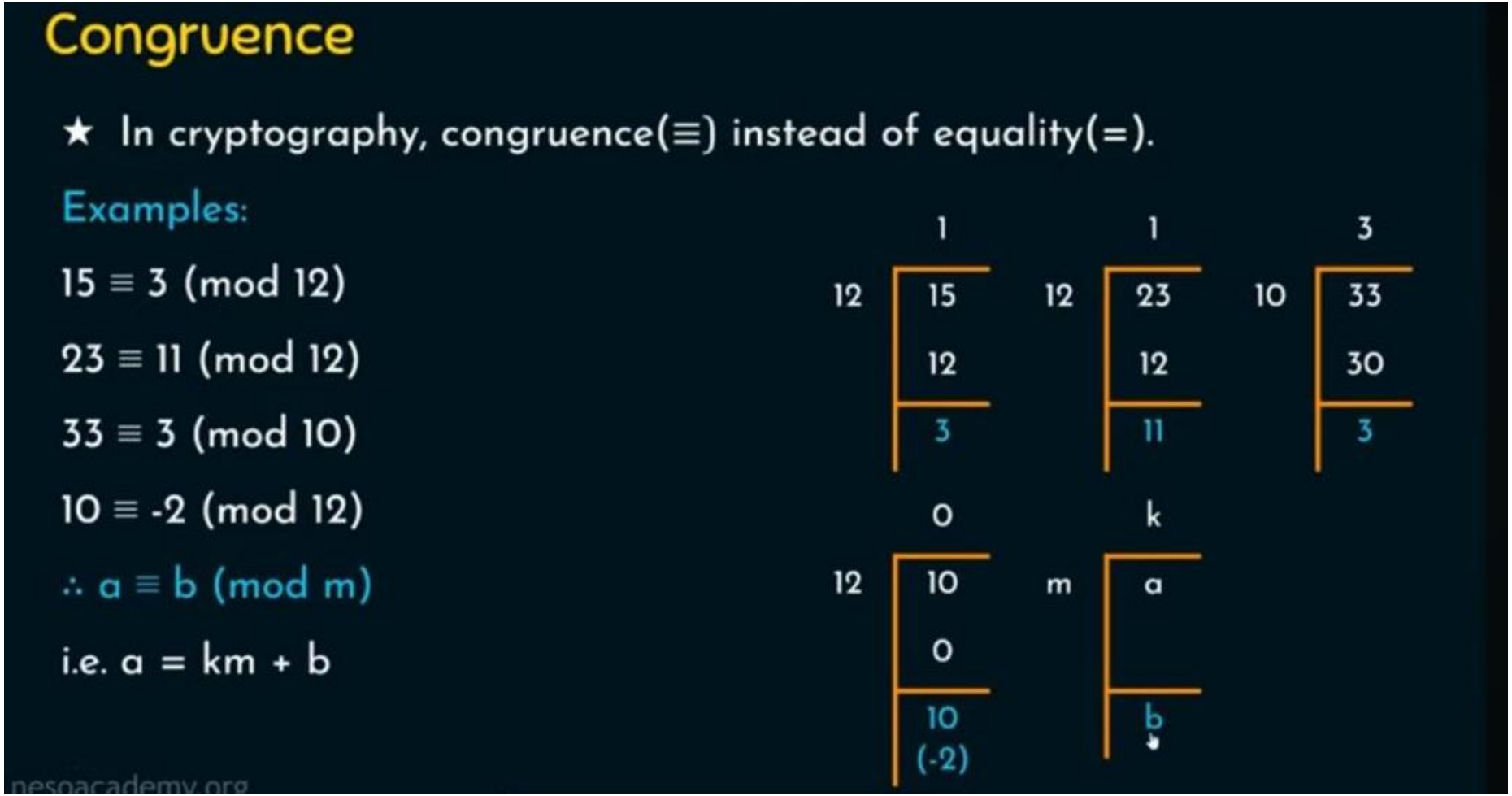
Basic Concepts in Number Theory and Finite Fields



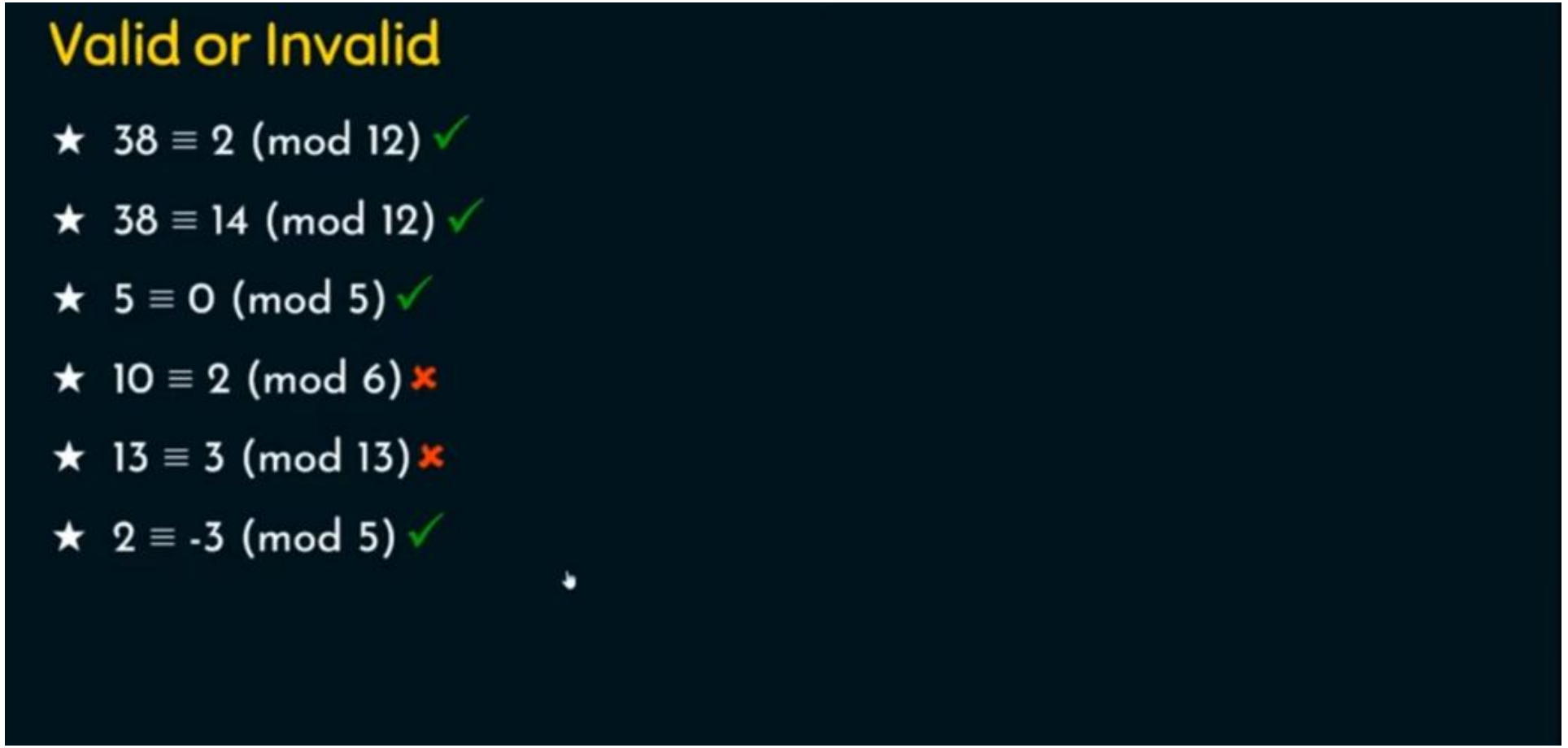
Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

Properties of Modular Arithmetic

1. $[(a \bmod n) + (b \bmod n)] \bmod n = (a + b) \bmod n$
2. $[(a \bmod n) - (b \bmod n)] \bmod n = (a - b) \bmod n$
3. $[(a \bmod n) \times (b \bmod n)] \bmod n = (a \times b) \bmod n$

Example:

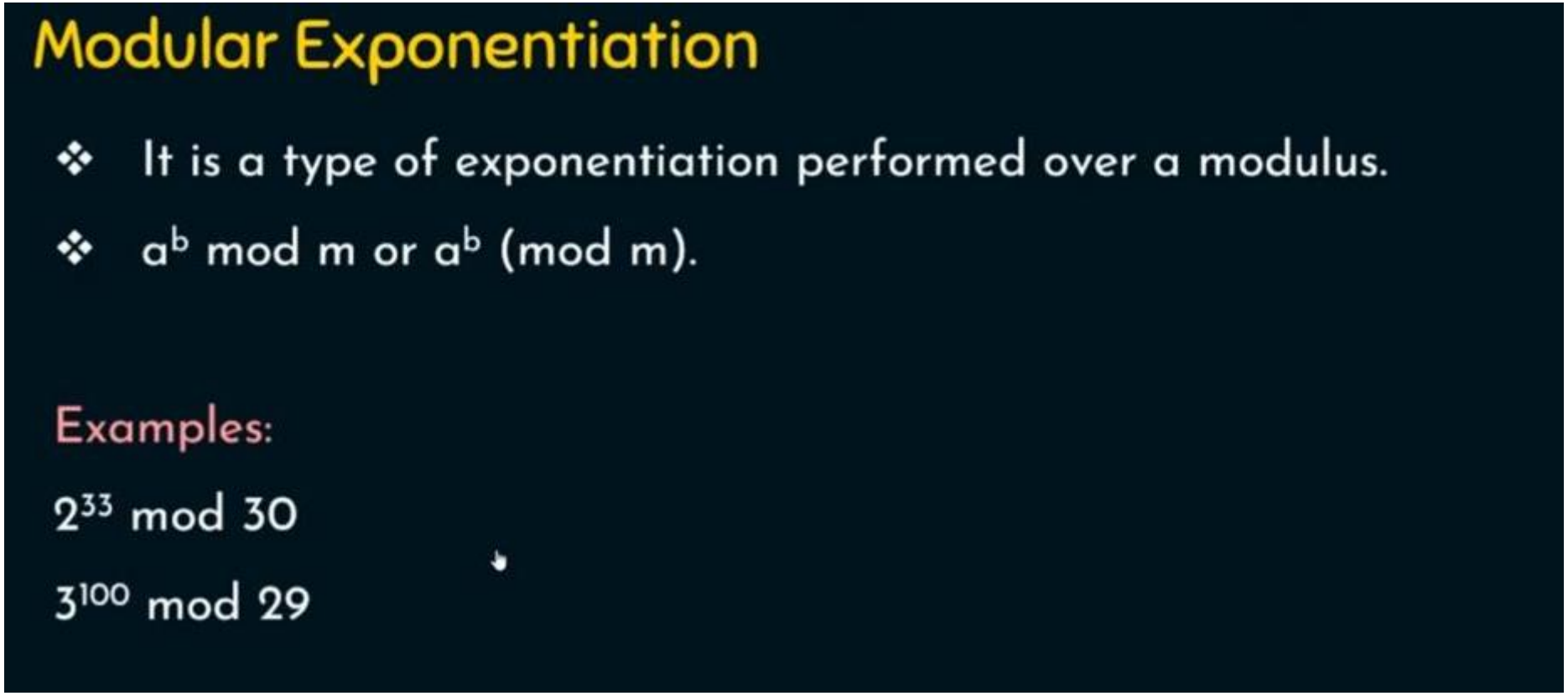
$$[(15 \bmod 8) \times (11 \bmod 8)] \bmod 8 = (15 \times 11) \bmod 8$$
$$= 165 \bmod 8$$
$$= 5$$

Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

Properties of Modular Arithmetic

Property	Expression
Commutative Laws	$(a + b) \bmod n = (b + a) \bmod n$ $(a \times b) \bmod n = (b \times a) \bmod n$
Associative Laws	$[(a + b) + c] \bmod n = [a + (b + c)] \bmod n$ $[(a \times b) \times c] \bmod n = [a \times (b \times c)] \bmod n$
Distributive Laws	$[a \times (b + c)] \bmod n = [(a \times b) + (a \times c)] \bmod n$
Identities	$(0 + a) \bmod n = a \bmod n$ $(1 \times a) \bmod n = a \bmod n$
Additive Inverse	For each $a \in Z_n$, there exists a '-a' such that $a + (-a) \equiv 0 \bmod n$

Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

Example 1

Solve $23^3 \bmod 30$.

$23^3 \bmod 30 = -7^3 \bmod 30$ || 23 mod 30 can be 23 or -7.

$= -7^3 \bmod 30$

$= -7^2 \times -7 \bmod 30$

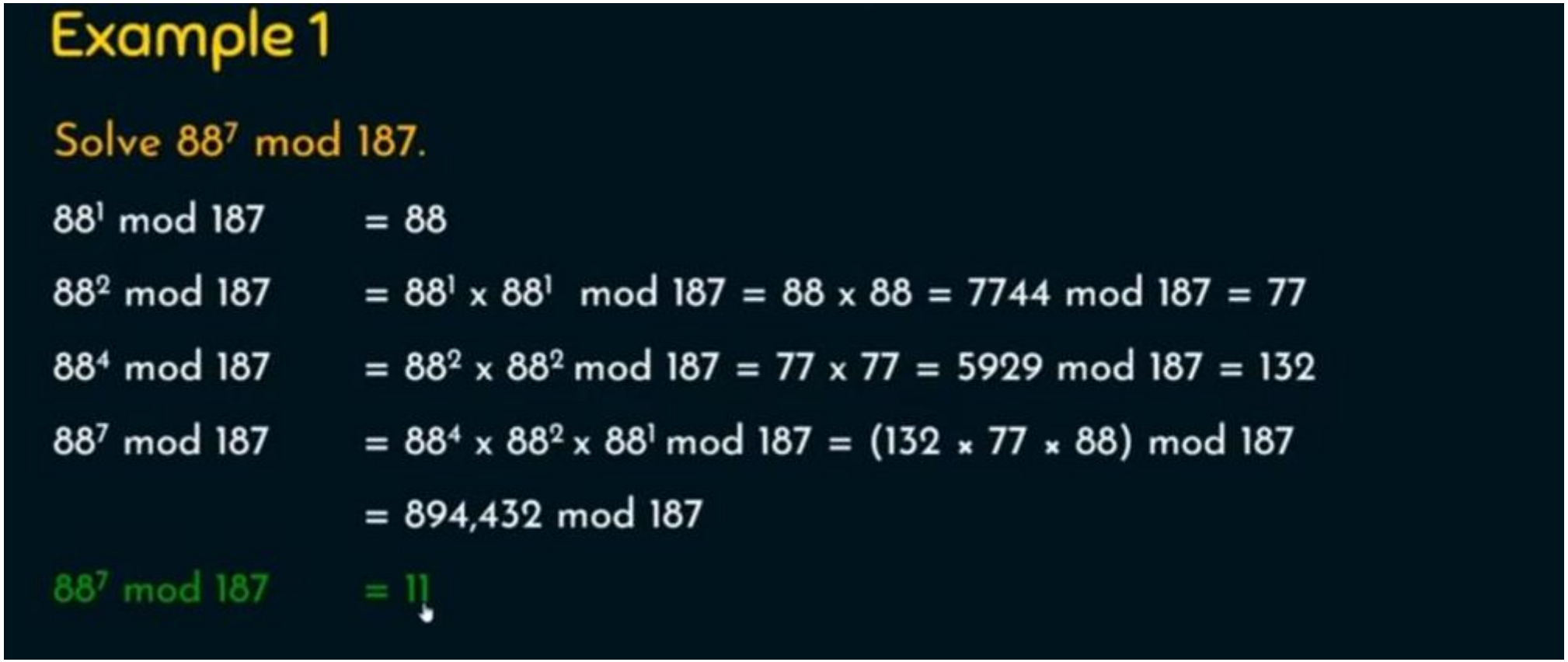
$= 49 \times -7 \bmod 30$

$= -133 \bmod 30$

$= -13 \bmod 30$

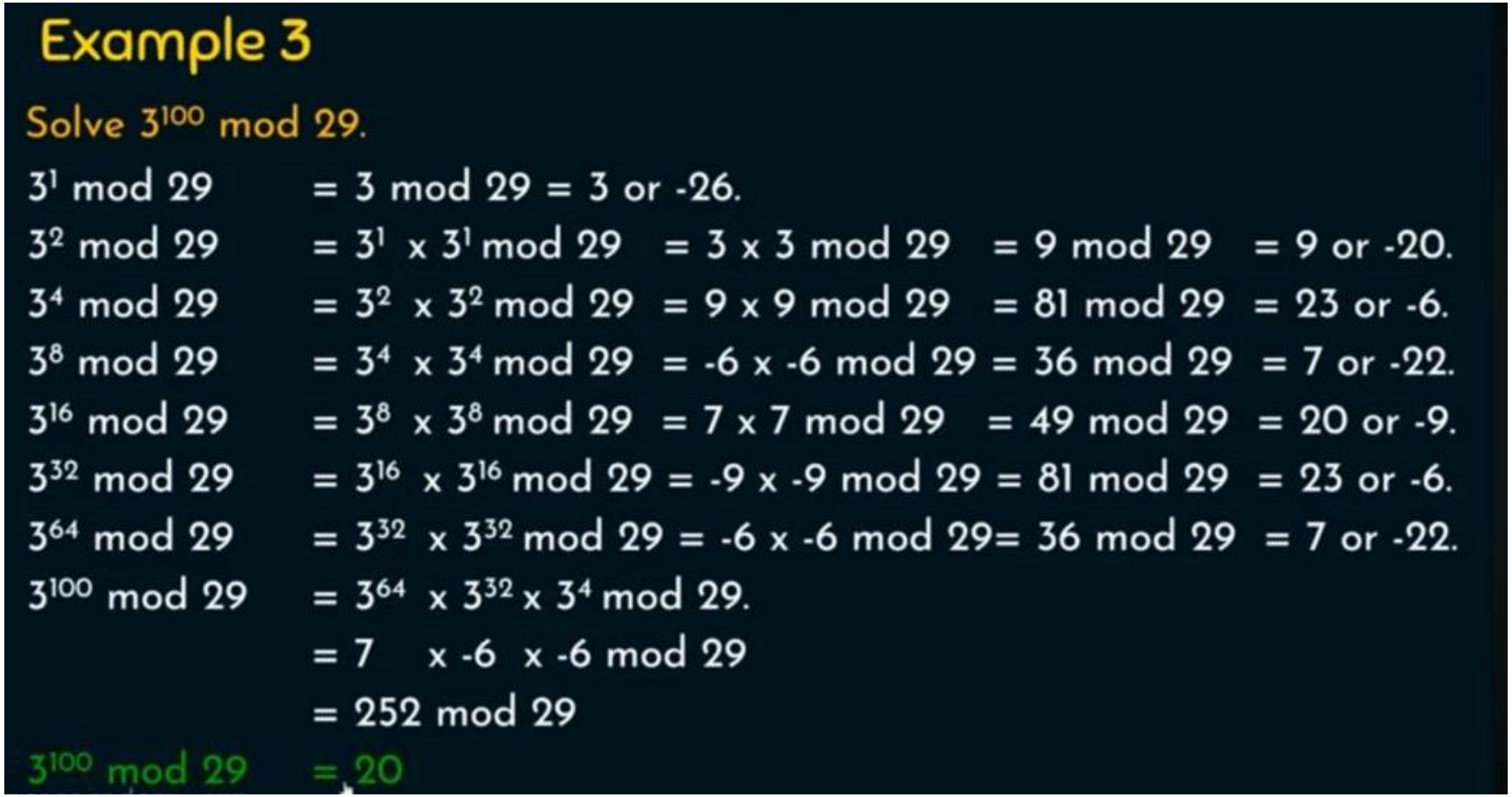
$= 17 \bmod 30$

Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

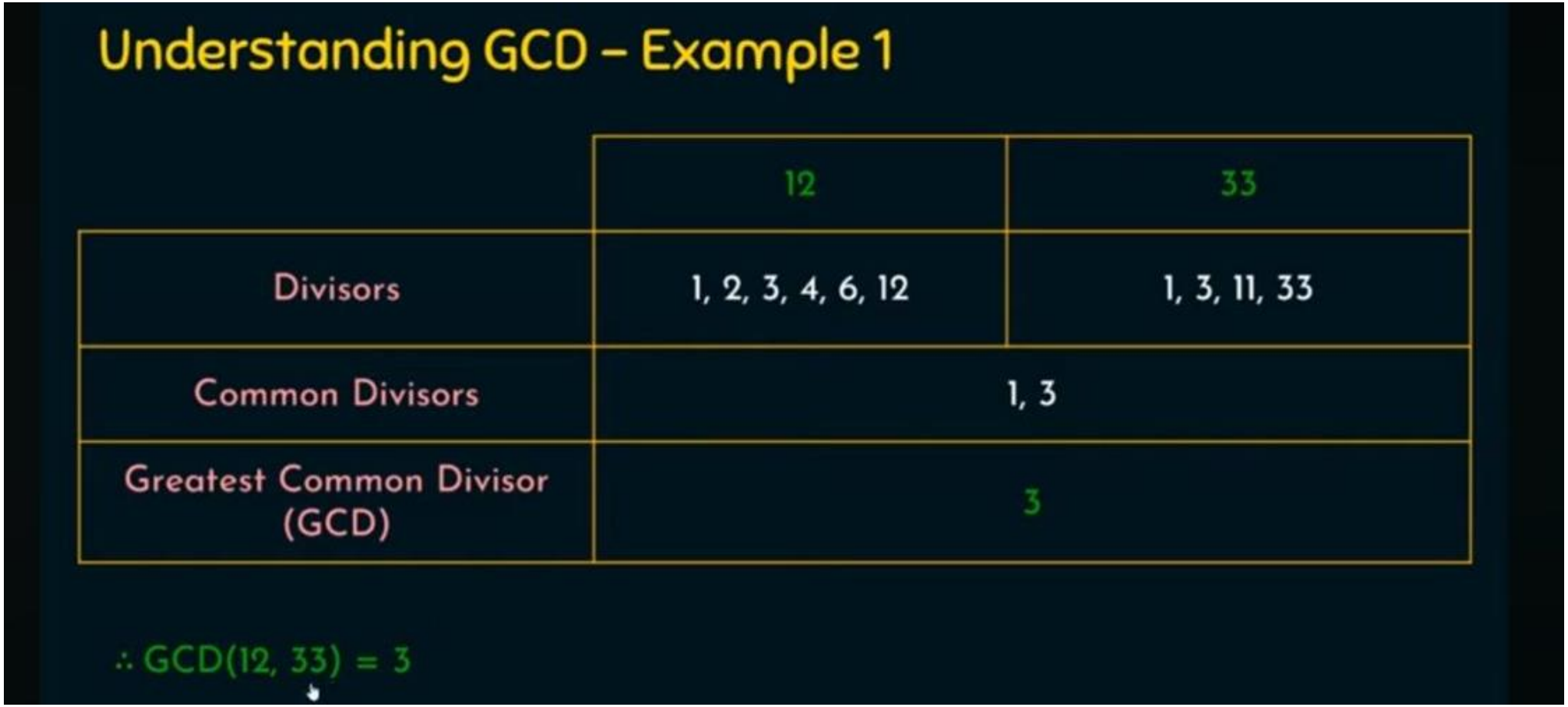


Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

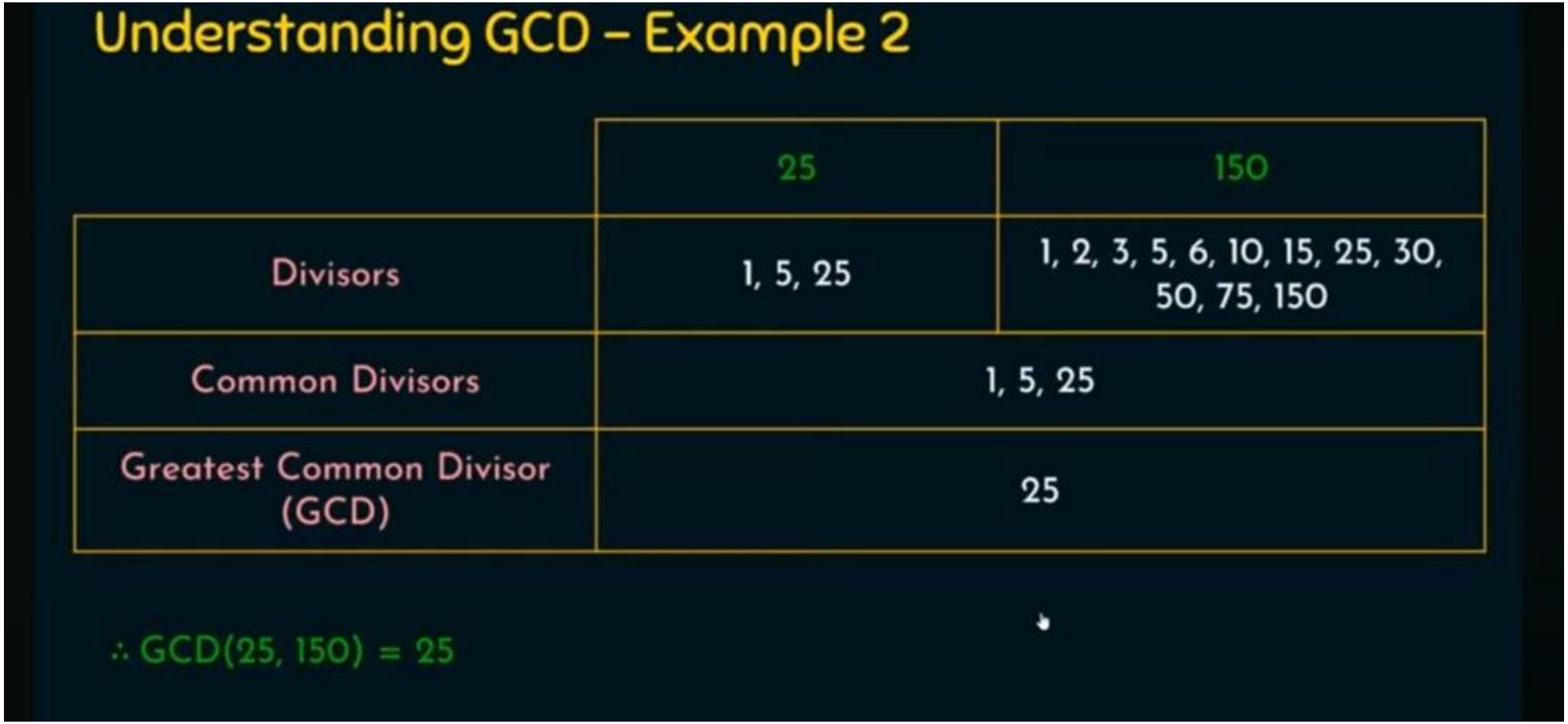
Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



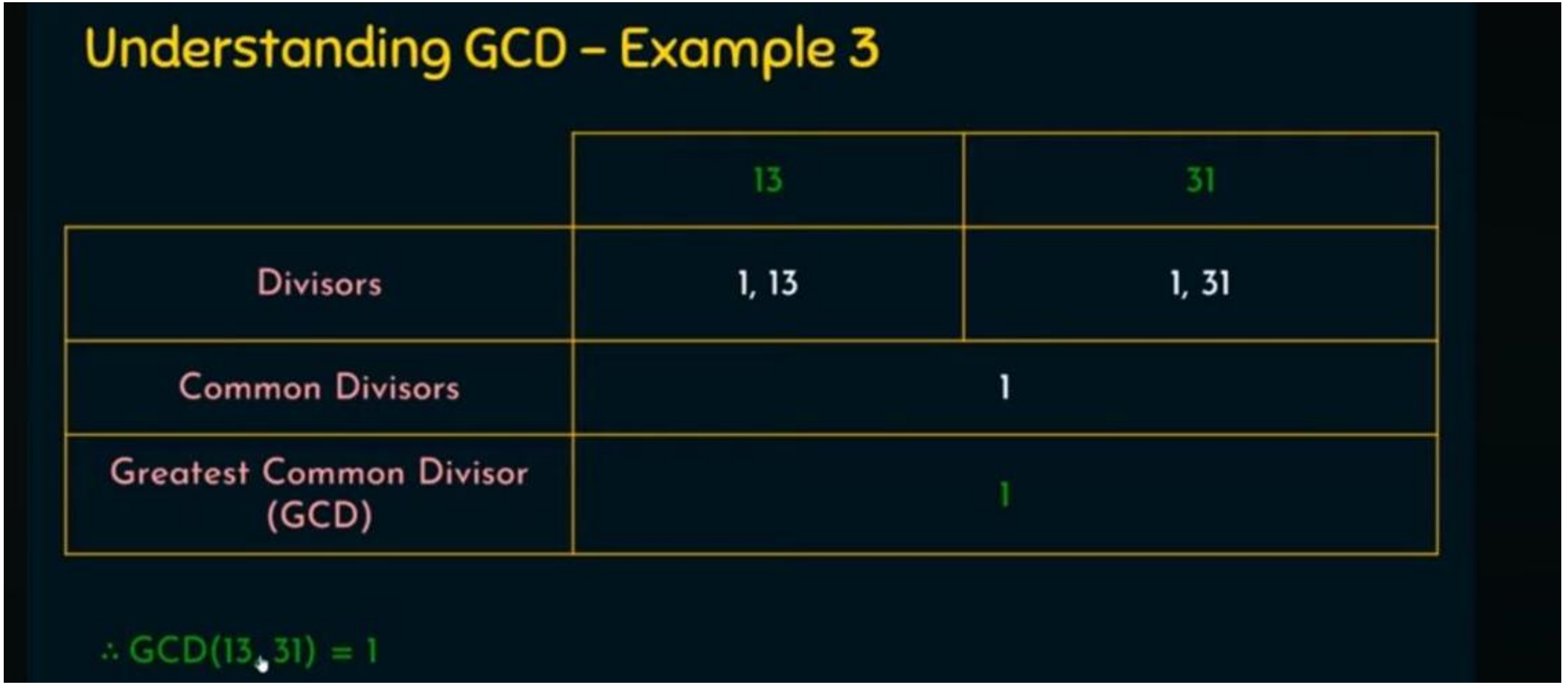
Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

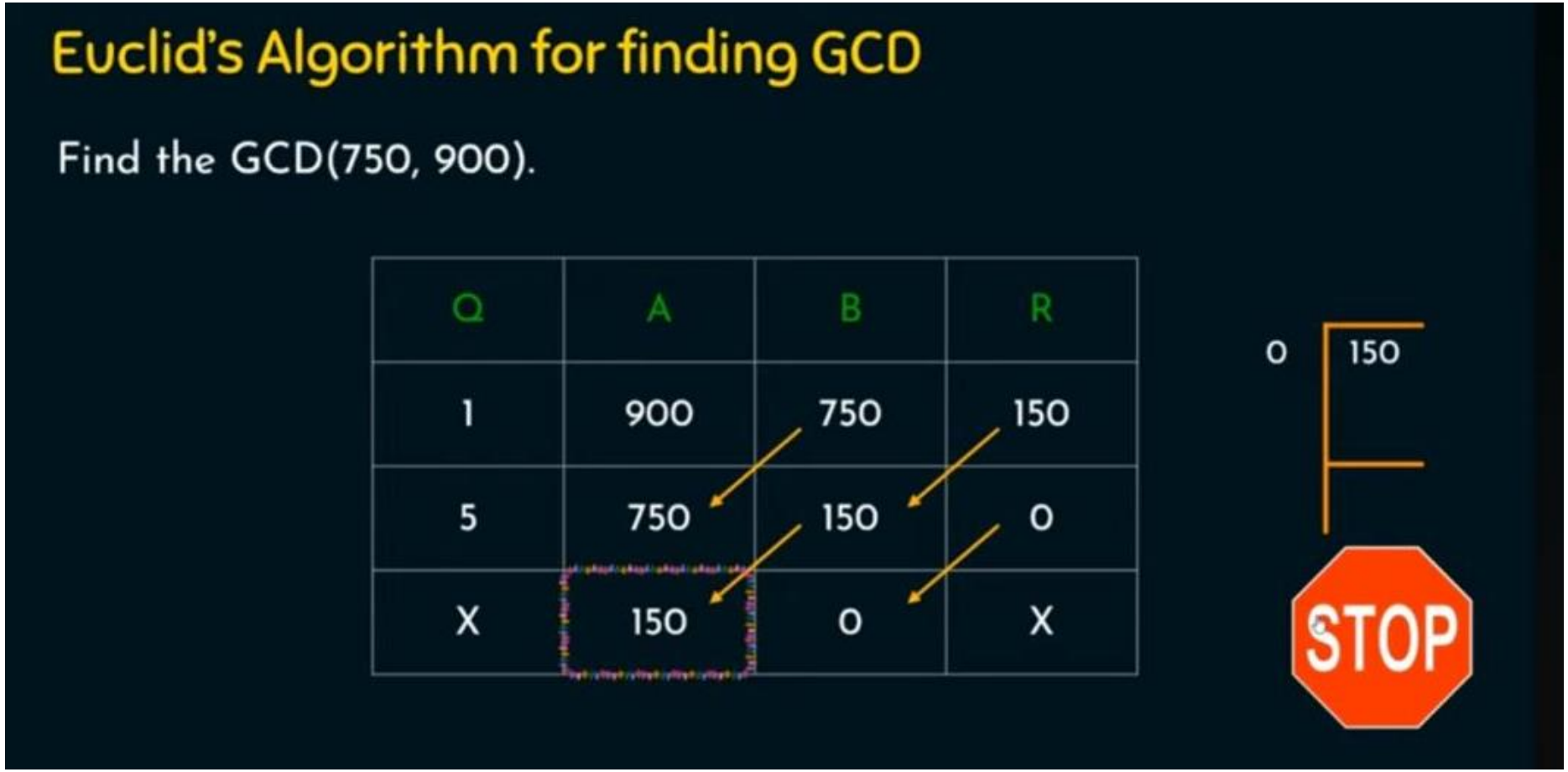


Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

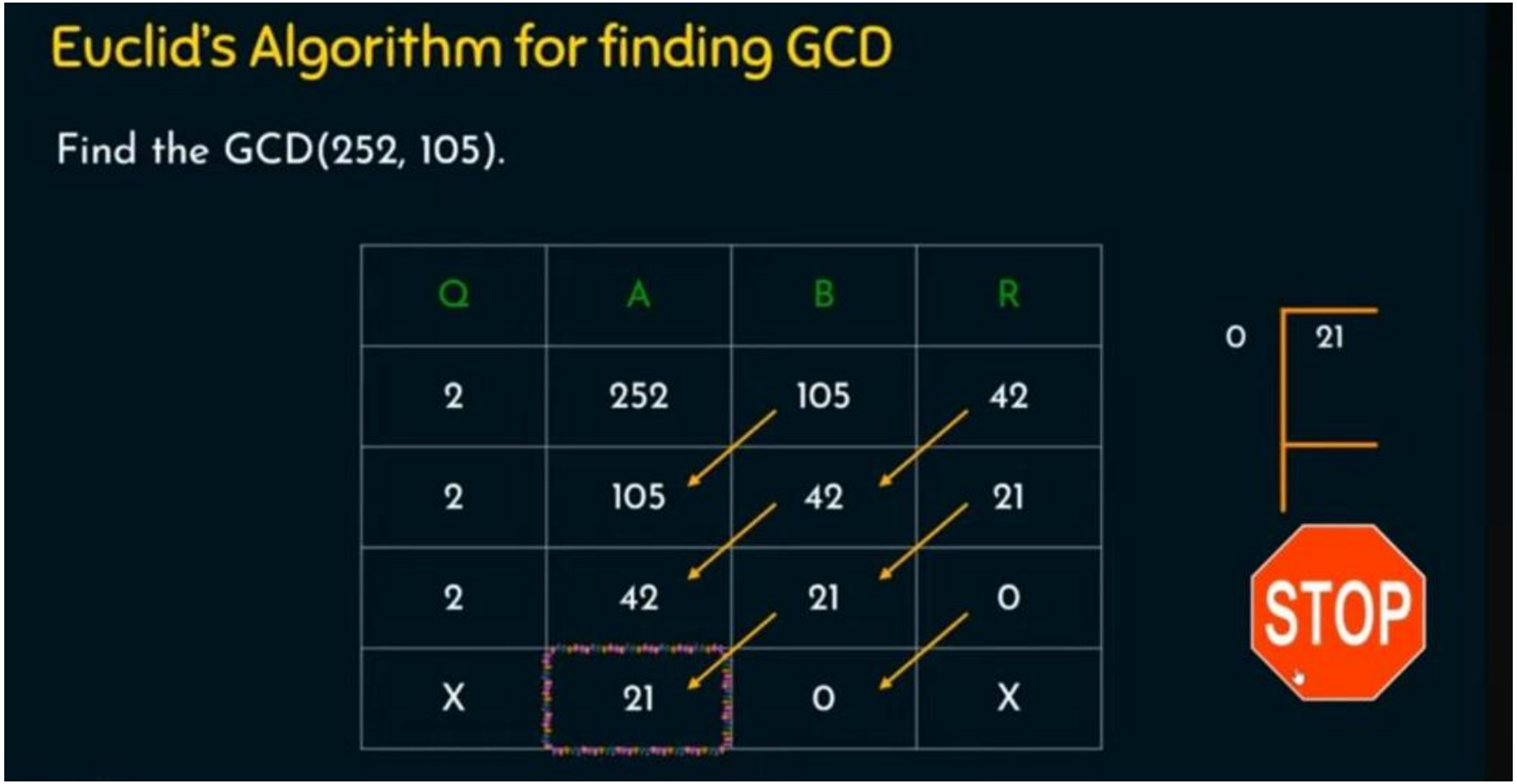


Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic

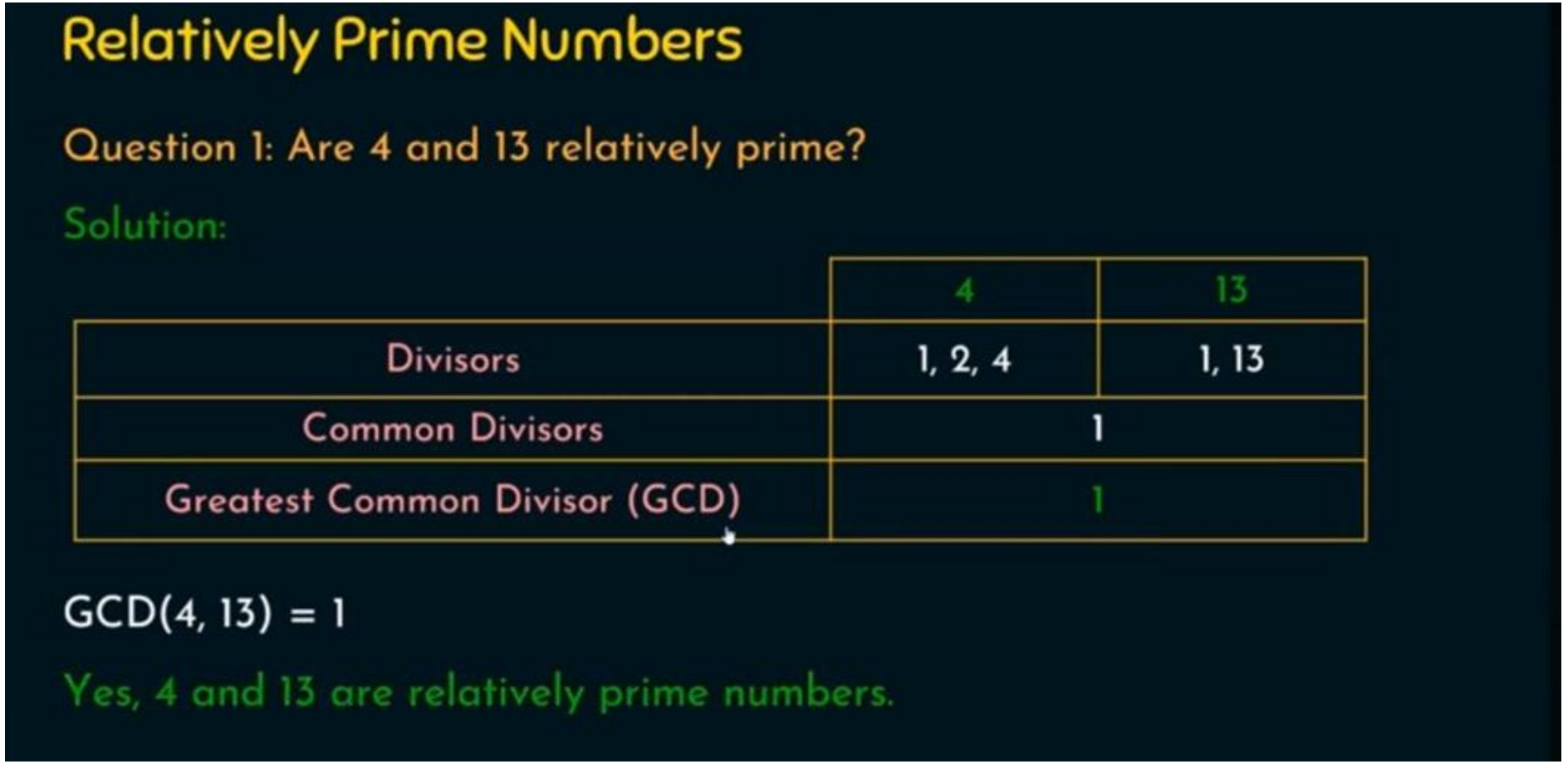
Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



Basic Concepts in Number Theory and Finite Fields – Modular Arithmatic



Basic Concepts in Number Theory and Finite Fields

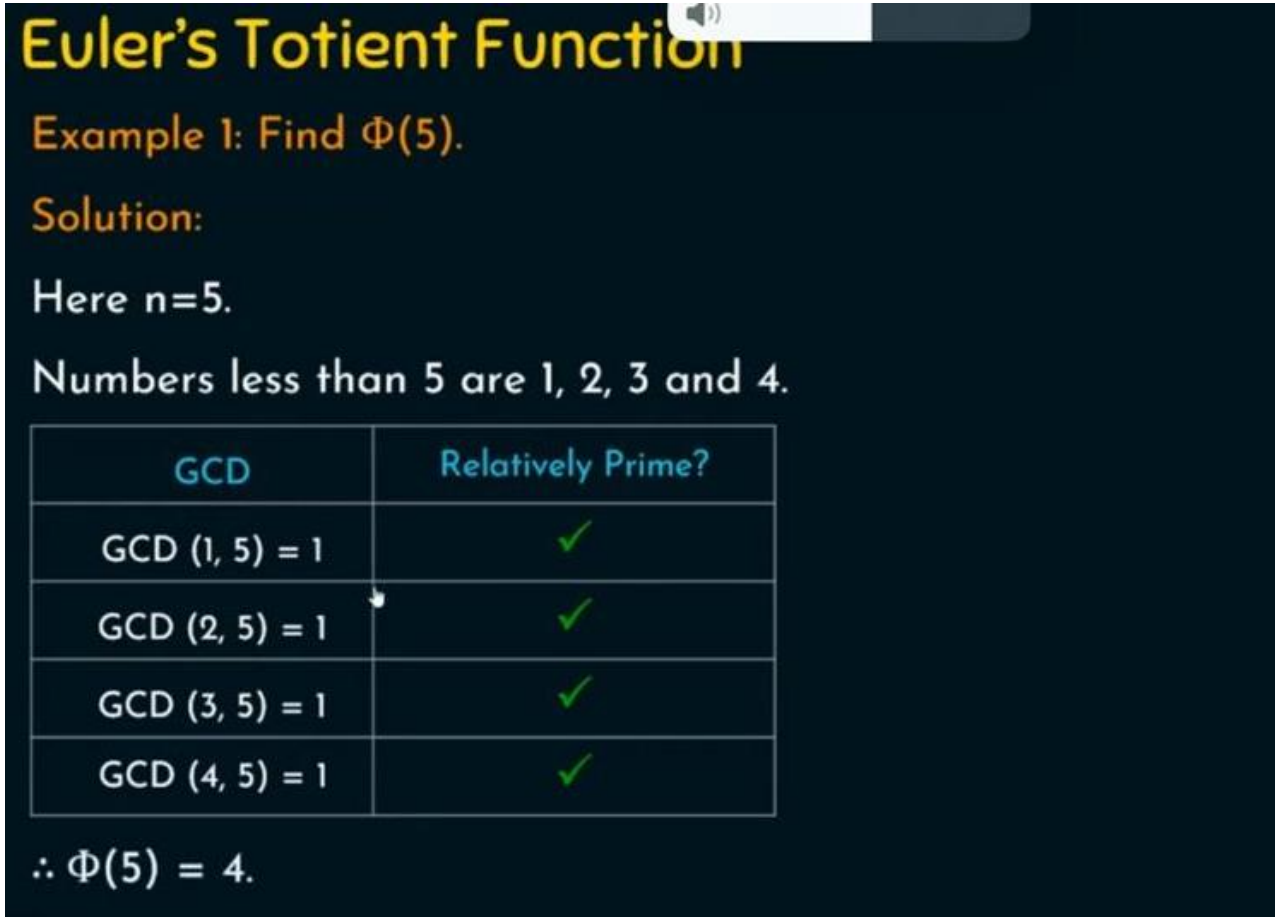


Basic Concepts in Number Theory and Finite Fields

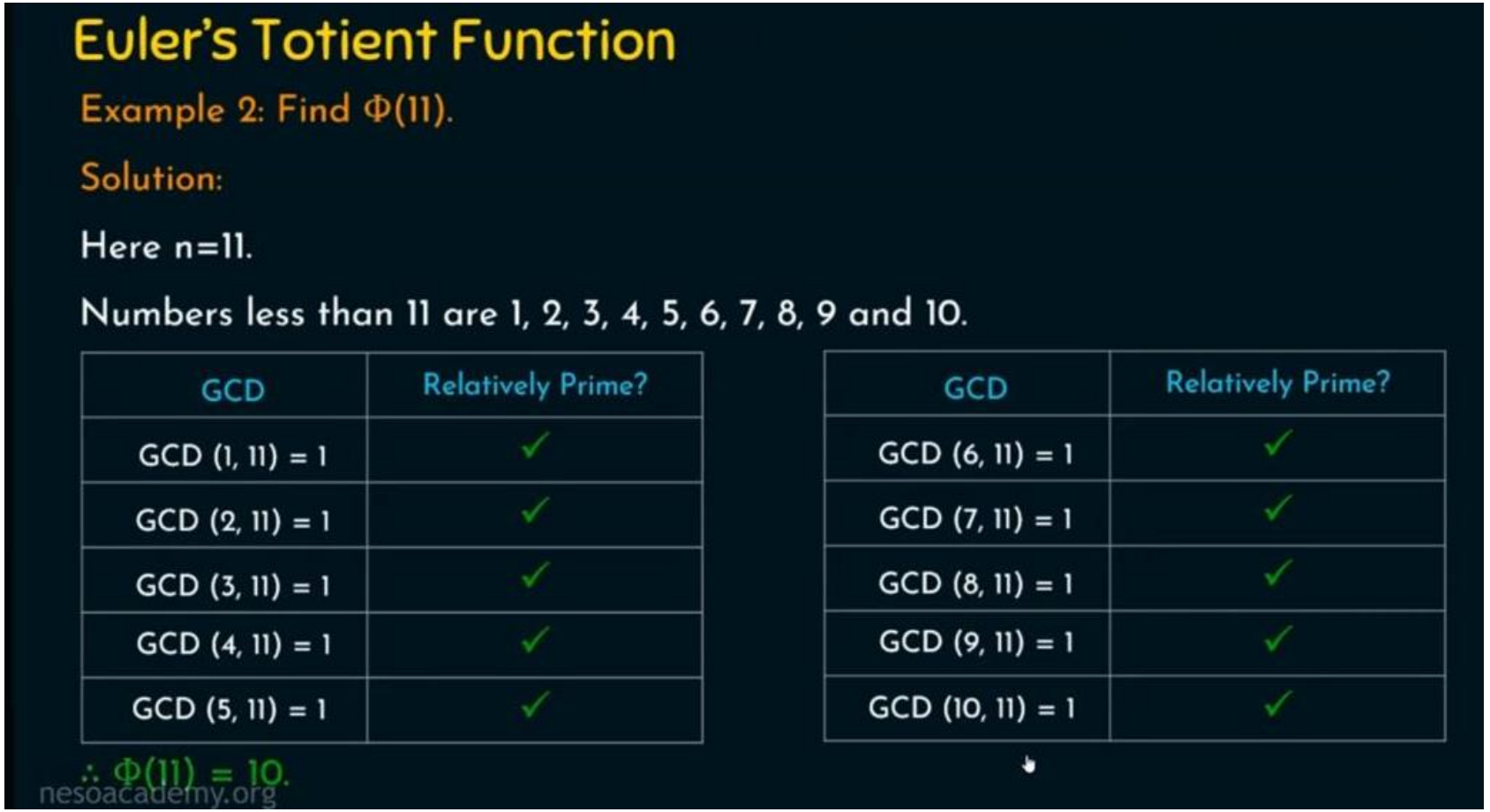
Relatively Prime Numbers

a	b	GCD(a, b)	Relatively Prime?	Remarks
11	17	1	Yes	'a' and 'b' are prime
11	21	1	Yes	'a' is prime and 'b' is composite
12	77	1	Yes	'a' and 'b' are composite

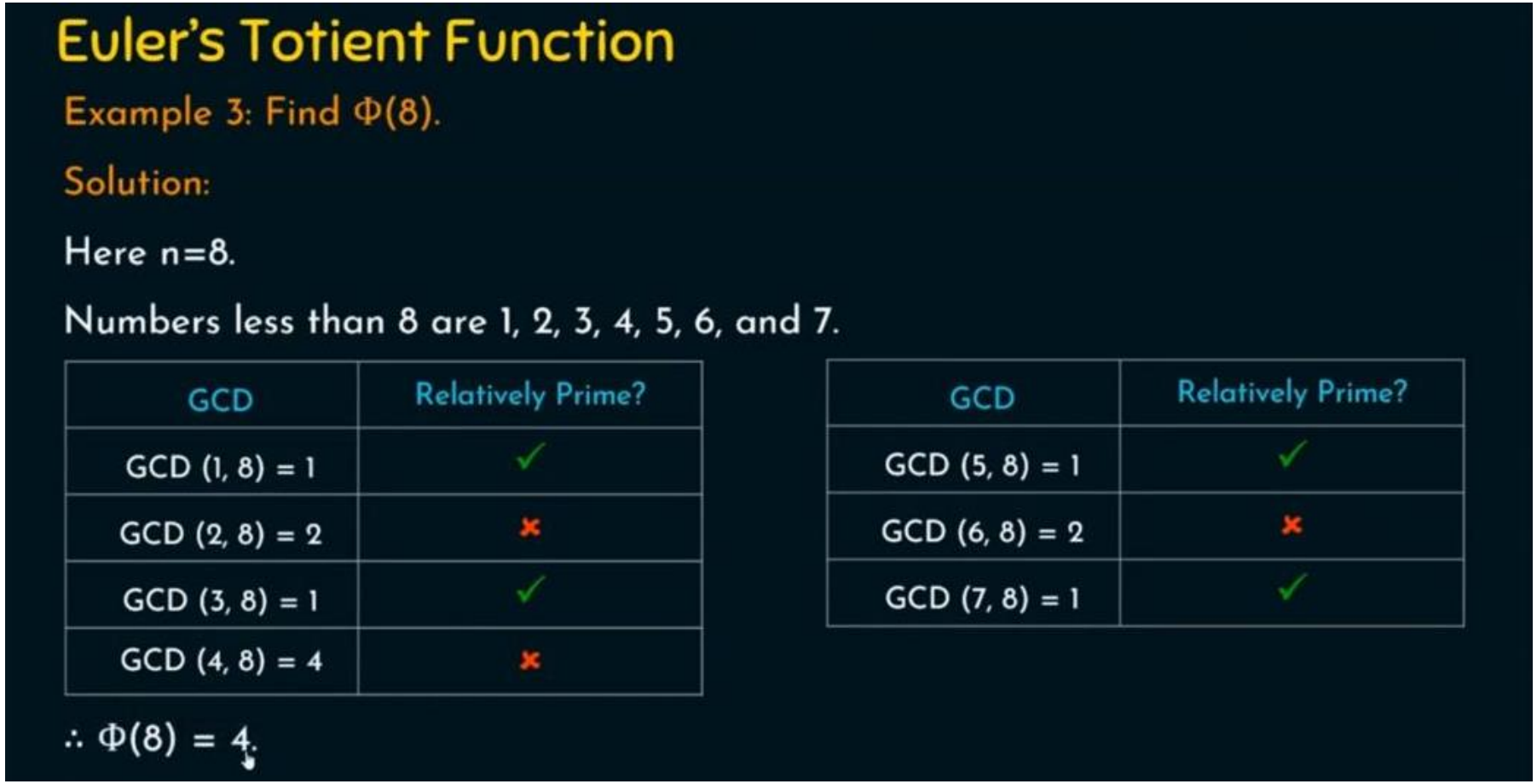
Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

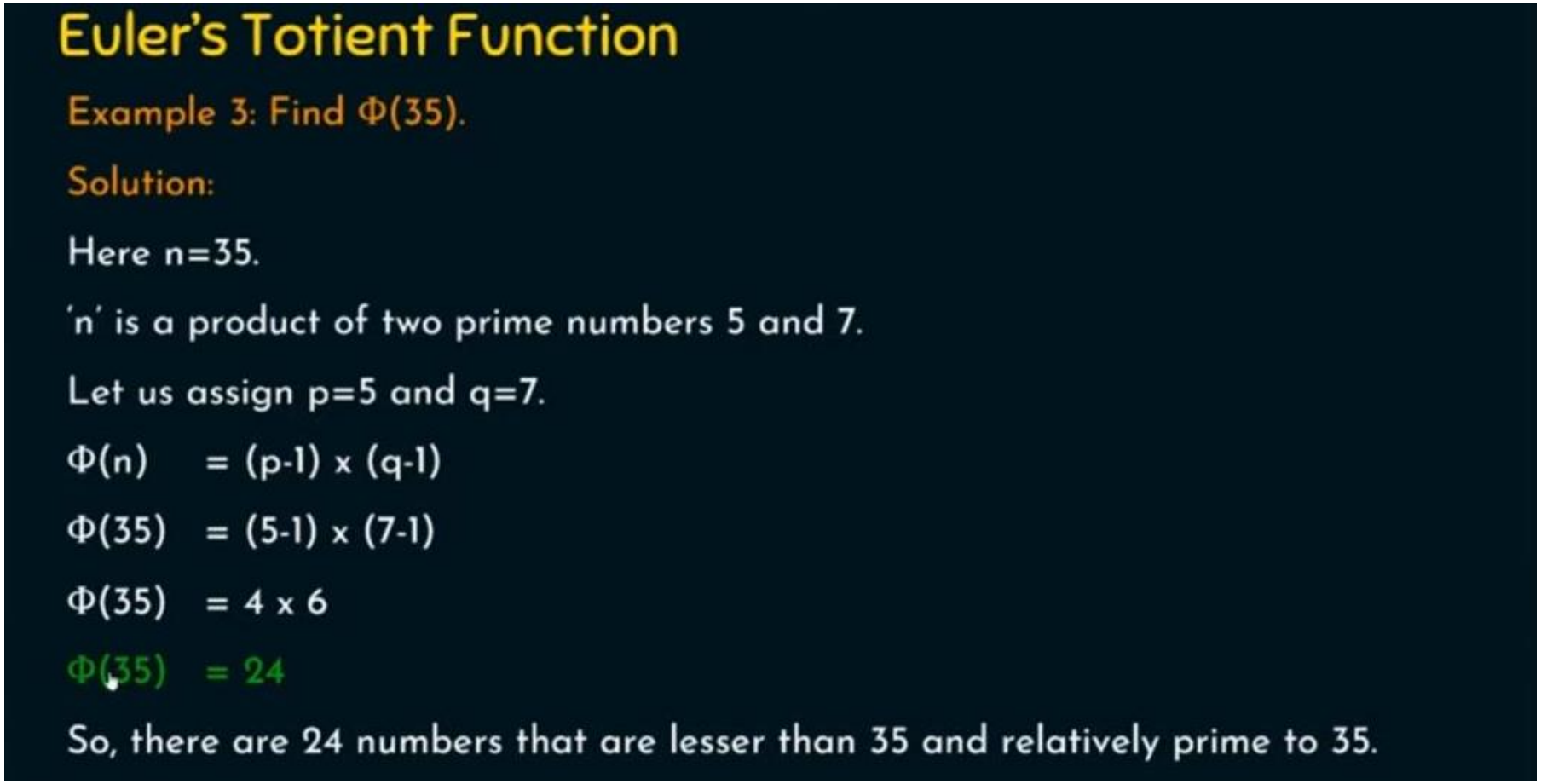


Basic Concepts in Number Theory and Finite Fields

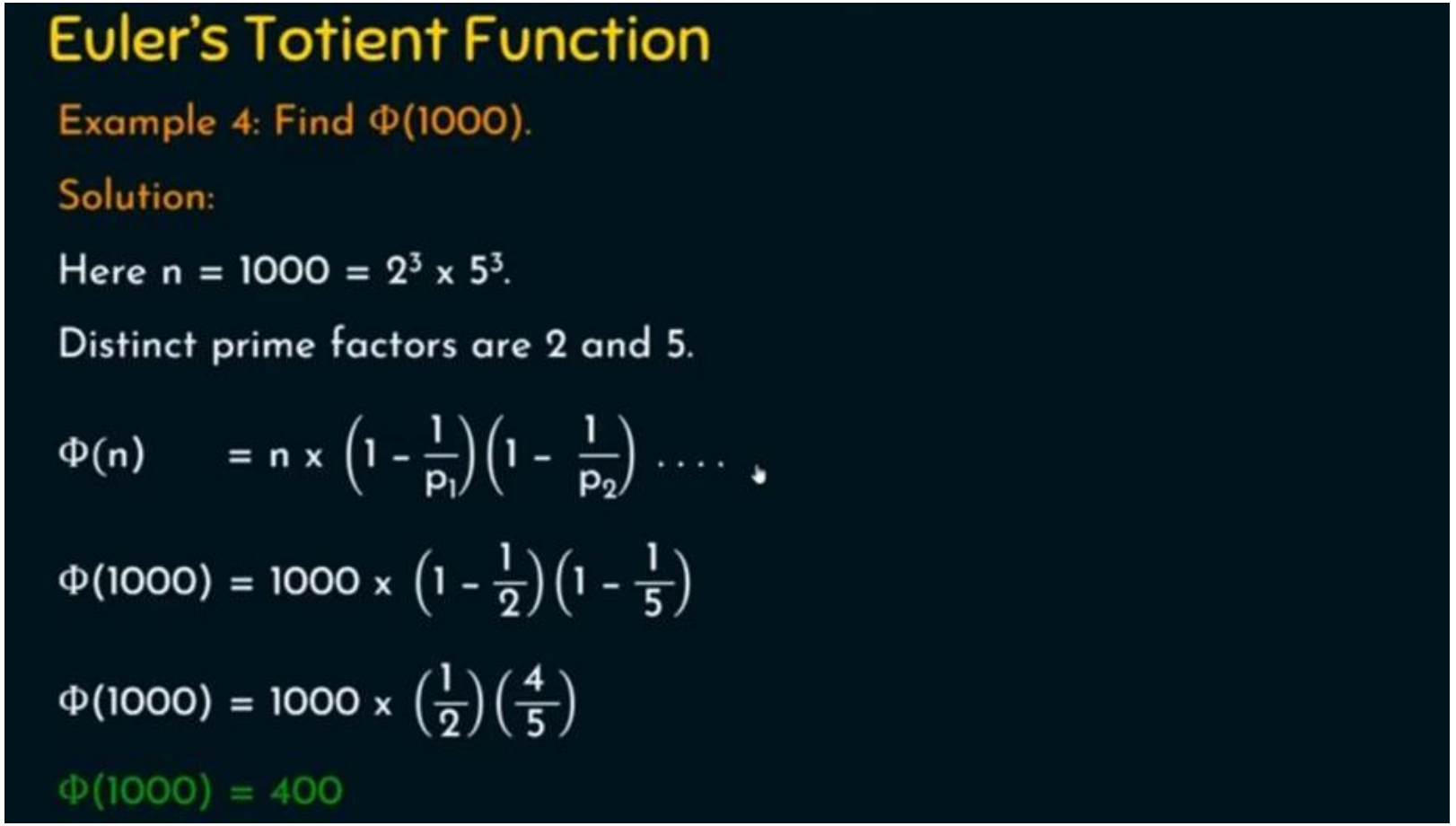
Euler's Totient Function

	Criteria of 'n'	Formula
$\Phi(n)$	'n' is prime.	$\Phi(n) = (n-1)$
	$n = p \times q$. 'p' and 'q' are primes.	$\Phi(n) = (p-1) \times (q-1)$
	$n = a \times b$. Either 'a' or 'b' is composite. Both 'a' and 'b' are composite.	$\Phi(n) = n \times \left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right) \ldots$ where $p_1, p_2, \ldots$ are distinct primes.

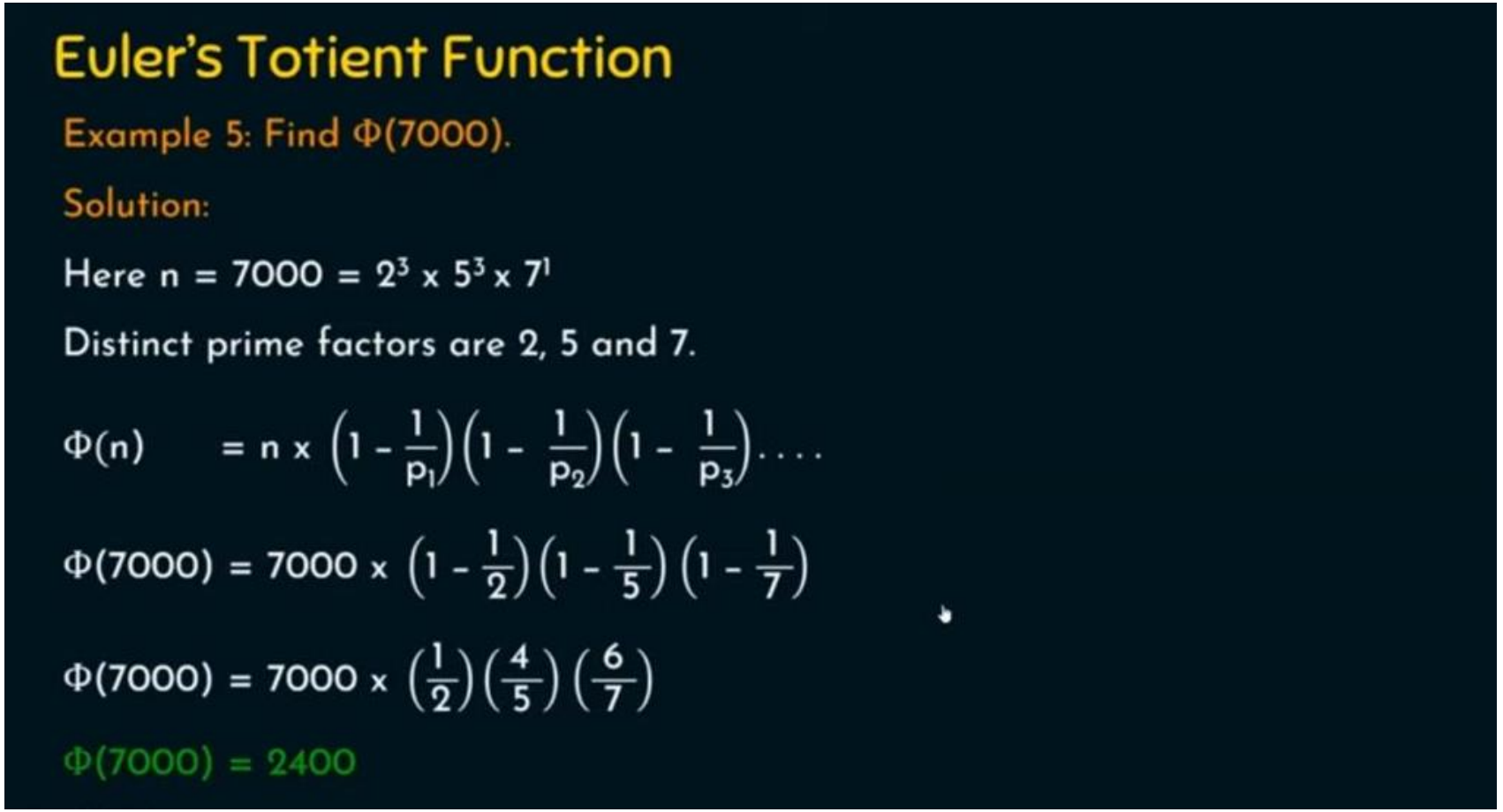
Basic Concepts in Number Theory and Finite Fields



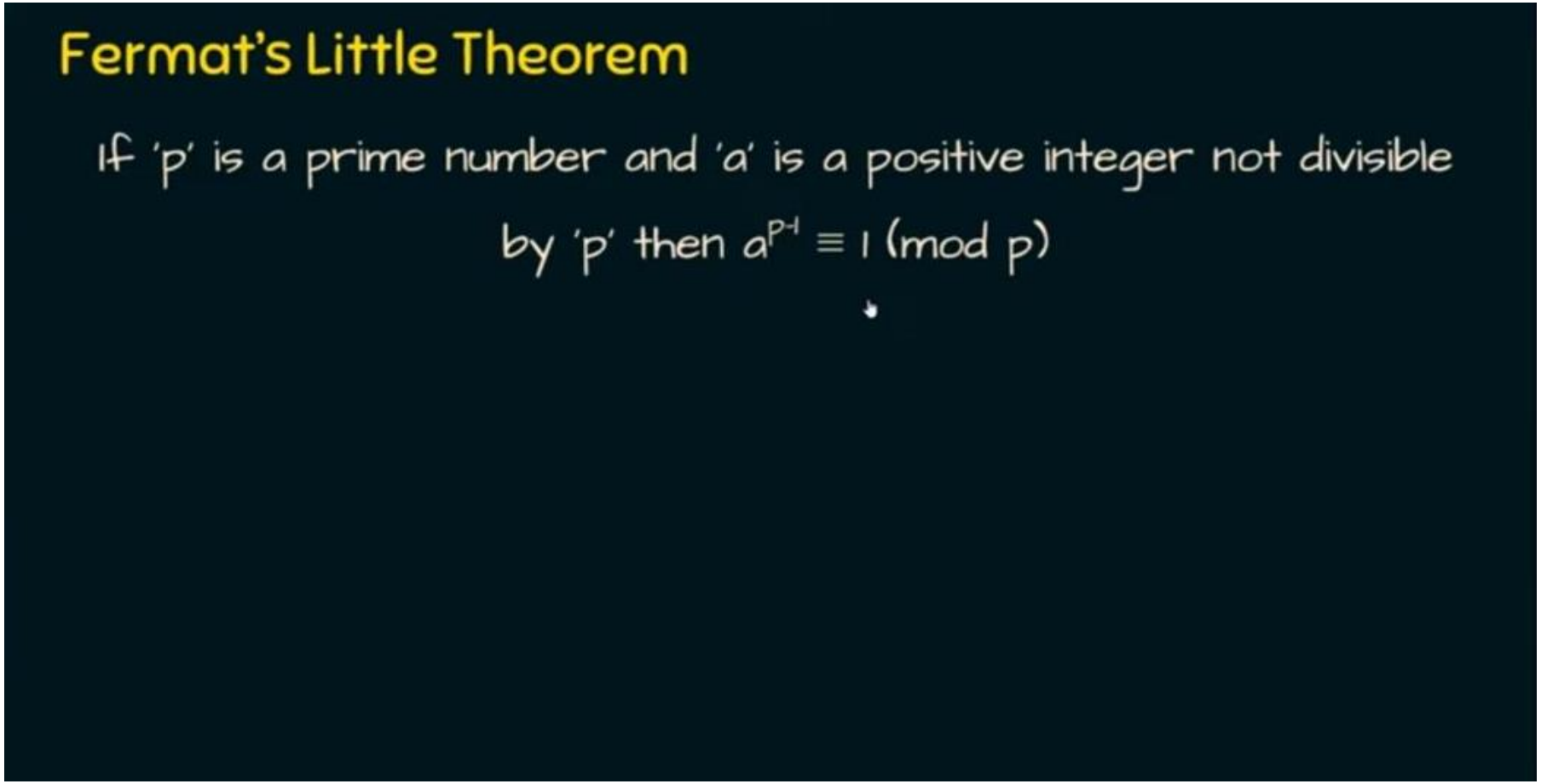
Basic Concepts in Number Theory and Finite Fields



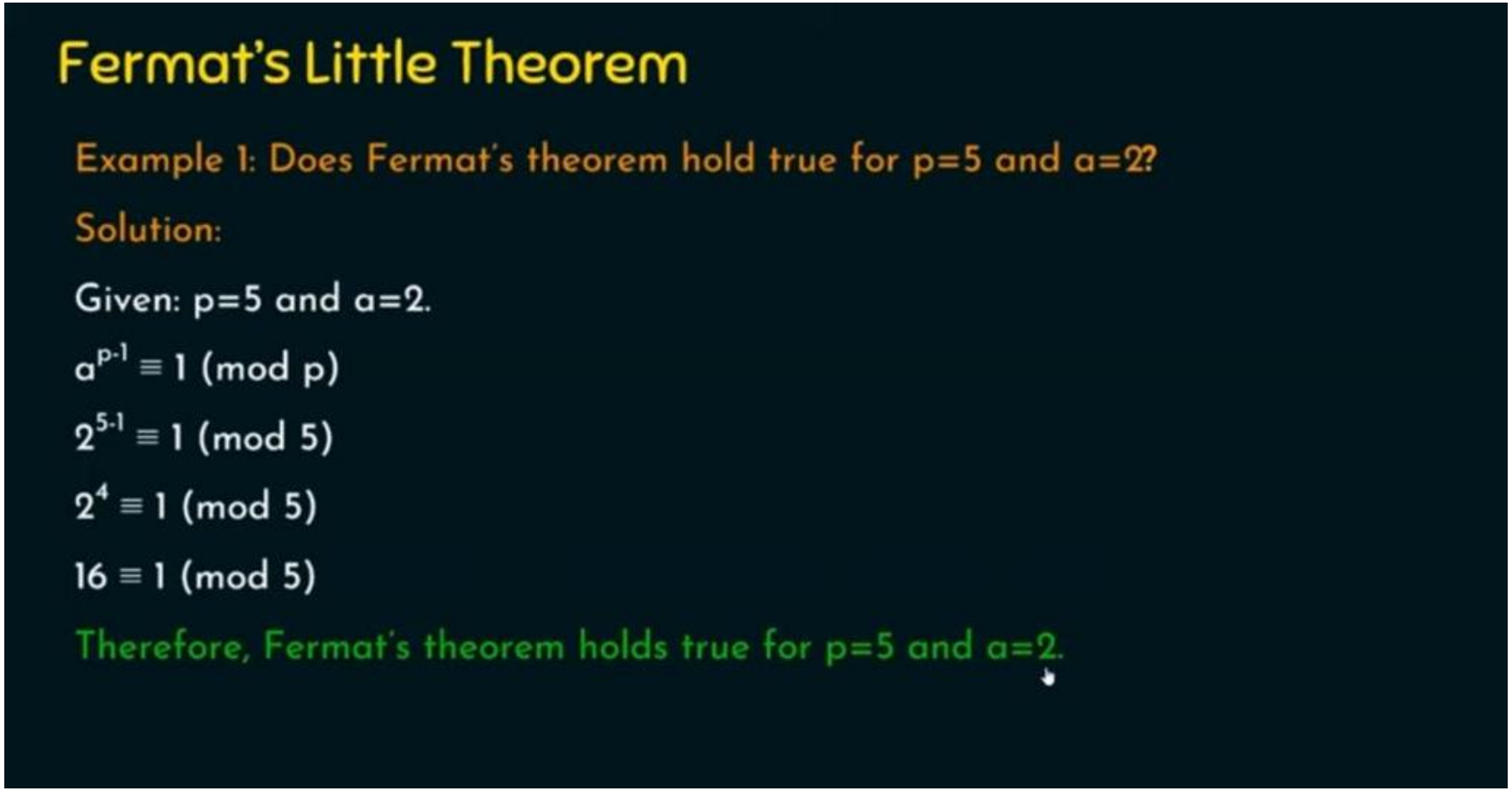
Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

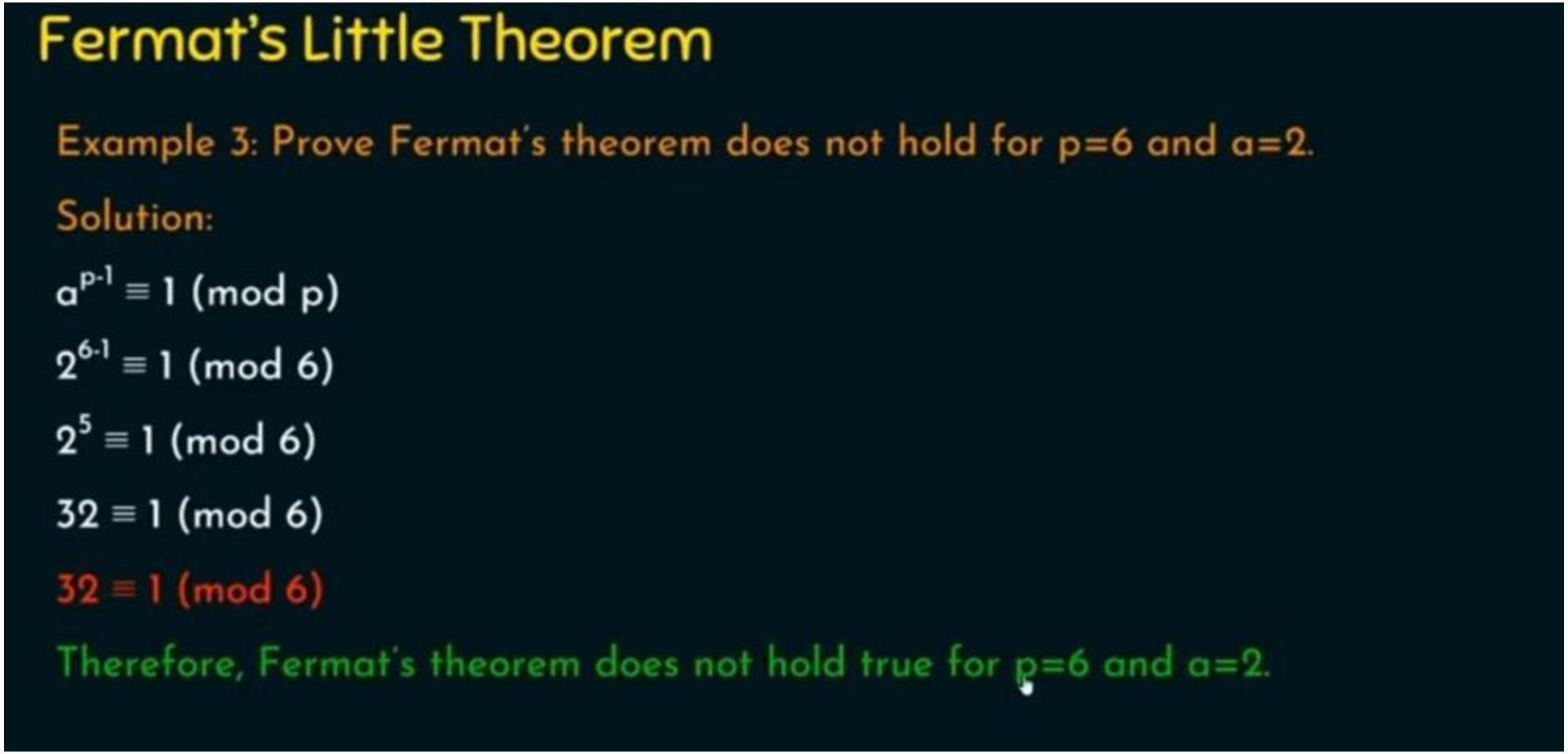


Basic Concepts in Number Theory and Finite Fields

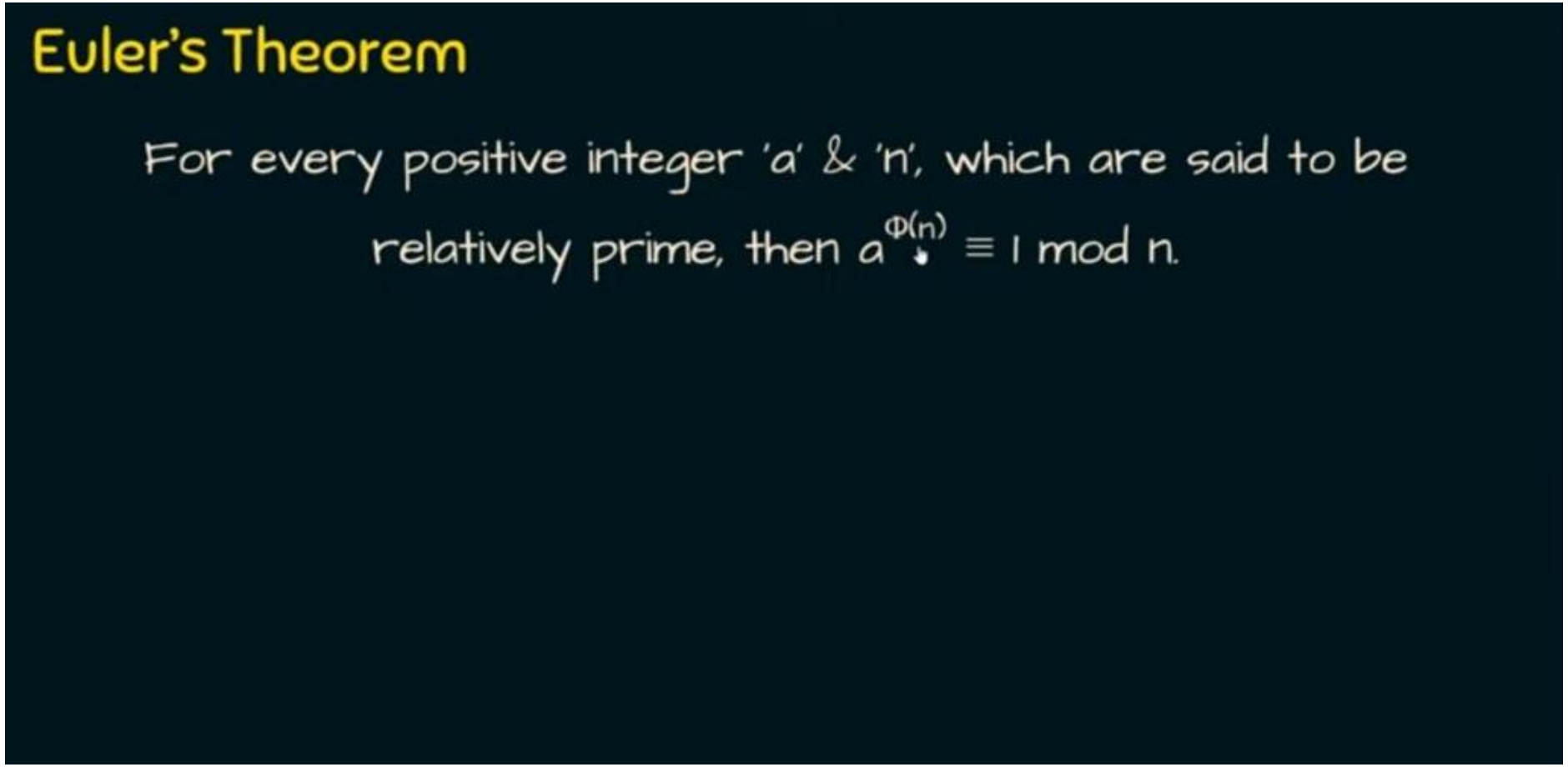


Basic Concepts in Number Theory and Finite Fields

Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

Euler's Theorem

Example 1: Prove Euler's theorem hold true for a=3 and n=10.

Solution:

Given: a=3 and n=10.

$a^{\Phi(n)} \equiv 1 \pmod{n}$

$3^{\Phi(10)} \equiv 1 \pmod{10}$

$\Phi(10) = 4$

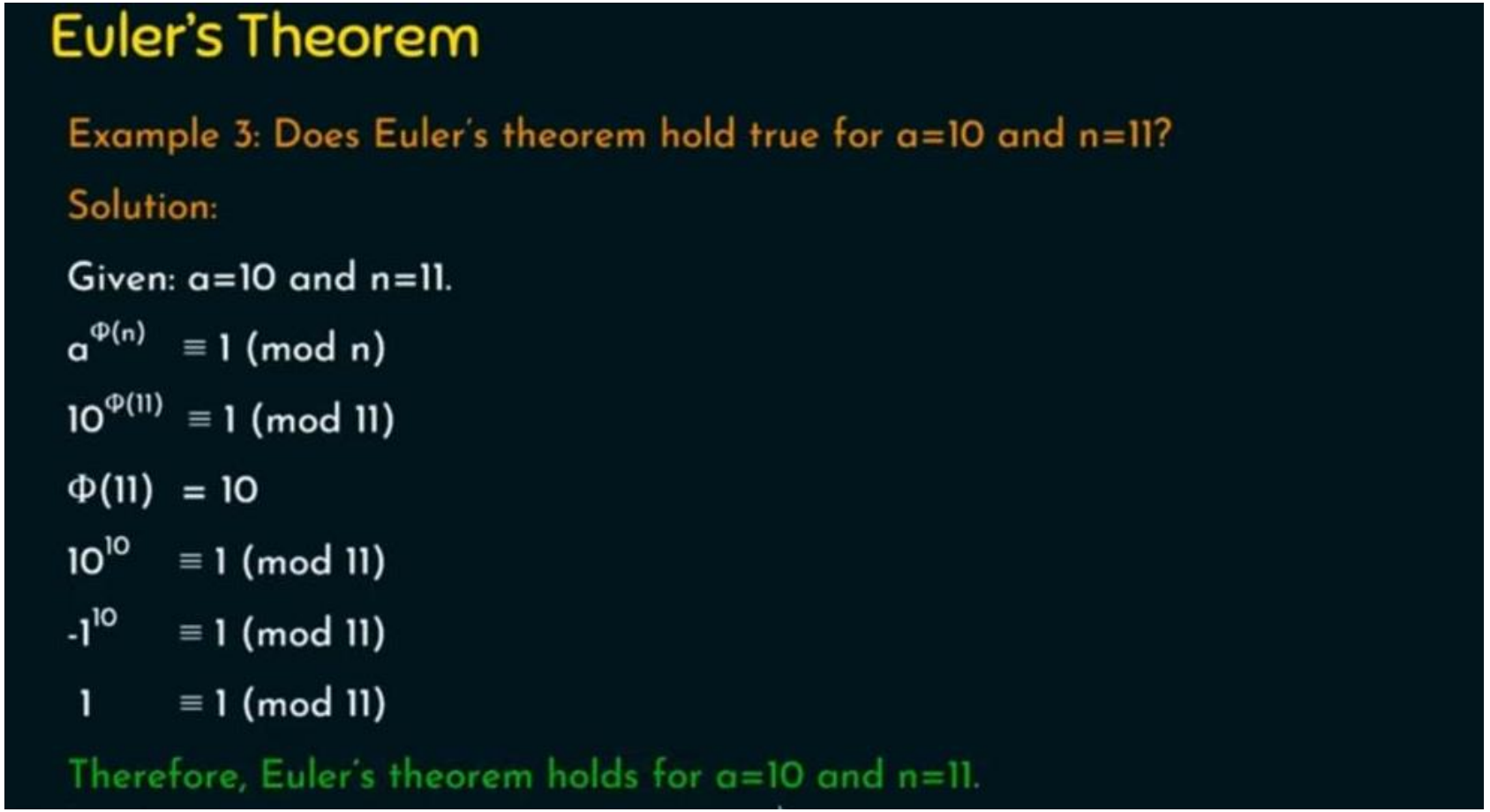
$3^4 \equiv 1 \pmod{10}$

$81 \equiv 1 \pmod{10}$

Therefore, Euler's theorem holds true for a=3 and n=10.

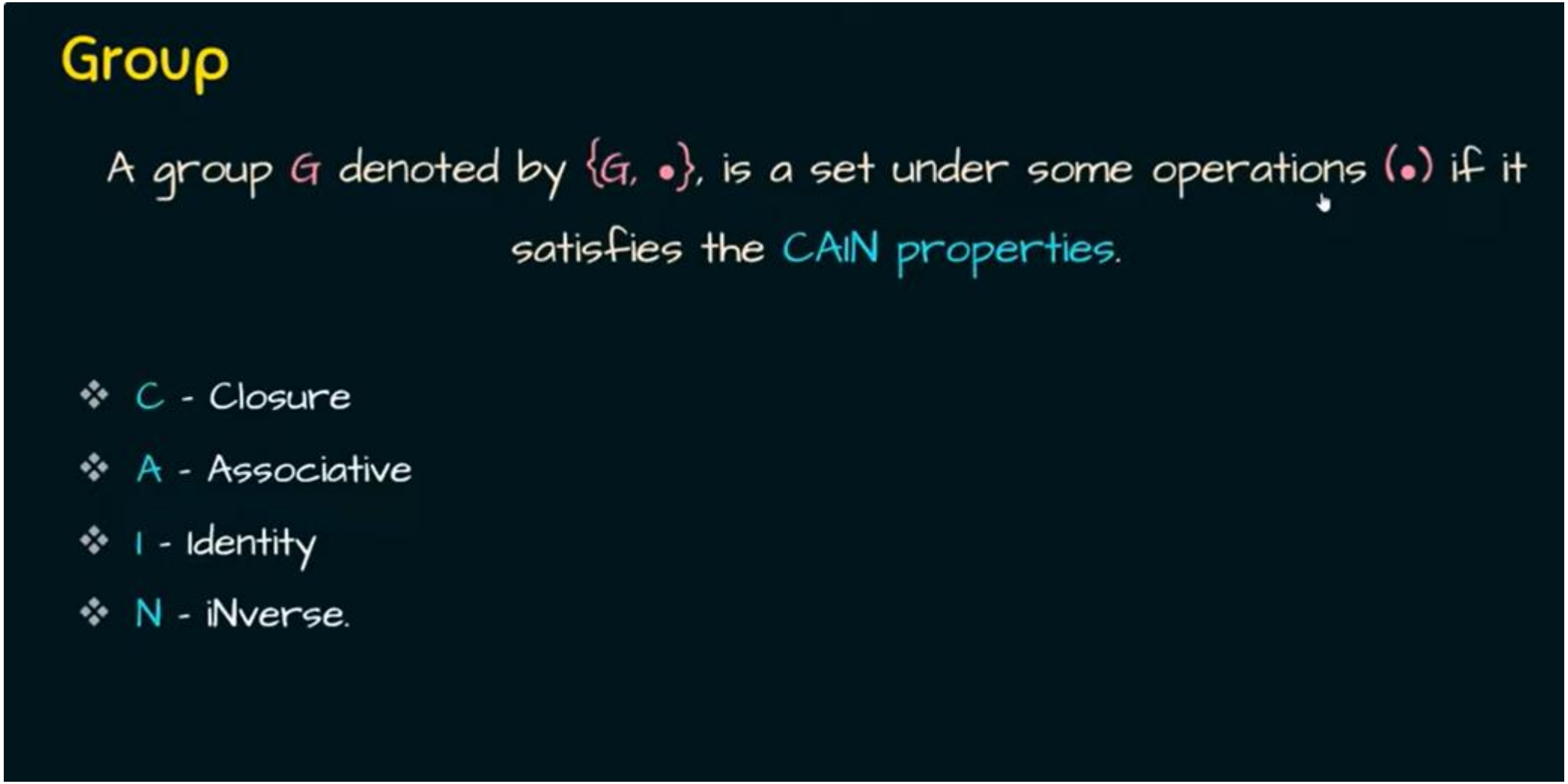
Basic Concepts in Number Theory and Finite Fields

Basic Concepts in Number Theory and Finite Fields

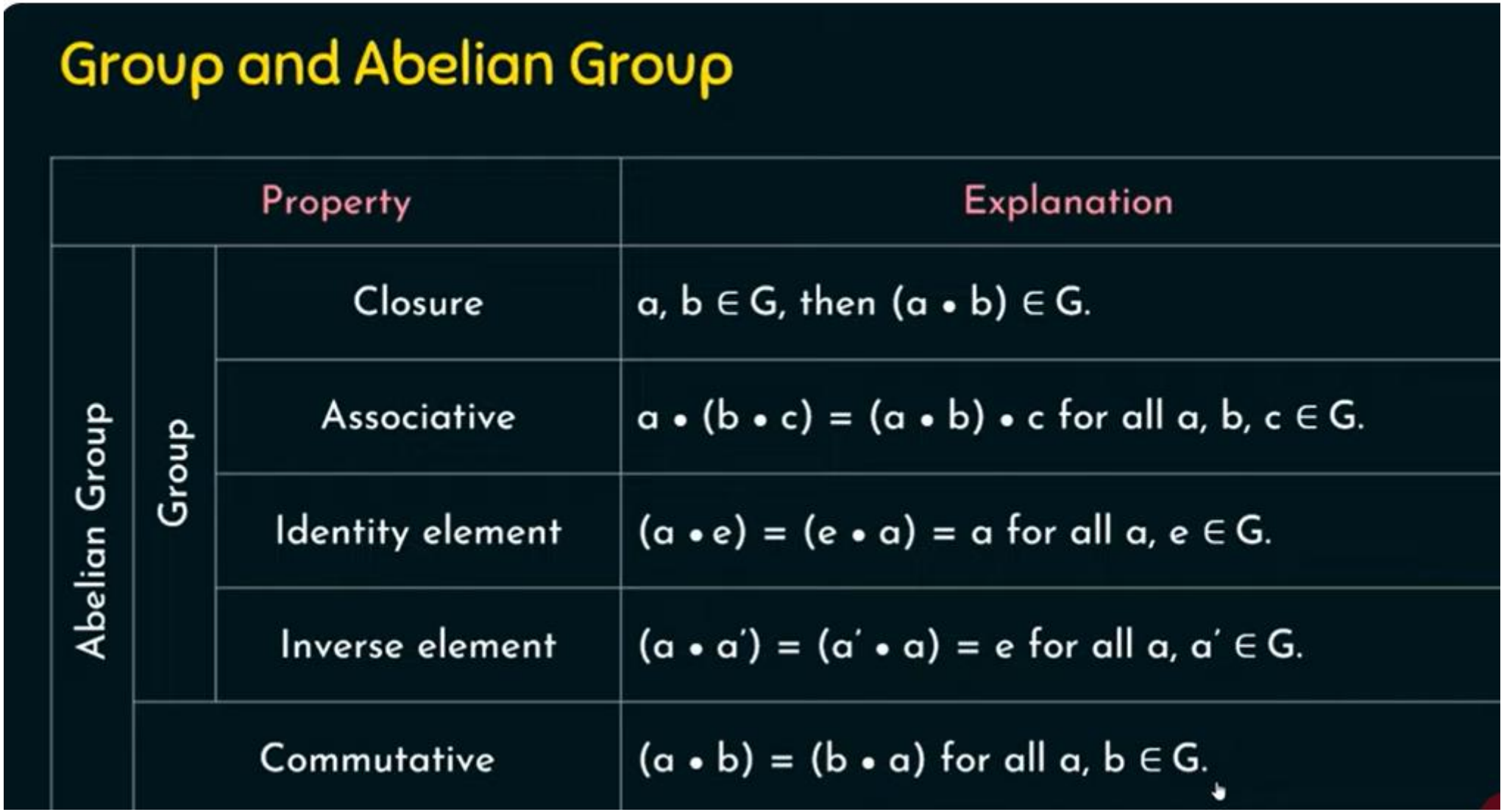


Basic Concepts in Number Theory and Finite Fields

Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

Example

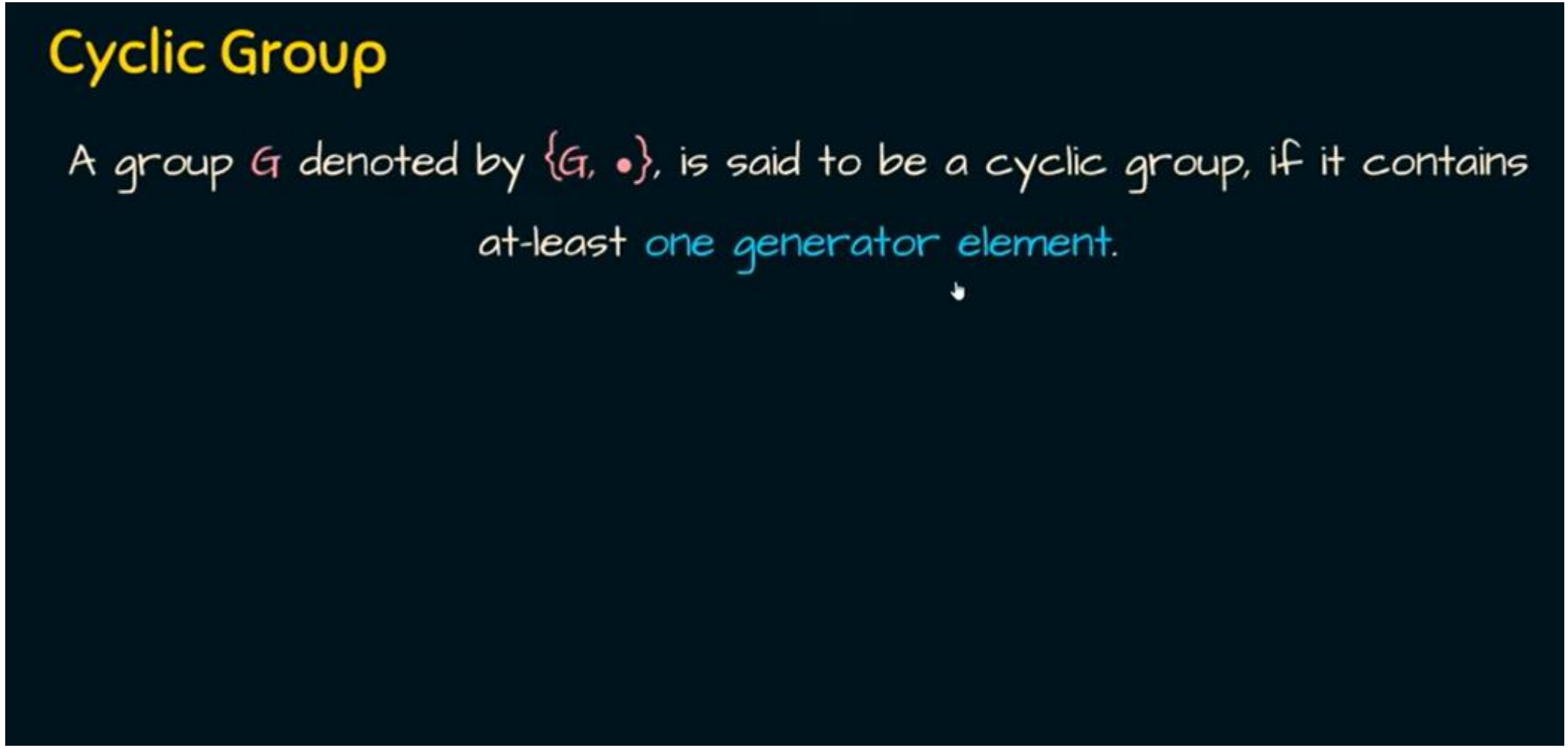
Question: Is (Z, +) a group?

Solution:

$Z = \{ \ldots, -3, -2, -1, 0, 1, 2, 3, \ldots \}$

CAIN Property	Explanation	Satisfied?
Closure	If $a, b \in G$, then $(a \bullet b) \in G$. If $a = 5, b = -2 \in Z$ then $(a + b) = -3 \in Z$	✓
Associative	$a \bullet (b \bullet c) = (a \bullet b) \bullet c$ for all $a, b, c \in G$. $5 + (3 + 7) = (5 + 3) + 7 \in Z$	✓
Identity element	$(a \bullet e) = (e \bullet a) = a$ for all $a \in G$. $(5 + 0) = (0 + 5) = 5$ for all $a \in G$.	✓
Inverse element	$(a \bullet a') = (a' \bullet a) = e$ for all $a, a' \in G$. $(5 + -5) = (-5 + 5) = 0$ for all $5, -5 \in Z$	✓
Commutative	$(a \bullet b) = (b \bullet a)$ for all $a, b \in G$. $(5 + 9) = (9 + 5)$ for all $9, 5 \in Z$.	

Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

Cyclic Group

Question 1: Prove that $(G, *)$ is a cyclic group, where $G = \{1, \omega, \omega^2\}$.

Solution:

Composition Table

*	1	ω	ω^2
1	1	ω	ω^2
ω	ω	ω^2	1
ω^2	ω^2	1	ω

$1^1 = 1$
$1^2 = 1*1 = 1$
$1^3 = 1*1*1 = 1$
$1^4 = 1*1*1*1 = 1$

Not a Generator

$\omega^1 = \omega$
$\omega^2 = \omega*\omega = \omega^2$
$\omega^3 = \omega^2*\omega = 1$
$\omega^4 = \omega^3*\omega = \omega$

Generator

$(\omega^2)^1 = \omega^2$
$(\omega^2)^2 = \omega^4 = \omega^3*\omega = \omega$
$(\omega^2)^3 = \omega^6 = \omega^3*\omega^3 = 1$
$(\omega^2)^4 = \omega^8 = \omega^3*\omega^3*\omega^2 = \omega^2$

Generator

The generators of $(G, *)$ are ω and ω^2.

$\therefore (G, *)$ is a cyclic group.

Basic Concepts in Number Theory and Finite Fields

Cyclic Group

Question 2: When does group G with operation 'x', is said to be a cyclic group?

Solution:

Let us take an element x

$G = \{ \ldots, x^{-4}, x^{-3}, x^{-2}, x^{-1}, 1, x, x^2, x^3, x^4, \ldots \}$

= Group generated by x

If $G = \langle x \rangle$ for some x, then we call G a cyclic group.

Basic Concepts in Number Theory and Finite Fields

Cyclic Group

Question 3: When does group G with operation '+', is said to be a cyclic group?

Solution:

Let us take an element y

G = { . . . . , -4y, -3y, -2y, -y, 0, y, 2y, 3y, 4y, ... }

= Group generated by y

If G = $\langle y \rangle$ for some y, then we call G a cyclic group.

Basic Concepts in Number Theory and Finite Fields

Rings

A ring R denoted by {R, +, *}, is a set of elements with two binary operations, called addition and multiplication, such that for all $a, b, c \in R$ the following axioms are obeyed:

- Group (A1-A4), Abelian Group(A5).
- Closure under multiplication (M1): If $a, b \in R$ then $ab \in R$
- Associativity of multiplication (M2): $a(bc) = (ab)c$ for all $a, b, c \in R$
- Distributive laws (M3) :

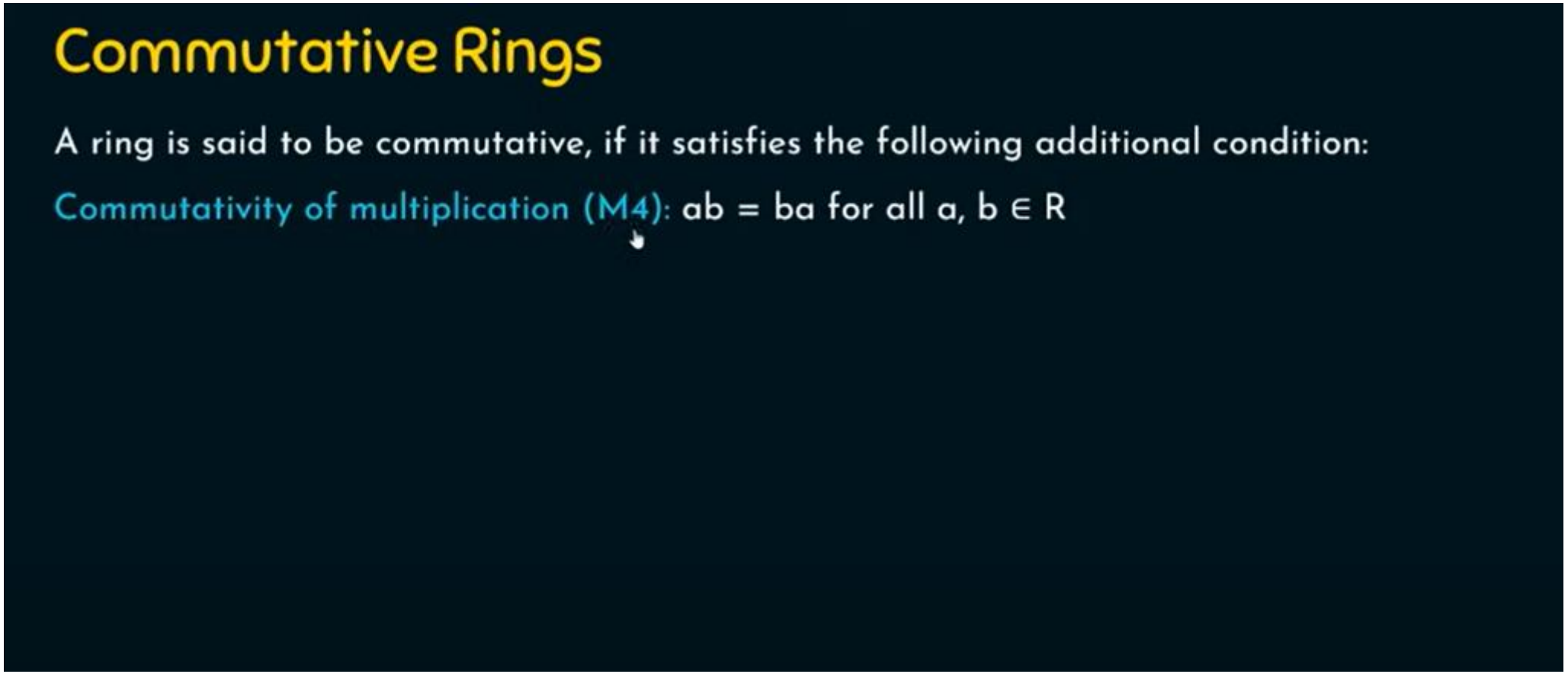
$$a(b + c) = ab + ac \quad \text{for all } a, b, c \in R$$

$$(a + b)c = ac + bc \quad \text{for all } a, b, c \in R$$

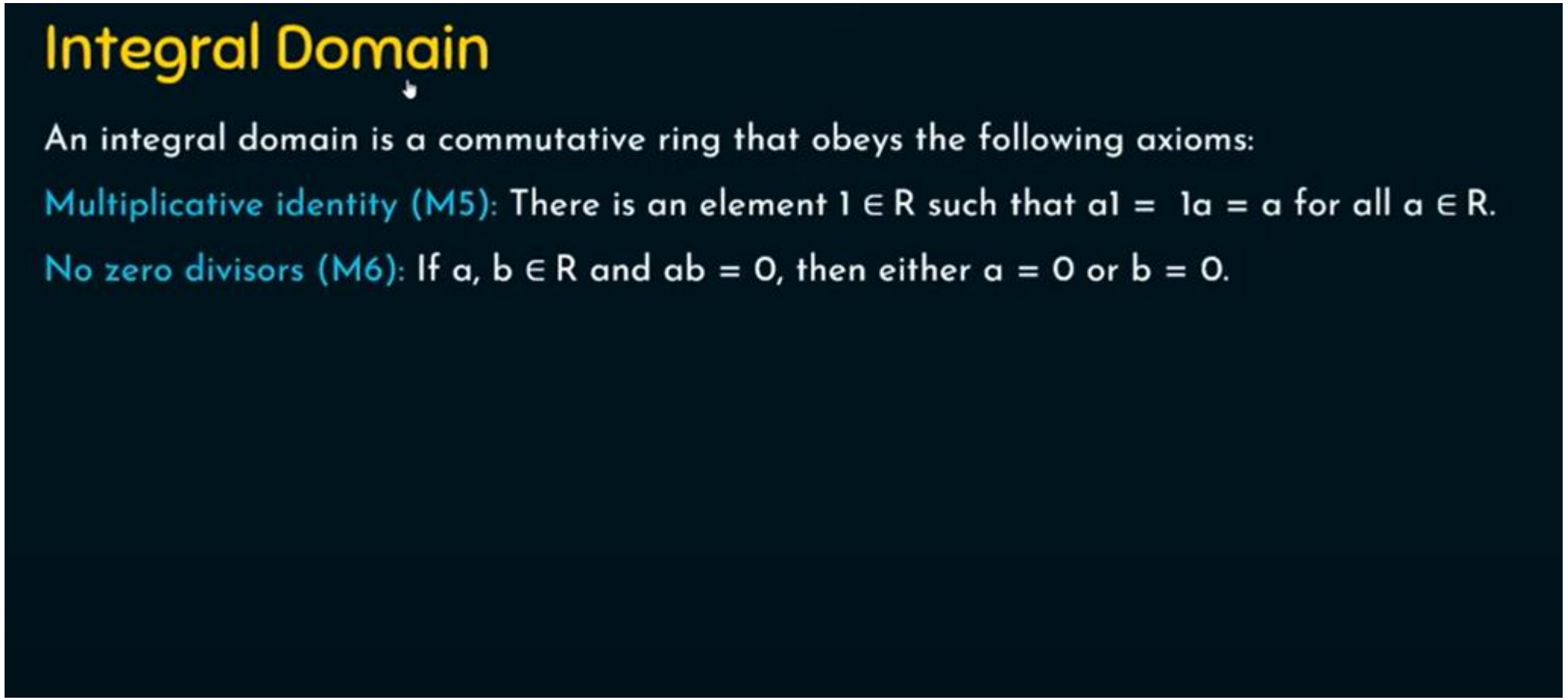
Note:

Subtraction [$a - b = a + (-b)$]

Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

Fields

A field F , sometimes denoted by {F, +,* }, is a set of elements with two binary operations, called addition and multiplication, such that for all a, b, c ∈ F the following axioms are obeyed:

(A1–M6): F is an integral domain; that is, F satisfies axioms A1 - A5 and M1 - M6.

(M7) Multiplicative inverse: For each a in F, except 0, there is an element a^{-1} in F such that

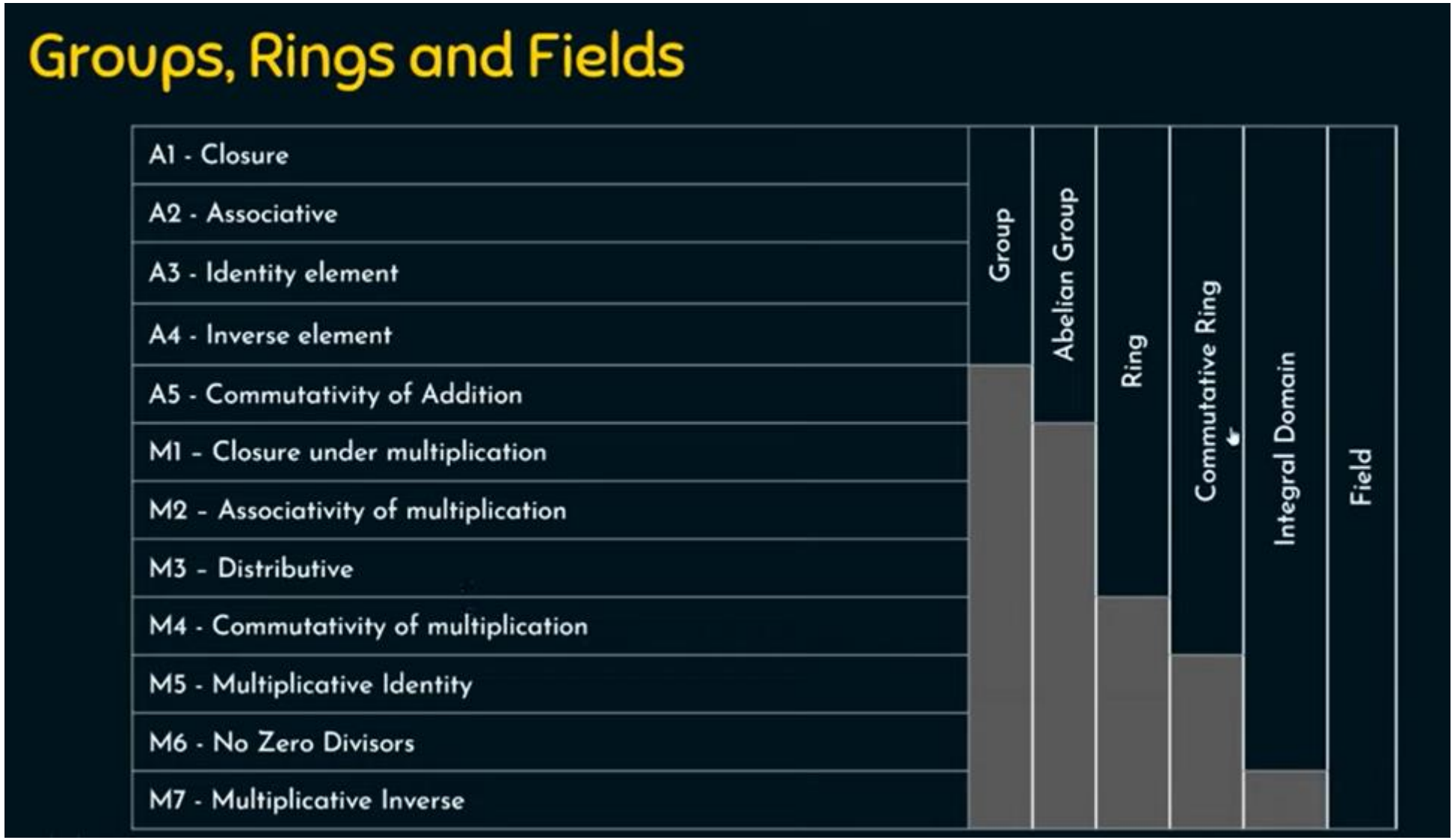
$$aa^{-1} = (a^{-1})a = 1$$

Note: $a/b = a(b^{-1})$.

Familiar examples of Fields:

- Rational numbers
- Real numbers
- Complex numbers

Basic Concepts in Number Theory and Finite Fields



Basic Concepts in Number Theory and Finite Fields

Finite Fields

- A finite field or Galois field (so-named in honor of Évariste Galois) 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.
- The most common examples of finite fields are given by the integers (mod p) when p is a prime number.

Application areas:

- Mathematics and computer science - Number theory, Algebraic geometry, Galois theory, Finite geometry, Cryptography and Coding theory.

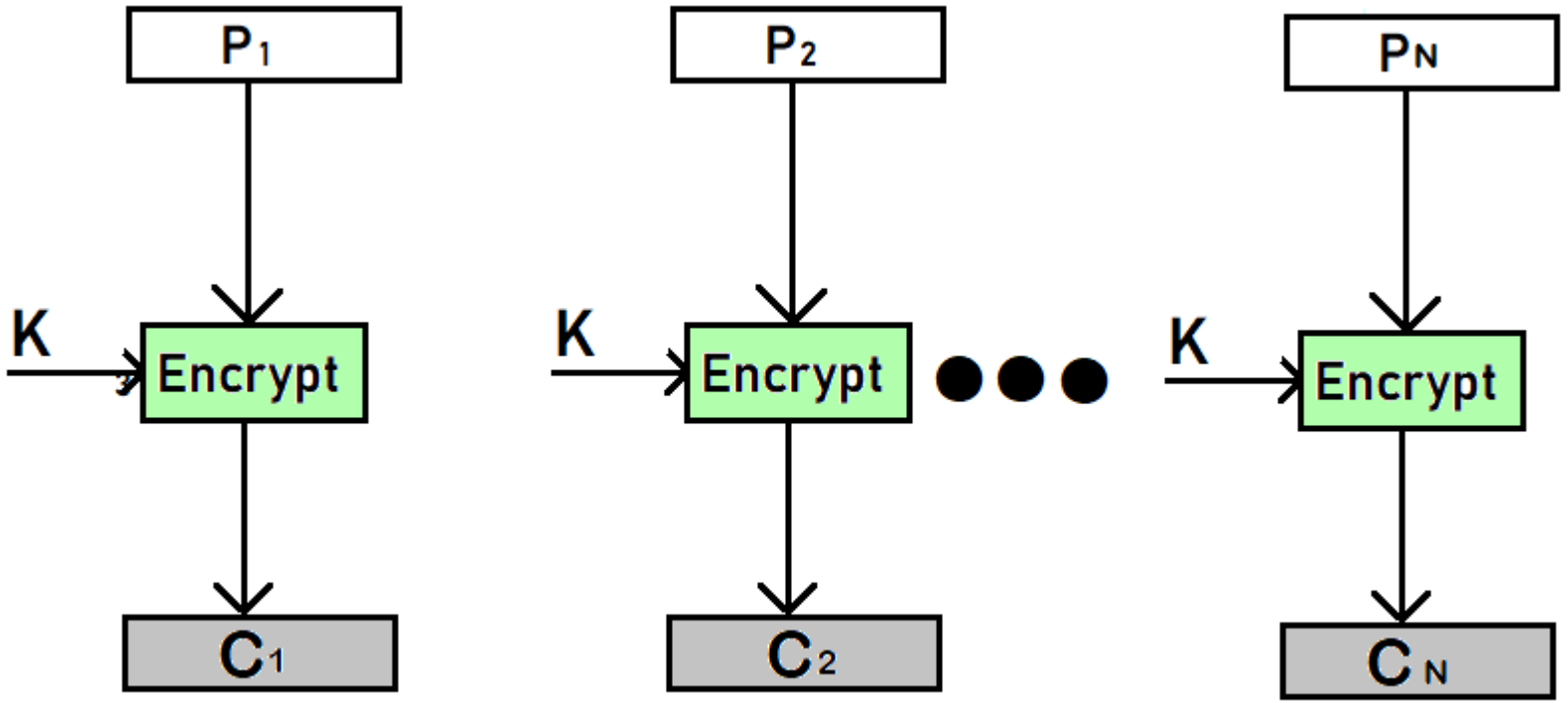
Block Cipher Operation Modes

- **Block Cipher Operation Modes:** Operation modes determine how blocks of plaintext are encrypted.
- **Common Modes:**
  1. **Electronic Codebook (ECB):**
  2. **Cipher Block Chaining (CBC):**
  3. **Cipher Feedback (CFB):**
  4. **Output Feedback (OFB):** Similar to CFB but pre-generates keystream blocks.
  5. **Counter (CTR) Mode:** Uses a counter that increments with each block, allowing parallel encryption and decryption.

Block Cipher Operation Modes

Electronic Codebook (ECB):

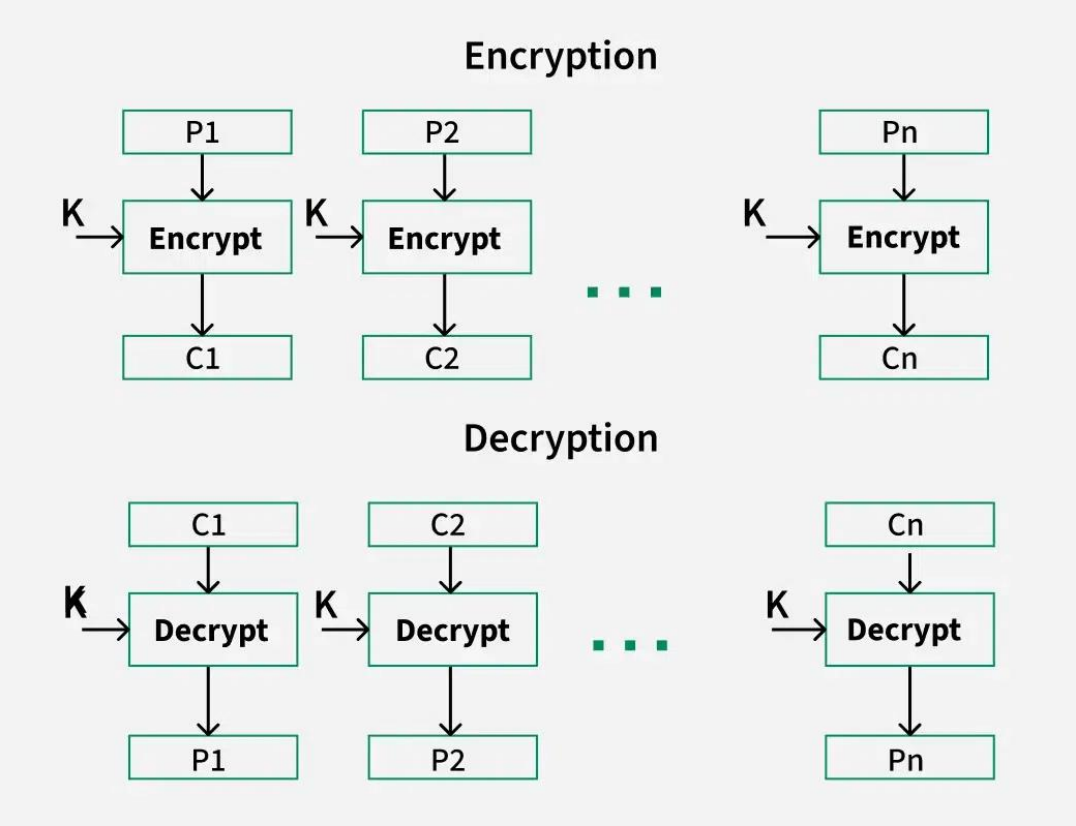
1. **Description:** Each block is encrypted independently.
2. **Weakness:** Identical plaintext blocks produce identical ciphertexts.



Block Cipher Operation Modes

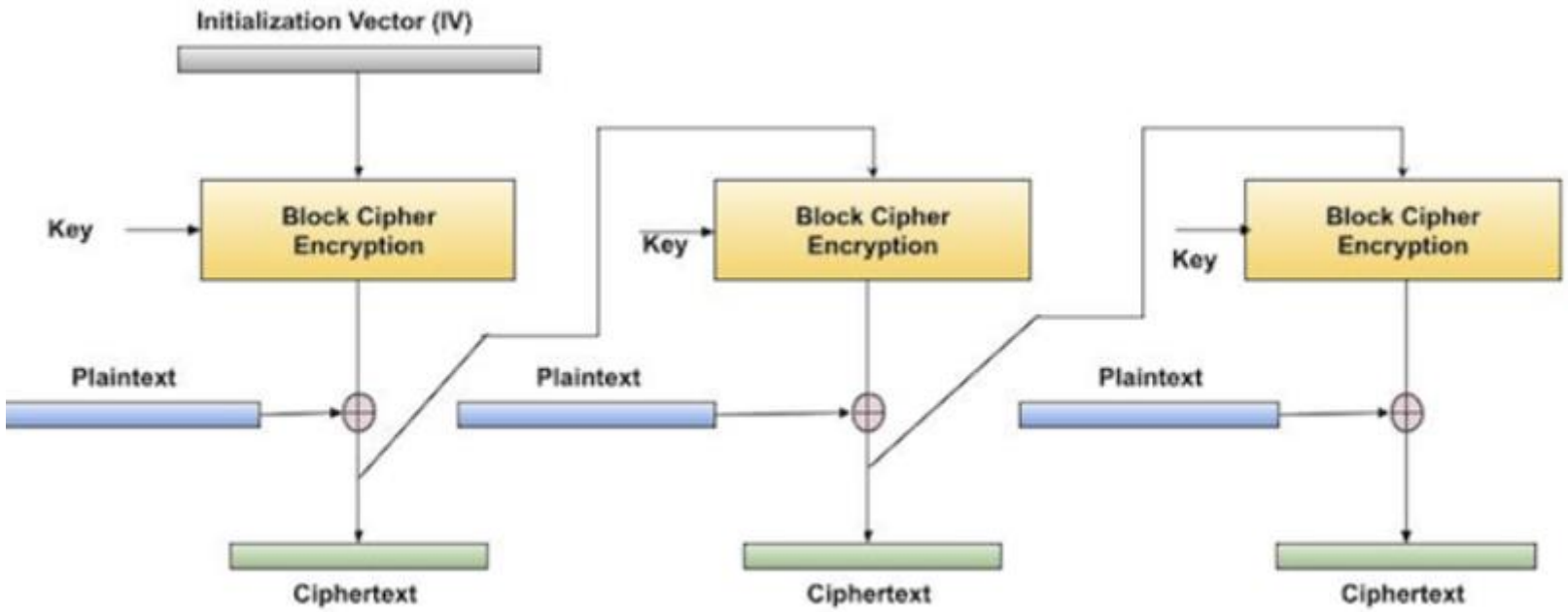
Cipher Block Chaining (CBC):

1. **Description:** Each plaintext block is XORed with the previous ciphertext block before encryption.
2. **Advantage:** Introduces randomness, preventing patterns.



Block Cipher Operation Modes

Cipher Feedback (CFB): Converts a block cipher into a stream cipher, suitable for streaming data.



Cipher Feedback (CFB) Mode Encryption

Asymmetric Ciphers

Public-key cryptography, also known as asymmetric cryptography, is a cryptographic system that uses a pair of keys:

- **Public Key:** Shared openly and used for encryption.
- **Private Key:** Kept secret and used for decryption.

- **Key Features:**
  - **Two-Key Mechanism:** One key encrypts, the other decrypts.
  - **Non-Repudiation:** Provides proof of the origin of data through digital signatures.
  - **Secure Key Distribution:** Eliminates the need to share secret keys over insecure channels.
- **Applications:**
  - **Secure Communication:** SSL/TLS for secure web browsing.
  - **Digital Signatures:** Ensuring authenticity and integrity of digital documents.
  - **Cryptocurrencies:** Used in Bitcoin and blockchain technologies.
- **Advantages:**
  1. **Enhanced Security:** No need to share private keys.
  2. **Scalability:** Easy to manage in large networks.
  3. **Authentication:** Ensures message authenticity and integrity.
- **Disadvantages:**
  1. **Slower than Symmetric Encryption:** Due to complex mathematical operations.
  2. **Key Length:** Requires longer keys for equivalent security compared to symmetric algorithms.

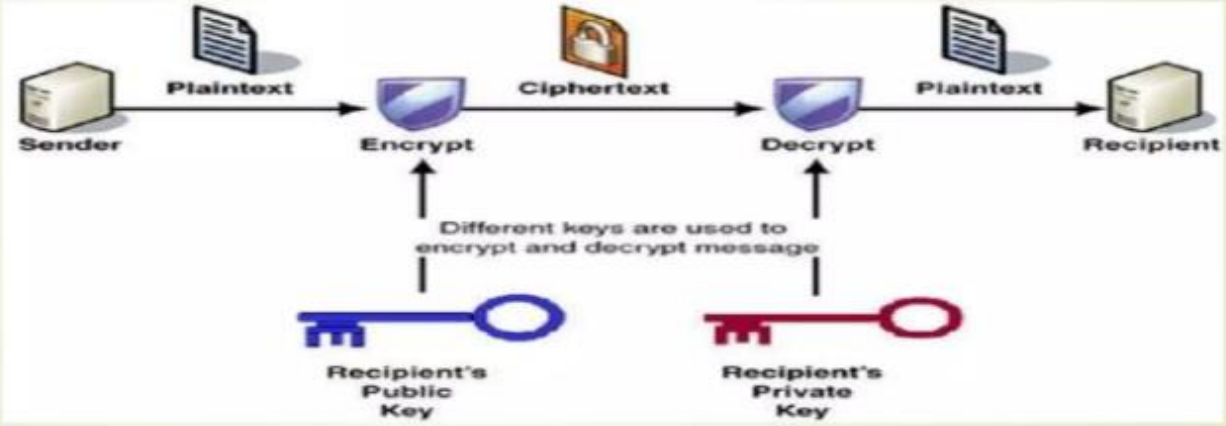
RSA Algorithm

- The **RSA algorithm** (named after Rivest, Shamir, and Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission.
- **Key Concepts:**
  - Based on the mathematical difficulty of **factoring large prime numbers**.
  - Involves **modular arithmetic** and **Euler's theorem**.
- **Key Generation:**
  1. Choose two large prime numbers, p and q.
  2. Compute n=p×q (modulus).
  3. Calculate ϕ(n)=(p−1)×(q−1) (Euler's totient function).
  4. Select public exponent e such that 1<e<ϕ(n) and gcd(e,ϕ(n))=1.
  5. Determine private key d such that d×e≡1mod ϕ(n).
- **Encryption:** Ciphertext $C = M^e \ mod \ n$, where M is the plaintext.
- **Decryption:** Plaintext $M = C^d \ mod \ n$.
- **Applications:**
  - Secure email encryption
  - Digital signatures for document verification.
- **Security:**
  - Based on the difficulty of factoring large numbers.
  - Vulnerable to quantum attacks (future concern).

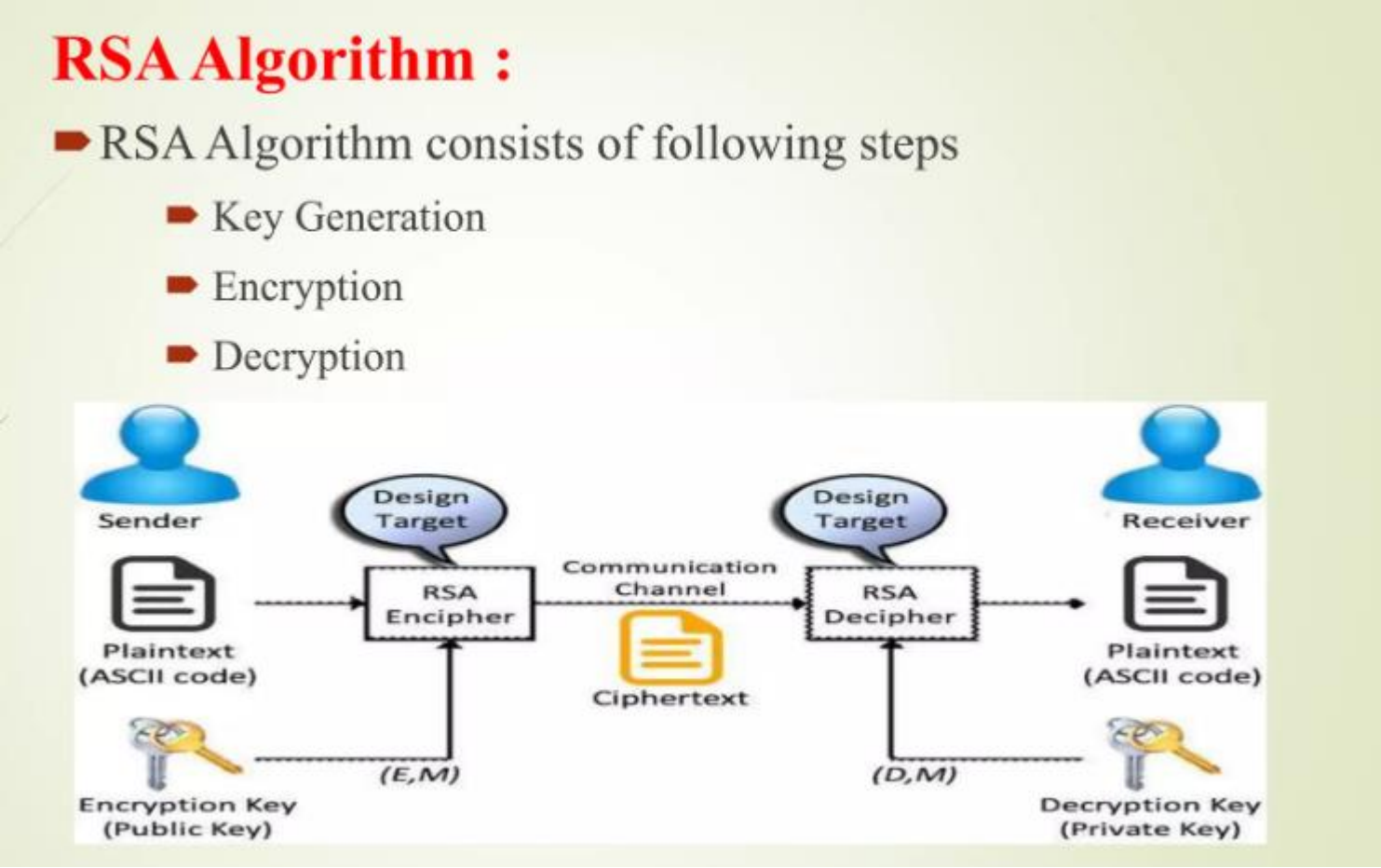
RSA Algorithm Example

RSA Algorithm :

- RSA is invented by Rivest, Shamir and Adleman of MIT.
- It is most widely used for secure data transmission.
- RSA Algorithm is known as Public key Cryptography.



RSA Algorithm Example



RSA Algorithm Example

Key Generation :

1. Select two prime numbers p & q
2. Calculate $n = p * q$
3. Calculate $m = \Phi(n) = (p - 1) * (q - 1)$
4. Choose a small number e , co prime to m , with
   $GCD(\Phi(n), e) = 1$, $1 < e < \Phi(n)$
5. Calculate $de \bmod \Phi(n) = 1$

6. d=((Ø(n)*i)+1)/e

RSA Algorithm Example

- **Encryption :**

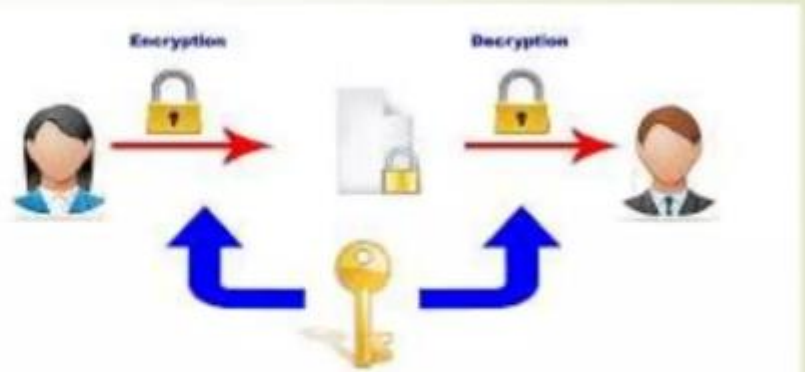
  Plaintext, $M < n$

  $C = M^e \bmod n$

- **Decryption :**

  Ciphertext, C

  $M = C^d \bmod n$



RSA Algorithm Example

RSA Algorithm Example :

1. Let p = 3, q = 11
2. n = p * q = 3 * 11 = 33
3. Φ (n) = (p-1)*(q-1) = 20
4. Let e =7 such that 1 < 7 < 20 , GCD (7, 20) = 1
5. ed = 1 mod Φ (n)

   7 * d = 1 mod 20

   So 7 * d mod 20 = 1

   - 7 * 3 mod 20 = 1
   - Here d = 3

RSA Algorithm Example

Encryption :

Plaintext, $M < e$

$C = M^e \bmod n$

Let $M = 31$

Then , $C = 3^{17} \bmod 33$

We get $C = 4$

Decryption :

Ciphertext, C

$M = C^d \bmod n$

$M = 4^3 \bmod 33$

$M = 31$

Diffie-Hellman algorithm

- The Diffie-Hellman algorithm is being used to establish a shared secret that can be used for secret communications while exchanging data over a public network using the elliptic curve to generate points and get the secret key using the parameters.
- For the sake of simplicity and practical implementation of the algorithm, we will consider only 4 variables, one prime P and G (a primitive root of P) and two private values a and b.
- P and G are both publicly available numbers. Users (say Alice and Bob) pick private values a and b and they generate a key and exchange it publicly. The opposite person receives the key and that generates a secret key, after which they have the same secret key to encrypt.

Input : 7 Output : Smallest primitive root = 3 Explanation: n = 7 3^0(mod 7) = 1 3^1(mod 7) = 3 3^2(mod 7) = 2 3^3(mod 7) = 6 3^4(mod 7) = 4 3^5(

Diffie-Hellman Setup

- **Primitive root of a prime number n modulo n**
- Given a prime number n, the task is to find its primitive root under modulo n. The primitive root of a prime number n is an integer r between[1, n-1] such that the values of r^x(mod n) where x is in the range[0, n-2] are different.

Input : 7 Output : Smallest primitive root = 3 Explanation: n = 7 3^0(mod 7) = 1 3^1(mod 7) = 3 3^2(mod 7) = 2 3^3(mod 7) = 6 3^4(mod 7) = 4 3^5(

Diffie-Hellman Setup

```
Input : 7
Output : Smallest primitive root = 3
Explanation: n = 7
3^0(mod 7) = 1
3^1(mod 7) = 3
3^2(mod 7) = 2
3^3(mod 7) = 6
3^4(mod 7) = 4
3^5(mod 7) = 5
```

Diffie-Hellman algorithm

Alice	Bob
Public Keys available = P, G	Public Keys available = P, G
Private Key Selected = a	Private Key Selected = b
Key generated = $x = G^a mod P$	Key generated = $y = G^b mod P$

Diffie-Hellman algorithm

Exchange of generated keys takes place	
Key received = y	key received = x
Generated Secret Key = $k_a = y^a mod P$	Generated Secret Key = $k_b = x^b mod P$
Algebraically, it can be shown that $k_a = k_b$	
Users now have a symmetric secret key to encrypt	

Diffie-Hellman algorithm

Example:

```
Step 1: Alice and Bob get public numbers P = 23, G = 9
Step 2: Alice selected a private key a = 4 and
        Bob selected a private key b = 3
Step 3: Alice and Bob compute public values
Alice:    x =(9^4 mod 23) = (6561 mod 23) = 6
        Bob:    y = (9^3 mod 23) = (729 mod 23)  = 16
Step 4: Alice and Bob exchange public numbers
Step 5: Alice receives public key y =16 and
        Bob receives public key x = 6
Step 6: Alice and Bob compute symmetric keys
        Alice:  ka = y^a mod p = 65536 mod 23 = 9
        Bob:    kb = x^b mod p = 216 mod 23 = 9
Step 7: 9 is the shared secret.
```

Introduction of ElGamal Encryption Algo.

1. ElGamal encryption is a public-key cryptosystem

1. ElGamal Algo. uses asymmetric key encryption for communicating between two parties and encrypting the message.

1. This cryptosystem is based on the difficulty of finding discrete logarithm in a cyclic group

1. It is based on the Diffie–Hellman key exchange And It was described by Taher Elgamal in 1985.

Components of ElGamal Encryption Algo.

ElGamal encryption consists of three components

1. Key Generation
2. Encryption
3. Decryption

Step 1 : Key Generation

Receiver Generates public and private keys.

- Select Large Prime No. (P)
- Select Decryption key/ private Key (D)
  gcd(D,P)=1
- Select Second part of Encryption key or public key (E1) & gcd(E1,P)=1
- Third part of the encryption key or public key (E2)
  $E2 = E1^{D} \bmod P$
- Public Key=(E1, E2, P) & Private key=D

Suppose :

1. P=11 , D=3, E1=2
2. Then E2= $2^3 \bmod 11=8$
3. Public key=(2, 8, 11) & Private key= 3.

Step 2 : Encryption

Sender Encrypts Data (PT) Using Receiver's Public Key

- Select Random Integer (R)
- $C1 = E1^R \bmod P$
- $C2 = (PT \times E2^R) \bmod P$
- C. T. =(C1,C2)

Continuous :-

1. $R=4, C1=2^4 \bmod 11=5, \ PT=7$
2. $C2=(7 \times 8^4) \bmod 11=6$
3. C.T. = (5,6)

Step 3 : Decryption

Receiver End Decrypts the Message

- $PT=[C2 \times (C1^{D})^{-1}] \bmod P$

Continuous :-

- $PT= (6 \times (5^{3}) ^{-1}) \bmod 11$
  $= 18 \bmod 11 = 7$

Quantum Cryptography: A New Era of Secure Communication

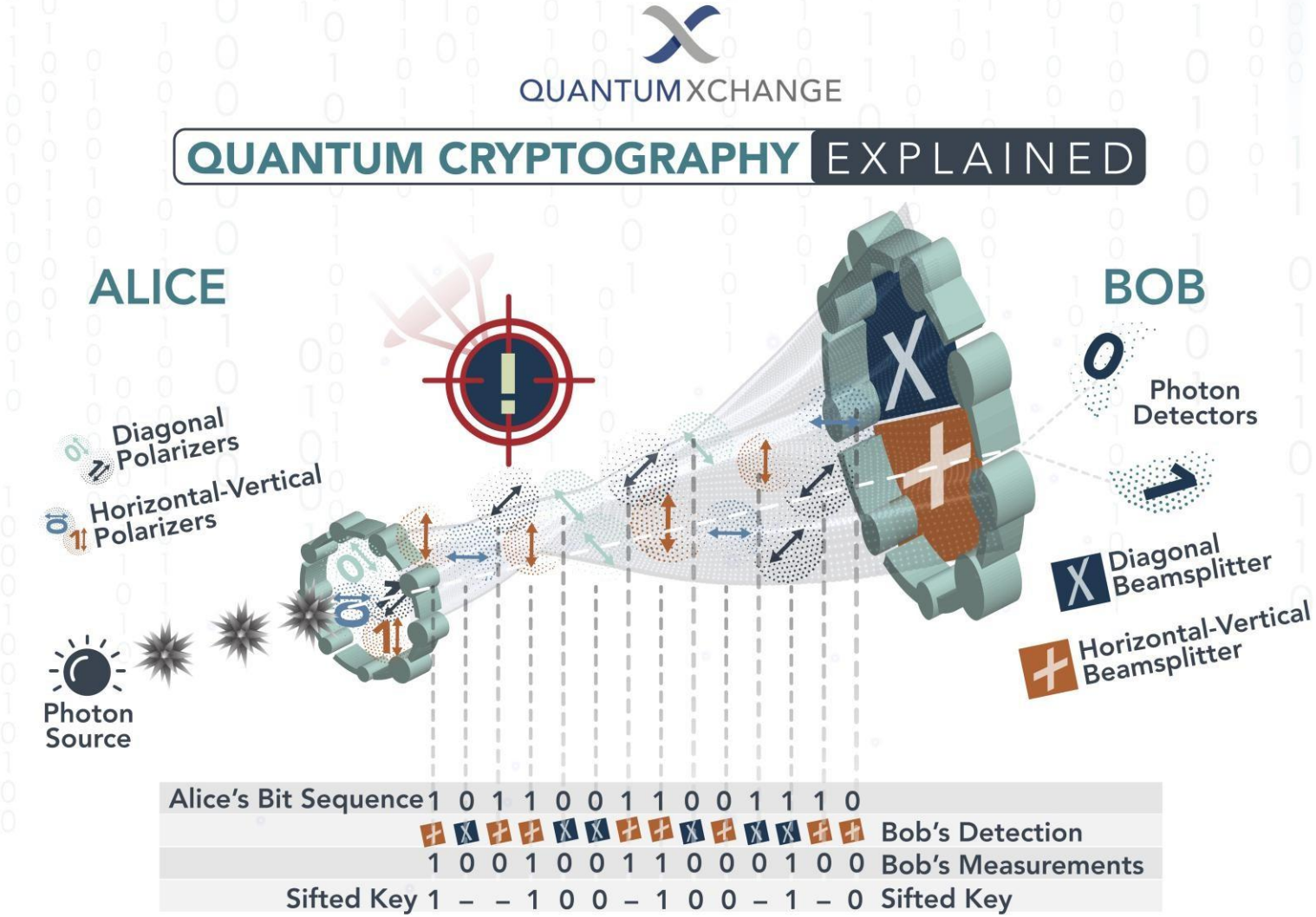
- **Introduction to Quantum Cryptography:** Quantum Cryptography leverages the principles of **quantum mechanics** to enable highly secure communication. Unlike classical cryptography, which relies on complex mathematical problems, quantum cryptography ensures security through the laws of physics.
- **Why Quantum Cryptography?**
  - Traditional encryption methods like RSA and ECC may become **vulnerable to quantum computers**.
  - Quantum cryptography offers **unbreakable security** due to its reliance on fundamental quantum properties.

Quantum Cryptography: A New Era of Secure Communication

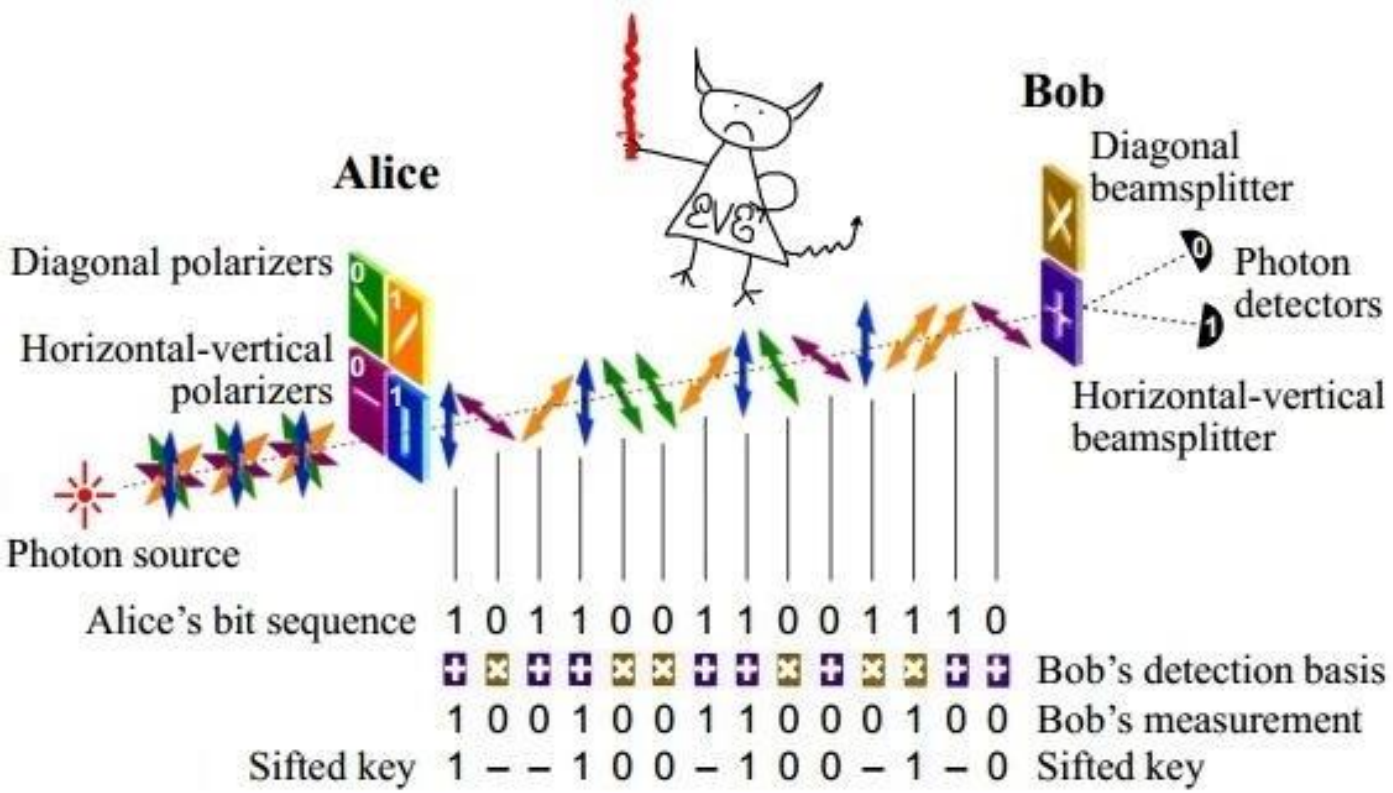
- **Quantum Key Distribution (QKD):** Quantum Key Distribution (QKD) enables two parties to generate and share a secret encryption key securely.
- **Why QKD is Secure?**
  - **No-Cloning Theorem:** Quantum states cannot be copied without detection.
  - **Observation Causes Disturbance:** Any eavesdropper attempting to measure qubits alters their state, making interception impossible.

- The sender transmits photons through a filter (or polarizer) which randomly gives them one of four possible polarizations and bit designations:
  - Vertical (One bit),
  - Horizontal (Zero bit),
  - 45 degree right (One bit), or
  - 45 degree left (Zero bit).
- The photons travel to a receiver, which uses two beam splitters (horizontal/vertical and diagonal) to “read” the polarization of each photon. The receiver does not know which beam splitter to use for each photon and has to guess which one to use.
- Once the stream of photons has been sent, the receiver tells the sender which beam splitter was used for each of the photons in the sequence they were sent, and the sender compares that information with the sequence of polarizers used to send the key. The photons that were read using the wrong beam splitter are discarded, and the resulting sequence of bits becomes the key.

Quantum Cryptography



Quantum Cryptography



- If an eavesdropper, named Eve, tries to listen in on the conversation, she has to read each photon to read the secret. Then she must pass that photon on to Bob. By reading the photon, Eve alters the photon's quantum state, which introduces errors into the quantum key. This alerts Alice and Bob that someone is listening and the key has been compromised, so they discard the key. Alice has to send Bob a new key that isn't compromised.

How Quantum Cryptography Works

- Quantum Key Cryptography uses quantum mechanics (that explains the aspect of nature at small (atomic and subatomic) scales) to guarantee secure communication.
- It enables two parties to produce a shared random secret key known only to them
  - The key can then be used to encrypt and decrypt messages.
- Quantum cryptography, or quantum key distribution (QKD), uses a series of photons (light particles) to transmit data from one location to another over a fiber optic cable

Working of BB84 Protocol

- BB84 is a quantum key distribution scheme developed by Charles Bennett and Gilles Brassard in 1984.
  - It is the first quantum cryptography protocol.
- Alice uses a light source to create a photon.
- The photon is sent through a polarizer and randomly given one of four possible polarization and bit designations-Vertical (One bit), Horizontal (Zero bit),45 degree right (One bit),or 45 degree left (Zero).
- The photon travels to Bob's location, where Bob has two beamsplitters -- a diagonal and vertical/horizontal - and two photon detectors.
- Bob randomly chooses one of the two beamsplitters and checks the photon detectors.
- The process is repeated until the entire key has been transmitted to Bob.
- Bob then tells Alice in sequence which beamsplitter he used.
- Alice compares this information with the sequence of polarizers she used to send the key.
- Alice tells Bob where in the sequence of sent photons he used the right beamsplitter.
- Bboth Alice and Bob have a sequence of bits (sifted key) they both know.

Implementation Challenges and Real-World Applications

- **Challenges:**
  - **Photon Loss:** Signal degradation over long distances.
  - **Quantum Noise:** Measurement errors due to environmental factors.
  - **Hardware Limitations:** Requires specialized quantum hardware.
- **Real-World Applications:**
  - **Secure Banking Transactions:** Implemented in financial sectors to protect sensitive data.
  - **Government Communications:** Used for secure military and diplomatic channels.
  - **Quantum Networks:** China's **Micius satellite** demonstrated the first global QKD communication.

Quantum Physics and Photon Reception

- **Quantum States of Photons:**
  - **Polarization Basis:** Used to encode quantum information.
  - **Quantum Bit (Qubit):** Represented by the polarization state of a single photon.
- **Measurement Techniques and Photon Polarization**
  1. **Linear Polarization:** Photons aligned horizontally or vertically.
  2. **Circular Polarization:** Rotating polarization states used for secure transmission.
  3. **Quantum Eraser Experiment:** Demonstrates that quantum information changes based on measurement choices.

Cryptography with Photons

- **Secure Communication Using Quantum Channels**
  - **Quantum Repeaters:** Enhance long-distance QKD without classical relays.
  - **Quantum Teleportation:** Securely transfers quantum information over distances.
- **Photon-Based Encryption Techniques**
  - **Quantum One-Time Pad:** Uses QKD-generated keys to encrypt messages unconditionally.
  - **Quantum Secure Direct Communication (QSDC):** Eliminates the need for post-key exchange encryption.

Implementation of Quantum Cryptographic Systems

- **Hardware Requirements (Quantum Sources, Detectors)**
  - **Quantum Random Number Generators (QRNGs):** Generate truly random keys.
  - **Single-Photon Detectors:** Essential for accurately measuring quantum states.
- **Practical Limitations and Future Developments**
  1. **Scalability Issues:** Quantum systems are expensive and require specialized infrastructure.
  2. **Quantum Computing Threat:** Future quantum computers may break existing encryption, necessitating **post-quantum cryptography**.
  3. **Integration with Classical Networks:** Research continues on **hybrid quantum-classical encryption**.

Conclusion: The Future of Quantum Cryptography

- **Unbreakable Security:** Quantum cryptography provides a **physically secure** alternative to mathematical encryption.
- **Global Adoption:** Countries and organizations are investing in **quantum-safe technologies**.
- **Next Steps:** Research is progressing in **quantum internet**, where secure global communication is based entirely on quantum principles.

Introduction to Hash Functions

- A **hash function** is a cryptographic algorithm that takes an input (message) and produces a **fixed-length hash value (digest)**. This output uniquely represents the original data in a condensed form.
- **Mathematical Representation:** H(M)=h where:
  - M = Input message
  - h = Hash output (digest)
  - H = Hash function
- **Key Features of Cryptographic Hash Functions:**
  1. **Deterministic**: The same input always produces the same output.
  2. **Fixed Output Size**: Regardless of input size, the hash output has a fixed length (e.g., 256-bit for SHA-256).
  3. **Fast Computation**: Hashing should be computationally efficient.
  4. **Irreversible (One-Way Function)**: It should be computationally infeasible to derive the original input from the hash.
  5. **Unique (Collision Resistance)**: Different inputs should produce different hash outputs.

Properties of Cryptographic Hash Functions

A secure cryptographic hash function must satisfy the following properties:

1. Pre-Image Resistance (One-Way Property)

- Given a hash value h, it should be computationally infeasible to find an input M such that: H(M)=h
- **Purpose:** Prevents attackers from retrieving original data (used in password storage).
- **Security Concern:** If a hash function is not pre-image resistant, brute-force attacks can reveal the input.

- **2. Collision Resistance**
  - It should be infeasible to find two distinct inputs M1 and M2 that produce the same hash output: H(M1)=H(M2)
  - **Purpose:** Ensures data integrity and prevents tampering.
  - **Security Concern:** If a collision exists, an attacker can replace data with malicious content having the same hash.

Properties of Cryptographic Hash Functions

- **3. Avalanche Effect**
- A small change in input should produce a **completely different** hash output.
- **Example:**

H("Hello") = 2CF24DBA5FB0A30E26E83B2AC5BCD773

H("hello") = 5D41402ABC4B2A76B9719D911017C592

- The change in only **one letter** (H → h) produces a vastly different hash.

Role of Hash Functions in Security

Cryptographic hash functions are used to ensure **data integrity** and **security** in various applications:

A. Data Integrity

- Hashing ensures that data remains unaltered by comparing hash values before and after transmission.
- Used in **checksums, file verification, and digital signatures**.

B. Digital Signatures

- Hashing is used in signing documents and verifying authenticity.
- A sender signs a document by hashing it and encrypting the hash with their private key.
- The recipient verifies it by decrypting the hash and comparing it to the computed hash of the received document.

Common Cryptographic Hash Algorithms

Algorithm	Output Size	Security Status
MD5	128-bit	Broken (Collision Attacks)
SHA-1	160-bit	Broken (Practical Collisions Found)
SHA-2	224, 256, 384, 512-bit	Secure (Recommended)
SHA-3	224, 256, 384, 512-bit	More Secure than SHA-2

Common Cryptographic Hash Algorithms

- **A. MD5 (Message Digest Algorithm 5)**
  - **Developed by**: Ronald Rivest (1991)
  - **Key Features**:
    - Produces a 128-bit hash.
    - Fast but vulnerable to collision attacks.
  - **Security Issue**: Collisions can be generated in seconds, making it insecure for cryptographic use.
  - **Usage Today**: Used in file integrity verification but **not recommended for security applications**.
- **B. SHA-1 (Secure Hash Algorithm 1)**
  - **Developed by**: NSA (1995)
  - **Key Features**:
    - Produces a 160-bit hash.
    - Used in SSL, TLS, and digital certificates.
  - **Security Issue**: In 2017, Google demonstrated a practical **collision attack** (SHA-1 shattered attack).
  - **Usage Today**: Deprecated and replaced by SHA-2.

Common Cryptographic Hash Algorithms

- **C. SHA-2 (Secure Hash Algorithm 2)**
  - **Developed by**: NSA (2001)
  - **Key Features**:
    - Provides **multiple hash sizes**: SHA-224, SHA-256, SHA-384, SHA-512.
    - Resistant to collision attacks.
  - **Usage Today**: Widely used in **TLS, SSL, cryptocurrencies (Bitcoin uses SHA-256), and digital signatures**.
- **D. SHA-3 (Secure Hash Algorithm 3)**
  - **Developed by**: NIST (2015)
  - **Key Features**:
    - Uses **Keccak algorithm**, making it more resistant to attacks.
    - Has variants SHA3-224, SHA3-256, SHA3-384, SHA3-512.
  - **Usage Today**: Recommended for post-quantum security applications.

Applications of Hash Functions

- **A. Password Storage**
- Hashing is used to store passwords securely.
- Example:
  - A password "mysecurepassword" is stored as: H(mysecurepassword) = 5F4DCC3B5AA765D61D8327DEB882CF99
  - To protect against **rainbow table attacks**, a **random salt** is added before hashing: H("mysecurepassword" + salt) = 8D969EEF6ECAD3C29A3A629280E686CF

Applications of Hash Functions

- **B. Data Verification**
- Hashing is used to verify **file integrity**.
- Example:
  - A file’s hash is computed before transmission.
  - The receiver recomputes the hash after downloading.
  - If both hashes match, the file was **not tampered** with.
- **C. Blockchain and Cryptocurrencies**
- **Proof of Work (PoW)**: Bitcoin uses **SHA-256** to create new blocks in the blockchain.
- **Immutable Records**: Every block in a blockchain contains the hash of the previous block, ensuring data integrity.

Applications of Hash Functions

- **Example: Bitcoin Block Hashing**
  - **Block 1 Hash: 0000XABCD1234...**
  - **Block 2 Hash: 0000EFG56789...**
- Any change in Block 1 will alter its hash, making **tampering impossible**.

Why Are Hash Functions Crucial?

- Provide **data integrity** by detecting unauthorized modifications.
- Used in **password security**, **digital signatures**, and **cryptocurrencies**.
- **SHA-3 is the future** due to its enhanced security over SHA-2.

Concept of Message Authentication

- **What is Message Authentication?**
  - Message Authentication ensures the **integrity** and **authenticity** of a message. It verifies that:
  - The message has **not been altered** during transmission.
  - The message is from the **intended sender** (authentic).

Concept of Message Authentication

- A **Message Authentication Code (MAC)** is a cryptographic checksum generated from a message and a secret key. It is used to:
- **Authenticate** the sender of the message.
- **Verify** the integrity of the message.
- **Mathematical Representation:**
- MAC=F(K,M)
- Where:
  - M = Message
  - K = Secret Key
  - F = MAC generation algorithm

Difference Between MACs and Hash Functions

Criteria	MAC (Message Authentication Code)	Hash Function
Key Requirement	Requires a **secret key**	**No key** required
Purpose	Ensures both **integrity** and **authenticity**	Ensures **integrity** only
Usage	Used in secure communications (e.g., APIs, banking)	Used for data verification (e.g., file checksums)
Security Level	Stronger security due to key-based verification	Weaker if not combined with a key

Key Point:

- **MAC = Hash Function + Secret Key** (in HMAC)
- MACs provide **authentication**, while hashes provide **integrity**.

Importance of MACs in Data Integrity and Authentication

- **Data Integrity:** Detects any unauthorized modifications to the message during transmission.
- **Authentication:** Ensures the message comes from a trusted sender because only the sender and receiver know the secret key.
- **Non-Repudiation (in certain cases):** Provides proof that the message was sent by a specific entity.
- **Real-World Applications:**
  - Secure email protocols (e.g., **SSL/TLS**).
  - Financial transactions (e.g., **banking APIs**).
  - Secure communication in **IoT devices**.

Types of MACs

A. HMAC (Hash-Based Message Authentication Code)

- HMAC is a MAC that uses a **cryptographic hash function** (like SHA-256) and a **secret key** to produce the authentication code.
- **Formula:**
- $HMAC(K,M)=H((K' \oplus opad) \,||\, H((K' \oplus ipad) \,||\, M))$
- Where:
  - K′ = Secret key (padded/truncated)
  - H = Hash function (e.g., SHA-256)
  - opad = Outer padding (0x5c)
  - ipad = Inner padding (0x36)
  - || = Concatenation operator
  - ⊕ = XOR operation

Types of MACs

- **How HMAC Works:**
  1. The secret key is **padded** to the block size of the hash function.
  2. It is combined with the **inner padding** and the message, then hashed.
  3. The result is combined with the **outer padding** and hashed again to produce the final HMAC.
- **Security Strength:**
  - Depends on the **strength of the hash function** (e.g., SHA-256).
  - Resistant to **length extension attacks** that affect plain hashes.

Types of MACs

B. CMAC (Cipher-Based Message Authentication Code)

- CMAC is a MAC that uses a **symmetric block cipher** (like AES) instead of a hash function.
- **Formula:**
- CMAC(K,M)=AES(K,M)⊕Subkey
- Where:
  - K = Symmetric secret key
  - M = Message
  - **Subkey** = Derived from the encryption process

Types of MACs

- **How CMAC Works:**
  1. The message is divided into blocks.
  2. Each block is processed using a **block cipher** (e.g., AES).
  3. The final block is XORed with a **subkey** to generate the CMAC.
- **Security Strength:**
  - **Strong security** due to reliance on AES (or similar block ciphers).
  - More efficient than HMAC when hardware-based encryption is available.

Working of HMAC and CMAC

Key Generation and Verification Processes

- **HMAC Generation:**
  1. **Key Agreement:** Both sender and receiver share a secret key K.
  2. **Message Input:** The sender applies HMAC using K and the message M.
  3. **HMAC Generation:** The output (digest) is appended to the message and sent to the receiver.
- **HMAC Verification:**
  1. The receiver uses the same key K to generate an HMAC for the received message.
  2. If the computed HMAC matches the received HMAC, the message is **authentic** and **unaltered**.

Working of HMAC and CMAC

- **CMAC Generation:**
  1. **Key Sharing:** A symmetric key KKK is securely shared.
  2. **Encryption Process:** The message is encrypted using AES, and a subkey is applied.
  3. **CMAC Output:** The final encrypted block acts as the MAC.
- **CMAC Verification:**
  1. The receiver re-encrypts the received message using the same key and subkey.
  2. If the computed CMAC matches the received CMAC, the message is **authenticated**.

Working of HMAC and CMAC

B. Security Considerations

- **HMAC Security:**
  - Strong if the **underlying hash function** (e.g., SHA-256) is secure.
  - Resistant to collision attacks due to the key-based double hashing process.
  - Vulnerable if the **key is weak** or poorly managed.
- **CMAC Security:**
  - Inherits the security of AES (or the underlying cipher).
  - Resistant to forgery if the key is kept secret.
  - Efficient in **hardware-based systems** (like embedded devices).

Practical Applications of HMAC and CMAC

Application Area	HMAC	CMAC
Network Security	SSL/TLS authentication	IPsec encryption
Data Integrity	Verifying software updates	Secure firmware validation
Banking & Finance	API authentication (OAuth)	Secure PIN verification
Cryptocurrencies	Blockchain transaction signing	Payment processing in EMV cards
IoT Devices	Device-to-cloud authentication	Lightweight encryption in smart devices

Why Are MACs Important?

- **MACs = Integrity + Authentication:** They ensure messages are **authentic** and **unchanged**.
- **HMAC** is versatile for software-based systems (secure APIs, emails).
- **CMAC** excels in hardware applications (smart cards, IoT devices).
- Both play a vital role in modern cryptography, securing **financial transactions**, **communications**, and **digital identities**.

What is a Digital Signature?

- A **Digital Signature** is a cryptographic technique used to:
  - **Verify the authenticity** of digital messages or documents.
  - **Ensure data integrity**, confirming that the content has not been altered.
  - **Provide non-repudiation**, meaning the sender cannot deny their involvement.
- It works using **asymmetric cryptography**, where:
  - A **private key** is used to **sign** the message.
  - A **public key** is used to **verify** the signature.

What is a Digital Signature?

- Purpose and Importance in Cybersecurity
  1. **Authentication:** Verifies the sender's identity.
  2. **Data Integrity:** Ensures the message hasn't been tampered with.
  3. **Non-Repudiation:** Prevents the sender from denying the origin of the message.
- **Real-World Use Cases:**
  - Secure emails (e.g., **PGP** encryption).
  - Legal documents and contracts.
  - Blockchain transactions.

Digital Signatures vs. Handwritten Signatures

Criteria	Digital Signature	Handwritten Signature
Technology	Based on cryptographic algorithms	Ink-based signature
Verification	Verified using public-key cryptography	Verified visually
Tamper Detection	Alteration invalidates the signature	Alterations can go unnoticed
Security Level	High (depends on encryption strength)	Low (can be forged easily)
Non-Repudiation	Strong, cryptographic proof	Weak, can be denied or disputed
Usage	Emails, software, digital contracts	Paper documents, physical agreements

Working of Digital Signature Algorithms

A. Signing and Verification Processes

Step 1: Signing Process (Sender Side)

1. **Hashing the Message:** The original message M is passed through a **hash function** (e.g., SHA-256) to create a unique digest H(M).
2. **Encrypting the Hash (Signing):** The hash is encrypted using the sender's **private key** to generate the **digital signature**: Signature=EncryptPrivate Key(H(M))
3. **Sending:** The sender sends both the **original message** and the **digital signature** to the receiver.

Working of Digital Signature Algorithms

Step 2: Verification Process (Receiver Side)

1. **Hashing the Received Message:** The receiver hashes the received message to generate H(M’).
2. **Decrypting the Signature:** The receiver decrypts the digital signature using the sender’s **public key** to retrieve the original hash: H(M)=DecryptPublic Key(Signature)
3. **Comparison:** The receiver compares H(M) and H(M’):
    - If they match, the message is **authentic** and **unaltered**.
    - If they don’t match, the message has been **tampered with** or the signature is **invalid**.

Key Generation, Hashing, and Encryption Integration

1. **Key Generation:** A pair of **asymmetric keys** is generated:
    1. **Private Key:** Used for signing.
    2. **Public Key:** Shared with others to verify signatures.
2. **Hashing:** A **cryptographic hash function** ensures data integrity.
3. **Encryption:** The hash is encrypted with the sender's private key to create the signature.

Common Digital Signature Algorithms

A. RSA Digital Signature Algorithm

- **Based on:** The difficulty of **factoring large prime numbers**.
- **Process:**
  - Hash the message using SHA-256 (or similar).
  - Encrypt the hash with the sender's **private RSA key**.
  - Verify by decrypting the signature with the sender's **public RSA key**.
- **Applications:** SSL/TLS certificates, software signing, secure emails.

Common Digital Signature Algorithms

B. DSA (Digital Signature Algorithm)

- **Based on: Discrete logarithm problem**.
- **Process:**
  - Generate a signature using a combination of private key and random numbers.
  - Verification involves mathematical operations with the public key and hash.
- **Faster** than RSA in signature generation but **slower** in verification.
- **Applications:** Government communications, secure protocols.

Common Digital Signature Algorithms

C. ECDSA (Elliptic Curve Digital Signature Algorithm)

- **Based on: Elliptic Curve Cryptography (ECC)**.
- **Advantages:**
  - **Smaller key sizes** with **stronger security**.
  - Faster signature generation and verification.
- **Applications:** Cryptocurrencies (e.g., Bitcoin), mobile security, IoT devices.

Applications of Digital Signatures

- **A. Secure Emails**
  - Ensures that emails are **authentic** and **unmodified**.
  - Tools: **PGP (Pretty Good Privacy)** and **S/MIME**.
- **B. Software Distribution: Code Signing Certificates** verify that software comes from a trusted source and hasn't been tampered with.
- **C. Blockchain Technology**
  - Used in **cryptocurrencies** (like Bitcoin) to sign transactions.
  - Ensures data integrity and prevents double-spending.

Legal Frameworks and Compliance Requirements

1. eIDAS Regulation (EU)

- **eIDAS (Electronic Identification, Authentication, and Trust Services)** is an EU regulation that standardizes electronic signatures.
- **Types of Electronic Signatures:**
  - **Simple Electronic Signature (SES)**
  - **Advanced Electronic Signature (AES)**
  - **Qualified Electronic Signature (QES):** Equivalent to handwritten signatures in legal contexts.

2. Other Global Standards

- **ESIGN Act (USA):** Legal recognition of electronic signatures.
- **UNCITRAL Model Law (International):** Framework for global electronic transactions.

The Importance of Digital Signatures

- **Secure Communication:** Verifies sender identity and message integrity.
- **Global Acceptance:** Legal recognition in many countries (eIDAS, ESIGN).
- **Future Trends:** Integration with **blockchain**, **IoT**, and **post-quantum cryptography**.

Conclusion

Key Takeaways

- **Symmetric Ciphers:** Fast encryption with a single key (e.g., DES, AES).
- **Asymmetric Ciphers:** Public-private key encryption for secure communication (e.g., RSA, ECC).
- **Quantum Cryptography:** Unbreakable security based on quantum mechanics (e.g., BB84 Protocol).
- **Hash Functions:** Ensuring data integrity (e.g., SHA-2, SHA-3).
- **Message Authentication Codes (MACs):** Verifying message authenticity (e.g., HMAC, CMAC).
- **Digital Signatures:** Providing non-repudiation and authentication (e.g., RSA, ECDSA).

Conclusion

Core Principles

- **Confidentiality:** Protecting data through encryption.
- **Integrity:** Ensuring data is unaltered using hashes and MACs.
- **Authentication:** Verifying sender identity with digital signatures.
- **Non-Repudiation:** Preventing denial of actions via cryptographic proofs.

Conclusion

Future Trends

- **Post-Quantum Cryptography:** Preparing for quantum threats.
- **Blockchain Security:** Leveraging cryptography for decentralized applications.
- **Zero Trust Security:** Continuous verification in dynamic environments.