Applied Cryptographic Mathematics

Advanced Number Theory and Elliptic Curve Mathematics

Jason Weiss Zeledón

July 27, 2025

Abstract

This document presents a sophisticated cryptographic challenge that demonstrates the practical application of advanced mathematical concepts in modern cryptography. The cipher employs six distinct mathematical transformations: elliptic curve scalar multiplication over finite fields, Carmichael lambda function computation, quadratic residue encoding with Legendre symbols, Fermat's Little Theorem applications, Chinese Remainder Theorem systems, and steganographic techniques. This challenge serves as both an educational tool and a demonstration of how mathematical complexity can provide cryptographic security without relying on computational secrecy.

1 Introduction

The Mathematical Labyrinth Cipher represents a convergence of pure mathematics and applied cryptography, designed to challenge participants' understanding of advanced number theory while demonstrating real-world cryptographic principles. Unlike traditional ciphers that rely on key secrecy, this system achieves security through mathematical complexity—all algorithms are publicly available, yet the solution requires deep mathematical insight.

1.1 Challenge Overview

This challenge combines multiple layers of advanced mathematics and cryptography to conceal a famous quotation from cryptographic literature. Participants must understand and reverse-engineer several mathematical transformations to reveal the concealed message.

- **Estimated Completion Time:** 20–60 hours for experienced cryptographers
- **Prerequisites:** Understanding of modular arithmetic, programming experience recommended
- **Difficulty Level:** Graduate-level mathematics and cryptography

1.2 Challenge Rules

1. **No Brute Force Attacks:** The cipher is designed to be mathematically challenging, not computationally intensive
2. **Open Source Implementation:** All encoding algorithms are publicly available—the difficulty lies in their mathematical reversal
3. **Mathematical Reasoning Required:** Success depends entirely on understanding the underlying mathematics
4. **Unique Solution:** There is exactly one correct answer

2 Mathematical Foundations

2.1 Modular Arithmetic

Modular arithmetic forms the foundation of modern cryptography. For integers a, b, and positive integer m, we write:

$$a \equiv b \pmod{m} \tag{1}$$

if m divides $(a-b)$. This relationship partitions the integers into equivalence classes modulo m.

Example: $25 \equiv 4 \pmod{7}$ because $25 = 3 \times 7 + 4$.

2.2 Elliptic Curves

An elliptic curve over a finite field $\mathbb{F}_p$ is defined by the Weierstrass equation:

$$y^2 \equiv x^3 + ax + b \pmod{p} \tag{2}$$

where $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$ to ensure non-singularity.

For this challenge, we employ the `secp256k1` curve:

$$y^2 \equiv x^3 + 7 \pmod{p} \tag{3}$$

where $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$.

2.2.1 Key Properties

- **Point Addition:** Points on the curve form an abelian group under a geometric addition law
- **Scalar Multiplication:** For point P and integer k, we compute kP via repeated addition
- **Discrete Logarithm Problem:** Given points P and $Q = kP$, finding k is computationally infeasible

2.3 Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) provides a method for solving systems of linear congruences with pairwise coprime moduli.

[Chinese Remainder Theorem] Let $m_1, m_2, \ldots, m_k$ be pairwise coprime positive integers, and let $a_1, a_2, \ldots, a_k$ be arbitrary integers. Then the system of congruences:

$$x \equiv a_1 \pmod{m_1} \tag{4}$$
$$x \equiv a_2 \pmod{m_2} \tag{5}$$
$$\vdots \tag{6}$$
$$x \equiv a_k \pmod{m_k} \tag{7}$$

has a unique solution modulo $M = m_1 m_2 \cdots m_k$.

Construction: The solution is given by:

$$x \equiv \sum_{i=1}^{k} a_i M_i y_i \pmod{M} \tag{8}$$

where $M_i = M/m_i$ and y_i is the modular inverse of M_i modulo m_i.

2.4 Fermat's Little Theorem

[Fermat's Little Theorem] If p is prime and a is not divisible by p, then:

$$a^{p-1} \equiv 1 \pmod{p} \tag{9}$$

This theorem implies that $a^p \equiv a \pmod{p}$ for any integer a, providing the foundation for efficient modular exponentiation algorithms.

2.5 Quadratic Residues and the Legendre Symbol

A number a is a *quadratic residue* modulo prime p if there exists an integer x such that:

$$x^2 \equiv a \pmod{p} \tag{10}$$

The Legendre symbol $\left(\frac{a}{p}\right)$ determines quadratic residuosity:

$$\left(\frac{a}{p}\right) = \begin{cases} 1 & \text{if } a \text{ is a quadratic residue mod } p \\ -1 & \text{if } a \text{ is a quadratic non-residue mod } p \\ 0 & \text{if } a \equiv 0 \pmod{p} \end{cases} \tag{11}$$

The Legendre symbol can be computed using:

$$\left(\frac{a}{p}\right) \equiv a^{(p-1)/2} \pmod{p} \tag{12}$$

2.6 Carmichael Lambda Function

The Carmichael function $\lambda(n)$ is the smallest positive integer such that:

$$a^{\lambda(n)} \equiv 1 \pmod{n} \tag{13}$$

for all integers a coprime to n.
For prime powers: $\lambda(p^k) = p^{k-1}(p-1)$
For composite numbers: $\lambda(n) = \text{lcm}(\lambda(p_1^{k_1}), \lambda(p_2^{k_2}), \ldots)$

3 Cipher Architecture

3.1 Encoding Pipeline

For each character c in the secret message, the encoding process applies six mathematical transformations:

1. **Elliptic Curve Mapping:**

$$\text{char\_val} = \text{ASCII}(c) \tag{14}$$
$$\text{seed} = \text{char\_val} \times 10^6 + 314159 \tag{15}$$
$$P = \text{seed} \times G \text{ (scalar multiplication on secp256k1)} \tag{16}$$
$$\text{ec\_x} = P.x \tag{17}$$

2. **Carmichael Lambda Calculation:**

$$\lambda\_\text{result} = \lambda(\text{char\_val}) \tag{18}$$

3. **Quadratic Residue Encoding:** Find the smallest integer ≥ char_val that is a quadratic residue modulo 1009:

$$\text{qr\_encoded} = \min\{x \geq \text{char\_val} : \left(\frac{x}{1009}\right) = 1\} \tag{19}$$

4. **Fermat Transform:**

$$\text{fermat\_result} = \text{char\_val}^{\text{position}+1} \bmod 170141183460469231731687303715884105727 \tag{20}$$

5. **Chinese Remainder Theorem Combination:** Solve the system:

$$x \equiv \text{ec\_x} \pmod{97} \tag{21}$$
$$x \equiv \lambda\text{\_result} \pmod{101} \tag{22}$$
$$x \equiv \text{qr\_encoded} \pmod{103} \tag{23}$$

6. **Final Prime Field Operation:**

$$\text{final\_encoded} = (\text{crt\_result} \times \text{fermat\_result}) \bmod (2^{521} - 1) \tag{24}$$

Random noise is added for obfuscation:

$$\text{output} = \text{final\_encoded} + \text{random\_noise} \times (2^{521} - 1) \tag{25}$$

3.2 Key Mathematical Constants

- **Elliptic Curve:** secp256k1 ($y^2 = x^3 + 7$)
- **Generator Point** G**:**

$$G_x = 55066263022277343669578718895168534326250603453777594175500187360389116729240 \tag{26}$$

$$G_y = 32670510020758816978083085130507043184471273380659243275938904335757337482424 \tag{27}$$

- **CRT Moduli:** $[97, 101, 103]$
- **Fermat Prime:** 170141183460469231731687303715884105727
- **QR Prime:** 1009
- **Final Field:** $2^{521} - 1$ (Mersenne prime)

4 Solution Methodology

4.1 Phase 1: Understanding the Structure

Participants must first analyze the open-source implementation to understand:

- The mathematical operations applied to each character
- The order and interdependencies of transformations
- The encoding pipeline from character to final number

4.2 Phase 2: Reverse Engineering

The solution requires working backwards from the encoded numbers:

1. **Remove Noise:** Apply modulo Mersenne prime $2^{521} - 1$
2. **Separate Components:** Factor the result into CRT and Fermat components
3. **Solve CRT System:** Extract the three remainders
4. **Reverse Transformations:** Map each remainder back to its source

4.3 Phase 3: Character Recovery

For each character position:

1. Use mathematical relationships to determine the original ASCII value
2. Verify consistency across all transformations
3. Build the complete message incrementally

4.4 Phase 4: Validation

1. Reconstruct the complete message
2. Verify using the provided hash verification system
3. Ensure the result matches the expected format

5 Verification System

The correct solution's SHA256 hash begins with a specific prefix. The solution should be a famous quote from cryptographic literature, submitted with proper capitalization and spacing.

Format Requirements:

- Submit the exact text with proper capitalization and spacing
- Example: If the answer is "Hello World", submit: `Hello World`

6 Hints and Guidance

6.1 Mathematical Hints

1. Look for patterns in the Carmichael lambda values—they might reveal character properties
2. The elliptic curve points follow predictable patterns based on the character encoding
3. Consider the relationship between ASCII values and the mathematical transformations

6.2 Programming Implementation

1. Use arbitrary precision arithmetic—regular integers will overflow
2. Implement modular inverse using the Extended Euclidean Algorithm
3. Test with small examples before working on the full dataset
4. Verify each step of reverse engineering with the forward encoding

6.3 Debugging Strategies

1. Start with character 1 and trace through all transformations
2. Compare intermediate results with provided example sequences
3. Use the mathematical constants exactly as specified
4. Verify your CRT implementation with simple test cases

7 Educational Value

This challenge serves multiple educational purposes:

7.1 Mathematical Understanding

- **Abstract Algebra:** Group theory through elliptic curve operations
- **Number Theory:** Practical applications of modular arithmetic and multiplicative functions
- **Algorithmic Complexity:** Understanding polynomial vs. exponential time problems

7.2 Cryptographic Concepts

- **Modern Cryptography:** Real-world applications of mathematical primitives
- **Security Analysis:** Understanding computational vs. information-theoretic security
- **Implementation Skills:** Translating theoretical concepts into working code

8 Computational Resources

8.1 Recommended Tools

- **Wolfram Alpha:** For mathematical calculations and verification
- **SageMath:** For elliptic curve computations and number theory
- **Python with SymPy:** For implementing mathematical algorithms

8.2 Required Algorithms

1. **Extended Euclidean Algorithm:** For computing modular inverses
2. **Chinese Remainder Theorem Solver:** For combining modular constraints
3. **Elliptic Curve Arithmetic:** Point addition and scalar multiplication
4. **Legendre Symbol Computation:** For quadratic residue testing

9 Security Considerations

Important Notice: This cipher is designed for educational purposes and should never be used for real-world cryptographic applications. Several design choices prioritize learning over security:

- Small moduli allow exhaustive search attacks
- Deterministic encoding enables frequency analysis
- No key derivation or authentication mechanisms

10 Conclusion

The Mathematical Labyrinth Cipher demonstrates the profound connection between pure mathematics and applied cryptography. Through the implementation of elliptic curve operations, number theoretic functions, and algebraic structures, we observe how mathematical abstractions translate into practical security mechanisms.

The challenge's design philosophy emphasizes transparency in method while maintaining complexity in execution—a hallmark of well-designed cryptographic educational tools. By making all algorithms publicly available, we demonstrate that security emerges from mathematical hardness rather than algorithmic secrecy.

Success in this challenge requires not just computational skill, but deep mathematical understanding and systematic problem-solving abilities. These skills form the foundation of modern cryptographic research and practice.

This challenge was designed to demonstrate the intersection of pure mathematics and applied cryptography in an educational context.