

Applications of Number Theory in Digital Communication

Dr. Vijayalaxmi Patil¹, Miss. Pallavi G. Kulkarni²

¹Department of Mathematics, Raja Lakhamagouda Science Institute(Autonomous), Belagavi

²Department of P.G. Studies in Mathematics, Raja Lakhamagouda Science Institute(Autonomous), Belagavi

Abstract - This paper explores the role of Number Theory in digital communication and cyber security. It explains how concepts such as prime numbers and modular arithmetic support cryptographic methods like RSA, Diffie–Hellman key exchange, and digital signatures. Through ATM and WhatsApp case studies and Python implementations, the research demonstrates how Number Theory helps ensure secure communication, data protection, authentication and integrity.

Key Words: Number Theory, cyber security, RSA, Diffie–Hellman key exchange, data protection, authentication and integrity.

1. INTRODUCTION

Digital communication involves the exchange of information between computers, mobile devices, servers, and other electronic systems through the Internet or communication networks. Since sensitive information such as passwords, banking details, emails, and confidential documents is transmitted over public networks, protecting this data is essential.

The main applications of number theory in digital communication include:

- Cryptography and Encryption
- ATM Transaction
- Digital Certificate
- Digital Signature

2. CASE STUDY: APPLICATION OF NUMBER THEORY IN ATM TRANSACTIONS

2.1 INTRODUCTION:

Automated Teller Machines (ATMs) are one of the most widely used banking technologies. Every day, millions of people perform financial transactions such as cash withdrawal, balance inquiry, fund transfer, PIN change, and mini statement generation. Since these transactions involve confidential information such as account numbers, ATM PINs, transaction amounts, and customer identity, they must be protected from cybercriminals.

The security of ATM transactions is based on **Number Theory**, which provides the mathematical foundation for cryptographic algorithms. Concepts such as **Prime Numbers**, **Greatest Common Divisor (GCD)**, **Modular Arithmetic**, **Euler's Totient Function**, **Modular Inverse**, **RSA Encryption**, and **Digital Signatures** ensure that customer information remains confidential, authentic, and tamper-proof.

This case study explains the mathematical role of Number Theory in an ATM transaction using RSA encryption.

2.2 REAL-LIFE SCENARIO.

Suppose a customer named **Spandana** visits an ATM to withdraw **₹10,000**.

The ATM must securely send the following information to the bank:

- Card Number
- ATM PIN
- Withdrawal Amount
- ATM ID
- Date and Time
- Session Number

If an attacker intercepts this communication, they should not be able to:

- Read the PIN
- Change the withdrawal amount
- Modify the account number
- Pretend to be the bank

To prevent these attacks, the ATM encrypts all sensitive data using Number Theory.

2.3 MATHEMATICAL FOUNDATION: THE RSA CRYPTOSYSTEM DEPENDS UPON FIVE IMPORTANT CONCEPTS OF NUMBER THEORY.

(1) Prime Numbers:

Select two prime numbers

$$p = 61$$

$$q = 53$$

Both numbers are prime because they have exactly two factors.

(2) Compute n:

Multiply the two primes.

$$\begin{aligned}n &= p \times q \\&= 3233\end{aligned}$$

The value **3233** becomes part of both the public and private keys.

(3) Euler's Totient Function:

$$\begin{aligned}\varphi(n) &= (p - 1)(q - 1) \\&= (61 - 1)(53 - 1) \\&= 60 \times 52 = 3120\end{aligned}$$

Euler's Totient Function counts the numbers less than 3233 that are relatively prime to it.

(4) Greatest Common Divisor (GCD):

Choose a number

$$e = 17$$

Now verify

$$GCD(17, 3120) = 1$$

Using Euclid's Algorithm

$$3120 = 17(183) + 9$$

$$17 = 9(1) + 8$$

$$9 = 8(1) + 1$$

$$8 = 1(8) + 0$$

Hence

$$GCD(17, 3120) = 1$$

Therefore,

17 can be selected as the public exponent.

(5) Modular Inverse:

Find d such that

$$ed \equiv 1 \pmod{3120}$$

$$\text{Substitute } e = 17$$

$$17d \equiv 1 \pmod{3120}$$

Using the Extended Euclidean Algorithm,

$$d = 2753$$

Check

$$17 \times 2753 = 46801$$

$$46801 \div 3120$$

$$= 15 \text{ remainder } 1$$

Therefore

$$17 \times 2753 \equiv 1 \pmod{3120}$$

Hence,

Private Key

$$(d, n) = (2753, 3233)$$

Public Key

$$(e, n) = (17, 3233)$$

These keys are now stored.

2.4 ATM TRANSACTION

The customer inserts the ATM card.

The ATM reads

Card Number

$$5123456789012345$$

The customer enters

PIN

$$5832$$

Withdrawal Amount of ₹10,000

Instead of sending this information directly, the ATM converts it into numerical form.

Suppose

Withdrawal Request

M=123

where M represents the encrypted transaction block.

2.5 RSA ENCRYPTION

Encryption Formula

$$C = M^e \pmod{n}$$

Substitute the values of

$$M = 123$$

$$e = 17$$

$$n = 3233 \text{ in above equation}$$

Therefore

$$C = 123^{17} \pmod{3233}$$

Using repeated modular squaring,

$$123^2 = 15129$$

$$15129 \pmod{3233} = 2197$$

Now,

$$123^4 = 2197^2 \pmod{3233} = 3173$$

Next,

$$123^8 \pmod{3233} = 367$$

Next,

$$123^{16} \pmod{3233} = 2136$$

Finally,

$$123^{17} = 123^{16} \times 123$$

Therefore

$$C = (2136 \times 123) \pmod{3233}$$

$$C = 855$$

Hence Ciphertext,

$$C = 855$$

The ATM sends **855** instead of the original message.

If a hacker intercepts the communication, only **855** is visible.

The original transaction cannot be recovered without the private key.

2.6 RSA DECRYPTION

The bank receives

Ciphertext

$$C = 855$$

Using the private key,

$$d = 2753$$

the bank calculates,

$$M = C^d \pmod{n}$$

Substitute

$$M = 855^{2753} \pmod{3233}$$

After modular exponentiation,

$$M = 123$$

Thus the original transaction is recovered.

2.7 VERIFICATION OF EULER'S THEOREM

Euler's theorem states that,

$$M^{\phi(n)} \equiv 1 \pmod{n}$$

when

$$\text{GCD}(M, n) = 1$$

Since

$$\phi(3233) = 3120$$

and

123 is relatively prime to 3233,

$$123^{3120} \equiv 1 \pmod{3233}$$

This theorem guarantees that

$$(M^e)^d \equiv M \pmod{n}$$

Therefore,

Encryption followed by decryption always returns the original message.

2.8 DIGITAL SIGNATURE

After verifying Rahul's PIN and account balance, the bank sends

"Withdrawal Approved"

The bank signs the message using its private key.

Signature

$$S = H(M)^d \pmod{n}$$

where, $H(M)$ represents the hash value of the message.

The ATM verifies

$$H(M) = S^e \pmod{n}$$

If both of the value matches, then ATM confirms

- The message came from the genuine bank.
- The message was not modified.
- The transaction is authentic.

2.9 WHY LARGE PRIME NUMBERS ARE NECESSARY

In our example,

We took 61 and 53 as primes which are chosen only for simplicity

Real ATM systems use prime numbers having approximately

1024 bits each.

Therefore,

N becomes a 2048-bit integer.

Factoring such a huge number into its original prime factors is computationally infeasible with current classical computers.

3. CASE STUDY: APPLICATION OF NUMBER THEORY IN WHATSAPP END-TO-END ENCRYPTION

3.1 INTRODUCTION

WhatsApp is one of the world's most popular messaging applications. Every day, billions of text messages, voice calls, videos, and documents are exchanged securely. The security of WhatsApp is based on **End-to-End Encryption (E2EE)**, which ensures that only the sender and the intended receiver can read the message. Even WhatsApp cannot read the contents of the messages.

The mathematical foundation of WhatsApp encryption includes:

- Prime Numbers
- Modular Arithmetic
- Modular Exponentiation
- Finite Fields
- Diffie-Hellman Key Exchange
- Elliptic Curve Cryptography (ECC)

For simplicity, this case study uses a **small numerical example** to demonstrate the mathematics. Real Whatsapp uses much larger numbers and elliptic curves.

3.2 STEP 1: ESTABLISHING THE SHARED SECRET KEY

Suppose two users want to communicate securely.

Choose a prime number

$$p = 23$$

Choose a primitive root

$$g = 5$$

These values are public.

User A chooses a private key as, $a = 6$

Compute the public value

$$A = g^a \pmod{p}$$

$$A = 5^6 \pmod{23}$$

Using modular arithmetic,

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

$$5^4 = 2^2 = 4 \pmod{23}$$

$$5^6 = 5^4 * 5^2$$

$$= 4 * 2 = 8 \pmod{23}$$

Hence, $A = 8$

User B chooses a private key as, $b = 15$

Compute,

$$B = 5^{15} \pmod{23}$$

Using modular exponentiation,

$$B = 19$$

Hence, $B = 19$

Computing the Shared Secret

User A computes

$$K = B^a \pmod{p}$$

$$K = 19^6 \pmod{23} = 2$$

User B computes

$$K = A^b \pmod{p}$$

$$K = 8^{15} \pmod{23} = 2$$

Thus, $K = 2$

Both users independently obtain the same secret key without transmitting it over the network.

3.3 STEP 2: CONVERTING THE MESSAGE INTO NUMBERS

Suppose the sender wants to send the message

HELLO

Using the ASCII code:

LETTER	ASCII
H	72
E	69
L	76
L	76
O	79

Thus,

$$\text{HELLO} \rightarrow (72, 69, 76, 76, 79)$$

3.4 STEP 3: ENCRYPTING THE MESSAGE

For this example, use the shared secret key

$$K = 2$$

Encrypt each character using

$$C = (M + K) \pmod{p}$$

where M = ASCII value of the plain text

$K = 2$ and C = ciphertext

Encryption Process

$$\text{For H : } 72 + 2 = 74$$

$$\text{For E : } 69 + 2 = 71$$

$$\text{For L : } 76 + 2 = 78$$

$$\text{For L : } 76 + 2 = 78$$

$$\text{For O : } 79 + 2 = 81$$

The encrypted numerical values are: (74,71,78,78,81)

Converting back to ASCII gives:

Plain Text	ASCII	Cipher Text
H	72	J (74)
E	69	G (71)
L	76	N (78)
L	76	N (78)
O	79	Q (81)

Therefore,

HELLO → JGNNQ

3.5 STEP 4: DECRYPTION THE MESSAGE

The receiver uses the same shared secret key

$$K = 2.$$

The decryption formula is

$$M = (C - K) \pmod{p}$$

Calculations:

$$74 - 2 = 72$$

$$71 - 2 = 69$$

$$78 - 2 = 76$$

$$78 - 2 = 76$$

$$81 - 2 = 79$$

The recovered ASCII values are: (72,69,76,76,79)

which correspond to HELLO

Hence, the receiver successfully recovers the original message.

3.6 MATHEMATICAL VERIFICATION

The Diffie–Hellman key exchange works because

$$K = (g^b)^a \pmod{p}$$

Applying the law of exponent,

$$K = (g^a)^b \pmod{p}$$

Similarly,

$$K = (g^b)^a \pmod{p}$$

Therefore,

$$K = (g^a)^b \pmod{p} = (g^b)^a \pmod{p} = g^{ab} \pmod{p}$$

which guarantees that both users compute the same shared secret key.

4. GENERATING RSA ON ATM TRANSACTION AND WHATSAPP

4.1 PROGRAM: RSA ENCRYPTION AND DECRYPTION FOR ATM TRANSACTION

```
p = 17
q = 11
n = p * q
phi = (p - 1) * (q - 1)
e = 7
d = pow(e, -1, phi)
print("RSA ATM TRANSCATION")
print("Public Key :", (e, n))
print("Private Key :", (d, n))
pin = input("\nEnter 4-digit ATM Pin:")
if len(pin) != 4 or not pin.isdigit():
    print("Please enter exactly 4 digits.")
else:
    ascii_values = [ord(x) for x in pin]
    encrypted = [pow(M, e, n) for M in ascii_values]
    decrypted_ascii = [pow(C, d, n) for C in encrypted]
    decrypted_pin = "".join(chr(x) for x in decrypted_ascii)
    print("\nOriginal PIN :", pin)
    print("ASCII Values :", ascii_values)
    print("Encrypted PIN :", encrypted)
```

```
print("Decrypted ASCII :", decrypted_ascii)
```

```
print("Decrypted PIN :", decrypted_pin)
```

```
if pin == decrypted_pin:
```

```
    print("Transaction Verified Successfully!")
```

```
else:
```

```
    print("Transaction Failed")
```

Output: RSA ATM TRANSCATION

Public Key : (7, 187)

Private Key : (23, 187)

Enter 4-digit ATM Pin:1234

Original PIN : 1234

ASCII Values : [49, 50, 51, 52]

Encrypted PIN : [25, 118, 17, 35]

Decrypted ASCII : [49, 50, 51, 52]

Decrypted PIN : 1234

Transaction Verified Successfully!

4.2 PROGRAM: WHATSAPP KEY GENERATION

```
p = 61
q = 53
n = p * q
phi = (p - 1) * (q - 1)
e = 17
d = pow(e, -1, phi)
print("RSA KEY GENERATION")
print("p =", p)
print("q =", q)
print("n =", n)
print("Euler's Totient phi(n) =", phi)
print("Public exponent e =", e)
```

```
print("Private exponent d =", d)

print("\nPublic Key =", (e, n))

print("Private Key =", (d, n))

message = "HELLO"

print("\nOriginal Message =", message)

numbers = [ord(character) for character in message]

print("ASCII Values =", numbers)

encrypted = []

for M in numbers:

    C= pow(M, e, n)

    encrypted.append(C)

print("\nEncrypted Message =")

print(encrypted)

decrypted_numbers = []

for C in encrypted:

    M= pow(C, d, n)

    decrypted_numbers.append(M)

decrypted_message = ".join(chr(M) for M in
decrypted_numbers)

print("\nDecrypted ASCII Values =")

print(decrypted_numbers)

print("\nDecrypted Message =",decrypted_message)
```

Output: RSA KEY GENERATION

P = 61
Q = 53
N = 3233
Euler's Totient $\phi(n)$ = 3120
Public exponent e = 17
Private exponent d = 2753

Public Key = (17, 3233)
Private Key = (2753, 3233)
Original Message = HELLO
ASCII Values = [72, 69, 76, 76, 79]
Encrypted Message =
[3000, 28, 2726, 2726, 1307]
Decrypted ASCII Values =
[72, 69, 76, 76, 79]
Decrypted Message = HELLO

5. CONCLUSION:

This paper demonstrates the important role of Number Theory in digital communication and cyber security. Concepts covered above provide the mathematical principles required to design secure cryptographic systems. The study of RSA encryption showed how these concepts are used for secure encryption, while ATM case study demonstrated the protection of PIN's and transaction data. Digital signatures provide authentication and data integrity. The Whatsapp case study explained Diffie-Hellman key exchange and demonstrated encryption and decryption using the message "HELLO". The Python programs provide a practical demonstration of RSA and connected mathematical theory with programming.

Overall, this paper shows that Number Theory plays a significant role in protecting digital information. Number Theory can be viewed as a powerful bridge between pure mathematics and modern technology, transforming mathematical properties of integers into practical tools for securing digital communication.

6. REFERENCES:

1. **A. J. Menezes, P. C. Van Oorschot and S. A. Vanstone**, *Handbook of Applied Cryptography*. CRC Press, 1996.
2. **C. Paar and J. Pelzl**, *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer, 2010.
3. **D. M. Burton**, *Elementary Number Theory*. McGraw-Hill Education, 2010.
4. **G. A. Jones and J. M. Jones**, *Elementary Number Theory*. Springer, 1998.
5. **J. Katz and Y. Lindell**, *Introduction to Modern Cryptography*. CRC Press, 2020.
6. **K. H. Rosen**, *Elementary Number Theory and Its Applications*. Pearson, 2011.
7. **N. Koblitz**, *A Course in Number Theory and Cryptography*. Springer, 1994.
8. **Python Software Foundation**, *Python Documentation*. Python Software Foundation.
9. **R. A. Mollin**, *An Introduction to Cryptography*. Chapman & Hall/CRC, 2007.

10. **R. L. Rivest, A. Shamir and L. Adleman**, "*A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.*" *Communications of the ACM*, Vol. 21, No. 2, 1978.
11. **W. Diffie and M. E. Hellman**, "*New Directions in Cryptography.*" *IEEE Transactions on Information Theory*, Vol. 22, No. 6, 1976.
12. **W. Stallings**, *Cryptography and Network Security: Principles and Practice*. Pearson, 2017.