

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы защиты информации

ОТЧЁТ
к лабораторной работе №3
на тему

АСИММЕТРИЧНАЯ КРИПТОГРАФИЯ. КРИПТОСИСТЕМА РАБИНА

Выполнил: студент гр.253501
Станишевский А.Д.

Проверил: ассистент кафедры информатики
Герчик А.В.

Минск 2025

СОДЕРЖАНИЕ

1 Цель работы	3
2 Теоретические сведения	4
3 Ход работы.....	5
Заключение	6
Приложение А (обязательное) Листинг программного кода	7

1 ЦЕЛЬ РАБОТЫ

В современном цифровом мире защита конфиденциальной информации становится все более актуальной задачей, что обуславливает необходимость изучения и практического применения криптографических методов. Одним из перспективных направлений в области защиты данных является асимметричная криптография, основанная на использовании парных ключей – открытого для шифрования и закрытого для расшифровки.

В рамках данной лабораторной работы необходимо изучить криптосистему Рабина, которая обладает доказанной криптостойкостью и основывается на вычислительной сложности задачи извлечения квадратных корней по модулю большого составного числа. Особенностью данной системы является ее высокая производительность при шифровании благодаря использованию операции возведения в квадрат, что делает ее перспективной для применения в системах, требующих быстрой обработки данных.

Основная цель работы заключается в разработке программного средства, реализующего полный цикл криптографических преобразований по алгоритму Рабина – от генерации ключевой пары до шифрования и корректного восстановления исходных текстовых данных.

В процессе работы предстоит решить ряд важных задач: изучить математические основы криптосистемы, включая условия генерации простых чисел; реализовать алгоритмы шифрования через возведение в квадрат по модулю n ; разработать процедуру расшифровки с применением китайской теоремы об остатках; создать механизм распознавания исходного сообщения; обеспечить удобный интерфейс для работы с текстовыми данными.

2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Криптосистема Рабина – криптографическая система с открытым ключом, безопасность которой обеспечивается сложностью поиска квадратных корней в кольце остатков по модулю составного числа. Безопасность системы, как и безопасность метода RSA, обусловлена сложностью разложения на множители больших чисел. Зашифрованное сообщение можно расшифровать 4 способами. Недостатком системы является необходимость выбора истинного сообщения из 4-х возможных.

Система Рабина, как и любая асимметричная криптосистема, использует открытый и закрытый ключи. Открытый ключ используется для шифрования сообщений и может быть опубликован для всеобщего обозрения. Закрытый ключ необходим для расшифровки и должен быть известен только получателям зашифрованных сообщений.

Процесс генерации ключей следующий:

1) выбираются два случайных числа p и q с учётом требований: 1) числа должны быть большими; 2) числа должны быть простыми; 3) должны выполняться условия: $p \equiv 3 \pmod{4}$; $q \equiv 3 \pmod{4}$ и $p \neq q$, где $k \in \mathbb{N}$.

2) вычисляется число $n = p \cdot q$;

3) число n – открытый ключ; числа p и q – закрытый. Базовые циклы заключаются в многократном выполнении для блока данных основного раунда, рассмотренного нами ранее, с использованием разных элементов ключа и отличаются друг от друга порядком использования ключевых элементов.

Исходное сообщение m (текст) шифруется с помощью открытого ключа – числа n по следующей формуле $c = m^2 \pmod{n}$. Благодаря использованию умножения по модулю скорость шифрования системы Рабина больше, чем скорость шифрования по методу RSA, даже если в последнем случае выбрать небольшое значение экспоненты. Пример (продолжение). Пусть исходным текстом является $m = 20$. Тогда зашифрованным текстом будет: $c = m^2 \pmod{n} = 20^2 \pmod{77} = 400 \pmod{77} = 15$. Для расшифровки сообщения необходим закрытый ключ — числа p и q . На рисунке 2.1 представлена формула китайской теоремы об остатках для нахождения четырех чисел.

$$\begin{aligned} 1) & r = (y_p * p * m_q + y_q * q * m_p) \pmod{n} \\ 2) & -r = n - r \\ 3) & s = (y_p * p * m_q - y_q * q * m_p) \pmod{n} \\ 4) & -s = n - s \end{aligned}$$

Рисунок 2.1 – Вычисление четырех чисел

Одно из этих чисел является истинным открытым текстом m .

3 ХОД РАБОТЫ

Программное средство реализовано при помощи языка программирования JavaScript. На рисунке 3.1 изображен процесс шифрования исходного сообщения.

Шифрование

Исходный текст

hello man 228 привет

ЗАШИФРОВАТЬ

Зашифрованный текст

28142814477990552704846374935960002804639238305140270374028098585764025817449945786396378587302049408432010470706607766996484

Число до шифрования (m)

167758202416425983545757646961155390410227562387404387498971522

Рисунок 3.1 – Процесс шифрования исходного сообщения

На рисунке 3.2 изображен процесс дешифрования зашифрованного исходного сообщения.

✔ **Вариант 1:** "hello man 228 привет"

Число: 167758202416425983545757646961155390410227562387404387498971522

⚠ **Вариант 2:** "Zn65-XPk2^QkjsNS"0t,ee\hnd"!v>tB^iWMM%Uf\q\lf"7 o"

Число: 129457250596218830000704943689605755562950901386552414309756361309119539113991282101675778274871520788695

⚠ **Вариант 3:** "fZ賀W 6p&huO)E8q-5(5RhBG%a~slzei掙P#lb|"Q9)dpC I T k ; ā"

Число: 230757408829765329724329343875960149143060363946796532247309825425623276850470902282368499784240721811215

⚠ **Вариант 4:** "~E ,z;>GQv4t* P ; / 'YO PaA<sO k J G ^ j m " d y f v ` m + \$ q J k"

Число: 106381509713242297028272009302009740648644864991872761085025378766557211428944191873438928296447448607574

Рисунок 3.2 – Процесс дешифрования исходного сообщения

Таким образом, программа успешно справляется с поставленными задачами.

ЗАКЛЮЧЕНИЕ

В результате выполнения лабораторной работы была успешно реализована криптосистема Рабина, позволяющая осуществлять шифрование и дешифрование текстовых данных.

В процессе работы были изучены теоретические основы алгоритма, включая принципы генерации ключей, математические операции возведения в квадрат по модулю и извлечения квадратных корней с использованием китайской теоремы об остатках.

Практическая реализация системы включала разработку основных функций для работы с простыми числами специального вида, создания ключевой пары и выполнения криптографических преобразований. Особое внимание было уделено решению характерной для системы Рабина проблемы множественности расшифрования через реализацию механизма выбора минимального корня.

Разработанное программное обеспечение с графическим интерфейсом пользователя продемонстрировало работоспособность алгоритма на практике. Тестирование системы подтвердило корректность выполнения операций шифрования и дешифрования, а также возможность обработки текстовых данных на русском и английском языках.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг программного кода

```
export async function isPrimeMillerRabin(n, k = 10) {
  if (n < 2n) return false;
  if (n === 2n) return true;
  if (n % 2n === 0n) return false;

  let d = n - 1n;
  let s = 0n;
  while (d % 2n === 0n) {
    d /= 2n;
    s++;
  }

  for (let i = 0; i < k; i++) {
    let a;
    do {
      a = await randomBigIntInRange(2n, n - 2n);
    } while (a <= 1n || a >= n - 1n);

    let x = modPow(a, d, n);
    if (x === 1n || x === n - 1n) continue;

    let composite = true;
    for (let r = 1n; r < s; r++) {
      x = mod(x * x, n);
      if (x === n - 1n) {
        composite = false;
        break;
      }
    }

    if (composite) return false;
  }

  return true;
}

export async function randomBigIntInRange(min, max) {
  const range = max - min + 1n;
  if (range < 1n) throw new Error('Некорректный диапазон');

  const bitLength = range.toString(2).length;
  const bytes = Math.ceil(bitLength / 8);
  const array = new Uint8Array(bytes);

  if (typeof crypto !== 'undefined' && crypto.getRandomValues) {
    crypto.getRandomValues(array);
  } else {
    throw new Error('crypto.getRandomValues не поддерживается');
  }

  let rand = 0n;
  for (let i = 0; i < bytes; i++) {
    rand = (rand << 8n) + BigInt(array[i]);
  }

  rand = min + (rand % range);
}
```

```

    return rand;
}

(mod 4)
export async function generateLargePrime(bitLength = 512) {
    let candidate;

    do {
        const min = 1n << (BigInt(bitLength) - 1n);
        const max = (1n << BigInt(bitLength)) - 1n;

        candidate = await randomBigIntInRange(min, max);

        if (candidate % 2n === 0n) candidate++;

        if (candidate % 4n !== 3n) {
            candidate = candidate + (4n - (candidate % 4n)) - 1n;
            if (candidate % 2n === 0n) candidate++;
        }

        if (await isPrimeMillerRabin(candidate, 10)) {
            return candidate;
        }

        candidate += 4n;
        if (candidate > max) candidate = min + 3n;
    } while (true);
}

export function extendedGCD(a, b) {
    if (a === 0n) return [b, 0n, 1n];
    const [g, x, y] = extendedGCD(b % a, a);
    return [g, y - (b / a) * x, x];
}

export function mod(n, m) {
    const result = n % m;
    return result >= 0n ? result : result + m;
}

export function modPow(base, exponent, modulus) {
    if (modulus === 1n) return 0n;
    let result = 1n;
    base = mod(base, modulus);
    while (exponent > 0n) {
        if (exponent % 2n === 1n) result = mod(result * base, modulus);
        exponent /= 2n;
        base = mod(base * base, modulus);
    }
    return result;
}

export function modSqrt(c, p) {
    if (p % 4n === 3n) return modPow(c, (p + 1n) / 4n, p);
    throw new Error("Модуль не в форме 4k+3");
}

export function stringToBigInt(str) {
    const encoder = new TextEncoder();
    const bytes = encoder.encode(str);
    let result = 0n;
    for (let i = 0; i < bytes.length; i++) {
        result = result * 256n + BigInt(bytes[i]);
    }
}

```



```

    }
    return result;
}

export function bigIntToString(num) {
    if (num === 0n) return "";

    const bytes = [];
    let temp = num;
    while (temp > 0n) {
        bytes.push(Number(temp % 256n));
        temp = temp / 256n;
    }

    bytes.reverse();

    try {
        const decoder = new TextDecoder('utf-8');
        return decoder.decode(new Uint8Array(bytes));
    } catch (e) {
        console.error('Ошибка декодирования UTF-8:', e);
        return '[Ошибка UTF-8 декодирования]';
    }
}

export async function generateKeys(bitLength = 512) {
    const p = await generateLargePrime(bitLength);
    let q = await generateLargePrime(bitLength);
    while (q === p) {
        q = await generateLargePrime(bitLength);
    }
    const n = p * q;
    return { p: p.toString(), q: q.toString(), n: n.toString() };
}

export function encrypt(plaintext, n) {
    const numN = BigInt(n);
    const m = stringToBigInt(plaintext);

    if (m >= numN) {
        throw new Error(`Текст слишком длинный! m=${m} >= n=${numN}. Сгенерируйте большие ключи.`);
    }

    const c = mod(m * m, numN);
    return c.toString();
}

export function decrypt(ciphertext, p, q) {
    const numP = BigInt(p);
    const numQ = BigInt(q);
    const numN = numP * numQ;
    const c = BigInt(ciphertext);

    const mp = modSqrt(mod(c, numP), numP);
    const mq = modSqrt(mod(c, numQ), numQ);

    const [_gcd, yp, yq] = extendedGCD(numP, numQ);

    const r = mod(yp * numP * mq + yq * numQ * mp, numN);
    const minusR = mod(numN - r, numN);
    const s = mod(yp * numP * mq - yq * numQ * mp, numN);
    const minusS = mod(numN - s, numN);

    const roots = [r, minusR, s, minusS];
}

```

```

return roots.map((root, idx) => {
  try {
    const text = bigIntToString(root);
    const isValid = [...text].every(char => {
      const code = char.charCodeAt(0);
      return (
        (code >= 32 && code <= 126) ||
        (code >= 1040 && code <= 1103) ||
        code === 1025 || code === 1105 ||
        code === 32 || code === 10 || code === 13
      );
    });
    return {
      value: root.toString(),
      text: text || '[пусто]',
      isValid: isValid,
      label: `Вариант ${idx + 1}`
    };
  } catch (e) {
    console.error(e);
    return {
      value: root.toString(),
      text: '[ошибка преобразования]',
      isValid: false,
      label: `Вариант ${idx + 1}`
    };
  }
});
}

```