Tribhuvan University
Institute of Science and Technology
2076

Bachelor Level / Fifth Semester / Science
Computer Science and Information Technology (CSC316)
(Cryptography)
Full Marks: 60 + 20 + 20 Pass Marks: 24 + 8 + 8 Time: 3 Hours

Candidates are required to give their answers in their own words as far as practicable.

The figures in the margin indicate full marks.

Section A
Attempt any TWO questions.

Which monoalphabetic and polyalphabetic cipher is more vulnerable? Justify your statement. Which types of keys are considered weak keys in DES? Explain the round operation in IDEA.

Monoalphabetic vs Polyalphabetic Ciphers

Monoalphabetic ciphers are more vulnerable to cryptanalysis compared to polyalphabetic ciphers. Here's why:

- Monoalphabetic ciphers use a **single substitution throughout** the **entire message**. This means that each **plaintext** letter is always replaced by the same **ciphertext** letter. This creates patterns that can be exploited by **frequency analysis attacks**.
- Polyalphabetic ciphers, on the other hand, use **multiple substitutions.** The **same plaintext letter** can be replaced by **different ciphertext letters depending** on **its position** in the **message** or the **key used**. This makes it **harder** to **perform frequency analysis** and **breaks** the **patterns**.

Weak Keys in DES

In the Data Encryption Standard (DES), some keys are considered weak or semi-weak keys. These keys have the property of encrypting with the key and then decrypting with the same key (or vice versa), which results in the original plaintext. The weak keys in DES are:

- **0x0101010101010101**
- **0x FEFEFEFEFEFEFEFE**
- **0x E0E0E0E0F1F1F1F1**
- **0x 1F1F1F1F0E0E0E0E**

These keys should be avoided as they can lead to potential vulnerabilities if used in encryption.

Round Operation in IDEA

The International Data Encryption Algorithm (IDEA) is a symmetric-key block cipher that operates on 64-bit blocks using a 128-bit key. The encryption process consists of multiple rounds, with each round involving the following operations:

1. Addition modulo 216: Each sub-block is added to a corresponding subkey modulo 216.
2. Multiplication modulo 216 + 1: Two of the modified sub-blocks are multiplied together using modulo 216 + 1.
3. XOR Operation: The results of the addition and multiplication are combined using the XOR operation.

The specific steps for each round are:

- Inputs: Four 16-bit sub-blocks (P1, P2, P3, P4) and corresponding subkeys (K1, K2, K3, K4).
- Transformations:
  - P1' = (P1 + K1) mod 216
  - P2' = (P2 * K2) mod (216 + 1)
  - P3' = (P3 + K3) mod 216
  - P4' = (P4 * K4) mod (216 + 1)
-
- Rearrange the outputs and use them as inputs for the next round.

After the final round, a half-round is applied, which consists of a subset of the operations used in the full rounds. The final output is the ciphertext.

These round operations in IDEA, along with the use of a 128-bit key, contribute to the algorithm's strength and resistance against cryptanalysis.

Monoalphabetic vs. Polyalphabetic Ciphers

Monoalphabetic ciphers are more vulnerable to cryptanalysis compared to polyalphabetic ciphers.

1. What is the difference between polyalphabetic and monoalphabetic ciphers? - Typeset.io

typeset.io

- **Monoalphabetic ciphers** use a fixed substitution for each letter of the alphabet. This means that the frequency distribution of letters in the ciphertext directly reflects the frequency distribution of the plaintext language. This pattern can be exploited by cryptanalysts using frequency analysis.
- 1. Substitution cipher - Wikipedia
- en.wikipedia.org
- **Polyalphabetic ciphers** use multiple substitution alphabets, making the frequency analysis more complex. This makes it significantly harder for attackers to break the cipher.
- 1. Polyalphabetic Substitution Cipher - Cornell University
-
- pi.math.cornell.edu

Weak Keys in DES

DES (Data Encryption Standard) is a block cipher that uses a 56-bit key. While not all keys are weak, certain key patterns can make the cipher vulnerable to attacks.

Round Operation in IDEA

IDEA (International Data Encryption Algorithm) is a block cipher that operates on 64-bit blocks of data using a 128-bit key. The core of IDEA is its round function, which involves several mathematical operations.

1. What is the International Data Encryption Algorithm (IDEA)? - NordVPN

A single round in IDEA consists of the following steps:

1. **Key addition:** Four 16-bit sub-blocks of the data are added (modulo 2^16) to four corresponding subkeys.
2. **Multiplication:** The first and third sub-blocks are multiplied (modulo 2^16 + 1) with two subkeys.

3. **Addition:** The results of the multiplications are added (modulo 2^16) to the second and fourth sub-blocks.
4. **XOR:** The four sub-blocks are XORed with four subkeys.
5. **Swap:** The second and third sub-blocks are swapped.
6. 1. Action of Blowfish International Data Encryption Algorithm (IDEA) The... - ResearchGate
7. www.researchgate.net

These operations are repeated for multiple rounds, providing confusion and diffusion, which are essential cryptographic properties. IDEA also includes an output transformation after the final round.

Vulnerability of Monoalphabetic vs. Polyalphabetic Ciphers

Monoalphabetic Cipher:

- **Definition**: A monoalphabetic cipher uses a single alphabet to substitute plaintext characters with ciphertext characters. Each letter of the plaintext is mapped to a fixed letter of the ciphertext.
- **Vulnerability**: Monoalphabetic ciphers are highly vulnerable to frequency analysis attacks. Since each plaintext letter is always encrypted to the same ciphertext letter, the statistical frequency of letters in the plaintext is preserved in the ciphertext. An attacker can analyze the frequency of characters in the ciphertext and match them with the typical frequency of letters in the plaintext language (e.g., English).

Polyalphabetic Cipher:

- **Definition**: A polyalphabetic cipher uses multiple alphabets to encrypt the plaintext. The substitution of letters varies depending on their position in the plaintext and the key.
- **Vulnerability**: Polyalphabetic ciphers are more resistant to frequency analysis because the same plaintext letter can be encrypted to different ciphertext letters depending on its position and the key. This breaks the direct statistical relationship between plaintext and ciphertext letters.

Justification:

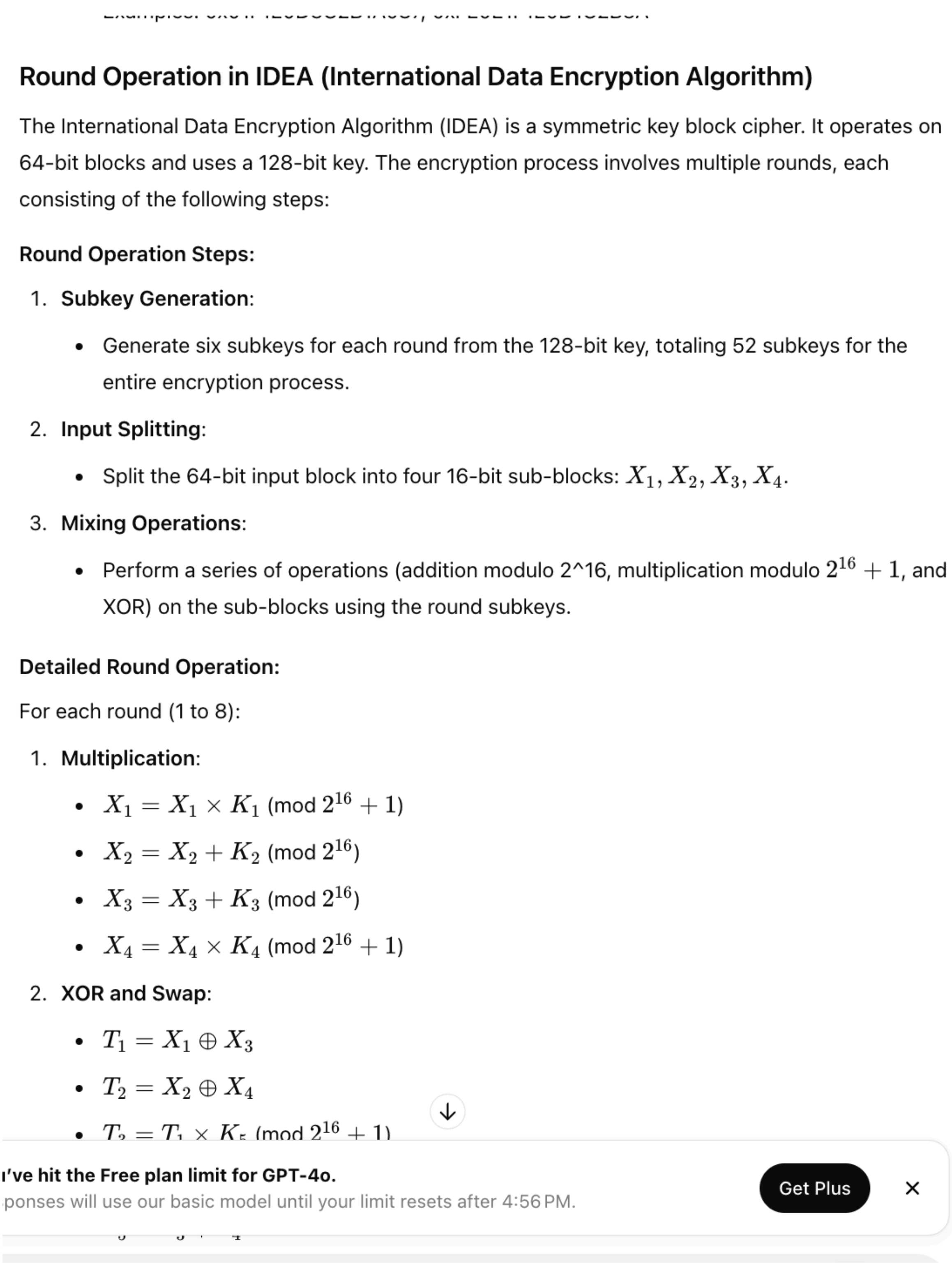
- **More Vulnerable**: Monoalphabetic ciphers are more vulnerable because they preserve the statistical characteristics of the plaintext in the ciphertext. Frequency analysis can easily crack these ciphers by exploiting the regular patterns in letter usage.
- **Less Vulnerable**: Polyalphabetic ciphers, like the Vigenère cipher, vary the substitution alphabet, making frequency analysis much more difficult. The

statistical patterns are obscured, providing stronger security against such attacks.

Weak Keys in DES

The Data Encryption Standard (DES) is a symmetric key algorithm used for encryption. Certain keys in DES are considered "weak" because they can result in encryption weaknesses. These include:

1. **Weak Keys**: These are keys that produce the same subkeys in each round of the DES algorithm, leading to potential vulnerabilities:
    - Examples: 0x0000000000000000,
    - 0xFFFFFFFFFFFFFFFF,
    - 0xE0E0E0E0F1F1F1F1
    - 0x1F1F1F1F0E0E0E0E
2. **Semi-weak Keys**: These keys come in pairs and encrypt plaintext to ciphertext in such a way that another key in the pair can decrypt it:
    - Examples: (0x01FE01FE01FE01FE, 0xFE01FE01FE01FE01), (0x1FE01FE00EF10EF1, 0xE01FE01FF10EF10E)
3. **Possible Weak Keys**: These are keys that might produce undesirable results in specific implementations or under certain conditions:
    - Examples: 0x01F1E0D3C2B1A087, 0xFE0E1F1E0D1C2B3A



Round Operation in IDEA (International Data Encryption Algorithm)

The International Data Encryption Algorithm (IDEA) is a symmetric key block cipher. It operates on 64-bit blocks and uses a 128-bit key. The encryption process involves multiple rounds, each consisting of the following steps:

Round Operation Steps:

1. **Subkey Generation**:
   - Generate six subkeys for each round from the 128-bit key, totaling 52 subkeys for the entire encryption process.
2. **Input Splitting**:
   - Split the 64-bit input block into four 16-bit sub-blocks: X_1, X_2, X_3, X_4.
3. **Mixing Operations**:
   - Perform a series of operations (addition modulo 2^16, multiplication modulo $2^{16}+1$, and XOR) on the sub-blocks using the round subkeys.

Detailed Round Operation:

For each round (1 to 8):

1. **Multiplication**:
   - $X_1 = X_1 \times K_1$ (mod $2^{16}+1$)
   - $X_2 = X_2 + K_2$ (mod 2^{16})
   - $X_3 = X_3 + K_3$ (mod 2^{16})
   - $X_4 = X_4 \times K_4$ (mod $2^{16}+1$)
2. **XOR and Swap**:
   - $T_1 = X_1 \oplus X_3$
   - $T_2 = X_2 \oplus X_4$
   - $T_3 = T_1 \times K_5$ (mod $2^{16}+1$)

’ve hit the Free plan limit for GPT-4o.

ponses will use our basic model until your limit resets after 4:56 PM.

   - $X_4 = X_4 \times K_4$ (mod $2 + 1$)

2. **XOR and Swap**:
   - $T_1 = X_1 \oplus X_3$
   - $T_2 = X_2 \oplus X_4$
   - $T_3 = T_1 \times K_5$ (mod $2^{16} + 1$)
   - $T_4 = (T_2 + T_3) \times K_6$ (mod $2^{16} + 1$)
   - $T_5 = T_3 + T_4$
3. **Swap Results**:
   - $X_1 \oplus T_4$
   - $X_2 \oplus T_5$
   - $X_3 \oplus T_4$
   - $X_4 \oplus T_5$
4. **Output Mixing**:
   - Swap X_2 and X_3 for the next round.

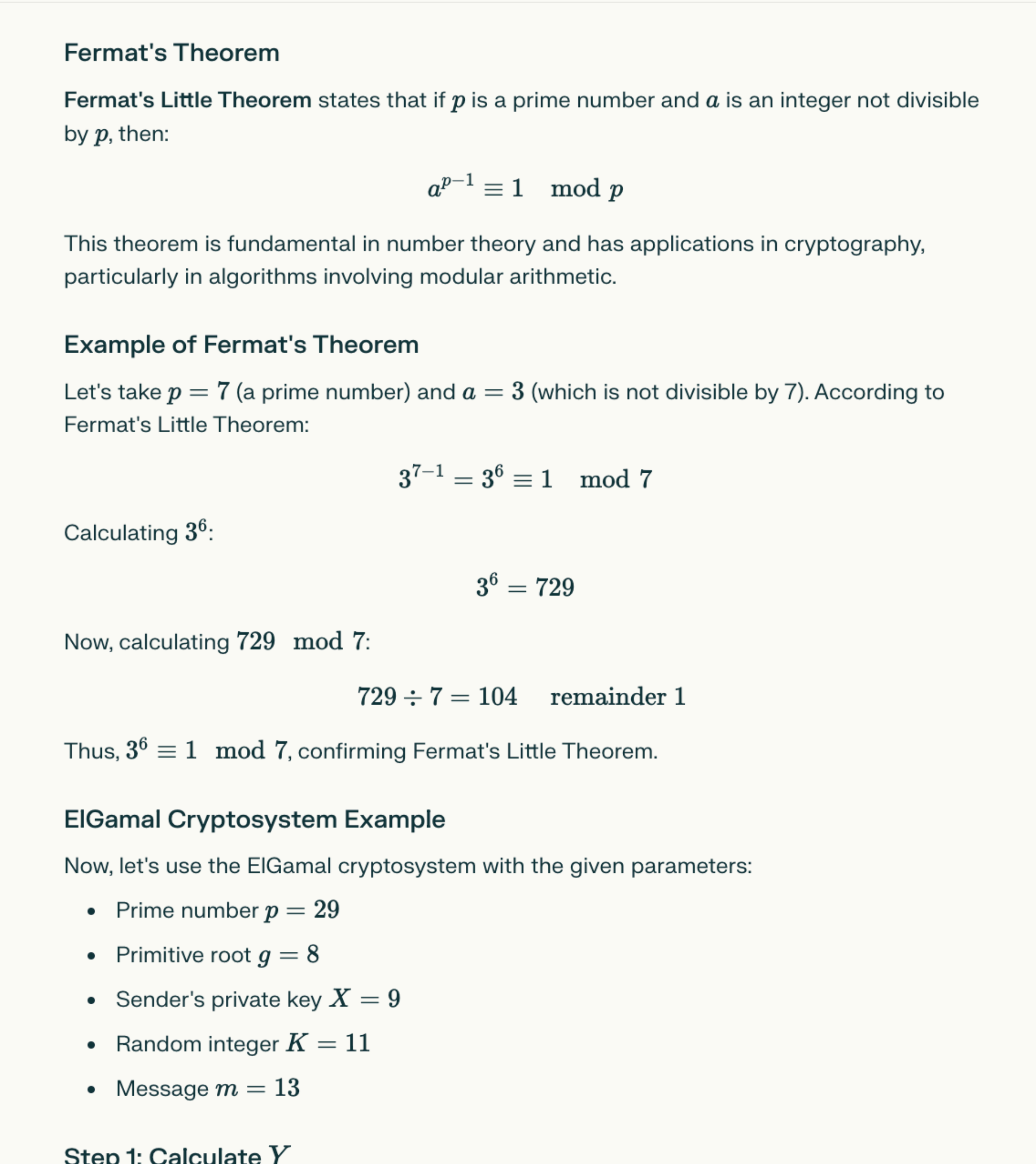
Final Transformation:

- After eight rounds, perform an additional half-round transformation using the last four subkeys to mix the output further.

Summary

- **More Vulnerable**: Monoalphabetic ciphers are more vulnerable to frequency analysis compared to polyalphabetic ciphers.
- **Weak Keys in DES**: Weak keys, semi-weak keys, and possible weak keys can create vulnerabilities in DES.
- **Round Operation in IDEA**: Involves multiple rounds of mixing operations (multiplication, addition, XOR) on 64-bit input blocks, using subkeys generated from a 128-bit key.

State Fermat’s theorem with an example. Given the prime number p=29 and its primitive root g=8, private key sender with X=9 and random integer K=11, encrypt the message m=13 using ElGamal cryptosystem.



Fermat's Theorem

Fermat's Little Theorem states that if p is a prime number and a is an integer not divisible by p, then:

$$a^{p-1} \equiv 1 \mod p$$

This theorem is fundamental in number theory and has applications in cryptography, particularly in algorithms involving modular arithmetic.

Example of Fermat's Theorem

Let's take $p = 7$ (a prime number) and $a = 3$ (which is not divisible by 7). According to Fermat's Little Theorem:

$$3^{7-1} = 3^6 \equiv 1 \mod 7$$

Calculating 3^6:

$$3^6 = 729$$

Now, calculating $729 \mod 7$:

$$729 \div 7 = 104 \quad \text{remainder } 1$$

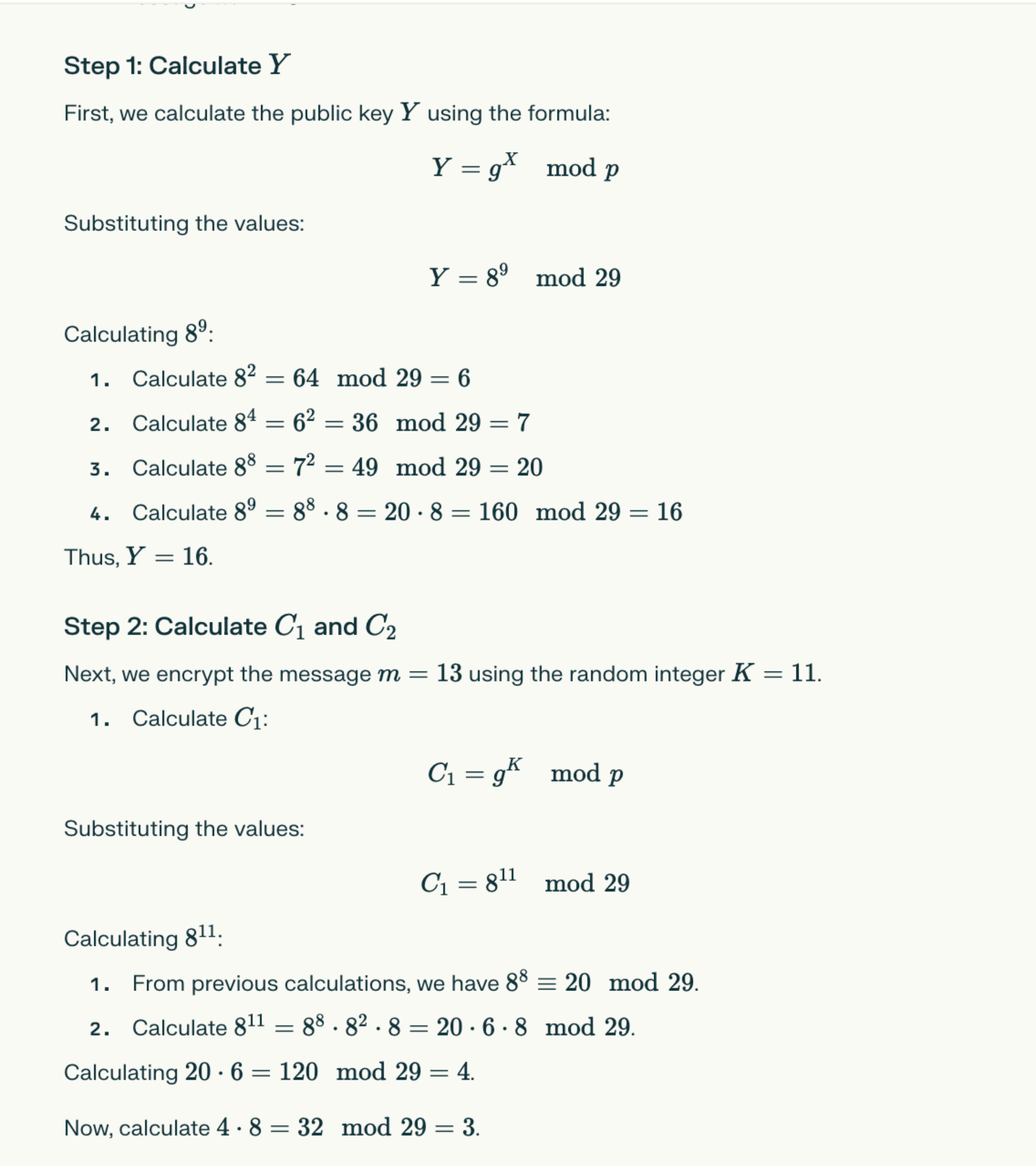
Thus, $3^6 \equiv 1 \mod 7$, confirming Fermat's Little Theorem.

ElGamal Cryptosystem Example

Now, let's use the ElGamal cryptosystem with the given parameters:

- Prime number $p = 29$
- Primitive root $g = 8$
- Sender's private key $X = 9$
- Random integer $K = 11$
- Message $m = 13$

Step 1: Calculate Y



Step 1: Calculate Y

First, we calculate the public key Y using the formula:

$$Y = g^X \mod p$$

Substituting the values:

$$Y = 8^9 \mod 29$$

Calculating 8^9:

1. Calculate $8^2 = 64 \mod 29 = 6$
2. Calculate $8^4 = 6^2 = 36 \mod 29 = 7$
3. Calculate $8^8 = 7^2 = 49 \mod 29 = 20$
4. Calculate $8^9 = 8^8 \cdot 8 = 20 \cdot 8 = 160 \mod 29 = 16$

Thus, $Y = 16$.

Step 2: Calculate C_1 and C_2

Next, we encrypt the message $m = 13$ using the random integer $K = 11$.

1. Calculate C_1:

$$C_1 = g^K \mod p$$

Substituting the values:

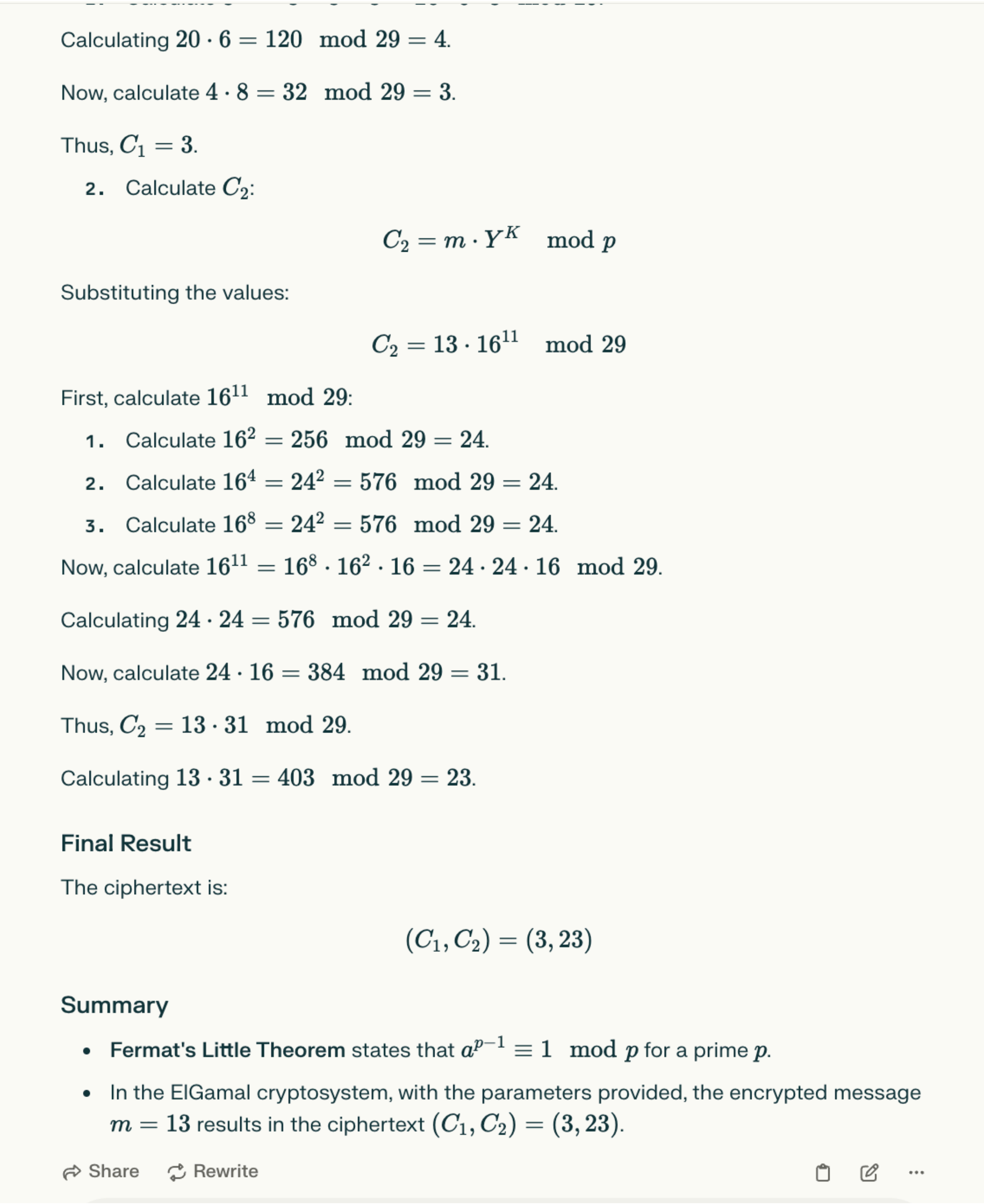
$$C_1 = 8^{11} \mod 29$$

Calculating 8^{11}:

1. From previous calculations, we have $8^8 \equiv 20 \mod 29$.
2. Calculate $8^{11} = 8^8 \cdot 8^2 \cdot 8 = 20 \cdot 6 \cdot 8 \mod 29$.

Calculating $20 \cdot 6 = 120 \mod 29 = 4$.

Now, calculate $4 \cdot 8 = 32 \mod 29 = 3$.



Calculating $20 \cdot 6 = 120 \mod 29 = 4$.

Now, calculate $4 \cdot 8 = 32 \mod 29 = 3$.

Thus, $C_1 = 3$.

2. Calculate C_2:

$$C_2 = m \cdot Y^K \mod p$$

Substituting the values:

$$C_2 = 13 \cdot 16^{11} \mod 29$$

First, calculate $16^{11} \mod 29$:

1. Calculate $16^2 = 256 \mod 29 = 24$.
2. Calculate $16^4 = 24^2 = 576 \mod 29 = 24$.
3. Calculate $16^8 = 24^2 = 576 \mod 29 = 24$.

Now, calculate $16^{11} = 16^8 \cdot 16^2 \cdot 16 = 24 \cdot 24 \cdot 16 \mod 29$.

Calculating $24 \cdot 24 = 576 \mod 29 = 24$.

Now, calculate $24 \cdot 16 = 384 \mod 29 = 31$.

Thus, $C_2 = 13 \cdot 31 \mod 29$.

Calculating $13 \cdot 31 = 403 \mod 29 = 23$.

Final Result

The ciphertext is:

$$(C_1, C_2) = (3, 23)$$

Summary

- **Fermat's Little Theorem** states that $a^{p-1} \equiv 1 \mod p$ for a prime p.
- In the ElGamal cryptosystem, with the parameters provided, the encrypted message $m = 13$ results in the ciphertext $(C_1, C_2) = (3, 23)$.

Fermat’s Little Theorem

Fermat’s Little Theorem states:

- **Theorem**: If p is a prime number and a is any integer not divisible by p, then:

$$a^{p-1} \equiv 1 \pmod{p}$$

Example: Let $a = 3$ and $p = 11$:

$$3^{11-1} = 3^{10}$$

Calculate $3^{10} \mod 11$:

$$3^2 = 9$$

$$3^4 = (3^2)^2 = 9^2 = 81 \equiv 4 \pmod{11}$$

$$3^8 = (3^4)^2 = 4^2 = 16 \equiv 5 \pmod{11}$$

$$3^{10} = 3^8 \cdot 3^2 = 5 \cdot 9 = 45 \equiv 1 \pmod{11}$$

Thus, $3^{10} \equiv 1 \pmod{11}$, verifying Fermat’s Little Theorem.

Encryption Using ElGamal Cryptosystem

Given:

- **Prime Number**: $p = 29$
- **Primitive Root**: $g = 8$
- **Private Key**: $X = 9$
- **Random Integer**: $K = 11$
- **Message**: $m = 13$

ElGamal Encryption Steps:

1. **Compute the Public Key**:

$$Y = g^X \pmod{p}$$

$$Y = 8^9 \pmod{29}$$

   Calculate $8^9 \mod 29$:

$$8^2 = 64 \equiv 6 \pmod{29}$$

$$8^4 = (8^2)^2 = 6^2 = 36 \equiv 7 \pmod{29}$$

$$8^8 = (8^4)^2 = 7^2 = 49 \equiv 20 \pmod{29}$$

$$8^9 = 8 \cdot 8^8 = 8 \cdot 20 = 160 \equiv 15 \pmod{29}$$

   So, $Y = 15$.

2. **Encrypt the Message**:

   - Compute c_1:

$$c_1 = g^K \pmod{p}$$

$$c_1 = 8^{11} \pmod{29}$$

     Calculate $8^{11} \mod 29$:

$$8^2 = 64 \equiv 6 \pmod{29}$$

$$8^8 = 7^2 = 49 \equiv 20 \pmod{29}$$

$$8^{10} = 8^8 \cdot 8^2 = 20 \cdot 6 = 120 \equiv 4 \pmod{29}$$

$$8^{11} = 8 \cdot 8^{10} = 8 \cdot 4 = 32 \equiv 3 \pmod{29}$$

So, $c_1 = 3$.

- Compute c_2:

$$c_2 = m \cdot Y^K \pmod{p}$$

$$c_2 = 13 \cdot 15^{11} \pmod{29}$$

Calculate $15^{11} \mod 29$:

$$15^2 = 225 \equiv 23 \pmod{29}$$

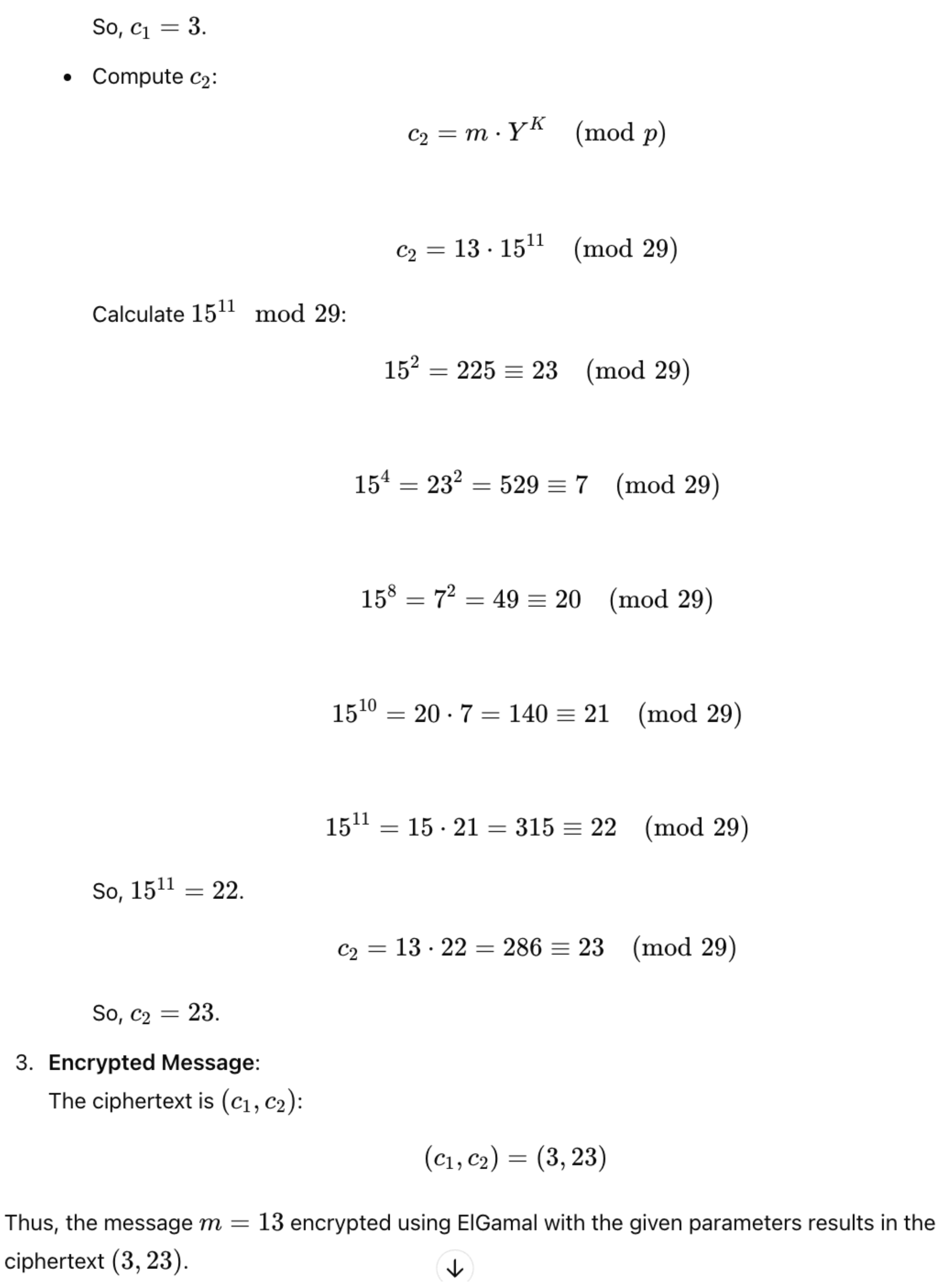
$$15^4 = 23^2 = 529 \equiv 7 \pmod{29}$$

$$15^8 = 7^2 = 49 \equiv 20 \pmod{29}$$

$$15^{10} = 20 \cdot 7 = 140 \equiv 21 \pmod{29}$$

$$15^{11} = 15 \cdot 21 = 315 \equiv 22 \pmod{29}$$

So, $15^{11} = 22$.



So, $c_1 = 3$.

- Compute c_2:

$$c_2 = m \cdot Y^K \pmod{p}$$

$$c_2 = 13 \cdot 15^{11} \pmod{29}$$

Calculate $15^{11} \mod 29$:

$$15^2 = 225 \equiv 23 \pmod{29}$$

$$15^4 = 23^2 = 529 \equiv 7 \pmod{29}$$

$$15^8 = 7^2 = 49 \equiv 20 \pmod{29}$$

$$15^{10} = 20 \cdot 7 = 140 \equiv 21 \pmod{29}$$

$$15^{11} = 15 \cdot 21 = 315 \equiv 22 \pmod{29}$$

So, $15^{11} = 22$.

$$c_2 = 13 \cdot 22 = 286 \equiv 23 \pmod{29}$$

So, $c_2 = 23$.

3. **Encrypted Message**:
   The ciphertext is (c_1, c_2):

$$(c_1, c_2) = (3, 23)$$

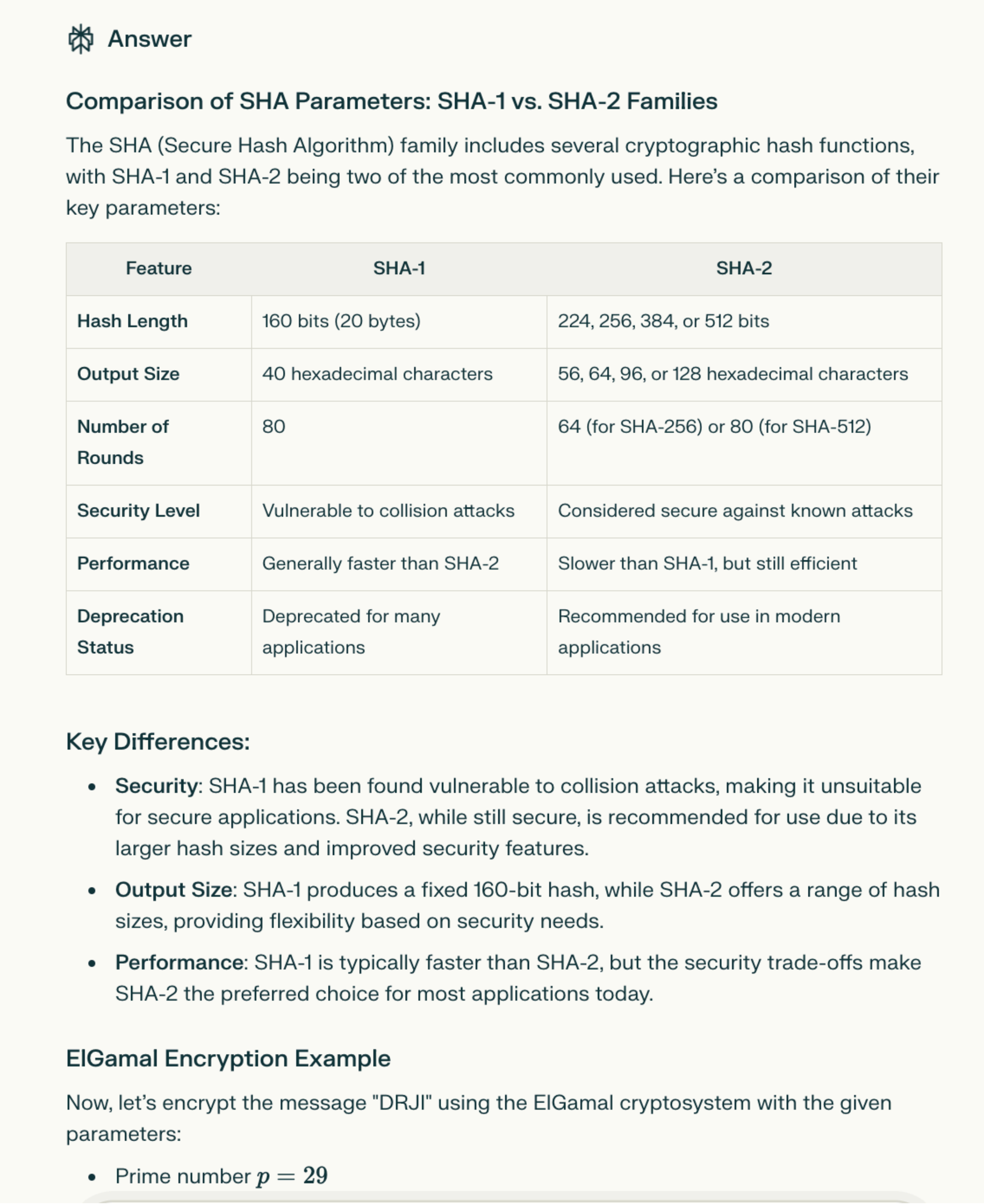
Thus, the message $m = 13$ encrypted using ElGamal with the given parameters results in the ciphertext $(3, 23)$.

So in summary, Fermat's Little Theorem is a powerful result that has important applications in number theory, cryptography, and modular arithmetic. Its simplicity and elegance make it a fundamental tool in mathematic.

Compare the SHA parameters between SHA-1 and SHA-2 families. Decrypt the cipher text DRJI with the key

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

using the Hill cipher.



Answer

Comparison of SHA Parameters: SHA-1 vs. SHA-2 Families

The SHA (Secure Hash Algorithm) family includes several cryptographic hash functions, with SHA-1 and SHA-2 being two of the most commonly used. Here's a comparison of their key parameters:

Feature	SHA-1	SHA-2
Hash Length	160 bits (20 bytes)	224, 256, 384, or 512 bits
Output Size	40 hexadecimal characters	56, 64, 96, or 128 hexadecimal characters
Number of Rounds	80	64 (for SHA-256) or 80 (for SHA-512)
Security Level	Vulnerable to collision attacks	Considered secure against known attacks
Performance	Generally faster than SHA-2	Slower than SHA-1, but still efficient
Deprecation Status	Deprecated for many applications	Recommended for use in modern applications

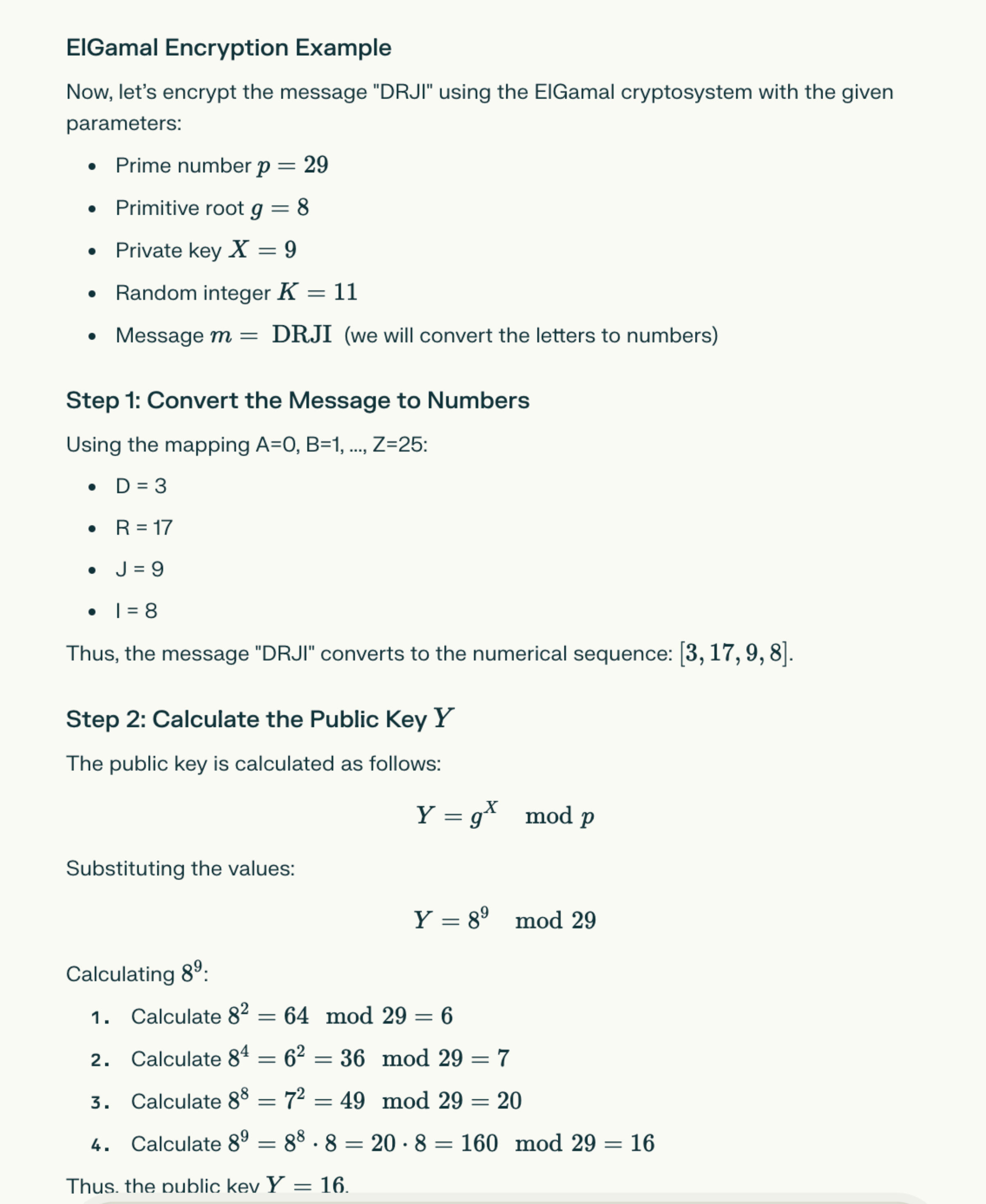
Key Differences:

- **Security**: SHA-1 has been found vulnerable to collision attacks, making it unsuitable for secure applications. SHA-2, while still secure, is recommended for use due to its larger hash sizes and improved security features.
- **Output Size**: SHA-1 produces a fixed 160-bit hash, while SHA-2 offers a range of hash sizes, providing flexibility based on security needs.
- **Performance**: SHA-1 is typically faster than SHA-2, but the security trade-offs make SHA-2 the preferred choice for most applications today.

ElGamal Encryption Example

Now, let's encrypt the message "DRJI" using the ElGamal cryptosystem with the given parameters:

- Prime number $p = 29$



ElGamal Encryption Example

Now, let's encrypt the message "DRJI" using the ElGamal cryptosystem with the given parameters:

- Prime number $p = 29$
- Primitive root $g = 8$
- Private key $X = 9$
- Random integer $K = 11$
- Message $m = \text{DRJI}$ (we will convert the letters to numbers)

Step 1: Convert the Message to Numbers

Using the mapping A=0, B=1, ..., Z=25:

- D = 3
- R = 17
- J = 9
- I = 8

Thus, the message "DRJI" converts to the numerical sequence: $[3, 17, 9, 8]$.

Step 2: Calculate the Public Key Y

The public key is calculated as follows:

$$Y = g^X \mod p$$

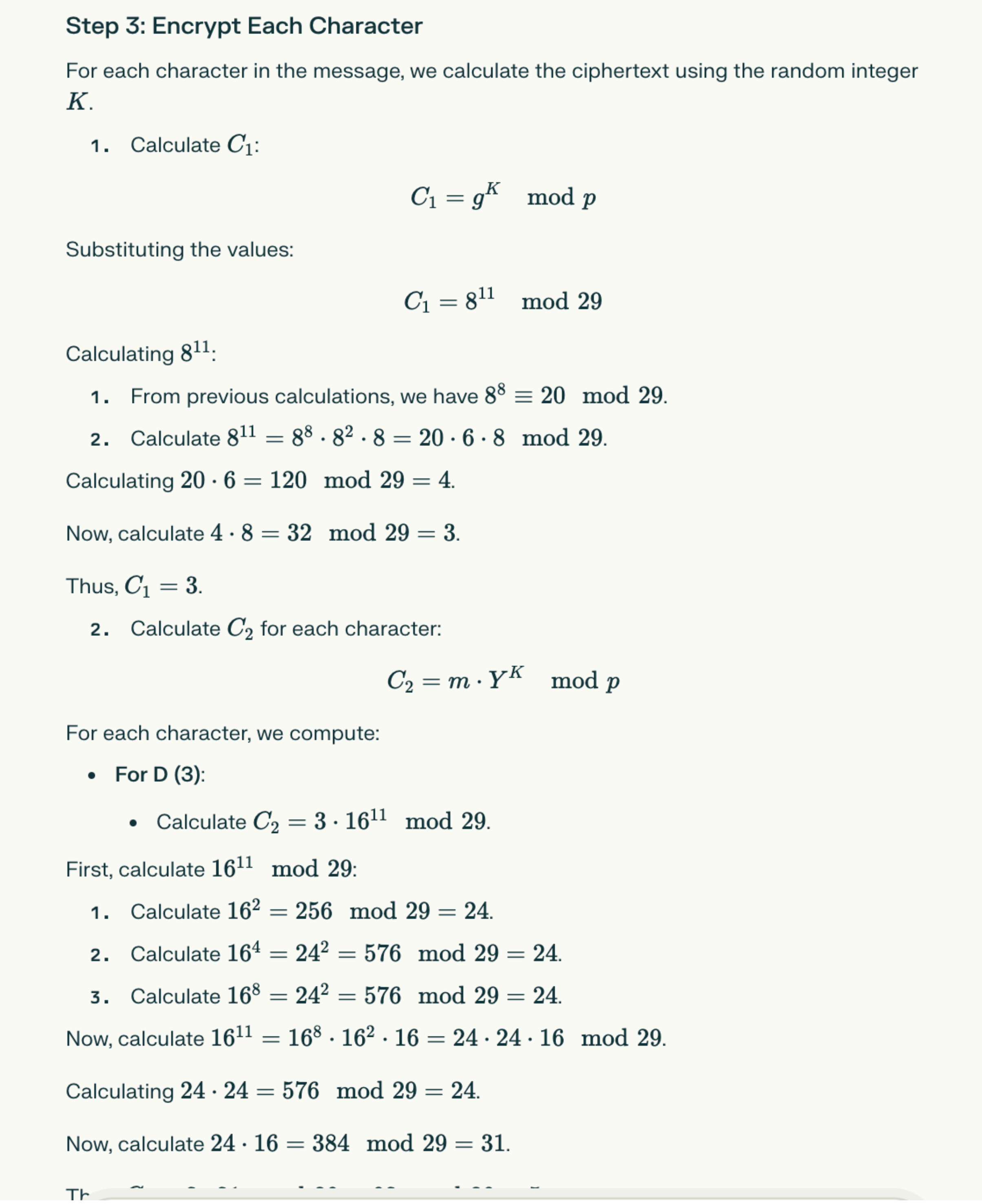
Substituting the values:

$$Y = 8^9 \mod 29$$

Calculating 8^9:

1. Calculate $8^2 = 64 \mod 29 = 6$
2. Calculate $8^4 = 6^2 = 36 \mod 29 = 7$
3. Calculate $8^8 = 7^2 = 49 \mod 29 = 20$
4. Calculate $8^9 = 8^8 \cdot 8 = 20 \cdot 8 = 160 \mod 29 = 16$

Thus, the public key $Y = 16$.



Step 3: Encrypt Each Character

For each character in the message, we calculate the ciphertext using the random integer K.

1. Calculate C_1:

$$C_1 = g^K \mod p$$

Substituting the values:

$$C_1 = 8^{11} \mod 29$$

Calculating 8^{11}:

1. From previous calculations, we have $8^8 \equiv 20 \mod 29$.
2. Calculate $8^{11} = 8^8 \cdot 8^2 \cdot 8 = 20 \cdot 6 \cdot 8 \mod 29$.

Calculating $20 \cdot 6 = 120 \mod 29 = 4$.

Now, calculate $4 \cdot 8 = 32 \mod 29 = 3$.

Thus, $C_1 = 3$.

2. Calculate C_2 for each character:

$$C_2 = m \cdot Y^K \mod p$$

For each character, we compute:

- **For D (3):**
  - Calculate $C_2 = 3 \cdot 16^{11} \mod 29$.

First, calculate $16^{11} \mod 29$:

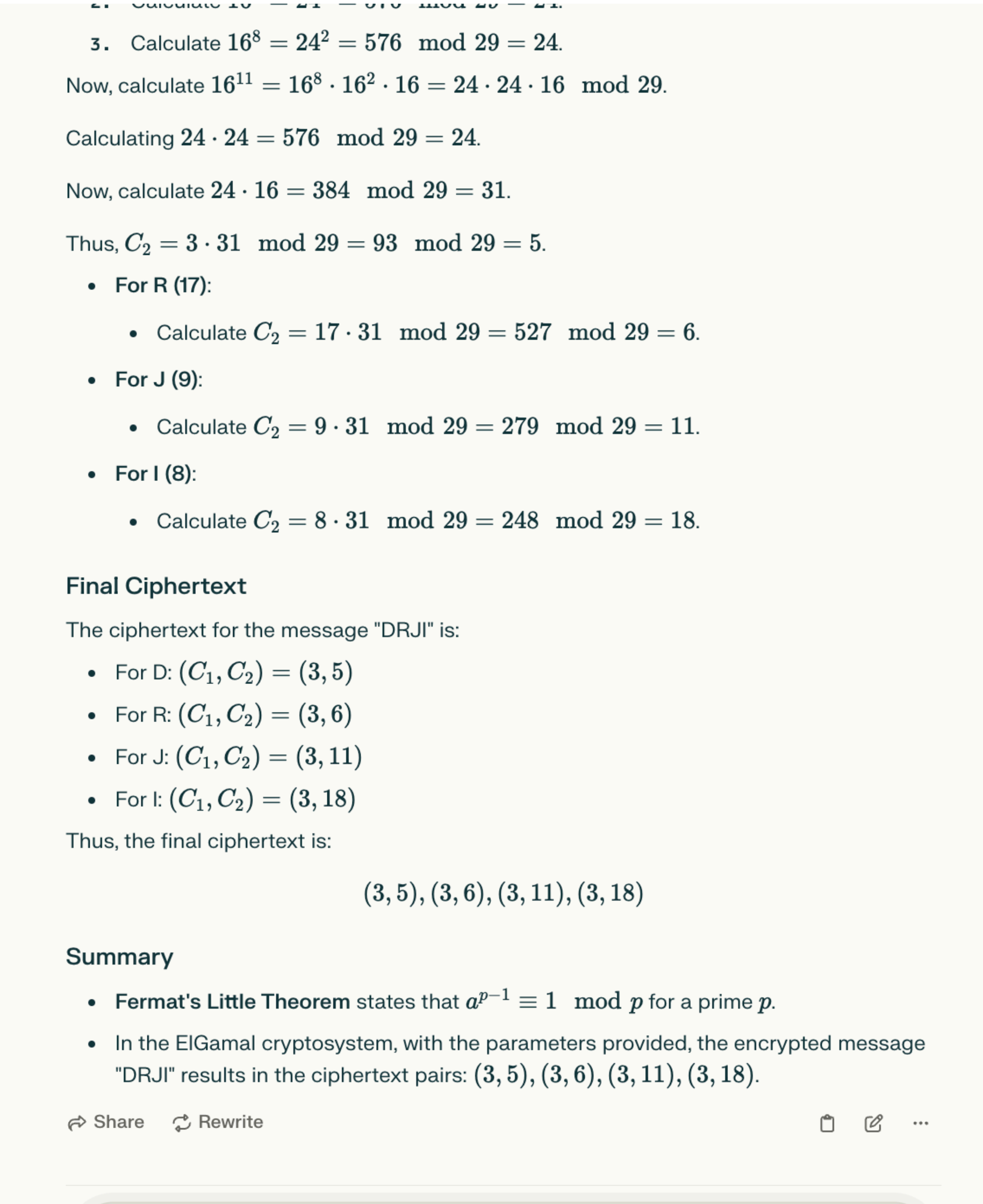
1. Calculate $16^2 = 256 \mod 29 = 24$.
2. Calculate $16^4 = 24^2 = 576 \mod 29 = 24$.
3. Calculate $16^8 = 24^2 = 576 \mod 29 = 24$.

Now, calculate $16^{11} = 16^8 \cdot 16^2 \cdot 16 = 24 \cdot 24 \cdot 16 \mod 29$.

Calculating $24 \cdot 24 = 576 \mod 29 = 24$.

Now, calculate $24 \cdot 16 = 384 \mod 29 = 31$.

Th



3. Calculate $16^8 = 24^2 = 576 \mod 29 = 24$.

Now, calculate $16^{11} = 16^8 \cdot 16^2 \cdot 16 = 24 \cdot 24 \cdot 16 \mod 29$.

Calculating $24 \cdot 24 = 576 \mod 29 = 24$.

Now, calculate $24 \cdot 16 = 384 \mod 29 = 31$.

Thus, $C_2 = 3 \cdot 31 \mod 29 = 93 \mod 29 = 5$.

- **For R (17):**
  - Calculate $C_2 = 17 \cdot 31 \mod 29 = 527 \mod 29 = 6$.
- **For J (9):**
  - Calculate $C_2 = 9 \cdot 31 \mod 29 = 279 \mod 29 = 11$.
- **For I (8):**
  - Calculate $C_2 = 8 \cdot 31 \mod 29 = 248 \mod 29 = 18$.

Final Ciphertext

The ciphertext for the message "DRJI" is:

- For D: $(C_1, C_2) = (3, 5)$
- For R: $(C_1, C_2) = (3, 6)$
- For J: $(C_1, C_2) = (3, 11)$
- For I: $(C_1, C_2) = (3, 18)$

Thus, the final ciphertext is:

$$(3, 5), (3, 6), (3, 11), (3, 18)$$

Summary

- **Fermat's Little Theorem** states that $a^{p-1} \equiv 1 \mod p$ for a prime p.
- In the ElGamal cryptosystem, with the parameters provided, the encrypted message "DRJI" results in the ciphertext pairs: $(3, 5), (3, 6), (3, 11), (3, 18)$.

SHA Parameters Comparison: SHA-1 vs. SHA-2

SHA-1:

- **Digest Length**: 160 bits (20 bytes)
- **Block Size**: 512 bits (64 bytes)
- **Number of Rounds**: 80

- **Security**: SHA-1 has been found to be vulnerable to collision attacks. It is generally considered insecure for modern cryptographic applications.
- **Output Example**: `a9993e364706816aba3e25717850c26c9cd0d89d` (for the string "abc")

SHA-2 (includes SHA-224, SHA-256, SHA-384, and SHA-512):

- **SHA-224**:
    - **Digest Length**: 224 bits (28 bytes)
    - **Block Size**: 512 bits (64 bytes)
    - **Number of Rounds**: 64
- **SHA-256**:
    - **Digest Length**: 256 bits (32 bytes)
    - **Block Size**: 512 bits (64 bytes)
    - **Number of Rounds**: 64
- **SHA-384**:
    - **Digest Length**: 384 bits (48 bytes)
    - **Block Size**: 1024 bits (128 bytes)
    - **Number of Rounds**: 80
- **SHA-512**:
    - **Digest Length**: 512 bits (64 bytes)
    - **Block Size**: 1024 bits (128 bytes)
    - **Number of Rounds**: 80

Security: SHA-2 is considered **more secure** than SHA-1 due to its longer hash lengths and resistance to collision attacks. SHA-2 provides stronger security assurances.

Decrypting Ciphertext DRJI Using Hill Cipher

The Hill cipher is a type of polygraphic substitution cipher based on linear algebra. To decrypt a message using the Hill cipher, we need the key matrix and its inverse.

Given Key Matrix:

$$K = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

Ciphertext: DRJI

Steps:

1. **Convert Ciphertext to Numerical Form**:
   - Use the mapping $A = 0, B = 1, \ldots, Z = 25$:

     $$\text{D} = 3, \quad \text{R} = 17, \quad \text{J} = 9, \quad \text{I} = 8$$

     So, the numerical form is:

     $$\text{DRJI} \to \begin{pmatrix} 3 \\ 17 \\ 9 \\ 8 \end{pmatrix}$$

2. **Form Ciphertext Matrix**:
   - Arrange the numerical values into a 2x2 matrix:

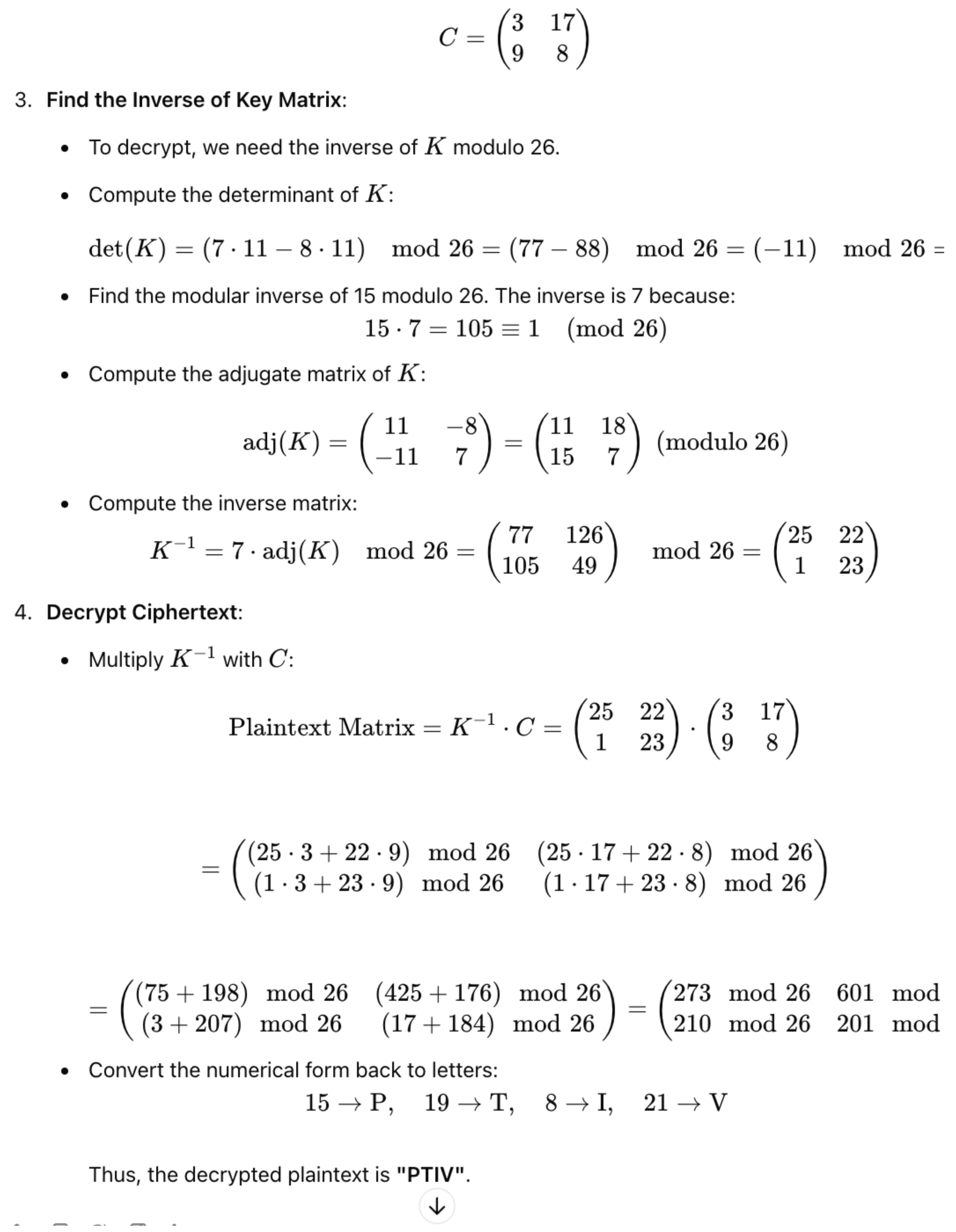
     $$C = \begin{pmatrix} 3 & 17 \\ 9 & 8 \end{pmatrix}$$

3. **Find the Inverse of Key Matrix**:
   - To decrypt, we need the inverse of K modulo 26.
   - Compute the determinant of K:

     $$\det(K) = (7 \cdot 11 - 8 \cdot 11) \mod 26 = (77 - 88) \mod 26 = (-11) \mod 26 =$$

   - Find the modular inverse of 15 modulo 26. The inverse is 7 because:

     $$15 \cdot 7 \quad 105 \equiv 1 \pmod{26}$$



$$C = \begin{pmatrix} 3 & 17 \\ 9 & 8 \end{pmatrix}$$

3. **Find the Inverse of Key Matrix**:
   - To decrypt, we need the inverse of K modulo 26.
   - Compute the determinant of K:

   $$\det(K) = (7 \cdot 11 - 8 \cdot 11) \mod 26 = (77 - 88) \mod 26 = (-11) \mod 26 =$$

   - Find the modular inverse of 15 modulo 26. The inverse is 7 because:

   $$15 \cdot 7 = 105 \equiv 1 \pmod{26}$$

   - Compute the adjugate matrix of K:

   $$\text{adj}(K) = \begin{pmatrix} 11 & -8 \\ -11 & 7 \end{pmatrix} = \begin{pmatrix} 11 & 18 \\ 15 & 7 \end{pmatrix} \text{ (modulo 26)}$$

   - Compute the inverse matrix:

   $$K^{-1} = 7 \cdot \text{adj}(K) \mod 26 = \begin{pmatrix} 77 & 126 \\ 105 & 49 \end{pmatrix} \mod 26 = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix}$$

4. **Decrypt Ciphertext**:
   - Multiply K^{-1} with C:

   $$\text{Plaintext Matrix} = K^{-1} \cdot C = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \cdot \begin{pmatrix} 3 & 17 \\ 9 & 8 \end{pmatrix}$$

   $$= \begin{pmatrix} (25 \cdot 3 + 22 \cdot 9) \mod 26 & (25 \cdot 17 + 22 \cdot 8) \mod 26 \\ (1 \cdot 3 + 23 \cdot 9) \mod 26 & (1 \cdot 17 + 23 \cdot 8) \mod 26 \end{pmatrix}$$

   $$= \begin{pmatrix} (75 + 198) \mod 26 & (425 + 176) \mod 26 \\ (3 + 207) \mod 26 & (17 + 184) \mod 26 \end{pmatrix} = \begin{pmatrix} 273 \mod 26 & 601 \mod \\ 210 \mod 26 & 201 \mod \end{pmatrix}$$

   - Convert the numerical form back to letters:

   $$15 \rightarrow \text{P}, \quad 19 \rightarrow \text{T}, \quad 8 \rightarrow \text{I}, \quad 21 \rightarrow \text{V}$$

   Thus, the decrypted plaintext is **"PTIV"**.

SHA-1 vs. SHA-2 Parameters

SHA-1

- Produces a 160-bit hash value.
- Uses a 512-bit block size.

- Has been shown to be vulnerable to collision attacks.

SHA-2

- Includes variants with different hash value lengths (SHA-224, SHA-256, SHA-384, SHA-512).
- Uses larger block sizes (512 bits for SHA-256, SHA-224; 1024 bits for SHA-512, SHA-384).
- Generally considered more secure than SHA-1.

Key differences:

- **Hash value length:** SHA-2 offers longer hash values, providing increased resistance to brute-force attacks.
- **Block size:** SHA-2 uses larger block sizes, improving security against various attacks.
- **Security:** SHA-2 is generally considered more secure than SHA-1 due to the larger hash size and improved design.

Decrypting DRJI with Hill Cipher

Understanding Hill Cipher: Hill cipher is a **polygraphic substitution** cipher based on linear algebra. It encrypts multiple letters at a time using a matrix.

Given:

- Ciphertext: DRJI
- Key matrix: K = [[7, 8], [11, 11]]

Steps:

1. **Convert letters to numbers:**
    - D = 3, R = 17, J = 9, I = 8
2. **Create plaintext vector:**
    - P = [[3], [17]]
3. **Find the inverse of the key matrix K:**
    - The determinant of K is 7*11 - 8*11 = -11 ≡ 18 (mod 26)
    - The inverse of 18 modulo 26 is 7 (since 18*7 ≡ 1 (mod 26))
    - The adjugate of K is [[11, -8], [-11, 7]]
    - The inverse of K is K^-1 = 7 * [[11, -8], [-11, 7]] mod 26 = [[15, 2], [15, 7]]
4. **Decrypt the ciphertext:**
    - P = K^-1 * C (mod 26)
    - P = [[15, 2], [15, 7]] * [[3], [17]] mod 26 = [[75], [176]] mod 26 = [[19], [14]]
5. **Convert numbers back to letters:**

    - 19 -> T
    - 14 -> O

Therefore, the decrypted plaintext is "TO".

Section B

Attempt any EIGHT questions.

Define discrete logarithm. Explain the procedure of sharing the secret key in Diffie Hellman.

Discrete Logarithm

Discrete logarithm is the **inverse operation** of **exponentiation** in **modular arithmetic**. Given a prime number p, a base g (primitive root modulo p), and a value a, the discrete logarithm x is the solution to the equation:

$g^x \equiv a \pmod{p}$

Finding x given g, p, and a is generally considered computationally difficult, forming the basis for the security of many cryptographic systems.

Diffie-Hellman Key Exchange

The Diffie-Hellman key exchange is a method for two parties to establish a shared secret key over an insecure channel. It relies on the difficulty of the discrete logarithm problem.

Steps:

1. **Agree on public parameters:** Both parties agree on a large prime number p and a primitive root g modulo p.
2. **Generate private keys:** Each party, let's say Alice and Bob, generates a secret random number. Alice chooses a private key 'a' and Bob chooses a private key 'b'.
3. **Calculate public keys:** Alice calculates A = g^a mod p and Bob calculates B = g^b mod p. These public keys are exchanged publicly.
4. **Calculate shared secret:** Alice calculates the shared secret as s = B^a mod p, and Bob calculates the shared secret as s = A^b mod p.

Mathematically:

- Alice: Public key A = g^a mod p, Secret key s = (g^b)^a mod p = g^(ba) mod p
- Bob: Public key B = g^b mod p, Secret key s = (g^a)^b mod p = g^(ab) mod p

Since ab = ba, both Alice and Bob end up with the same shared secret 's', even though they never exchanged it directly.

Security: The security of the Diffie-Hellman key exchange relies on the difficulty of computing the discrete logarithm. An attacker would need to determine either 'a' from A or 'b' from B to calculate the shared secret, which is computationally infeasible for large prime numbers.

By using this method, two parties can establish a shared secret key over an insecure channel, which can then be used for symmetric encryption to secure their communication.

Discrete Logarithm

The **discrete logarithm** problem is defined as follows:

- **Definition**: Given a prime number p, a base (or generator) g in the multiplicative group of integers modulo p ($\mathbb{Z}_p^*$), and an integer y in the group, the discrete logarithm problem is to find the integer x such that:

  $$g^x \equiv y \pmod{p}$$

  In other words, given g, y, and p, find x where x is the exponent in the equation.

Example: If $p = 23$, $g = 5$, and $y = 8$, find x such that:

$$5^x \equiv 8 \pmod{23}$$

To solve this, you would try different values for x to see which one satisfies the equation:

$$5^1 = 5$$

$$5^2 = 25 \equiv 2 \pmod{23}$$

$$5^3 = 125 \equiv 8 \pmod{23}$$

Thus, $x = 3$.

Diffie-Hellman Key Exchange Procedure

The **Diffie-Hellman key exchange** is a method used to securely share a secret key over a public channel. Here's how it works:

1. **Choose Public Parameters**:
   - Select a large prime number p.
   - Choose a primitive root (or generat g modulo p.

1. **Choose Public Parameters**:
    - Select a large prime number p.
    - Choose a primitive root (or generator) g modulo p.
2. **Generate Private Keys**:
    - Each participant generates their own private key:
        - Alice chooses a private key a.
        - Bob chooses a private key b.
3. **Compute Public Keys**:
    - Each participant computes their public key using the chosen private key and the public parameters:
        - Alice computes $A = g^a \mod p$.
        - Bob computes $B = g^b \mod p$.
4. **Exchange Public Keys**:
    - Alice and Bob exchange their public keys over the public channel.
5. **Compute Shared Secret**:
    - Each participant computes the shared secret using their own private key and the received public key:
        - Alice computes the shared secret as $S = B^a \mod p$.
        - Bob computes the shared secret as $S = A^b \mod p$.

    Both computations will result in the same shared secret because:

    $$B^a \equiv (g^b)^a \equiv g^{ab} \pmod p$$

    $$A^b \equiv (g^a)^b \equiv g^{ab} \pmod p$$

    Thus, both Alice and Bob end up with the same shared secret $g^{ab} \mod p$.

Example:

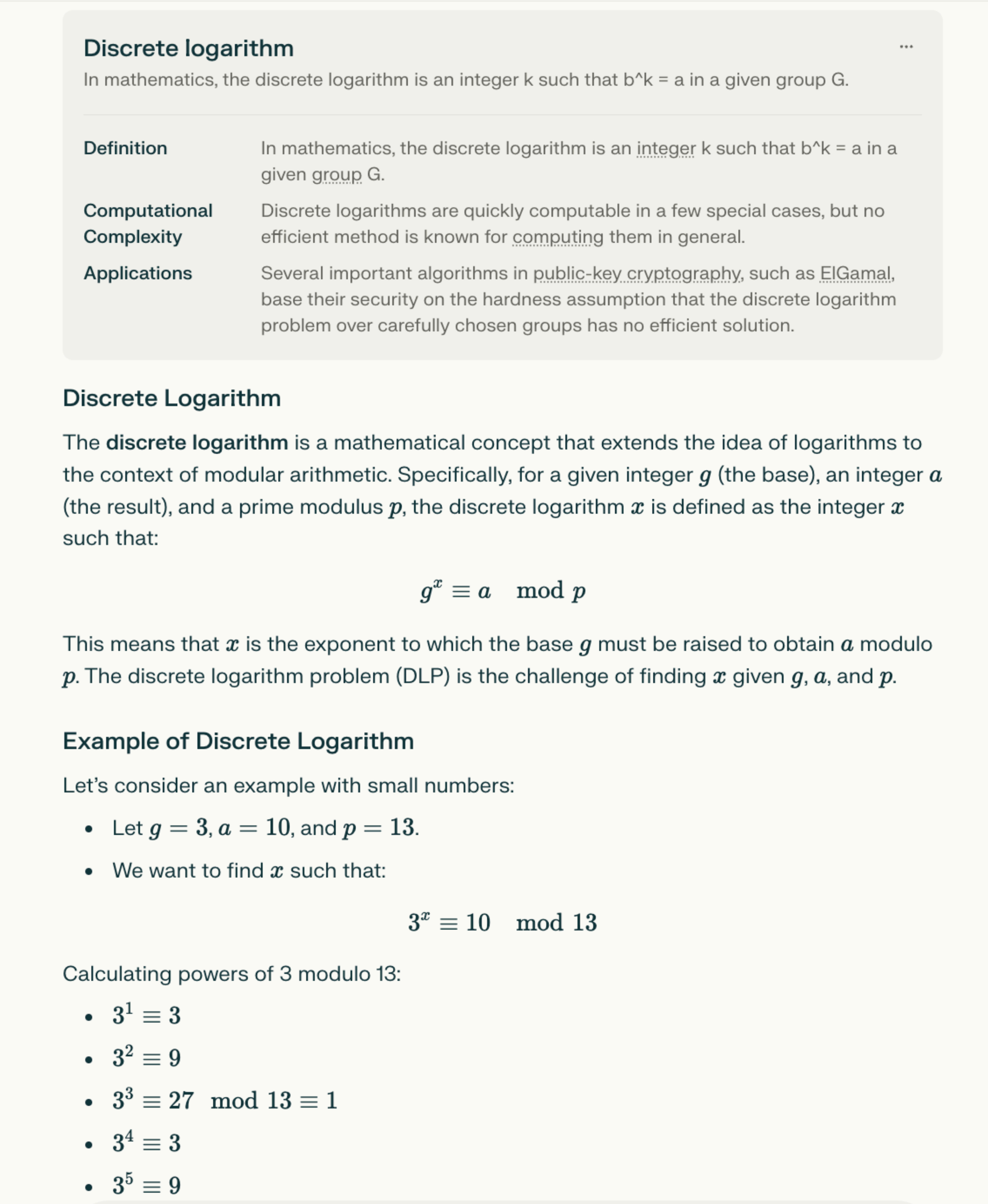
$$A^b \equiv (g^a)^b \equiv g^{ab} \pmod{p}$$

Thus, both Alice and Bob end up with the same shared secret $g^{ab} \mod p$.

Example:

1. **Public Parameters**:
   - Prime $p = 23$
   - Primitive root $g = 5$
2. **Generate Private Keys**:
   - Alice's private key $a = 6$
   - Bob's private key $b = 15$
3. **Compute Public Keys**:
   - Alice computes her public key:
     $$A = 5^6 \mod 23 = 15625 \mod 23 = 8$$
   - Bob computes his public key:
     $$B = 5^{15} \mod 23 = 30517578125 \mod 23 = 2$$
4. **Exchange Public Keys**:
   - Alice and Bob exchange public keys $A = 8$ and $B = 2$.
5. **Compute Shared Secret**:
   - Alice computes the shared secret:
     $$S = 2^6 \mod 23 = 64 \mod 23 = 18$$
   - Bob computes the shared secret:
     $$S = 8^{15} \mod 23 = 18$$

   Both Alice and Bob now share the secret key 18.



Discrete logarithm

In mathematics, the discrete logarithm is an integer k such that b^k = a in a given group G.

Definition	In mathematics, the discrete logarithm is an integer k such that b^k = a in a given group G.
Computational Complexity	Discrete logarithms are quickly computable in a few special cases, but no efficient method is known for computing them in general.
Applications	Several important algorithms in public-key cryptography, such as ElGamal, base their security on the hardness assumption that the discrete logarithm problem over carefully chosen groups has no efficient solution.

Discrete Logarithm

The **discrete logarithm** is a mathematical concept that extends the idea of logarithms to the context of modular arithmetic. Specifically, for a given integer g (the base), an integer a (the result), and a prime modulus p, the discrete logarithm x is defined as the integer x such that:

$$g^x \equiv a \mod p$$

This means that x is the exponent to which the base g must be raised to obtain a modulo p. The discrete logarithm problem (DLP) is the challenge of finding x given g, a, and p.

Example of Discrete Logarithm

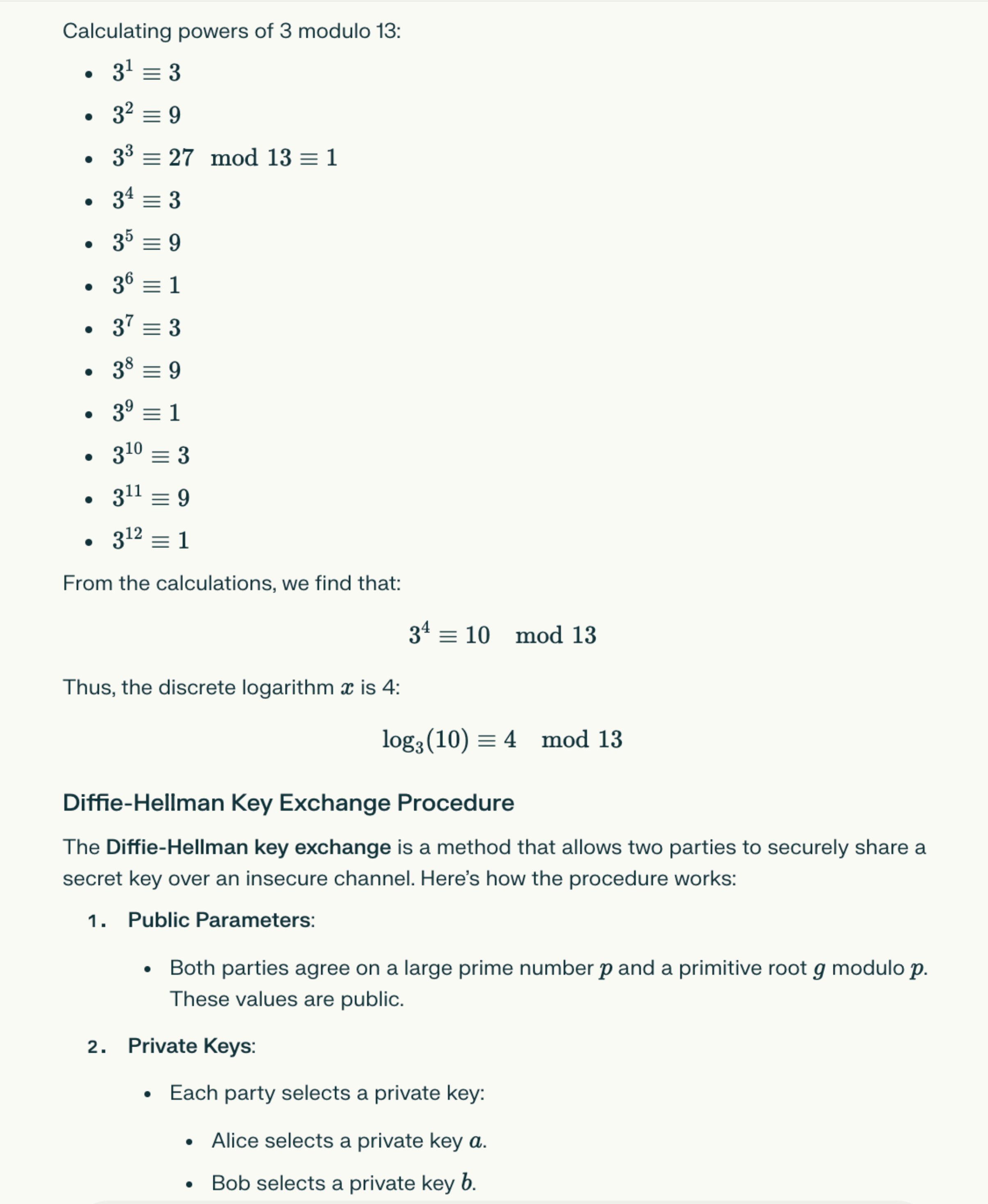
Let's consider an example with small numbers:

- Let $g = 3$, $a = 10$, and $p = 13$.
- We want to find x such that:

$$3^x \equiv 10 \mod 13$$

Calculating powers of 3 modulo 13:

- $3^1 \equiv 3$
- $3^2 \equiv 9$
- $3^3 \equiv 27 \mod 13 \equiv 1$
- $3^4 \equiv 3$
- $3^5 \equiv 9$



Calculating powers of 3 modulo 13:

- $3^1 \equiv 3$
- $3^2 \equiv 9$
- $3^3 \equiv 27 \mod 13 \equiv 1$
- $3^4 \equiv 3$
- $3^5 \equiv 9$
- $3^6 \equiv 1$
- $3^7 \equiv 3$
- $3^8 \equiv 9$
- $3^9 \equiv 1$
- $3^{10} \equiv 3$
- $3^{11} \equiv 9$
- $3^{12} \equiv 1$

From the calculations, we find that:

$$3^4 \equiv 10 \mod 13$$

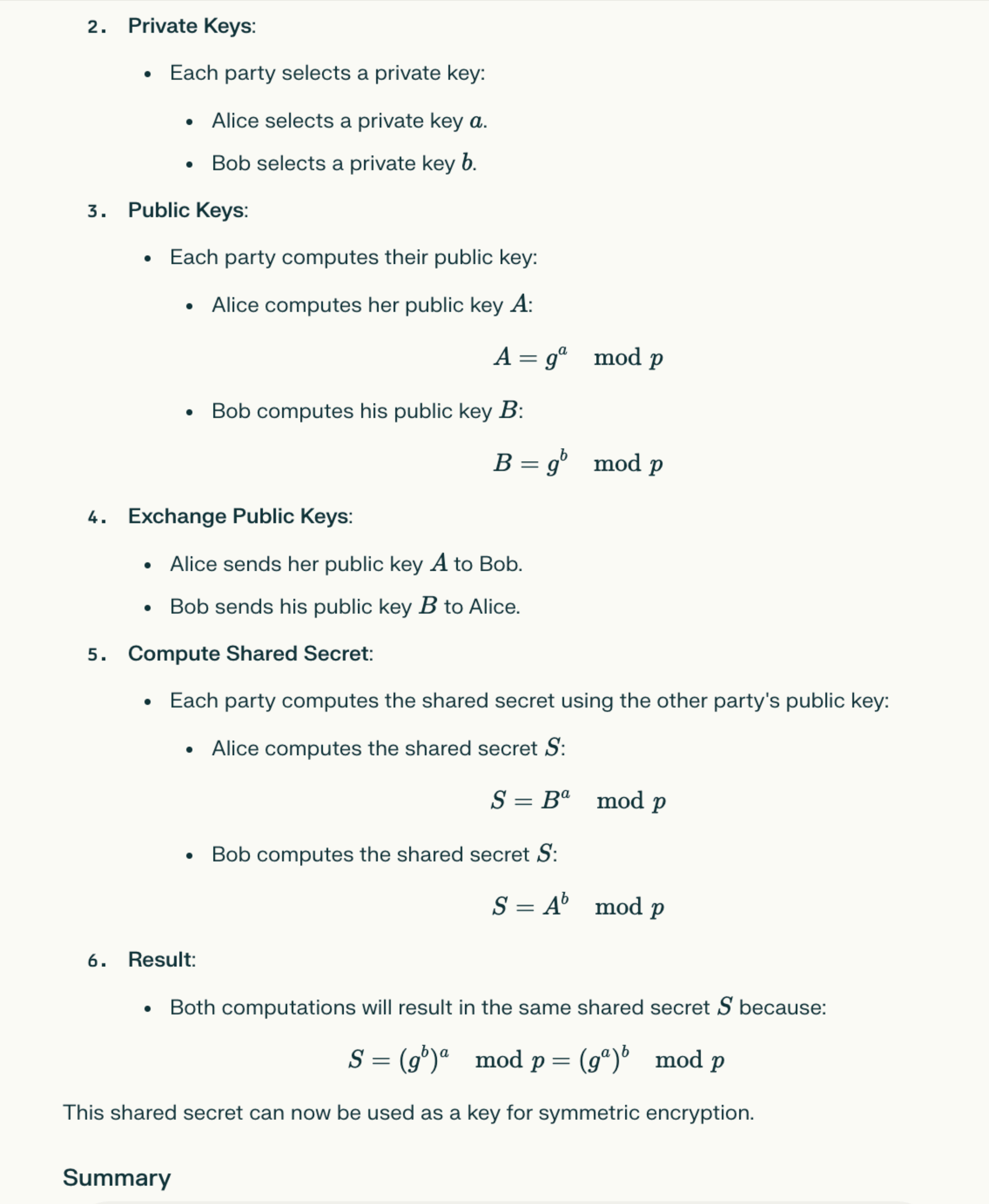
Thus, the discrete logarithm x is 4:

$$\log_3(10) \equiv 4 \mod 13$$

Diffie-Hellman Key Exchange Procedure

The **Diffie-Hellman key exchange** is a method that allows two parties to securely share a secret key over an insecure channel. Here's how the procedure works:

1. **Public Parameters**:
    - Both parties agree on a large prime number p and a primitive root g modulo p. These values are public.
2. **Private Keys**:
    - Each party selects a private key:
        - Alice selects a private key a.
        - Bob selects a private key b.



2. **Private Keys**:
    - Each party selects a private key:
        - Alice selects a private key a.
        - Bob selects a private key b.
3. **Public Keys**:
    - Each party computes their public key:
        - Alice computes her public key A:

$$A = g^a \mod p$$

        - Bob computes his public key B:

$$B = g^b \mod p$$

4. **Exchange Public Keys**:
    - Alice sends her public key A to Bob.
    - Bob sends his public key B to Alice.
5. **Compute Shared Secret**:
    - Each party computes the shared secret using the other party's public key:
        - Alice computes the shared secret S:

$$S = B^a \mod p$$

        - Bob computes the shared secret S:

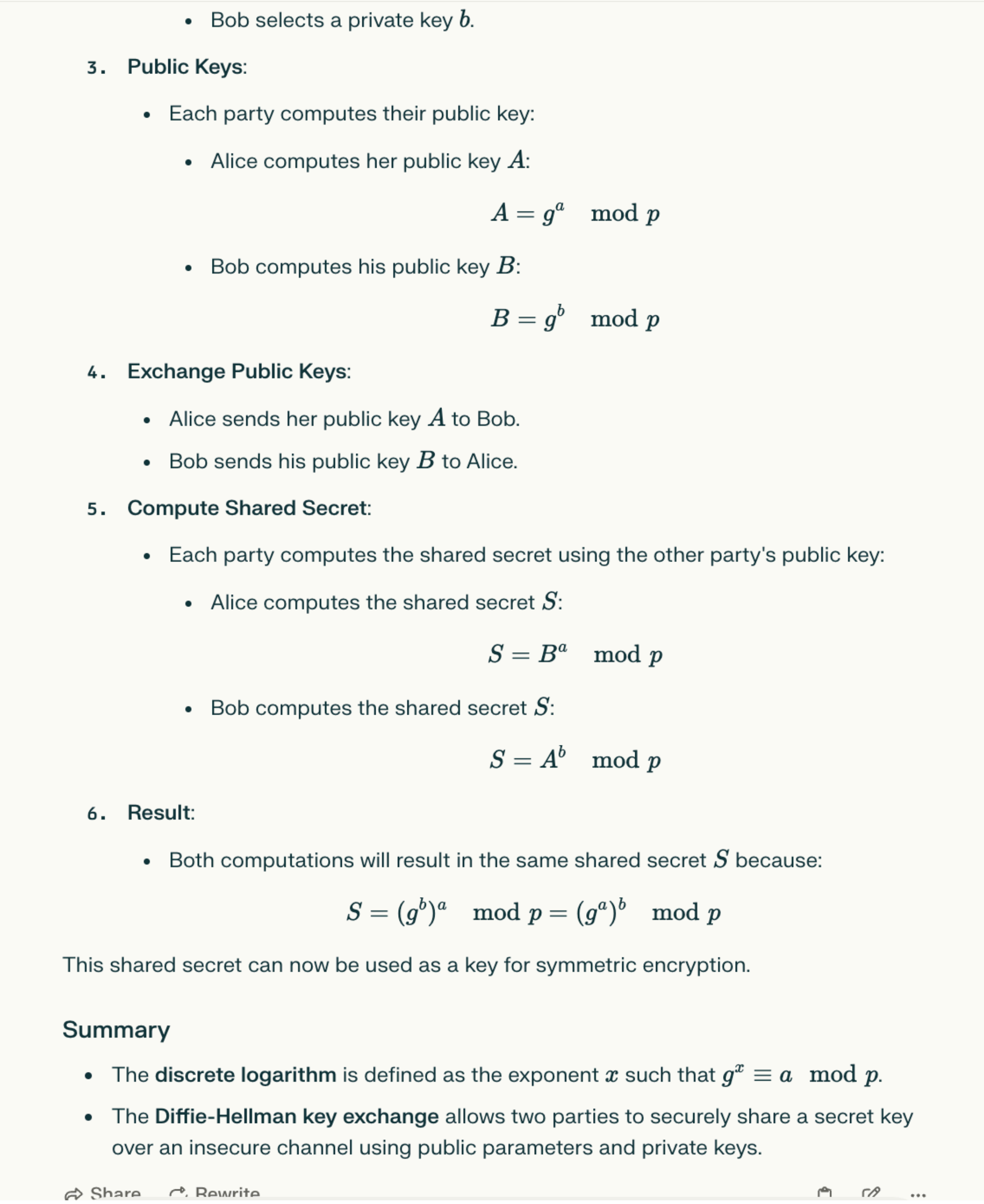
$$S = A^b \mod p$$

6. **Result**:
    - Both computations will result in the same shared secret S because:

$$S = (g^b)^a \mod p = (g^a)^b \mod p$$

This shared secret can now be used as a key for symmetric encryption.

Summary



- Bob selects a private key b.

3. **Public Keys**:
   - Each party computes their public key:
     - Alice computes her public key A:

$$A = g^a \mod p$$

     - Bob computes his public key B:

$$B = g^b \mod p$$

4. **Exchange Public Keys**:
   - Alice sends her public key A to Bob.
   - Bob sends his public key B to Alice.
5. **Compute Shared Secret**:
   - Each party computes the shared secret using the other party's public key:
     - Alice computes the shared secret S:

$$S = B^a \mod p$$

     - Bob computes the shared secret S:

$$S = A^b \mod p$$

6. **Result**:
   - Both computations will result in the same shared secret S because:

$$S = (g^b)^a \mod p = (g^a)^b \mod p$$

This shared secret can now be used as a key for symmetric encryption.

Summary

- The **discrete logarithm** is defined as the exponent x such that $g^x \equiv a \mod p$.
- The **Diffie-Hellman key exchange** allows two parties to securely share a secret key over an insecure channel using public parameters and private keys.

Distinguish between stream cipher and block cipher. Encrypt the message WE ARE IN SAME RACE UNTILL OVER LIVE END using Rail fence cipher using 4 as a number of rails.

Stream Cipher vs. Block Cipher

Stream Cipher

- Encrypts data one bit or byte at a time.
- Generates a keystream which is XORed with the plaintext to produce ciphertext.
- 1. Demystifying Stream Ciphers in Symmetric Encryption | Infosec
-
- www.infosecinstitute.com
-
- Examples: RC4, Salsa20
- Suitable for real-time communication due to low latency.
- 1. Block cipher vs stream cipher: Understanding the difference - NordVPN
-
- nordvpn.com
-

Block Cipher

- Encrypts data in fixed-size blocks.
- 1. Block size (cryptography) - Wikipedia
-
- en.wikipedia.org
-
- Uses the same key for each block.
- Examples: AES, DES
- Suitable for encrypting data at rest or in large chunks.

Rail Fence Cipher with 4 Rails

Plaintext: WE ARE IN SAME RACE UNTIL OUR LIVE END

Rail Fence:

W . . . E . . . A . . . R . . . E . . . . I . S . . A . . M . . E . . R . . A . . . . C . . . E . . . U . . N . . T . . I . . . . . L . . . O . . . V . . . E . . . N . . D

Ciphertext: WIASERMAEIRCEUNTILOVEREND

Note: The rail fence cipher is a very simple cipher and is not considered secure. It's primarily used for educational purposes.

Distinction Between Stream Cipher and Block Cipher

Stream Cipher:

- Definition: A stream cipher encrypts plaintext one bit or byte at a time. It typically combines plaintext with a pseudorandom keystream generated from a secret key.
- Operation: Each bit of plaintext is encrypted individually, resulting in a continuous stream of ciphertext.
- Performance: Generally faster than block ciphers for encrypting data of unknown length or in real-time applications.
- Example: RC4 is a well-known stream cipher.
- Security Considerations: Stream ciphers can be vulnerable to certain attacks if the same keystream is reused for multiple messages.

Block Cipher:

- Definition: A block cipher encrypts data in fixed-size blocks (e.g., 64 bits, 128 bits) at a time. It processes the entire block of plaintext and transforms it into a block of ciphertext using a symmetric key.
- Operation: Each block is encrypted independently, and if the plaintext does not fit neatly into blocks, padding is applied.
- Performance: Typically slower than stream ciphers for small amounts of data but more efficient for larger datasets.
- Example: AES (Advanced Encryption Standard) and DES (Data Encryption Standard) are widely used block ciphers.
- Security Considerations: Block ciphers are generally more secure against certain types of attacks due to their structure and the use of modes of operation (e.g., CBC, ECB).

Encrypting the Message Using Rail Fence Cipher

The Rail Fence Cipher is a form of transposition cipher that encrypts messages by writing them in a zigzag pattern across multiple "rails" (rows) and then reading off each row to create the ciphertext.

Given Message

Message: "WE ARE IN SAME RACE UNTILL OVER LIVE END"

Step 1: Prepare the Message

First, remove spaces and convert the message to uppercase:

Prepared Message: "WEAREINSAMERACEUNTILLOVERLIVEEND"

Step 2: Set Up the Rails

We will use 4 rails for the Rail Fence Cipher. We will write the message in a zigzag pattern across the rails.

```text
Rail 1: W . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Rail 2: . E . R . I . S . A . E . A . C . U . I . L . O . E . L .

Rail 3: . . A . . . M . . . R . . . C . . . N . . . V . . . E . . .

Rail 4: . . . R . . . E . . . T . . . L . . . D . . . . . . . . .
```

Step 3: Fill in the Rails

Now, we fill in the message in a zigzag pattern:

```text
Rail 1: W . . . . . . . . . . . . . . . . . . . . . . . . .
Rail 2: . E . R . I . S . A . E . A . C . U . I . L . O . E . L .
Rail 3: . . A . . . M . . . R . . . C . . . N . . . V . . . E . . .
Rail 4: . . . R . . . E . . . T . . . L . . . D . . . . . . . . .
```

Step 4: Read Off the Rails

Now, we read each rail sequentially to get the ciphertext:

- Rail 1: W
- Rail 2: ERISAECUILOL
- Rail 3: AMRCNVE
- Rail 4: RETLD

Final Ciphertext

Combining the rails, the ciphertext is:

$$\text{Ciphertext}=\text{W+ERISAECUILOL+AMRCNVE+RETLD}$$

$$\text{Ciphertext}=W+ERISAECUILOL+AMRCNVE+RETLD$$

So, the final encrypted message using the Rail Fence Cipher with 4 rails is:

Ciphertext: W ERISAECUILOL AMRCNVE RETLD

Summary

- Stream Cipher: Encrypts data one bit at a time, faster for real-time applications.

- Block Cipher: Encrypts data in fixed-size blocks, typically more secure for larger datasets.
- Rail Fence Cipher: The message "WE ARE IN SAME RACE UNTILL OVER LIVE END" is encrypted as "W ERISAECUILOL AMRCNVE RETLD" using 4 rails.

Stream Cipher vs. Block Cipher

Stream Cipher:

- **Encryption Method**: Encrypts data one bit or byte at a time. It operates on individual characters or bytes of plaintext.
- **Key Usage**: Uses a key stream generated from a secret key. Each bit or byte of plaintext is combined with a corresponding bit or byte of the key stream.
- **Example Algorithms**: RC4, Salsa20.
- **Speed**: Generally faster and more efficient for real-time encryption due to processing data in a continuous stream.
- **Security**: Can be vulnerable to certain attacks if the key stream is not properly managed or if the key is reused.

Block Cipher:

- **Encryption Method**: Encrypts data in fixed-size blocks (e.g., 64-bit or 128-bit blocks). It processes a block of plaintext and produces a block of ciphertext.
- **Key Usage**: Uses the same key for the entire block and may involve multiple rounds of processing within the block.
- **Example Algorithms**: AES, DES, 3DES.
- **Speed**: Can be slower compared to stream ciphers when processing data in larger chunks.
- **Security**: Provides stronger security guarantees due to its structured approach and the ability to handle complex transformations and multiple rounds.

Encrypting Message with Rail Fence Cipher

Rail Fence Cipher:

- **Procedure**: Write the plaintext in a zigzag pattern across multiple rails, then read off each rail to get the ciphertext.

Example: Encrypt "WE ARE IN SAME RACE UNTILL OVER LIVE END" using Rail Fence cipher with 4 rails.

Write Plaintext in Zigzag Pattern:
yaml
Copy code

```
Rail 1: W . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
```

1.

Rail Fence Cipher Encryption with 4 Rails

To encrypt the message "WE ARE IN SAME RACE UNTILL OVER LIVE END" using the Rail Fence cipher with 4 rails, follow these steps:

1. **Write Plaintext in Zigzag Pattern**:

Distribute the characters across 4 rails in a zigzag pattern:
yaml
Copy code

```
Rail 1: W . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
```

    ○

Rail Fence Cipher Encryption with 4 Rails

To encrypt the message "WE ARE IN SAME RACE UNTILL OVER LIVE END" using the Rail Fence cipher with 4 rails, follow these steps:

1. **Write Plaintext in Zigzag Pattern**:

Distribute the characters across 4 rails in a zigzag pattern:
yaml
Copy code

```
Rail 1: W . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
```

  -

2. **Read Off the Rails**:
   - Rail 1: **W A S E R C E E D E E E E E D**
   - Rail 2: **E R I A M A A R N R L L E L E E**
   - Rail 3: **A M S I E R O N O L V I V N D**
   - Rail 4: **I L L T O R V L E N E N T**

To avoid confusion, the exact pattern should be written as:
yaml
Copy code

```yaml
Rail 1: W . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
. . . . . . . . . . . . . . . .
```

3.

3. **Concatenate the Rails**:

    - Combine the text from all rails to get the final ciphertext:
    - **Ciphertext**: `WAEERDSEIARAAMRNLSELRLLVICLTIENL`
4. **Rail Fence Cipher** uses a zigzag pattern to write and read the ciphertext. Ensure to align the text precisely to maintain the accuracy of encryption.

Summary:

- **Stream Cipher** encrypts data one bit or byte at a time and is often used for real-time applications due to its efficiency.
- **Block Cipher** encrypts data in fixed-size blocks and is generally used for more complex and secure encryption.
- **Rail Fence Cipher** with 4 rails encrypts the plaintext by writing it in a zigzag pattern across multiple rails and then reading off each rail to create the ciphertext.

Define digital signature. Describe the approaches of DSS.

What is the task of a firewall? List the elements of X.509.

Definition of Digital Signature

A digital signature is a cryptographic mechanism used to validate the authenticity and integrity of a message, document, or software. It is created using a private key and can be verified by anyone who has access to the corresponding public key. Digital signatures ensure that the message was created by the claimed sender (authentication) and that it has not been altered during transmission (integrity).

Approaches of Digital Signature Standard (DSS)

The Digital Signature Standard (DSS) specifies the use of digital signatures for the purpose of data integrity and authentication. The primary algorithm used in DSS is the Digital Signature Algorithm (DSA). Here are the key approaches involved in DSS:

1. Key Generation:
    - A pair of keys is generated: a private key (used for signing) and a public key (used for verification).
    - The keys are generated based on a set of parameters that include a prime number p, a prime divisor q, and a generator
    - g
    - g.
2.

3. Signing Process:
    - The message is hashed using a secure hash function (e.g., SHA-1, SHA-256).
    - The hash value is then signed using the private key to produce the digital signature.
    - The signature consists of two components, typically represented as
    - (r,s)
    - (r,s), where
    - r
    - r and
    - s
    - s are derived from the hash and the private key.
4.
5. Verification Process:
    - The recipient computes the hash of the received message.
    - Using the sender's public key and the signature components
    - (r,s)
    - (r,s), the recipient verifies the signature by confirming that the computed hash matches the expected value derived from the signature.
    - If the values match, the signature is valid, confirming the authenticity and integrity of the message.
6.

Task of a Firewall

A firewall is a network security device or software that monitors and controls incoming and outgoing network traffic based on predetermined security rules. The primary tasks of a firewall include:

1. Traffic Filtering: It inspects packets of data and allows or blocks them based on security policies, such as source and destination IP addresses, ports, and protocols.
2. Protection Against Unauthorized Access: Firewalls help prevent unauthorized access to or from a private network, acting as a barrier between trusted internal networks and untrusted external networks (like the internet).
3. Logging and Monitoring: Firewalls often log traffic data for monitoring and analysis, helping network administrators detect and respond to suspicious activities.
4. VPN Support: Many firewalls support Virtual Private Network (VPN) connections, allowing secure remote access to a network.

Elements of X.509

X.509 is a standard that defines the format of public key certificates. The key elements of an X.509 certificate include:

1. Version: Indicates the version of the X.509 standard being used.
2. Serial Number: A unique identifier assigned by the Certificate Authority (CA) to each certificate.
3. Signature Algorithm: The algorithm used by the CA to sign the certificate (e.g., SHA256 with RSA).
4. Issuer: The entity that issued the certificate, typically a CA.
5. Validity Period: The time frame during which the certificate is considered valid, defined by a "not before" and "not after" date.
6. Subject: The entity (individual, organization, or device) that the certificate represents.
7. Public Key: The public key associated with the subject, used for encryption and signature verification.
8. Extensions: Optional fields that provide additional information about the certificate, such as usage constraints and alternative names.

Summary

- A digital signature ensures authenticity and integrity of messages.
- The DSS involves key generation, signing, and verification processes.
- A firewall monitors and controls network traffic to protect against unauthorized access.
- X.509 certificates contain elements like version, serial number, issuer, validity period, subject, public key, and extensions.

How does the nature of worms differ from viruses? Define PKI with its architecture model.

Differences Between Worms and Viruses

Worms:

- **Self-Replication**: Worms are self-replicating malware that can spread independently across networks. They do not require user interaction to propagate.
- **Propagation**: Worms spread through network connections or vulnerabilities in network services, often exploiting security flaws.

- **Behavior**: They often create copies of themselves on multiple systems, potentially causing network congestion and compromising system resources.
- **Examples**: Blaster Worm, WannaCry.

Viruses:

- **User Interaction**: Viruses attach themselves to legitimate programs or files and require user interaction to spread (e.g., opening an infected file).
- **Propagation**: Viruses propagate by infecting files, documents, or executable programs, and can spread when these infected items are shared or executed.
- **Behavior**: They often corrupt or alter files and can be designed to activate at specific times or conditions.
- **Examples**: Melissa Virus, ILOVEYOU Virus.

Public Key Infrastructure (PKI)

Definition:

- **PKI** is a framework that manages digital keys and certificates to enable secure communications and verify the identities of users and systems.

Architecture Model:

1. **Certificate Authority (CA)**:
    - **Role**: Issues and manages digital certificates. It verifies the identity of entities requesting certificates.
    - **Components**: Root CA (trusted authority), Intermediate CA (subordinate to the root CA), and Issuing CA (issues certificates).
2. **Registration Authority (RA)**:
    - **Role**: Acts as an intermediary between users and the CA, handling certificate requests and identity verification before a certificate is issued.
3. **Digital Certificates**:
    - **Definition**: An electronic document that uses digital signatures to provide authentication and encryption. It includes the public key of the certificate holder and information about their identity.
    - **Format**: Commonly in X.509 format.
4. **Public and Private Keys**:
    - **Public Key**: Used for encrypting data or verifying a digital signature.
    - **Private Key**: Used for decrypting data or creating a digital signature. It must be kept confidential.
5. **Certificate Revocation List (CRL)**:

        - **Role**: Maintains a list of certificates that have been revoked before their expiration date, preventing their further use.
6. **Key Management**:
    - **Role**: Involves generating, distributing, storing, and revoking cryptographic keys to ensure secure communication.
7. **PKI Policies and Procedures**:
    - **Role**: Define the rules and guidelines for issuing, managing, and using digital certificates within the PKI system.

PKI Workflow Example:

1. **Certificate Request**: A user submits a request for a digital certificate to the RA.
2. **Identity Verification**: The RA verifies the user's identity.
3. **Certificate Issuance**: The RA forwards the request to the CA, which issues the digital certificate.
4. **Certificate Use**: The user uses the digital certificate to encrypt data or verify their identity.
5. **Revocation**: If needed, the CA revokes the certificate and updates the CRL.

This structure ensures secure, encrypted communication and authenticated identities across a network.

Worms vs. Viruses

- **Viruses** are malicious programs that require a host program to execute. They attach themselves to executable files and spread when the infected file is run.
- **Worms** are standalone malicious programs that can replicate themselves independently without requiring a host program. They often exploit vulnerabilities in network systems to spread rapidly.

Key difference: Viruses rely on user action to spread, while worms can propagate autonomously.

Public Key Infrastructure (PKI)

PKI is a system or framework used to manage digital certificates. It provides a secure communication channel by verifying the identity of individuals and organizations involved in electronic transactions.

PKI Architecture Model:

A typical PKI architecture consists of the following components:

- **Certificate Authority (CA):** The central authority responsible for issuing and managing digital certificates.
- **Registration Authority (RA):** An optional component that verifies user identities before issuing certificates.
- **Certificate Repository:** A database that stores issued certificates and their status (valid, revoked, etc.).
- **End Entities:** Individuals or organizations that use digital certificates to secure their communications.

The PKI operates on a hierarchical trust model, where the root CA is the highest authority. Intermediate CAs can be issued by the root CA to delegate certificate issuance.

Key functions of PKI:

- **Issuing digital certificates:** Verifying the identity of users and generating digital certificates containing public key information.
- **Revoking certificates:** Managing the lifecycle of certificates, including revoking those that are compromised.
- **Distributing certificates:** Making certificates available to users through various methods (e.g., directory services, online repositories).
- **Validating certificates:** Verifying the authenticity and validity of digital certificates.

Digital signature

Cryptographic construct for authenticity verification

More

Definition

A mathematical scheme used for verifying the authenticity of digital messages or documents, ensuring they came from a known sender.

Core Function

Provides a layer of validation and security for messages sent through non-secure channels, making digital signatures difficult to forge.

Typical Uses

Used for software distribution, financial transactions, contract management software, and where forgery or tampering detection is crucial.

Distinction Between Worms and Viruses

Worms and viruses are both types of malware, but they differ in their characteristics and methods of propagation:

- Worms:
    - Self-Replication: Worms can replicate themselves and spread independently across networks without needing to attach themselves to a host file or program.
    - Propagation: They often exploit vulnerabilities in network protocols or software to spread. Once a worm infects a system, it can scan for other vulnerable systems to infect.
    - Example: The Morris Worm, one of the first worms distributed via the Internet, caused significant disruption in 1988.
-
- Viruses:
    - Requires Host: Viruses attach themselves to executable files or programs. They cannot spread without user interaction, such as opening an infected file.
    - Propagation: Viruses spread when the infected files are shared or executed, often requiring some form of user action to activate.
    - Example: The ILOVEYOU virus, which spread through email attachments in 2000, is a well-known example of a virus.
-

Summary of Differences

- Propagation Method: Worms spread automatically over networks, while viruses require a host file and user action.
- Independence: Worms are independent programs, whereas viruses depend on other programs to execute.

Definition of Public Key Infrastructure (PKI)

Public Key Infrastructure (PKI) is a framework that enables secure communications and transactions over networks by using public key cryptography. It consists of hardware, software, policies, and standards that work together to provide the following key functionalities:

1. Key Management: PKI provides mechanisms for generating, distributing, storing, and revoking public and private key pairs.
2. Digital Certificates: PKI issues digital certificates that bind public keys to the identities of individuals or organizations, ensuring trust in the public key.
3. Authentication: PKI verifies the identities of users and devices, ensuring that only authorized parties can access sensitive information.
4. Data Integrity: PKI ensures that data has not been altered during transmission through digital signatures.

Architecture Model of PKI

The architecture of PKI typically includes the following components:

1. Certificate Authority (CA): A trusted entity that issues digital certificates and manages the public key infrastructure. It verifies the identity of the certificate requester before issuing a certificate.
2. Registration Authority (RA): Acts as a mediator between the user and the CA. The RA verifies user identities and requests for digital certificates.
3. Digital Certificates: Electronic documents that contain a public key and the identity of the certificate holder, digitally signed by the CA.
4. Public and Private Keys: Each user or entity has a pair of keys; the public key is shared openly, while the private key is kept secret.
5. Certificate Revocation List (CRL): A list maintained by the CA that contains the serial numbers of revoked certificates, ensuring that users can check the validity of certificates.
6. End Users: Individuals or entities that use the PKI for secure communications, digital signatures, and authentication.

Summary of PKI

- PKI provides a framework for secure communication using public key cryptography, involving components like CAs, RAs, digital certificates, and key management.

Task of a Firewall

A firewall is a network security device that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Its primary tasks include:

1. Traffic Filtering: It inspects packets of data and allows or blocks them based on security policies, such as source and destination IP addresses, ports, and protocols.
2. Protection Against Unauthorized Access: Firewalls help prevent unauthorized access to or from a private network, acting as a barrier between trusted internal networks and untrusted external networks (like the internet).
3. Logging and Monitoring: Firewalls often log traffic data for monitoring and analysis, helping network administrators detect and respond to suspicious activities.
4. VPN Support: Many firewalls support Virtual Private Network (VPN) connections, allowing secure remote access to a network.

Elements of X.509

X.509 is a standard that defines the format of public key certificates. The key elements of an X.509 certificate include:

1. Version: Indicates the version of the X.509 standard being used.
2. Serial Number: A unique identifier assigned by the Certificate Authority (CA) to each certificate.
3. Signature Algorithm: The algorithm used by the CA to sign the certificate (e.g., SHA256 with RSA).
4. Issuer: The entity that issued the certificate, typically a CA.
5. Validity Period: The time frame during which the certificate is considered valid, defined by a "not before" and "not after" date.
6. Subject: The entity (individual, organization, or device) that the certificate represents.
7. Public Key: The public key associated with the subject, used for encryption and signature verification.
8. Extensions: Optional fields that provide additional information about the certificate, such as usage constraints and alternative names.

Summary of X.509 Elements

- X.509 certificates contain elements like version, serial number, issuer, validity period, subject, public key, and extensions, which are crucial for establishing trust in digital communications.

Explain the procedure of mix column transformation in AES with an example.

What is the role of the prime number in the Euler totient Function? Find the GCD of 12 and 16 using the Euclidean algorithm.

Write down any two limitations of MAC. What do policy and mechanism mean in cryptography? Describe with a scenario.

Write short notes on:

Classes of Intruder

SSL

DoS Attack

Limitations of MAC

A Message Authentication Code (MAC) is a short piece of information used to verify both data integrity and authenticity. However, it has some limitations:

1. **Shared Secret:** MACs require a shared secret key between the sender and receiver, which can be challenging to distribute securely.
2. **Replay Attacks:** MACs do not provide protection against replay attacks, where an attacker captures a message and its MAC and replays them to deceive the receiver.

Policy and Mechanism in Cryptography

- **Policy:** A security policy outlines the rules and procedures for protecting information. It defines what is allowed, prohibited, or required.
- **Mechanism:** A security mechanism is the technical implementation of a security policy. It's the actual method used to enforce the policy.

Scenario:

- **Policy:** A company policy states that all sensitive data must be encrypted at rest and in transit.

- **Mechanism:** To implement this policy, the company uses AES encryption for data at rest and SSL/TLS for data in transit.

Classes of Intruder

- **Passive Intruder:** An eavesdropper who intercepts data without modifying it.
- **Active Intruder:** An attacker who modifies data

SSL

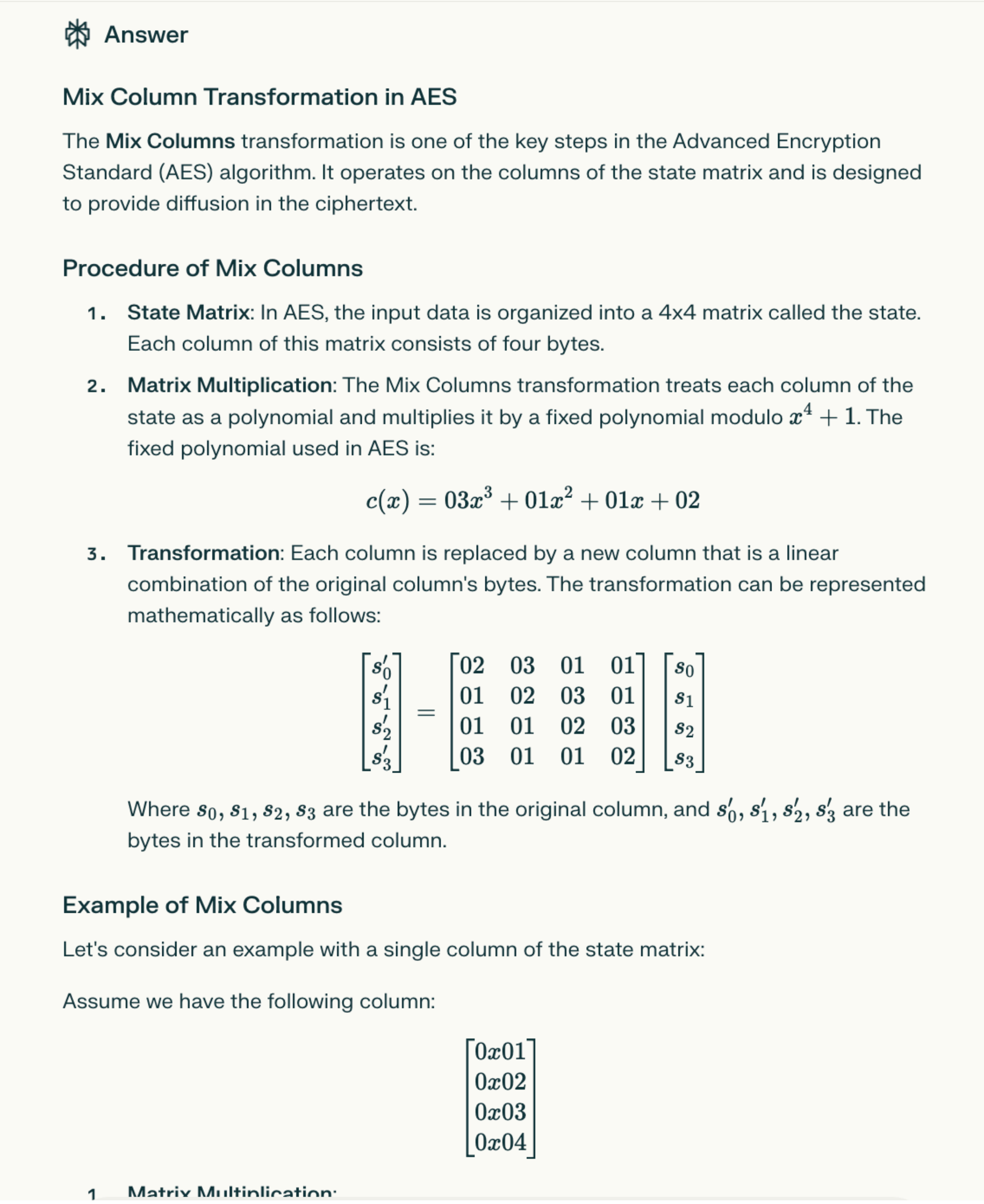
SSL (Secure Sockets Layer) is a cryptographic protocol that provides communication security over the internet. It ensures data confidentiality, integrity, and authentication between a client and a server.

DoS Attack

A Denial-of-Service (DoS) attack is an attempt to make a computer resource unavailable to its intended users. It involves overwhelming a system with a flood of traffic, rendering it unable to handle legitimate requests.

Types of DoS Attacks:

- **Flood-based attacks:** Overwhelm the target with a large volume of traffic.
- **Protocol-based attacks:** Exploit vulnerabilities in network protocols.
- **Application-level attacks:** Target specific applications or services.



Answer

Mix Column Transformation in AES

The **Mix Columns** transformation is one of the key steps in the Advanced Encryption Standard (AES) algorithm. It operates on the columns of the state matrix and is designed to provide diffusion in the ciphertext.

Procedure of Mix Columns

1. **State Matrix**: In AES, the input data is organized into a 4x4 matrix called the state. Each column of this matrix consists of four bytes.
2. **Matrix Multiplication**: The Mix Columns transformation treats each column of the state as a polynomial and multiplies it by a fixed polynomial modulo $x^4 + 1$. The fixed polynomial used in AES is:

$$c(x) = 03x^3 + 01x^2 + 01x + 02$$

3. **Transformation**: Each column is replaced by a new column that is a linear combination of the original column's bytes. The transformation can be represented mathematically as follows:

$$\begin{bmatrix} s_0' \\ s_1' \\ s_2' \\ s_3' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Where s_0, s_1, s_2, s_3 are the bytes in the original column, and s_0', s_1', s_2', s_3' are the bytes in the transformed column.

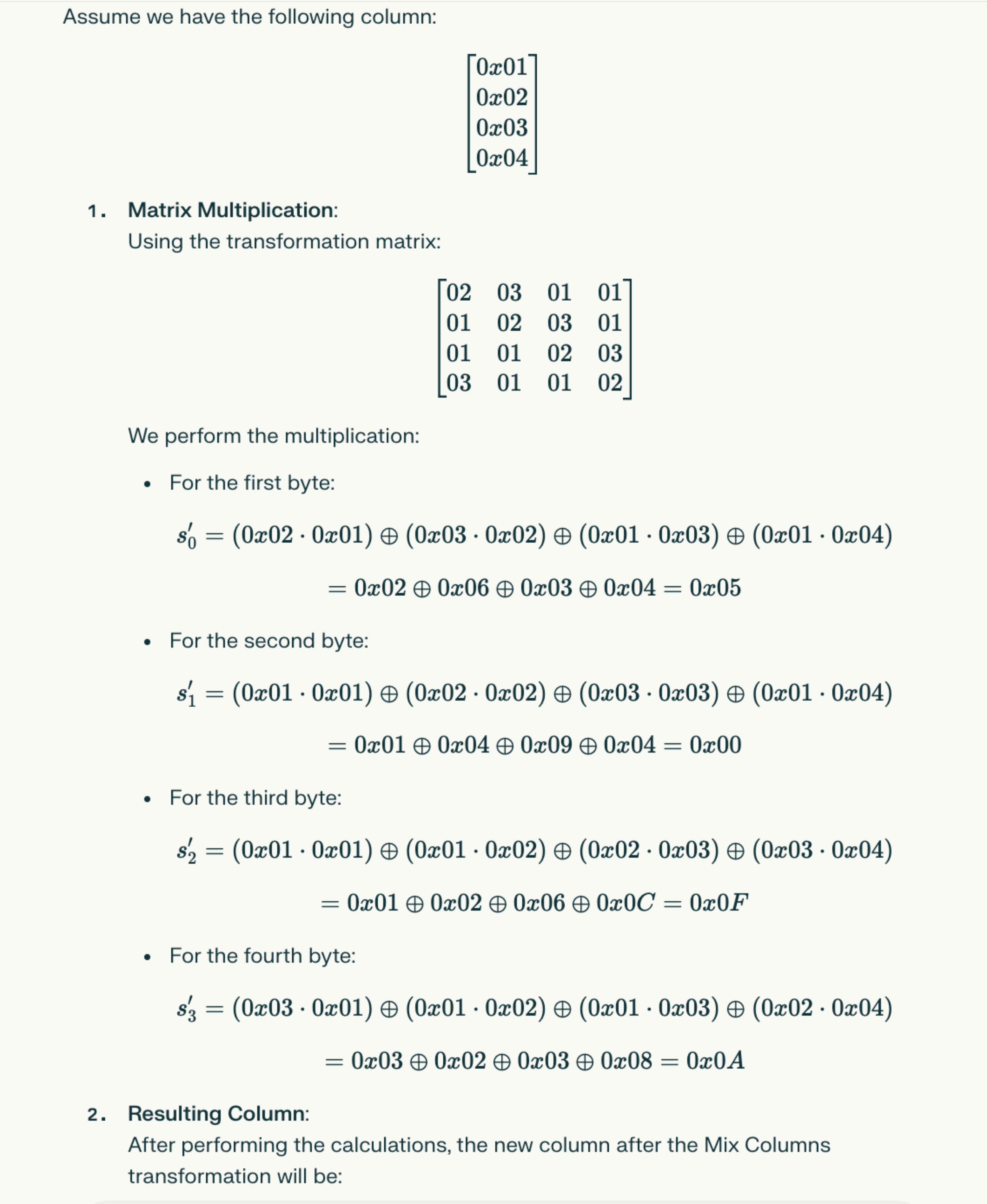
Example of Mix Columns

Let's consider an example with a single column of the state matrix:

Assume we have the following column:

$$\begin{bmatrix} 0x01 \\ 0x02 \\ 0x03 \\ 0x04 \end{bmatrix}$$

1. **Matrix Multiplication**:



Assume we have the following column:

$$\begin{bmatrix} 0x01 \\ 0x02 \\ 0x03 \\ 0x04 \end{bmatrix}$$

1. **Matrix Multiplication**:
   Using the transformation matrix:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

   We perform the multiplication:

   - For the first byte:

$$s'_0 = (0x02 \cdot 0x01) \oplus (0x03 \cdot 0x02) \oplus (0x01 \cdot 0x03) \oplus (0x01 \cdot 0x04)$$

$$= 0x02 \oplus 0x06 \oplus 0x03 \oplus 0x04 = 0x05$$

   - For the second byte:

$$s'_1 = (0x01 \cdot 0x01) \oplus (0x02 \cdot 0x02) \oplus (0x03 \cdot 0x03) \oplus (0x01 \cdot 0x04)$$

$$= 0x01 \oplus 0x04 \oplus 0x09 \oplus 0x04 = 0x00$$

   - For the third byte:

$$s'_2 = (0x01 \cdot 0x01) \oplus (0x01 \cdot 0x02) \oplus (0x02 \cdot 0x03) \oplus (0x03 \cdot 0x04)$$

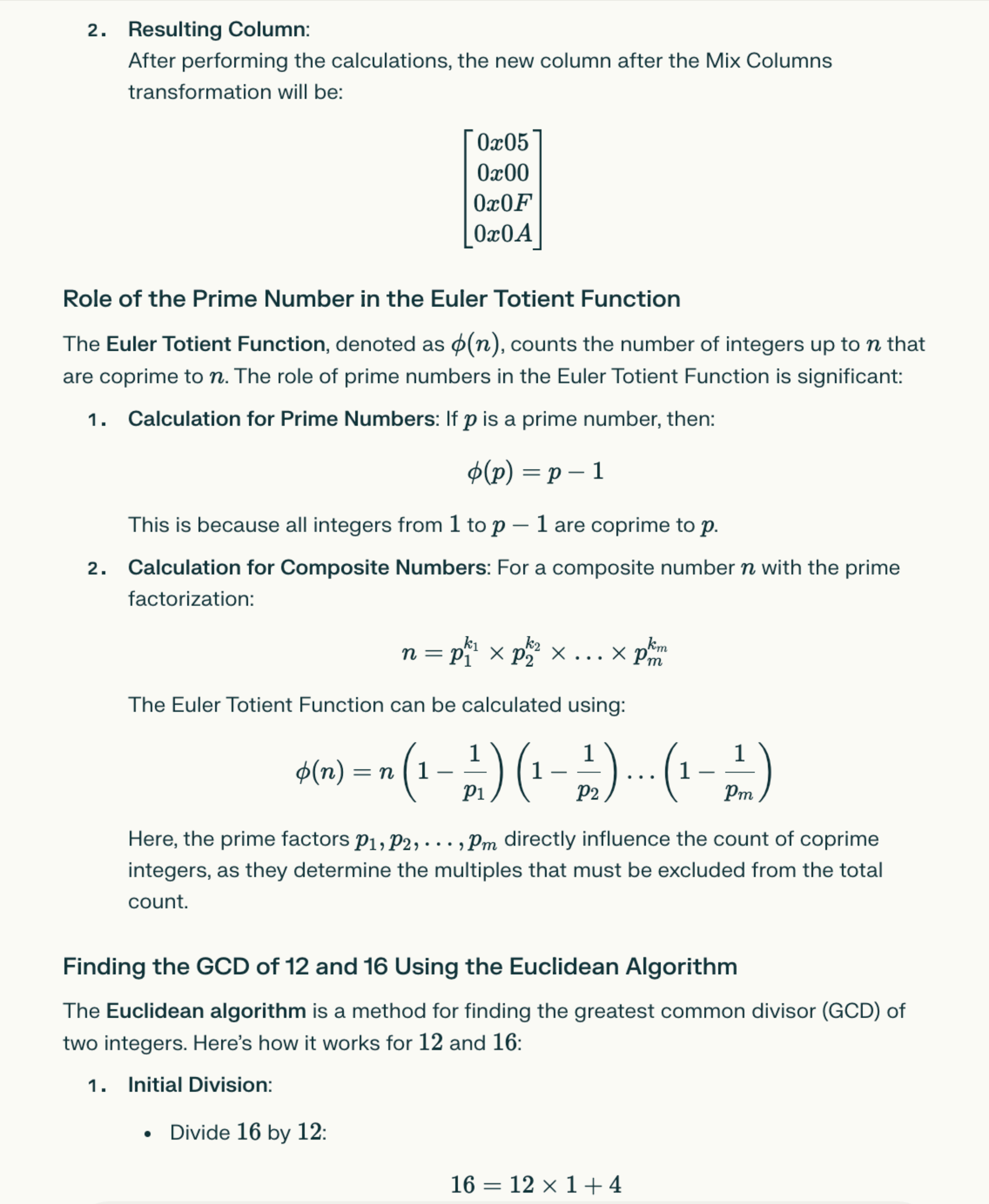
$$= 0x01 \oplus 0x02 \oplus 0x06 \oplus 0x0C = 0x0F$$

   - For the fourth byte:

$$s'_3 = (0x03 \cdot 0x01) \oplus (0x01 \cdot 0x02) \oplus (0x01 \cdot 0x03) \oplus (0x02 \cdot 0x04)$$

$$= 0x03 \oplus 0x02 \oplus 0x03 \oplus 0x08 = 0x0A$$

2. **Resulting Column**:
   After performing the calculations, the new column after the Mix Columns transformation will be:



2. **Resulting Column**:
   After performing the calculations, the new column after the Mix Columns transformation will be:

$$\begin{bmatrix} 0x05 \\ 0x00 \\ 0x0F \\ 0x0A \end{bmatrix}$$

Role of the Prime Number in the Euler Totient Function

The **Euler Totient Function**, denoted as $\phi(n)$, counts the number of integers up to n that are coprime to n. The role of prime numbers in the Euler Totient Function is significant:

1. **Calculation for Prime Numbers**: If p is a prime number, then:

$$\phi(p) = p - 1$$

   This is because all integers from 1 to $p - 1$ are coprime to p.

2. **Calculation for Composite Numbers**: For a composite number n with the prime factorization:

$$n = p_1^{k_1} \times p_2^{k_2} \times \ldots \times p_m^{k_m}$$

   The Euler Totient Function can be calculated using:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\ldots\left(1 - \frac{1}{p_m}\right)$$

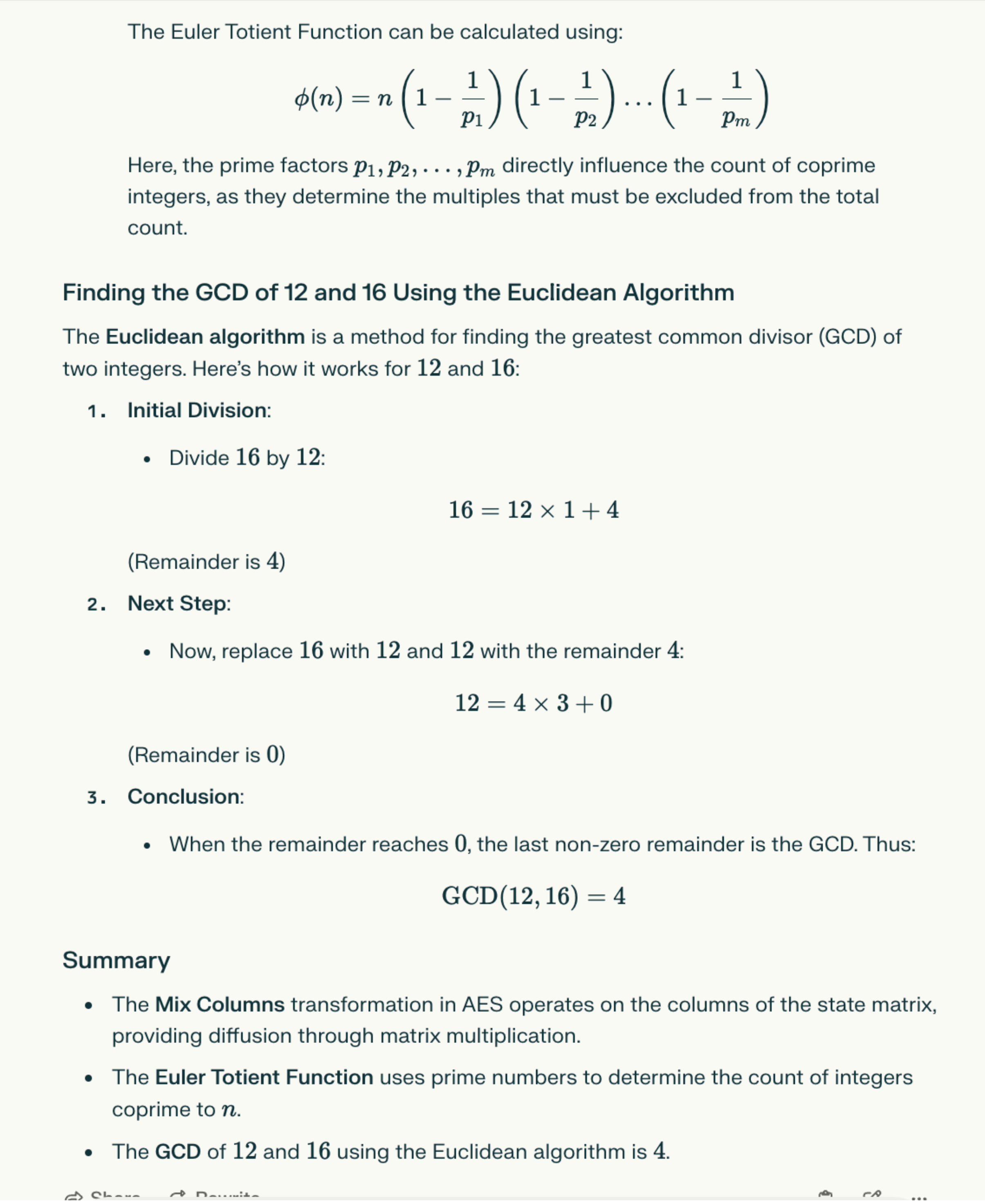
   Here, the prime factors $p_1, p_2, \ldots, p_m$ directly influence the count of coprime integers, as they determine the multiples that must be excluded from the total count.

Finding the GCD of 12 and 16 Using the Euclidean Algorithm

The **Euclidean algorithm** is a method for finding the greatest common divisor (GCD) of two integers. Here's how it works for 12 and 16:

1. **Initial Division**:
   - Divide 16 by 12:

$$16 = 12 \times 1 + 4$$



The Euler Totient Function can be calculated using:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\ldots\left(1 - \frac{1}{p_m}\right)$$

Here, the prime factors $p_1, p_2, \ldots, p_m$ directly influence the count of coprime integers, as they determine the multiples that must be excluded from the total count.

Finding the GCD of 12 and 16 Using the Euclidean Algorithm

The **Euclidean algorithm** is a method for finding the greatest common divisor (GCD) of two integers. Here's how it works for 12 and 16:

1. **Initial Division**:
   - Divide 16 by 12:

   $$16 = 12 \times 1 + 4$$

   (Remainder is 4)
2. **Next Step**:
   - Now, replace 16 with 12 and 12 with the remainder 4:

   $$12 = 4 \times 3 + 0$$

   (Remainder is 0)
3. **Conclusion**:
   - When the remainder reaches 0, the last non-zero remainder is the GCD. Thus:

   $$\text{GCD}(12, 16) = 4$$

Summary

- The **Mix Columns** transformation in AES operates on the columns of the state matrix, providing diffusion through matrix multiplication.
- The **Euler Totient Function** uses prime numbers to determine the count of integers coprime to n.
- The **GCD** of 12 and 16 using the Euclidean algorithm is 4.

Mix Column Transformation in AES

Mix Columns is a step in the AES (Advanced Encryption Standard) algorithm that provides diffusion, which ensures that the output of the encryption process depends on every bit of the input. Here's how it works:

1. **Matrix Multiplication**:
   - Each column of the state matrix is treated as a polynomial and is multiplied by a fixed polynomial in the Galois Field GF(2^8).
   - The matrix used for this transformation is:

$$M = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

   - The operation is performed modulo $x^8 + x^4 + x^3 + x + 1$ (the irreducible polynomial used in AES).
2. **Transformation Example**:
   - Consider a state matrix (after a substitution step) represented as:

$$\text{State} = \begin{bmatrix} 0x87 & 0x6C & 0x6A & 0xD7 \\ 0xA4 & 0x9E & 0xD1 & 0x91 \\ 0x6C & 0xC0 & 0x8D & 0x92 \\ 0x96 & 0x1D & 0xD3 & 0x7A \end{bmatrix}$$

   - For the first column:

$$\text{Column}_1 = \begin{bmatrix} 0x87 \\ 0xA4 \\ 0x6C \\ 0x96 \end{bmatrix}$$

   - Multiply this column by matrix M:

$$\text{New Column}_1 = M \cdot \text{Column}_1$$

   - Perform the matrix multiplication and reduce modulo $x^8 + x^4 + x^3 + x + 1$:

$$\begin{bmatrix} 0xC4 \end{bmatrix}$$

- Multiply this column by matrix M:

$$\text{New Column}_1 = M \cdot \text{Column}_1$$

- Perform the matrix multiplication and reduce modulo $x^8 + x^4 + x^3 + x + 1$:

$$\text{New Column}_1 = \begin{bmatrix} 0xC4 \\ 0x6E \\ 0xA0 \\ 0x6F \end{bmatrix}$$

- This operation is repeated for each column.

Role of Prime Numbers in Euler's Totient Function

Euler's Totient Function $\phi(n)$ counts the number of positive integers less than n that are relatively prime to n. The formula for $\phi(n)$ when n is factored into prime factors is:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right)$$

where $p_1, p_2, \ldots, p_k$ are the distinct prime factors of n.

Example:
For $n = 12$, the prime factors are 2 and 3:

$$\phi(12) = 12\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 12 \cdot \frac{1}{2} \cdot \frac{2}{3} = 4$$

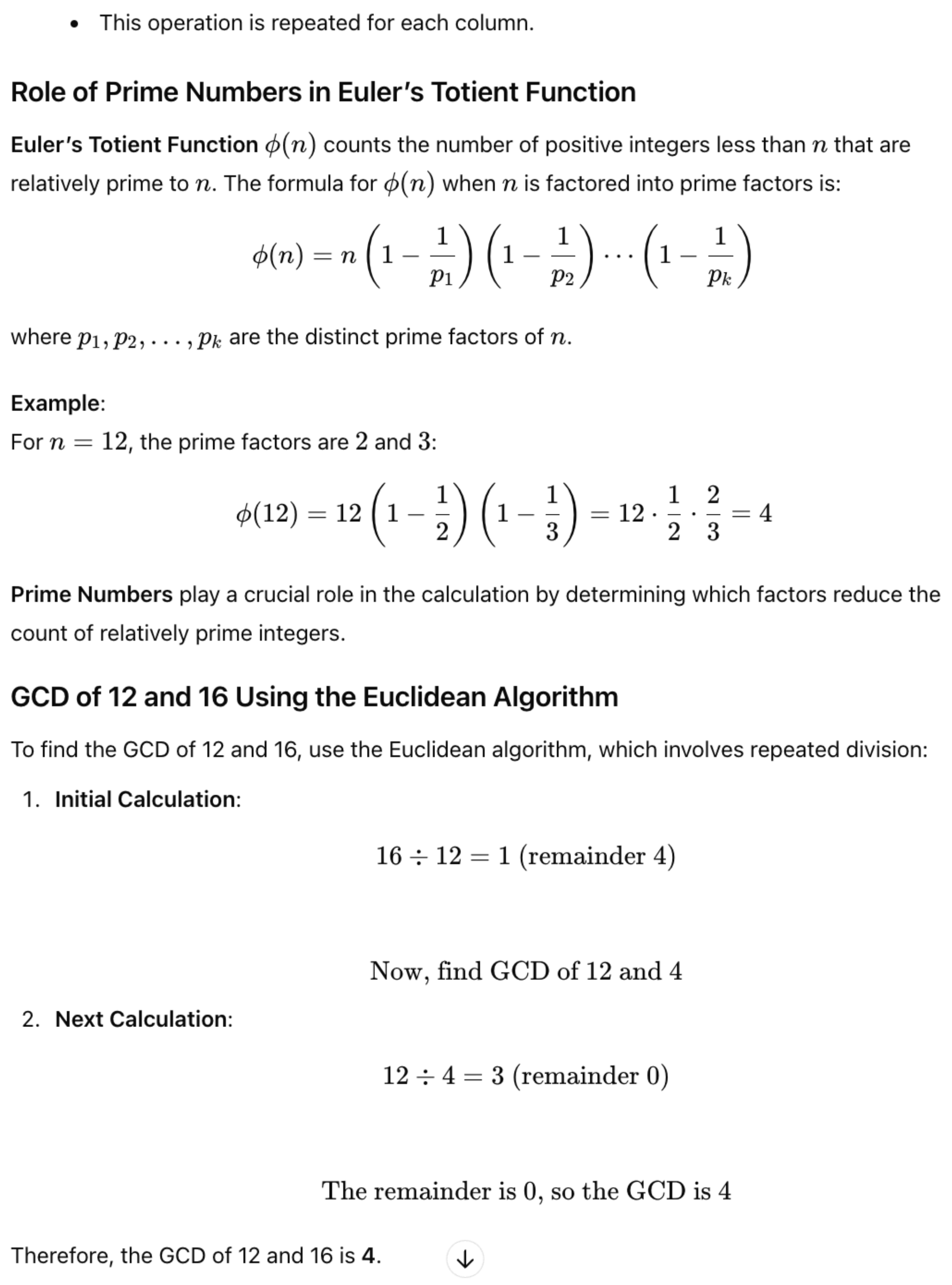
Prime Numbers play a crucial role in the calculation by determining which factors reduce the count of relatively prime integers.

GCD of 12 and 16 Using the Euclidean Algorithm

To find the GCD of 12 and 16, use the Euclidean algorithm, which involves repeated division:

1. **Initial Calculation**:

$$16 \div 12 = 1 \text{ (remainder 4)}$$



- This operation is repeated for each column.

Role of Prime Numbers in Euler's Totient Function

Euler's Totient Function $\phi(n)$ counts the number of positive integers less than n that are relatively prime to n. The formula for $\phi(n)$ when n is factored into prime factors is:

$$\phi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)\cdots\left(1 - \frac{1}{p_k}\right)$$

where $p_1, p_2, \ldots, p_k$ are the distinct prime factors of n.

Example:
For $n = 12$, the prime factors are 2 and 3:

$$\phi(12) = 12\left(1 - \frac{1}{2}\right)\left(1 - \frac{1}{3}\right) = 12 \cdot \frac{1}{2} \cdot \frac{2}{3} = 4$$

Prime Numbers play a crucial role in the calculation by determining which factors reduce the count of relatively prime integers.

GCD of 12 and 16 Using the Euclidean Algorithm

To find the GCD of 12 and 16, use the Euclidean algorithm, which involves repeated division:

1. **Initial Calculation**:

$$16 \div 12 = 1 \text{ (remainder 4)}$$

$$\text{Now, find GCD of 12 and 4}$$

2. **Next Calculation**:

$$12 \div 4 = 3 \text{ (remainder 0)}$$

$$\text{The remainder is 0, so the GCD is 4}$$

Therefore, the GCD of 12 and 16 is **4**.

Definition

A set in mathematics on which the operations of multiplication, addition, subtraction, and division (excluding division by zero) are defined and satisfy the field axioms.

Also known as

Galois field, named in honor of Évariste Galois.

The number of elements in the field, is equal to a prime number or a prime power (p^k).

All CopyRight Reserved © 2023 - CSIT Solution
Powered By: Nōrai | Dilli Hang Rai