УДК 004.942

А.А. Гайдучок
І.А. Клєопа

Оптимізація обчислювальних процесів у криптосистемі RSA на основі Китайської теореми про залишки

Вінницький національний технічний університет

Анотація

Метою даної статті є вивчення принципів асиметричного шифрування та методів підвищення обчислювальної ефективності, а також практична реалізація алгоритму RSA з використанням китайської теореми про залишки. У ході виконання роботи необхідно було реалізувати програмну демонстрацію роботи алгоритму, порівняти швидкість обчислень класичного методу RSA та оптимізованого варіанту, а також дослідити вплив застосування китайської теореми про залишки на операції дешифрування.

Ключові слова: мова програмування C++, алгоритм RSA, китайська теорема про залишки, асиметричне шифрування.

Abstract

The purpose of this article is to study the principles of asymmetric encryption and methods for increasing computational efficiency, as well as the practical implementation of the RSA algorithm using the chinese remainder theorem. In the course of the work, it was necessary to implement a software demonstration of the algorithm, compare the calculation speed of the classical RSA method and the optimized variant, and investigate the impact of applying the chinese remainder theorem on decryption.

Keywords: C++ programming language, RSA algorithm, chinese remainder theorem, asymmetric encryption.

Вступ

Алгоритм RSA є одним із найпоширеніших стандартів асиметричного шифрування, який широко використовується для захисту передачі даних та створення цифрових підписів. Його надійність ґрунтується на математичній складності факторизації великих цілих чисел. Проте, суттєвим недоліком класичної реалізації RSA є висока обчислювальна складність операцій піднесення до степеня за модулем, що стає критичним при використанні ключів великої довжини.
У даній роботі реалізовано алгоритм RSA мовою C++ та досліджено приріст продуктивності, який забезпечує застосування китайської теореми про залишки порівняно з класичним методом.

Результати дослідження

Програма реалізована мовою C++ і складається з набору функцій, що забезпечують генерацію ключів, шифрування та два методи дешифрування: класичний та оптимізований з використанням китайської теореми про залишки.

Під час запуску програми користувач вводить два великих простих числа p та q. Для перевірки їх простоти використовується функція isPrime. На основі введених даних обчислюється модуль системи n = p * q та значення функції Ейлера. Відкрита експонента зафіксована у коді значенням u=17, а секретна експонента v обчислюється за допомогою розширеного алгоритму Евкліда (modInverse) так, щоб виконувалась умова:

$$u * v = 1 \bmod phi(n)$$

Після генерації ключів відбувається шифрування введеного користувачем повідомлення m за формулою:

$$C = m^u \bmod n$$

Для реалізації процесу дешифрування програма пропонує два шляхи, що дозволяє наочно порівняти їхню обчислювальну складність.

1. Стандартний метод. Дешифрування виконується шляхом прямого піднесення шифротексту до степеня секретної експоненти за модулем n:

$$m = C^u \bmod n$$

Оскільки числа v та n є великими, ця операція вимагає значних ресурсів. У програмній реалізації функції power для цього методу встановлено ваговий коефіцієнт складності операції 4 (умовних одиниць), що відображає роботу з числами подвійної довжини (у бітах).

2. Метод на основі китайської теореми про залишки. Цей метод оптимізує процес, розбиваючи одне складне обчислення на два простіших за модулями p та q. Алгоритм діє наступним чином:

1. Попередньо обчислюються компоненти повідомлення: $C_p = C * \bmod p$ та $C_q = C * \bmod q$.
2. Обчислюються редуковані експоненти: $vp = v * \bmod(p-1)$ та $vq = v * \bmod(q-1)$.
3. Знаходяться частини вихідного повідомлення: $mp = Cp^{vp} * \bmod p$ та $mq = Cq^{vq} * \bmod q$.
4. Кінцевий результат збирається за формулою китайської теореми про залишки, реалізованою в коді: $m = (p*(p^{-1} \bmod q)*mq + q*(q^{-1} \bmod p)*mp) \bmod n$.
5. У цьому випадку операції виконуються з меншими числами, тому ваговий коефіцієнт складності у функції power дорівнює 1.

Повний код програми:

```
#include <iostream>
using namespace std;
long long operations_count = 0;
//Підрахунок біт
int countBits(long long n) {
    int bits = 0;
    while (n > 0) {
        n /= 2;
        bits++;
    }
    return bits;
}
bool isPrime(long long n) {
    if (n <= 1) return false;
    for (long long i = 2; i * i <= n; i++) {
        if (n % i == 0) return false;
    }
    return true;
}
// Функція піднесення до степеня з "вагою" (операції у стандартному способі
приблизно у 4 рази складніші для обрахунків
// при використані для обрахунків китайської теореми про залишки)
long long power(long long base, long long exp, long long mod, int weight) {
    long long res = 1;
    base = base % mod;
    while (exp > 0) {
        if (exp % 2 == 1) {
            res = (res * base) % mod;
            operations_count += weight;
        }
        exp = exp / 2;
        base = (base * base) % mod;
        operations_count += weight;
    }
    return res;
```

```
}
// Розширений алгоритм Евкліда
long long modInverse(long long a, long long m) {
    long long m0 = m;
    long long y = 0, x = 1;
    if (m == 1) return 0;
    while (a > 1) {
        long long q = a / m;
        long long t = m;
        m = a % m, a = t;
        t = y;
        y = x - q * y;
        x = t;
    }
    if (x < 0) x += m0;
    return x;
}
int main() {
    long long p, q, message;
    long long u = 17;
    cout << "========================================" << endl;
    cout << "      RSA DECIPHERING SIMULATION        " << endl;
    cout << "========================================" << endl;
    do{
        cout << "Enter prime P (rec: 40009): ";
        cin >> p;
        if (!isPrime(p)) cout << "Error: Not prime.\n";
    } while (!isPrime(p));
    do {
        cout << "Enter prime Q (rec: 40037): ";
        cin >> q;
        if (!isPrime(q)) cout << "Error: Not prime.\n";
    } while (!isPrime(q));
    long long n = p * q; // Обчислення модуля n
    long long phi = (p - 1) * (q - 1); // Функція Ейлера
    long long v = modInverse(u, phi); // Секретна експонента
    cout << "\n[System] Keys Generated:" << endl;
    cout << "   Public Key (u, n): (" << u << ", " << n << ")" << endl;
    cout << "   Private Key (v, n): (" << v << ", " << n << ")" << endl;
    if (v == 0) {
        cout << "Error: Invalid keys. Restart." << endl;
        return 1;
    }
    bool validMsg = false;
    do {
        cout << "\nEnter message (m < " << n << "): ";
        cin >> message;
        if (message >= n) cout << "Error: Too big!\n";
        else validMsg = true;
    } while (!validMsg);
    // Шифрування: C = m^u mod n
    operations_count = 0;
    long long c = power(message, u, n, 1);
    cout << "\n[Step 2] Encrypted C: " << c << endl;
    while (true) {
        cout << "\n----------------------------------------" << endl;
        cout << "Select Decryption Method:" << endl;
        cout << "1. Standard Method (Direct Exponentiation)" << endl;
        cout << "2. CRT Method (Chinese Remainder Theorem)" << endl;
```

```
            cout << "3. Exit" << endl;
            cout << "Choice: ";
            int choice;
            cin >> choice;
            if (choice == 3) break;
            long long result = 0;
            long long cost = 0;
            if (choice == 1) {
                // Стандартний спосіб: m = C^v mod n
                cout << "\n=== STANDARD RSA DECRYPTION ===" << endl;
                cout << "1. Formula application:" << endl;
                cout << "   m = C^v mod n" << endl;
                cout << "   m = " << c << "^" << v << " mod " << n << endl;
                cout << "\n2. Complexity Analysis:" << endl;
                cout << "   Exponent v: " << v << " (" << countBits(v) << " bits)"
<< endl;
                cout << "   Modulus n:  " << n << " (" << countBits(n) << " bits)"
<< endl;
                operations_count = 0;
                result = power(c, v, n, 4);
                cost = operations_count;
                cout << "   -> Decrypted Message m: " << result << endl;
                cout << "   -> Operations Cost: " << cost << " units" << endl;
            }
            else if (choice == 2) {
                // Китайська теорема про лишки
                cout << "\n=== CRT DECRYPTION (METHODOLOGY) ===" << endl;
                operations_count = 0;
                // Обчислення основ (c_p, c_q)
                long long cp = c % p;
                long long cq = c % q;
                // Обчислення показників за теоремою Ферма (v_gp, v_gq)
                long long v_gp = v % (p - 1);
                long long v_gq = v % (q - 1);
                cout << "1. Foundations and Exponents:" << endl;
                cout << "   cp = C mod p = " << cp << ", cq = C mod q = " << cq <<
endl;
                cout << "   v_gp = v mod (p-1) = " << v_gp << endl;
                cout << "   v_gq = v mod (q-1) = " << v_gq << endl;
                // Обчислення m_p, m_q
                long long m_p = power(cp, v_gp, p, 1);
                long long m_q = power(cq, v_gq, q, 1);
                cout << "\n2. Calculating Parts (Parallelizable):" << endl;
                cout << "   m_p = cp^v_gp mod p = " << m_p << endl;
                cout << "   m_q = cq^v_gq mod q = " << m_q << endl;
                // Коефіцієнти p_q^-1 та q_p^-1 за розширеним алгоритмом Евкліда
                long long p_inv_q = modInverse(p, q);
                long long q_inv_p = modInverse(q, p);
                // m = (p * p_inv_q * m_q + q * q_inv_p * m_p) mod n
                long long term1 = (p % n * p_inv_q % n) % n;
                term1 = (term1 * (m_q % n)) % n;
                long long term2 = (q % n * q_inv_p % n) % n;
                term2 = (term2 * (m_p % n)) % n;
                result = (term1 + term2) % n;
                cost = operations_count;
                cout << "\n3. Recombining Result (CRT Formula 3):" << endl;
                cout << "   m = " << result << endl;
                cout << "   Operations Cost: " << cost << " units" << endl;
            }
```

```
    }
    return 0;
}
```

Приклади роботи програми:

Генерація ключів та шифрування

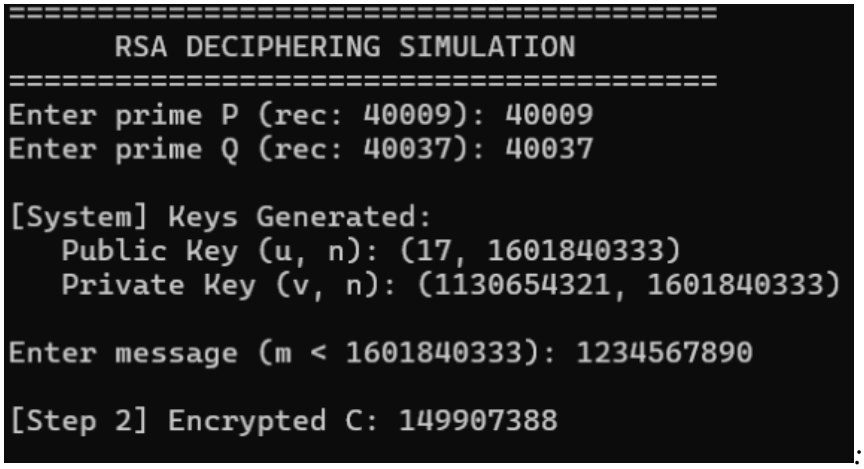

Рис. 1. Етап ініціалізації системи

Метод CRT та порівняння

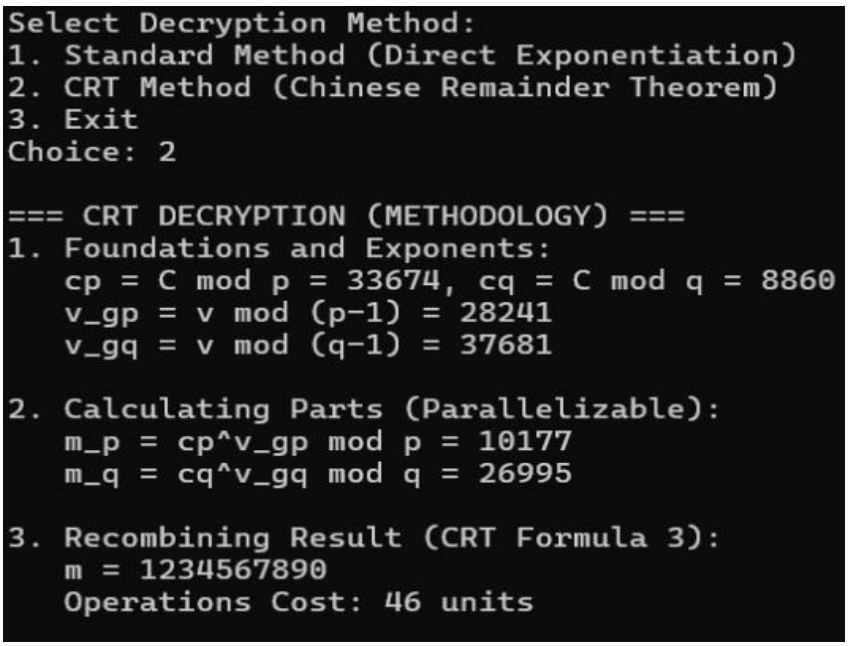

Рис. 2. Результат дешифрування за допомогою CRT.

Стандартний метод дешифрування

Рис. 3. Результат дешифрування стандартним методом.

Висновок

В ході виконання роботи було реалізовано та досліджено алгоритм оптимізації дешифрування RSA. Проведені тести показали, що використання Китайської теореми про залишки дозволяє:

1. Зменшити обчислювальну складність операції дешифрування (згідно з лічильником операцій — у кілька разів).
2. Забезпечити математичну точність відновлення даних.
3. Створити підґрунтя для реалізації паралельних обчислень, оскільки значення **m_p** та **m_q** можуть обчислюватися одночасно на різних ядрах процесора.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Маліновська, О. О. Вимоги до криптографічної системи захисту інформації / О. О. Маліновська, О. І. Зінченко ; наук. кер. Я. Ю. Усов // Новітні технології у науковій діяльності і навчальному процесі : матеріали тез доп. Всеукр. наук.- практ. конф. студентів, аспірантів та молодих учених (м. Чернігів, 10 -11 квітня 2019 р.). - Чернігів : ЧНТУ, 2019. – С. 113-116.
2. Jorgen Veisdal, Shannon Ciphers and Perfect Security/Cantor’s Paradise/2020.
3. Математичні методи криптології: Навчальний посібник [Електронний ресурс] (Для студентів техн. спец. вищ. навч. закл.) / [А.Д. Кожухівський, І.Д. Горбенко, Г.І. Гайдур, О.А. Кожухівська, В.В. Марченко]; М-во освіти і науки України, Державний університет телекомунікацій. Київ: ДУТ, 2021 – 244 с.

Гайдучок Анатолій Анатолійович – студент групи 2БС – 25б, факультет інформаційних технологій та комп’ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: tbadlyb0y@gmail.com

Науковий керівник: ***Клєопа Ірина Анатоліївна*** – pHd, доцент, кафедра вищої математики, Вінницький національний технічний університет, м. Вінниця, Хмельницьке шосе, 95, e-mail: paceka08@vntu.edu.ua

Haiduchok Anatolii Anatoliiovych – student of group 2SS – 25b, Faculty of Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: tbadlyb0y@gmail.com

Scientific supervisor: ***Klieopa Iryna Anatoliivna*** – PhD, Associate Professor, Department of Higher Mathematics, Vinnytsia National Technical University, Vinnytsia, Khmelnytske Shosse, 95, e-mail: paceka08@vntu.edu.ua