МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ:

по дисциплине «Информационная безопасность»

Исполнитель

студент (ка) 3 курса группы 6 Розель Станислав Александрович

(Ф.И.О.)

Минск 2025

### Алгоритм шифрования RSA.

Данная лабораторная работа была совмещена с лабораторной работой по цифровой подписи. В этом разделе мы отдельно расмотрим RSA.

Алгоритм RSA был реализован в оконом приложении. На выбор есть разные длины ключей(от 4 и 8 бит, которые нужны были для тестов, так и до 1024 и 2048 бит). На рисунке 1.1 представлен результат генерации публичного и приватных ключей. Так же в поле получателя вставлен такой же публичный ключ, так как я явлюсь отправителем и получателем одновременно.

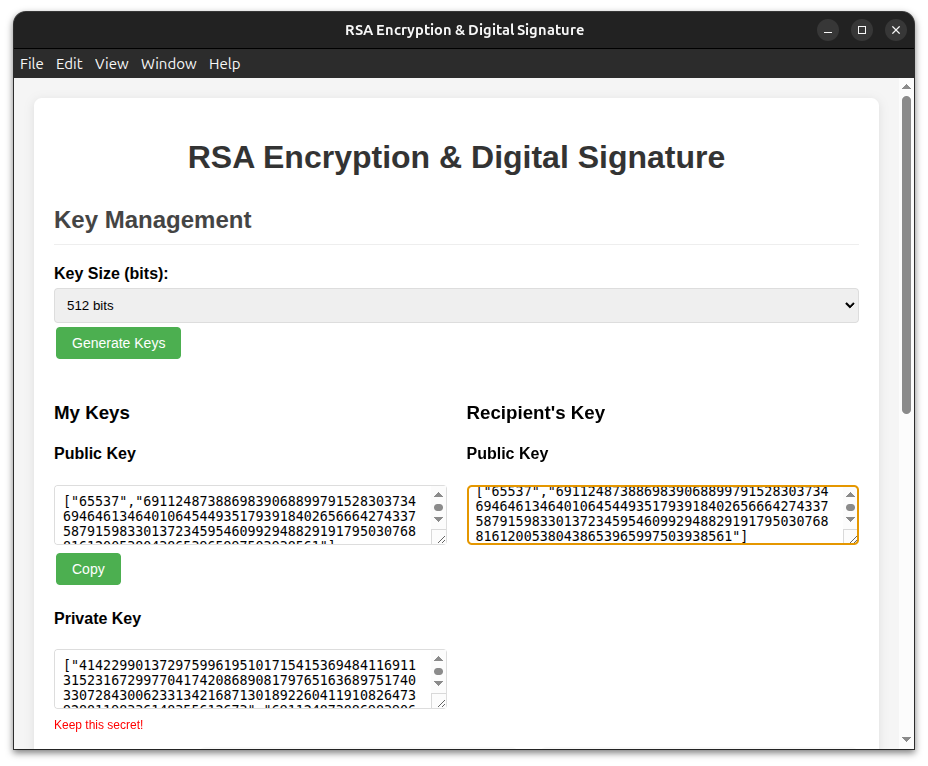
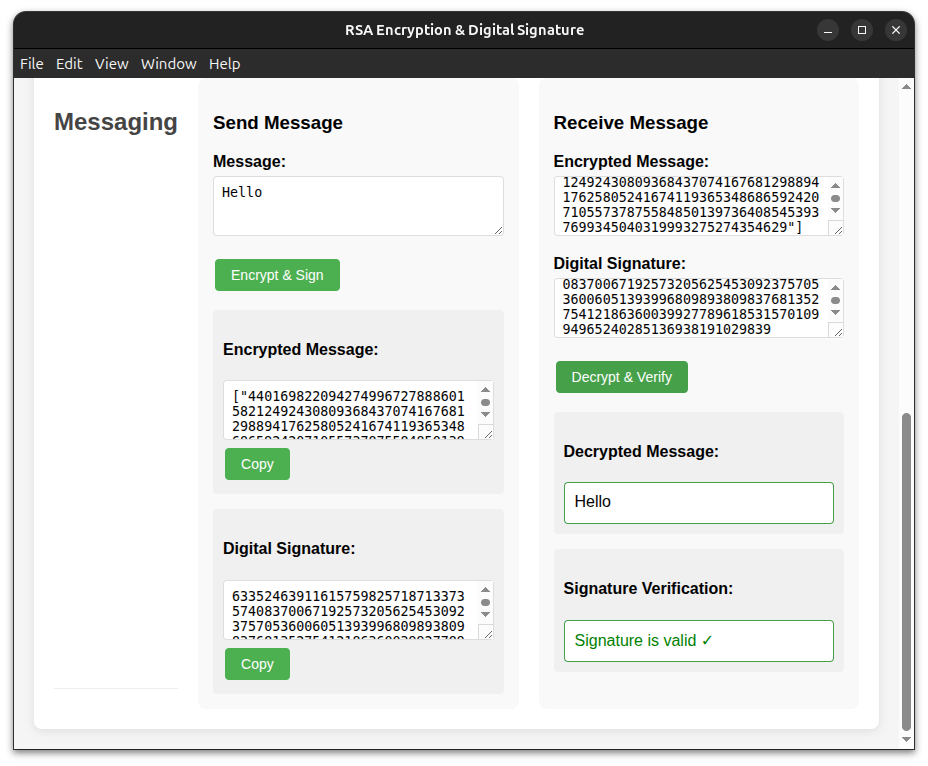


Рисунок 1.1 - Генерация ключей.

После этого можно ввести сообщение и зашифровать его. В результате мы получим зашифрованое сообщение и подпись. После этого так как я получатель и отправитель, я вставляю зашифрованное сообщение в поле для расшифровки и в соседнее поле подпись. Результат зашифрования и расшифрования представлен на рисунке 1.2.

Рисунок 1.2 - Зашифрование и расшифрование сообщения

Ниже представлен в листинге 1.1 реализация алгоритма RSA.

|  |
| --- |
| export function generateRSAKeyPair(bits: number = 2048): {  publicKey: [string, string];  privateKey: [string, string]  } {  const p = generateLargePrime(bits / 2);  const q = generateLargePrime(bits / 2);  const n = p \* q;  const phi = (p - 1n) \* (q - 1n);  const e = 65537n;  const d = modInverse(e, phi);  return {  publicKey: [e.toString(), n.toString()],  privateKey: [d.toString(), n.toString()],  };  } |

1. Цифровая подпись  
    Во второй части этой лабораторной работы была выполнена реализация с помощью кода электронно цифровой подписи. При шифровании мы хэшируем сообщение и шифруем с помощью публичного ключа получателя. При расшифровании мы расшифровываем не только сообщение, но и хэш, дальше мы хэшируем сообщение нам пришло и сверяем их, чтобы они сходились. На листинге 2.1 представлены функции для генерации подписи и их проверка.

|  |
| --- |
| export function signMessage(message: string, privateKey: [bigint, bigint]): bigint {  const [d, n] = privateKey;  const messageHash = createMessageHash(message, n);  return modExp(messageHash, d, n);  }  export function verifySignature(message: string, signature: bigint, publicKey: [bigint, bigint]): boolean {  const [e, n] = publicKey;  const messageHash = createMessageHash(message, n);  const decryptedSignature = modExp(signature, e, n);  return decryptedSignature === messageHash;  } |

На рисунке 2.1 представлена работа проверки подписи. Как можно увидеть, в колонке получателя, горит зеленый флаг, что означает, что подпись верна.

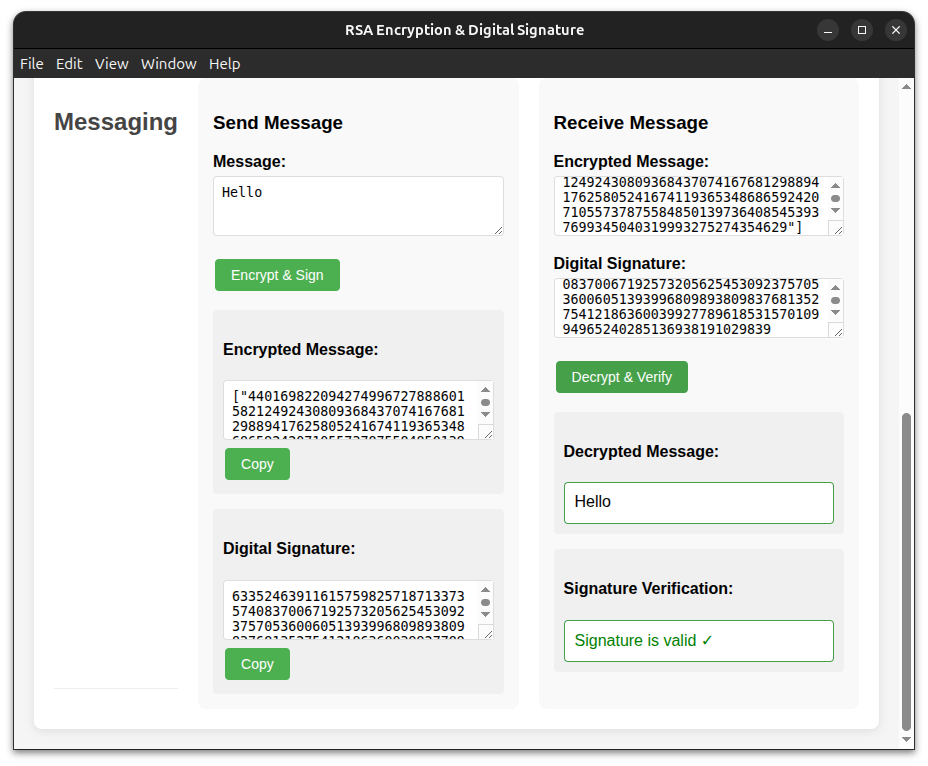


Рисунок 2.1 - Проверка подписи

### Индивидуальное задание.

Так же в виде индивидуального задания было нужно выполнить взлом RSA. Для этого был разработан скрипт, который методом подбора, проверял пары d и n, на основе известных e и n. Это выполняла функция факторизации, код которой представлен на листинге 3.1.

|  |
| --- |
| function factorize(n: bigint): [bigint | null, bigint | null] {  const startTime = Date.now();  const notificationInterval = 5 \* 60 \* 1000;  let lastNotification = startTime;  const sqrtN = BigInt(Math.floor(Math.sqrt(Number(n))));  for (let i = 2n; i <= sqrtN; i++) {  const currentTime = Date.now();  if (currentTime - lastNotification >= notificationInterval) {  const elapsedSeconds = Math.floor((currentTime - startTime) / 1000);  console.log(`Факторизация продолжается, прошло: ${formatTime(elapsedSeconds)}`);  lastNotification = currentTime;  }  if (n % i === 0n && isPrime(i) && isPrime(n / i)) {  const elapsedSeconds = Math.floor((Date.now() - startTime) / 1000);  console.log(`Факторизация завершена за: ${formatTime(elapsedSeconds)}`);  return [i, n / i];  }  }  return [null, null];  } |

Листинг 3.1 - Код функции factorize

Так же было проведено замер времени для разного количества бит. На рисунке 3.1 представлено время подбора приватного ключа длиной 64 бит.

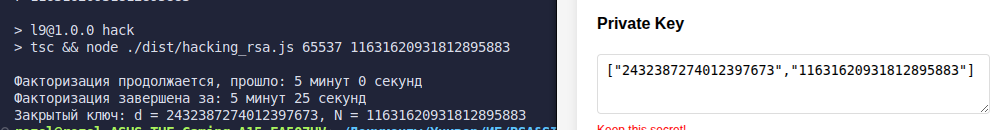


Рисунок 3.1 - Время для длины 64 бит

На рисунке 3.2 представлено время подбора приватного ключа длиной 128 бит.

На рисунке 3.3 представлено время подбора приватного ключа длиной 256 бит.