

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп'ютерних
систем**

«На правах рукопису»

УДК 519.688

«До захисту допущено»

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«__» _____ 2023 р.

**Магістерська дисертація
на здобуття ступеня магістра
за освітньо-професійною програмою
зі спеціальності 123 «Комп'ютерна інженерія»
на тему:
«Засіб передачі інформації, що зашифрована асиметричними
алгоритмами, в комп'ютерних мережах»**

Виконав:

студент II курсу, групи КВ-22мп

Глеб Владислав Юрійович _____

Керівник:

Доцент кафедри СПСКС, к.т.н., доцент

Тарасенко-Клятченко Оксана Володимирівна _____

Рецензент: _____

Засвідчую, що у цій магістерській
дисертації немає запозичень з праць
інших авторів без відповідних посилань.

Студент _____

Київ 2023

**Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»**

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність 123 комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

«__» _____ 2023 р.

**З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ**

Глебу Владиславу Юрійовичу

Тема дисертації: «Засіб передачі інформації, що зашифрована асиметричними алгоритмами, в комп'ютерних мережах», науковий керівник дисертації: доцент кафедри СПСКС, к.т.н., доцент Тарасенко-Клятченко О. В. затверджений наказом по університету від «9» листопада 2023 р. №5217-с

2. Термін подання студентом дисертації: «14» грудня 2023 р.

3. Об'єкт дослідження: застосунок для шифрування даних асиметричними алгоритмами.

4. Предмет дослідження: принцип роботи різних асиметричних алгоритмів. Застосунок для передачі інформації, зашифрованої асиметричними алгоритмами, в комп'ютерних мережах

5. Перелік завдань, які потрібно розробити:

- Провести аналіз і класифікацію існуючих асиметричних алгоритмів шифрування.
- Дослідити принципи та архітектури застосунку для передачі даних в комп'ютерних мережах на основі асиметричного шифрування.

6. Орієнтовний перелік ілюстративного матеріалу:

- Схеми збереження інформації про користувачів в базі даних.
- Користувацький інтерфейс застосунку і його використання.

7. Перелік публікацій:

- «Передача інформації, захищеної за допомогою асиметричних алгоритмів шифрування(RSA та ін.)». V Міжнародна науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2023),

8. Дата видачі завдання: “10” вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Літературний огляд тематики проекту	15.09.2023	
2.	Дослідження існуючих рішень та їх аналіз	23.09.2023	
3.	Підготовка матеріалів першого розділу магістерської дисертації	01.10.2023	
4.	Підготовка матеріалів другого розділу магістерської дисертації	17.10.2023	
5.	Підготовка матеріалів третього розділу магістерської дисертації	03.11.2023	
6.	Підготовка матеріалів четвертого розділу магістерської дисертації	16.11.2023	
7.	Підготовка графічної частини дипломного проекту	28.11.2023	
8.	Оформлення документації дипломного проекту	05.12.2023	
9.	Попередній розгляд магістерської дисертації на засіданні кафедри	21.12.2023	

Студент

Глеб В.Ю.

Науковий керівник дисертації

Тарасенко-Клятченко О.В.

РЕФЕРАТ

Актуальність теми. З поглибленням цифровізації сучасного суспільства виникають нові виклики у забезпеченні конфіденційності та безпеки інформації, яка обмінюється через комп'ютерні мережі. Асиметричні алгоритми шифрування стають ключовими інструментами у захисті даних від несанкціонованого доступу.

Об'єктом дослідження є процес передачі інформації в комп'ютерних мережах, де дані зашифровані асиметричними алгоритмами.

Предметом дослідження є вивчення та аналіз ефективності асиметричних алгоритмів у забезпеченні безпеки передачі інформації через комп'ютерні мережі.

Мета роботи. дослідження та оцінка застосування асиметричних алгоритмів у передачі зашифрованої інформації в комп'ютерних мережах, а також розробка рекомендацій щодо покращення процесів шифрування та дешифрування даних.

Наукова новизна. Розроблений застосунок дозволяє відправляти повідомлення, зашифровані асиметричними алгоритмами. Застосунок демонструє наглядність роботи асиметричного шифрування та можливість використовувати ці алгоритми через готовий інтерфейс.

Практична цінність. Результати дослідження можуть бути використані для розробки та удосконалення засобів захисту інформації в різних сферах, включаючи фінанси, медицину та галузі, пов'язані із зберіганням особистих даних.

Апробація роботи. Основні положення та результати роботи були представлені та обговорювались на V Міжнародній науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2023)

Структура та обсяг роботи. Дипломна робота складатиметься з вступу, чотирьох розділів, висновків та списку використаних джерел літератури. Загальний обсяг роботи буде складати 63 сторінки.

Ключові слова: асиметричне шифрування, шифрування, асиметричні алгоритми шифрування

ABSTRACT

Relevance of the topic. With the deepening digitization of contemporary society, new challenges arise in ensuring the confidentiality and security of information exchanged through computer networks.

The object of research is the process of transmitting information in computer networks where data is encrypted using asymmetric algorithms.

The subject of research is the study and analysis of the effectiveness of asymmetric algorithms in securing the transmission of encrypted information in computer networks, as well as the development of recommendations for improving data encryption and decryption processes.

The purpose of the study. The goal of this thesis is to investigate and evaluate the application of asymmetric algorithms in the transmission of encrypted information in computer networks, as well as to develop recommendations for improving encryption and decryption processes.

Scientific novelty. The developed application allows sending messages encrypted with asymmetric algorithms. The application demonstrates the transparency of asymmetric encryption and the ability to use these algorithms through a ready-made interface.

The practical value of obtained results can be used to develop and improve information security tools in various sectors, including finance, medicine, and areas related to the storage of personal data.

Approval of the work. The main provisions and results of the work were presented and discussed at the V International Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies" (SoftTech-2023).

Structure and scope of the work. The thesis will consist of an introduction, four chapters, conclusions, and a list of used literature sources. The total volume of the work will be 63 pages.

Keywords: asymmetric encryption, encryption, asymmetric encryption algorithms

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ОДИНИЦІ.....	9
ВСТУП.....	11
1.ТЕОРЕТИЧНИЙ ОГЛЯД ОБЛАСТІ ЗАХИСТУ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ МЕРЕЖАХ.....	12
1.1 Обґрунтування Важливості Безпеки Інформації	12
1.2 Основна Концепція Асиметричного Шифрування.....	13
1.3 Публічний та приватний ключі.....	14
1.4 Принципи Роботи Асиметричних Алгоритмів.....	15
1.4.1 Переваги та Застосування.....	16
1.5 Переваги та Застосування Асиметричних Алгоритмів.....	17
1.5.1 Переваги Асиметричних Алгоритмів.....	17
1.5.2 Застосування Асиметричних Алгоритмів.....	18
2. ОСОБЛИВОСТІ АСИМЕТРИЧНИХ АЛГОРИТМІВ	19
2.1 Види асиметричних алгоритмів.....	19
2.2 Принцип роботи асиметричного алгоритму RSA.....	20
2.2.1 Основні Компоненти RSA.....	20
2.2.2 Шифрування.....	20
2.2.3 Розшифрування.....	21
2.2.4 Застосування RSA	21
2.3 Принцип роботи асиметричного алгоритму DSA.....	21
2.3.1 Основні Кроки Роботи DSA.....	22
2.3.2.Перевірка Цифрового Підпису.....	22
2.3.3 Застосування DSA.....	22

2.4 Принцип роботи асиметричного алгоритму ECC.....	23
2.4.1 Основні Компоненти ECC.....	23
2.4.2 Шифрування.....	23
2.4.3 Розшифрування.....	24
2.4.4 Застосування ECC	24
2.5 Принцип роботи асиметричного алгоритму DH (Diffie-Hellman).....	25
2.5.1 Основні Кроки Роботи DH.....	25
2.5.2 Процес Захищеного Обміну Ключами.....	25
2.5.3 Застосування DH.....	26
2.6 Принцип роботи асиметричного алгоритму ElGamal даних.....	26
2.6.1 Основні Кроки Роботи ElGamal.....	26
2.6.2 Шифрування.....	27
2.6.3 Розшифрування.....	27
2.6.4 Застосування ElGamal.....	27
2.7 Принцип роботи асиметричного алгоритму EdDSA.....	28
2.7.1 Основні Кроки Роботи EdDSA.....	28
2.7.2 Підписування.....	29
2.7.3 Перевірка.....	29
2.7.4 Застосування EdDSA.....	29
2.8 Порівняльна характеристика асиметричних алгоритмів.....	30
3. НАПИСАННЯ ЗАСТОСУНКУ, ЯКИЙ РЕАЛІЗУЄ АСИМЕТРИЧНЕ ШИФРУВАННЯ.....	32
3.1 Архітектура застосунку.....	32
3.1.1 Основні принципи клієнтсько-серверної архітектури.....	32
3.1.2 Переваги клієнтсько-серверної архітектури	33
3.1.3 Недоліки клієнтсько-серверної архітектури.....	33
3.2 Мова написання застосунку (JavaScript).....	34
3.2.1 Основні аспекти використання JavaScript.....	34
3.2.2 Переваги використання JavaScript.....	35

3.2.3 Недоліки використання JavaScript.....	35
3.3 Клієнтська сторона застосунку (ReactJS).....	35
3.3.1 Основні характеристики ReactJS.....	36
3.3.2 Використання ReactJS в клієнтській частині.....	36
3.3.3 Переваги використання ReactJS.....	37
3.3.4 Недоліки використання ReactJS.....	37
3.4 Серверна сторона застосунку (NestJS).....	37
3.4.1 Основні характеристики NestJS.....	38
3.4.2 Використання NestJS на сервері.....	38
3.4.3 Переваги використання NestJS.....	39
3.4.4 Недоліки використання NestJS.....	39
3.5 База Даних Застосунку (PostgreSQL)	39
3.5.1 Порівняння з NoSQL Базами Даних.....	40
3.6 Схеми даних.....	40
3.6.1 Таблиця "users".....	41
3.6.2 Таблиця "chats".....	41
3.6.3 Таблиця "messages".....	42
3.6.4 Таблиця "users_chats"	42
3.7 HTTP протокол.....	44
3.7.1 Основні характеристики HTTP.....	44
3.7.2 Методи HTTP запитів.....	45
3.7.3 Статус-коди HTTP.....	45
3.7.4 Заголовки HTTP.....	46
3.7.5 Сесії та куки.....	46
3.7.6 Захист від атак.....	47
3.8 Зв'язок між клієнтом і сервером на основі GraphQL.....	47
3.8.1 Основні принципи GraphQL.....	47
3.8.2 Структура запитів та відповідей GraphQL.....	47
3.8.3 Переваги використання GraphQL в месенджерах.....	48

3.8.4 Реалізація механізму підписки (Subscriptions) в GraphQL.....	48
3.8.5 Виклики до сервера та оптимізація ресурсів.....	48
3.9 Запуск застосунку використовуючи docker і docker compose	48
4. ОГЛЯД ЗАСТОСУНКУ, ЯКИЙ РЕАЛІЗУЄ АСИМЕТРИЧНЕ ШИФРУВАННЯ.....	51
4.1 Сторінка реєстрації.....	51
4.2 Сторінка логіну.....	52
4.3 Головна сторінка додатку.....	52
4.3.1 Завантаження приватного ключа в програму.....	53
4.3.2 Створення нового чату.....	54
4.3.3 Відправка повідомлення користувачеві.....	54
ВИСНОВКИ.....	57
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ ТА ДОДАТКИ.....	58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ОДИНИЦЬ

У даному розділі наведено перелік умовних позначень, скорочень і одиниць, які використовуються в дипломній роботі для забезпечення чіткості та зрозумілості тексту.

1. Асиметричне шифрування (АЕ) - процес шифрування, що використовує ключі для шифрування та розшифрування інформації.
2. Алгоритм шифрування (AS) - математичний метод, який визначає процес шифрування або розшифрування даних.
3. Комп'ютерні мережі (КМ) - інфраструктура, що забезпечує обмін даними між комп'ютерами та іншими пристроями.
4. Мережевий протокол (МП) - набір правил і форматів для обміну даними в комп'ютерних мережах.
5. RSA-алгоритм (RSA) - асиметричний алгоритм шифрування, який використовується для створення криптосистем з відкритим ключем.
6. SSL/TLS-протокол (SSL/TLS) - протоколи, що забезпечують безпеку передачі даних в мережі Інтернет.
7. Elliptic Curve Cryptography (ECC): це криптографічний алгоритм, який використовує еліптичні криві для забезпечення безпеки при передачі даних і для різних криптографічних застосувань, таких як підписи та шифрування.
8. Diffie-Hellman (DH) : це криптографічний протокол для безпечного обміну ключами через ненадійний канал. Використовується для створення загального

секретного ключа між двома сторонами, який може бути використаний для шифрування комунікації.

9. ElGamal Encryption (ElGamal): це криптографічний алгоритм шифрування, який використовує математичні принципи теорії чисел. Названий на честь свого створювача Тахіра Ель-Гамалі.

10. Edwards-curve Digital Signature Algorithm (EdDSA): це алгоритм цифрового підпису, який використовує едвардські криві для створення ефективних та безпечних цифрових підписів.

ВСТУП

В епоху стрімкого розвитку інформаційних технологій, комп'ютерних мереж та високошвидкісного обміну даними, питання захисту конфіденційності та безпеки інформації стає більш нагальним та важливим, ніж будь-коли раніше. Забезпечення цілісності та конфіденційності даних у віртуальному просторі визначається не лише як технічна необхідність, але й як фундаментальний аспект забезпечення довіри до сучасних технологій.

Дипломний проект, присвячений дослідженню та аналізу засобів передачі інформації, що зашифрована асиметричними алгоритмами в комп'ютерних мережах, вирізняється важливістю теми в контексті сучасних викликів та можливостей. Сучасні технології змушують нас переглядати та покращувати існуючі засоби та методи захисту інформації, а асиметричне шифрування виявляється ключовим елементом цього завдання.

Мета дипломного проекту - глибше розібратися в принципах та ефективності асиметричних алгоритмів шифрування у відношенні до передачі інформації через комп'ютерні мережі. Дослідження спрямоване на вдосконалення та оптимізацію існуючих методів захисту, а також на висвітлення практичних застосувань цих технологій у різних галузях.

У контексті актуальності теми та потреби у невпинному вдосконаленні засобів інформаційної безпеки, цей дипломний проект спрямований на високоякісне дослідження, яке має не тільки академічне, але і практичне значення в контексті розвитку сучасних інформаційних технологій та забезпечення їхньої безпеки.

1. ТЕОРЕТИЧНИЙ ОГЛЯД ОБЛАСТІ ЗАХИСТУ ІНФОРМАЦІЇ В КОМП'ЮТЕРНИХ МЕРЕЖАХ

1.1 Обґрунтування важливості безпеки інформації

Безпека інформації стає однією з найбільш критичних аспектів в сучасному цифровому світі. В еру високотехнологічних засобів зберігання та обробки даних, де інформаційні потоки швидко зростають та перетинають глобальні мережі, забезпечення безпеки інформації стає завданням критичного значення.

Сучасні технології викликають зростаючу кількість загроз інформаційній безпеці. Хакерські атаки, фішинг, віруси, витоки даних та інші загрози стають все більш вдосконаленими та важкодоступними для виявлення. Втрати від кібератак вже не обмежуються простою матеріальною шкодою, але також включають репутаційний та фінансовий вплив.

Безпека інформації включає два ключові аспекти: конфіденційність та цілісність. Конфіденційність означає забезпечення того, що інформація доступна лише тим, хто має право на її отримання. Цілісність, у свою чергу, гарантує, що дані залишаються непошкодженими та не змінюються без дозволу.

Нещодавні приклади великих кібератак на фінансові установи та компанії свідчать про те, наскільки суттєвим є питання безпеки інформації. Втрати від таких атак можуть сягати мільйонів, а іноді й мільярдів доларів, позначаючи не тільки фінансовий вплив, але й загрозу для стабільності фінансових ринків.

Крім великих корпорацій, безпека інформації стає актуальною для кожного індивіда. Особиста інформація, така як фінансові дані, медичні записи та особисті листи, є цінним активом. Її втрата може призвести до крадіжки ідентичності, фінансових проблем і порушення особистої конфіденційності.

Безпека інформації грає ключову роль у забезпеченні довіри в інтернет-середовищі. Користувачі та компанії повинні бути впевнені, що їхні дані захищені від несанкціонованого доступу та зміни.

Висока важливість безпеки інформації визначається не лише економічними втратами, але й загрозою для особистого благополуччя та довіри в цифровому світі. Асиметричне шифрування, з його унікальною можливістю використання двох ключів, є важливим інструментом для забезпечення безпеки конфіденційності

1.2 Основна концепція асиметричного шифрування

Асиметричне шифрування - це ключовий компонент сучасних систем криптографії, що відрізняється від традиційних симетричних методів. Основна ідея полягає в тому, що для шифрування та розшифрування використовуються два взаємно пов'язаних ключі: публічний і приватний.

Публічний ключ відомий всім і використовується для шифрування даних, тоді як приватний ключ є конфіденційним і використовується для розшифрування отриманих зашифрованих даних.

Процес асиметричного шифрування розпочинається з обрання публічного ключа отримувача, який використовується для шифрування конфіденційного повідомлення. Зашифроване повідомлення потім надсилається отримувачеві.

Отримувач використовує свій приватний ключ для розшифрування отриманого зашифрованого повідомлення, відновлюючи оригінальне повідомлення в його чіткому вигляді.

Асиметричне шифрування широко використовується в сучасному світі для захисту інтернет-передач, таких як безпечні з'єднання (SSL/TLS), електронна пошта та створення цифрових підписів для підтвердження автентичності інформації.

Узагальнюючи, асиметричне шифрування є важливим інструментом для забезпечення безпеки та конфіденційності інформації в цифровому середовищі.

1.3 Публічний та приватний ключі

Публічний та приватний ключі є основою асиметричного шифрування, визначаючи спосіб захисту та обміну конфіденційною інформацією.

Публічний ключ відкритий для всіх і може бути розповсюджений відправниками та отримувачами. Цей ключ використовується для шифрування інформації перед її відправленням. Оскільки публічний ключ відомий глобально, будь-хто може використовувати його для шифрування повідомлень або файлів перед відправленням до власника приватного ключа.

Приватний ключ є суто конфіденційним для власника. Тільки особа, яка володіє відповідним приватним ключем, може розшифрувати дані, зашифровані з використанням її публічного ключа. Цей ключ зберігається в безпечному місці та ніколи не розголошується. Власник приватного ключа використовує його для розшифрування отриманих зашифрованих повідомлень.

У процесі асиметричного шифрування відправник використовує публічний ключ отримувача для шифрування повідомлення перед відправленням. Отримувач, в свою чергу, використовує свій приватний ключ для розшифрування цього зашифрованого повідомлення. Ця пара ключів дозволяє забезпечити конфіденційність та безпеку обміну інформацією відправників і одержувачів в сучасних криптографічних системах.

Використання публічних та приватних ключів у асиметричному шифруванні є фундаментальним для забезпечення безпеки та конфіденційності даних у цифровому середовищі. Ця система ключів використовується в різних сферах, включаючи електронну пошту, безпечні з'єднання та створення цифрових підписів.

1.4 Принципи роботи асиметричних алгоритмів

Асиметричні алгоритми шифрування базуються на унікальному принципі використання пари ключів: публічного та приватного. Ці алгоритми дозволяють здійснювати безпечний обмін конфіденційною інформацією у відкритих мережах, забезпечуючи конфіденційність та цілісність даних.

Генерація Ключів. Власник асиметричної ключової пари генерує свій публічний та приватний ключі. Публічний ключ розповсюджується, тоді як приватний залишається в безпеці.

Шифрування. Коли відправник хоче надіслати конфіденційне повідомлення отримувачеві, він використовує публічний ключ отримувача для шифрування даних. Таке шифрування гарантує, що лише власник приватного ключа зможе розшифрувати повідомлення.

Розшифрування. Отримувач використовує свій приватний ключ для розшифрування отриманого зашифрованого повідомлення. Такий підхід забезпечує конфіденційність інформації, оскільки лише власник приватного ключа може отримати доступ до оригінального повідомлення.

Цифровий Підпис. Пара ключів також використовується для створення цифрового підпису. Відправник може використовувати свій приватний ключ для створення підпису, а отримувач - публічний ключ для перевірки автентичності підписаного повідомлення.

1.4.1 Переваги та застосування.

- Конфіденційність. Асиметричне шифрування дозволяє захищати дані від несанкціонованого доступу завдяки використанню приватних ключів для розшифрування.
- Автентичність. Використання публічних та приватних ключів дозволяє створювати цифрові підписи для підтвердження автентичності повідомлень.
- Безпека у Відкритих Мережах. Асиметричні алгоритми ефективно застосовуються в Інтернеті, де забезпечення безпеки відкритих з'єднань та електронної пошти є критичним завданням.

Завдання Асиметричних Алгоритмів:

- Захист інформації від несанкціонованого доступу.
- Забезпечення конфіденційності та цілісності даних.
- Здійснення безпечного обміну інформацією в відкритих мережах.
- Застосування цифрових підписів для підтвердження автентичності та неперевершеності повідомлень.

Асиметричні алгоритми є важливим інструментом для забезпечення безпеки та конфіденційності в області криптографії, дозволяючи ефективно використовувати пару ключів для різноманітних цілей у сучасному цифровому середовищі.

1.5 Переваги та застосування асиметричних алгоритмів

Асиметричні алгоритми шифрування відіграють ключову роль у забезпеченні безпеки та конфіденційності інформації у віртуальному просторі. Їхні переваги та застосування включають різноманітні сфери, від захисту особистих даних до забезпечення безпеки великих корпоративних мереж.

1.5.1 Переваги асиметричних алгоритмів

- Конфіденційність та Безпека. Асиметричні алгоритми забезпечують високий рівень конфіденційності, оскільки приватний ключ залишається тільки у власника. Це робить надійним шляхом захисту важливих даних.
- Цифрові Підписи. Використання публічних та приватних ключів дозволяє створювати цифрові підписи, які підтверджують автентичність повідомлень та документів.
- Безпека в Відкритих Мережах. Асиметричне шифрування є ефективним засобом для забезпечення безпеки відкритих з'єднань, таких як безпечні з'єднання SSL/TLS в інтернеті.
- Гнучкість та Зручність. Можливість вільного розповсюдження публічного ключа робить асиметричні алгоритми гнучкими та зручними для використання в обширному спектрі застосувань.

1.5.2 Застосування асиметричних алгоритмів

- Захист Електронної Пошти. Використання асиметричного шифрування для шифрування та цифрового підпису електронних листів.
- Безпечні З'єднання. Шифрування інтернет-з'єднань з використанням асиметричних алгоритмів (SSL/TLS) для захисту онлайн-транзакцій та особистих даних.
- Цифрові Підписи та Аутентифікація. Застосування для створення цифрових підписів для підтвердження автентичності та неперевершеності документів.
- Криптовалюти та Блокчейн. Використання асиметричних алгоритмів для створення криптовалют та забезпечення безпеки транзакцій у блокчейні.
- Корпоративна Безпека. Захист корпоративних мереж та конфіденційної інформації великих підприємств.

Висновки

За останні роки потужність комп'ютерів виросла в мільйони разів, що відкриває можливість простого підбору для того, щоб отримати розшифрований текст. В такому випадку світу були потрібні зовсім іншого рівня зі сторони безпеки алгоритми шифрування. З метою забезпечення безпеки інформації були створені асиметричні алгоритми шифрування, які забезпечують зовсім інший рівень надійності.

2. ОСОБЛИВОСТІ АСИМЕТРИЧНИХ АЛГОРИТМІВ

2.1 Види асиметричних алгоритмів

На даний момент найпопулярнішими асиметричними алгоритмами шифрування є:

- RSA (Rivest–Shamir–Adleman): Це один з найбільш відомих та широко використовуваних асиметричних алгоритмів. Використовується для шифрування та підпису даних.
- DSA (Digital Signature Algorithm): Розроблений для генерації цифрових підписів і перевірки їхньої автентичності.
- ECC (Elliptic Curve Cryptography): Використовує математичні властивості еліптичних кривих для забезпечення безпеки. Ефективніше за RSA при різній довжині ключа.
- DH (Diffie-Hellman): Використовується для безпечного обміну ключами у відкритих мережах та заснований на проблемі дискретного логарифмування.
- ElGamal: Використовується для шифрування і підпису даних. Заснований на важкій задачі дискретного логарифмування.
- EdDSA (Edwards-curve Digital Signature Algorithm): Використовує криві Едвардса та забезпечує високу безпеку при меншому обсязі даних порівняно з іншими алгоритмами.
- Post-Quantum Cryptography Algorithms: Оскільки квантові комп'ютери можуть долати деякі сучасні асиметричні алгоритми, вивчається нове покоління алгоритмів, таких як NTRUEncrypt, Kyber, Falcon, інші.

2.2 Принцип роботи асиметричного алгоритму RSA

RSA (Rivest–Shamir–Adleman) є одним із найбільш відомих та широко використовуваних асиметричних алгоритмів шифрування та генерації цифрових підписів. Названий на честь винахідників – Рона Рівеста, Аді Шаміра та Леонарда Адлемана, алгоритм базується на математичних властивостях складних арифметичних задач та теорії чисел.

2.2.1 Основні компоненти RSA

- Генерація Ключів. Отримання двох простих великих чисел, скажімо, p та q . Обчислення їхнього добутку $n = pq$, який стає модулем для публічного та приватного ключів.
- Вибір Параметра e . Вибір відкритого експонента e , який є невеликим цілим числом і не є дільником (крім 1) значення функції Ейлера ($\phi(n) = (p-1)(q-1)$), де ϕ - функція Ейлера).
- Вибір Закритого Експонента d . Обчислення закритого експонента d , яке є мультиплікативно оберненим до e в модулі $\phi(n)$, тобто $ed \equiv 1 \pmod{\phi(n)}$.
- Публічний та Приватний Ключі.
 - Публічний ключ: (e, n)
 - Приватний ключ: (d, n)

2.2.2 Шифрування

- Отримання публічного ключа отримувача: (e, n) .
- Перетворення повідомлення M в ціле число m , де $0 < m < n$.
- Шифрування повідомлення за допомогою публічного ключа: $C \equiv m^e \pmod{n}$.

- Відправлення шифрованого тексту C отримувачу.

2.2.3 Розшифрування

- Отримання зашифрованого тексту C .
- Використання приватного ключа для розшифрування: $m \equiv C^d \pmod{n}$.
- Відновлення оригінального повідомлення M з отриманого числового значення m .

2.2.4 Застосування RSA

- Шифрування даних. Використовується для зашифрування чутливої інформації при передачі через ненадійні мережі.
- Цифрові підписи. Генерація та перевірка цифрових підписів для підтвердження автентичності повідомлень.
- Безпека інтернет-з'єднань (SSL/TLS). Використовується для захисту безпеки та конфіденційності даних під час онлайн-транзакцій.

RSA залишається важливим елементом криптографії, хоча в останні роки йому властива конкуренція від алгоритмів на основі еліптичних кривих, особливо при областях, де важливо зменшення обсягу обчислень.

2.3 Принцип роботи асиметричного алгоритму DSA

DSA (Digital Signature Algorithm) є асиметричним алгоритмом, розробленим для генерації цифрових підписів та перевірки їхньої автентичності. Цей алгоритм часто використовується для створення цифрових підписів в контексті безпеки інтернет-передач, електронного голосування та інших додатків, де важлива автентичність інформації.

2.3.1 Основні кроки роботи DSA

- Генерація параметрів. Вибір простого числа p (порядок групи) та числа q (безпечний простий дільник $p-1$). Обчислення генератора g (певний номер, де $1 < g < p-1$).
- Генерація ключів. Обирається випадковий приватний ключ x такий, що $0 < x < q$. Обчислюється публічний ключ $y = g^x \mod p$.
- Генерація цифрового підпису. Обирається випадковий тимчасовий ключ k такий, що $0 < k < q$. Обчислюється $r = (g^k \mod p) \mod q$. Обчислюється $s = k^{-1} (H(m) + xr) \mod q$, де $H(m)$ - хеш повідомлення m .

2.3.2 Перевірка цифрового підпису

- Отримуються підписані повідомлення (r, s) та відповідне публічне ключ y .
- Обчислюється $w = s^{-1} \mod q$.
- Обчислюється $u_1 = (H(m)w) \mod q$ та $u_2 = (rw) \mod q$.
- Обчислюється $v = ((g^{u_1} y^{u_2} \mod p) \mod q)$.
- Підпис є автентичним, якщо $v = r$.

2.3.3 Застосування DSA

- Цифрові підписи. Забезпечення автентичності та неперевершеності електронних документів та повідомлень.
- Безпека протоколів обміну ключами. Використовується для генерації цифрових підписів у протоколах обміну ключами, наприклад, в TLS/SSL

для безпечних з'єднань.

- Електронне голосування. Забезпечення автентичності електронних голосів.
- Безпека криптовалют. Деякі криптовалютні протоколи використовують DSA для підпису транзакцій.

DSA використовується там, де важлива перевірка автентичності, але останнім часом він втрачає популярність на користь алгоритмів на основі еліптичних кривих, які можуть забезпечувати аналогічний рівень безпеки при менших обчислювальних витратах.

2.4 Принцип роботи асиметричного алгоритму ECC

Еліптична криптографія (ECC - Elliptic Curve Cryptography) є асиметричним алгоритмом шифрування, який базується на використанні математичних властивостей еліптичних кривих над полями скінченного порядку. ECC володіє високою ефективністю та забезпечує високий рівень безпеки при коротших ключах порівняно з традиційними асиметричними алгоритмами, такими як RSA.

2.4.1 Основні компоненти ECC

- Еліптична крива. Визначається рівнянням $y^2 = x^3 + ax + b$, де a та b - параметри, існуючі над полем скінченного порядку.
- Публічний та приватний ключі. Кожен користувач має свій приватний ключ (скаляр) d . Відповідний публічний ключ (точка на кривій) обчислюється як $Q = d \cdot G$, де G - генераторна точка на еліптичній кривій.

2.4.2 Шифрування

- Вибір випадкової точки. Вибір випадкової точки P на еліптичній кривій.

- Обчислення ключа. Обчислення публічного ключа для отримувача $(Q = d \cdot G)$.
- Створення ключового тексту. Використання точки (P) та публічного ключа (Q) для створення ключового тексту.

2.4.3 Розшифрування

- Отримання секретного ключа. Власник приватного ключа обчислює скаляр (d) для точки (P) $(P = d \cdot G)$.
- Розшифрування ключового тексту. Використання отриманого скаляру та криптограми для розшифрування оригінального тексту.

2.4.4 Застосування ECC

- Безпека інтернет-з'єднань (SSL/TLS). Використовується для забезпечення конфіденційності та цілісності даних у веб-браузерах.
- Мобільні додатки та IoT. Ефективно використовується в обміні даними в мобільних додатках та в інтернеті речей (IoT).
- Безпека криптовалют. Використовується для генерації ключів та підписів в криптовалютних протоколах.
- Безпека електронних платежів. Застосовується для генерації цифрових підписів та забезпечення безпеки транзакцій.
- Корпоративна безпека. Використовується для захисту конфіденційної інформації у корпоративних мережах.

ECC стає все більш популярною завдяки своїй ефективності та безпеці, що робить її важливим компонентом сучасних криптографічних систем.

2.5 Принцип роботи асиметричного алгоритму DH (Diffie-Hellman)

Асиметричний алгоритм Diffie-Hellman (DH) використовується для безпечного обміну секретними ключами через ненадійні канали зв'язку. Його основний принцип полягає в тому, що дві сторони можуть створити спільний секретний ключ, не викриваючи його в іншим, навіть якщо сам обмін ключами протікає у відкритому доступі.

2.5.1 Основні кроки роботи DH

- Вибір параметрів. Обрання простого числа (p) та примітивного кореня за модулем (p) , що є генератором мультиплікативної групи.
- Генерація приватного ключа. Кожна сторона вибирає випадковий приватний ключ (a) або (b) .
- Генерація публічного ключа. Обчислення $(A = g^a \mod p)$ та $(B = g^b \mod p)$.
- Обмін публічними ключами. Сторони обмінюються значеннями (A) та (B) .
- Обчислення спільного секретного ключа. Кожна сторона обчислює спільний секретний ключ, який є $(K = B^a \mod p)$ або $(K = A^b \mod p)$.

2.5.2 Процес захищеного обміну ключами

- Аліса та Боб обирають параметри. Вони обирають просте число (p) та примітивний корінь (g) за модулем (p) . Кожна сторона обирає свій приватний ключ. Аліса обирає (a) , а Боб обирає (b) .
- Генерація та обмін публічними ключами. Аліса обчислює $(A = g^a \mod p)$ та відправляє (A) Бобу. Боб обчислює $(B = g^b \mod p)$ та відправляє (B) Алісі.

- Обчислення спільного секретного ключа. Аліса обчислює $(K = B^a \bmod p)$. Боб обчислює $(K = A^b \bmod p)$.
- Захищений обмін ключами. Тепер Аліса та Боб обидва володіють спільним секретним ключем (K) , яким можуть шифрувати та розшифровувати свої повідомлення.

2.5.3 Застосування DH

- SSL/TLS протокол. Використовується для безпечного обміну ключами між клієнтом та сервером.
- VPN тунелі. Застосовується для створення безпечних тунелів між вузлами VPN.
- Безпека електронної пошти. Використовується для забезпечення конфіденційності та цілісності електронних листів.
- Захист з'єднань в бездротових мережах. Використовується для безпечного обміну ключами у бездротових комунікаціях.

Алгоритм Diffie-Hellman є важливою складовою для забезпечення безпеки у безпечних з'єднаннях та інших сценаріях, де необхіден захищений обмін ключами.

2.6 Принцип роботи асиметричного алгоритму ElGamal

Асиметричний алгоритм шифрування ElGamal названий на честь свого творця Тахера Ель-Гамала та використовується для захищеного обміну повідомленнями та шифрування даних. Основна ідея алгоритму полягає в складних операціях з дискретним логарифмуванням на групі, заснованій на кінцевому полі.

2.6.1. Основні Кроки Роботи ElGamal

- Генерація параметрів. Вибір безпечного простого числа (p) та примітивного кореня (g) за модулем (p) для створення групи.

- Генерація ключів. Обирається випадковий приватний ключ (x) такий, що $(0 < x < p-1)$. Обчислюється відповідний публічний ключ $(y = g^x \bmod p)$.
- Шифрування. Кожен користувач обирає випадковий секретний ключ (k) та обчислює $(K = y^k \bmod p)$ та $(C_1 = g^k \bmod p)$. Кожен біт повідомлення (m) шифрується за допомогою операції $(C_2 = m \cdot K \bmod p)$.

2.6.2 Шифрування

- Вибір випадкового секретного ключа (k) .
- Обчислення $(K = y^k \bmod p)$ та $(C_1 = g^k \bmod p)$.
- Шифрування повідомлення (m) : $(C_2 = m \cdot K \bmod p)$.

2.6.3 Розшифрування

- Отримання секретного ключа $(K = C_1^x \bmod p)$.
- Відновлення оригінального повідомлення $(m = C_2 \cdot K^{-1} \bmod p)$.

2.6.4 Застосування ElGamal

- Захищений обмін ключами. Використовується для безпечного обміну секретними ключами у протоколах безпеки, таких як SSH.
- Безпечна електронна пошта. Використовується для шифрування електронних листів та захисту конфіденційної інформації.
- Криптографічні протоколи. Входить в склад різноманітних криптографічних протоколів для безпечної комунікації.

Алгоритм ElGamal використовується в різних сценаріях та є важливою складовою сучасної криптографії для забезпечення безпеки під час обміну та збереження даних.

2.7 Принцип роботи асиметричного алгоритму EdDSA (Edwards-curve Digital Signature Algorithm)

EdDSA є асиметричним алгоритмом для генерації цифрових підписів на основі кривих Едвардса. Цей алгоритм надає ефективність та безпеку, що робить його популярним для використання в сучасних криптографічних системах.

2.7.1 Основні кроки роботи EdDSA

- Генерація кривої. Вибір параметрів для кривої Едвардса, яка задовольняє певним властивостям безпеки.
- Генерація ключів. Обирається випадковий приватний ключ (k) та обчислюється відповідний публічний ключ $(A = [k]B)$, де (B) - генераторна точка на кривій.
- Підписування:
 - Обчислюється геш підписуваного повідомлення $(H(m))$.
 - Генерується тимчасовий ключ (r) та обчислюється $(R = [r]B)$.
 - Обчислюється $(S = r + H(R, A, m) \cdot k \mod q)$, де (q) - порядок групи точок на кривій.
- Перевірка Підпису.
 - Отримуючи підпис $((R, S))$, обчислюється геш повідомлення $(H(m))$.
 - Обчислюється $(U_1 = [H(m) \cdot S^{-1}]B + [R \cdot S^{-1}]A)$.
 - Підпис вважається дійсним, якщо координата (x) точки (U_1) дорівнює координаті (x) точки (R) .

2.7.2 Підписування

- Обирання приватного ключа (k) .
- Обчислення відповідного публічного ключа $(A = [k]B)$.
- Обчислення геша повідомлення $(H(m))$.
Генерація тимчасового ключа (r) та обчислення $(R = [r]B)$.
- Обчислення підпису $(S = r + H(R, A, m) \cdot k \mod q)$.

2.7.3 Перевірка

- Отримання підпису $((R, S))$ та геша повідомлення $(H(m))$.
- Обчислення оберненого елемента (S^{-1}) за модулем (q) .
- Обчислення точок $(U_1 = [H(m) \cdot S^{-1}]B + [R \cdot S^{-1}]A)$ та $(U_2 = [H(R) \cdot S^{-1}]B)$.
- Перевірка, чи рівні координати (x) точок (U_1) та (U_2) .

2.7.4 Застосування EdDSA

- Криптовалютні протоколи. Використовується для генерації цифрових підписів у криптовалютних транзакціях (наприклад, в протоколі Libra Blockchain).
- SSL/TLS протокол. Застосовується для забезпечення безпеки у протоколах безпечної комунікації в мережі Інтернет.
- Інтернет в речах (IoT). Використовується для захисту комунікацій в мережах IoT.

Алгоритм EdDSA відомий своєю безпекою та ефективністю, що робить його популярним у різних областях, де потрібен надійний асиметричний алгоритм.

2.8 Порівняльна характеристика асиметричних алгоритмів

Розглянемо порівняльну характеристику алгоритмів RSA, DSA, ECC, DH, ElGamal і EdDSA:

RSA (Rivest–Shamir–Adleman)

- Тип алгоритму. Криптосистема з відкритим ключем (асиметричний).
- Обчислювальна складність. Залежить від розміру ключа; операції шифрування та розшифрування можуть бути витратними.
- Використання. Підпис та шифрування.

DSA (Digital Signature Algorithm).

- Тип алгоритму. Асиметричний для цифрового підпису.
- Обчислювальна складність. Залежить від розміру ключа; швидший у порівнянні з RSA.
- Використання. Цифровий підпис.

ECC (Elliptic Curve Cryptography)

- Тип алгоритму. Асиметричний.
- Обчислювальна складність. Вимагає менше ресурсів порівняно з RSA та DSA.
- Використання. Підпис, шифрування, обмін ключами.

DH (Diffie-Hellman)

- Тип алгоритму. Асиметричний для обміну ключами.
- Обчислювальна складність. Використовується для безпечного обміну ключами.
- Використання. Обмін ключами у криптографічних протоколах.

ElGamal

- Тип алгоритму. Асиметричний для шифрування та підпису.
- Обчислювальна складність. Вимагає обчислень у групі.
- Використання. Шифрування та цифровий підпис.

EdDSA (Edwards-curve Digital Signature Algorithm)

- Тип алгоритму. Асиметричний для цифрового підпису.
- Обчислювальна складність. Відносно легший у порівнянні з RSA та DSA.
- Використання. Цифровий підпис.

Ці алгоритми мають різні характеристики, такі як ефективність обчислень, вимоги до ресурсів та відомі застосування. Вибір конкретного алгоритму повинен враховувати конкретні вимоги до безпеки, продуктивності та контексту використання.

Висновки

У даній частині дипломної роботи було розглянуто особливості асиметричних алгоритмів шифрування. Було детально проаналізовано різні види асиметричних алгоритмів, такі як RSA, DSA, ECC, Diffie-Hellman, ElGamal і EdDSA.

Досліджено принцип роботи кожного з них, розкрито основні компоненти та етапи їхньої реалізації. Зокрема, надано інформацію про шифрування та розшифрування в контексті кожного алгоритму, а також про їхнє застосування у практиці.

Порівняльна характеристика асиметричних алгоритмів дозволяє зробити висновки щодо їхньої ефективності, швидкодії та безпеки. Ця інформація є важливою для вибору оптимального алгоритму з урахуванням конкретних

вимог та завдань системи шифрування. Результати порівняння надають змогу зрозуміти, в яких випадках краще використовувати конкретний асиметричний алгоритм.

3. НАПИСАННЯ ЗАСТОСУНКУ, ЯКИЙ РЕАЛІЗУЄ АСИМЕТРИЧНЕ ШИФРУВАННЯ

3.1 Архітектура застосунку

Клієнтсько-серверна архітектура є однією з найбільш поширених моделей для розробки та розгортання застосунків. Ця архітектура базується на розподілених системах, де функції та відповідальності розділені між двома основними компонентами: клієнтом і сервером.

3.1.1 Основні принципи клієнтсько-серверної архітектури

Розділення відповідальностей:

- Сервер. Відповідає за обробку та зберігання даних, виконання бізнес-логіки, та забезпечення доступу до ресурсів.
- Клієнт. Виконує інтерфейсну частину, надаючи зручний інтерфейс для користувача та взаємодіючи з сервером.

Забезпечення відмінності:

- Інтерфейс. Клієнт відповідає за реалізацію графічного інтерфейсу та взаємодії з користувачем.
- Бізнес-логіка. Сервер містить бізнес-логіку, що визначає обробку даних та логіку застосунку.

Масштабованість:

- Горизонтальна масштабованість. Спрощується розширення шляхом додавання нових серверів або клієнтів.
- Вертикальна масштабованість. Збільшення потужності серверів чи клієнтів для покращення продуктивності.

Незалежність інтерфейсу:

- Різноманітність клієнтів. Різні типи клієнтів (наприклад, веб-браузери, мобільні додатки) можуть використовувати один сервер.

Комунікація через мережу:

- Протоколи. Використання мережевих протоколів для обміну даними між клієнтом та сервером (наприклад, HTTP для веб-додатків).

3.1.2 Переваги клієнтсько-серверної архітектури

- Розподілена робота. Дозволяє ефективно розподілити завдання між клієнтом та сервером.
- Спрощена розробка. Розподілені обов'язки спрощують розробку та підтримку застосунків.
- Масштабованість. Можливість масштабування клієнтів та серверів для відповіді на зростання обсягів даних або користувачів.
- Забезпечення безпеки. Можливість обмеження доступу до ресурсів та контролю за безпекою на рівні сервера.

3.1.3 Недоліки клієнтсько-серверної архітектури

- Залежність від мережі. Робота системи вимагає наявності стабільного мережевого з'єднання.
- Серверна навантаженість. Сервер повинен обробляти запити багатьох клієнтів, що може стати джерелом навантаження.
- Синхронність. Клієнт чекає на відповідь сервера, що може спричинити затримки.
- Складність підтримки. Розподілені системи можуть бути складними для виявлення та усунення несправностей.

Клієнтсько-серверна архітектура застосувань є важливою у світі розподілених систем, де вона дозволяє ефективно розподіляти ресурси та забезпечувати зручний інтерфейс для користувачів у різних засобах доступу. Правильне проектування цієї архітектури дозволяє покращити продуктивність та масштабованість системи.

3.2 Мова написання застосунку (JavaScript)

Вибір мови програмування для написання застосунку - важливий етап розробки, який впливає на продуктивність, швидкість розробки та обслуговування коду. У вашому випадку, мова програмування - JavaScript.

3.2.1 Основні аспекти використання JavaScript

- Універсальність. JavaScript є універсальною мовою, яка підтримується браузерами для реалізації клієнтської логіки та Node.js для створення серверної сторони.
- Реалізація клієнтської логіки. JavaScript використовується для написання клієнтської логіки в браузері, забезпечуючи динамічний та інтерактивний інтерфейс.
- Серверна сторона. JavaScript може бути використаний для розробки серверної сторони застосунків за допомогою платформи Node.js.
- Швидкість розробки. Зручність роботи з браузерними DevTools, багатofункціональність бібліотек та фреймворків прискорюють розробку.
- Багатопоточність. JavaScript використовує події та зворотні виклики для реалізації асинхронного програмування, що полегшує роботу з паралельними задачами.
- Велика спільнота. JavaScript має велику та активну спільноту розробників, що дозволяє швидко знаходити рішення та розв'язувати проблеми.

3.2.2 Переваги використання JavaScript

- Кросплатформенність. Можливість використовувати JavaScript як на клієнтській, так і на серверній стороні, що полегшує створення універсальних застосунків.
- Швидкість розробки. Доступність багатofункціональних бібліотек та фреймворків сприяє швидкій розробці.
- Широкі можливості. Велика кількість інструментів та ресурсів для розробки різних типів застосунків.

3.2.3 Недоліки використання JavaScript

- Визначеність типів. JavaScript є слаботипізованою мовою, що може призводити до непередбачуваного поведінки.
- Швидкість виконання. У порівнянні з мовами низькорівневого рівня (наприклад, C++), JavaScript може бути менш ефективним з точки зору швидкості виконання.
- Керування пам'яттю. JavaScript автоматично керує пам'яттю, що може призводити до проблем із збіркою сміття.

Використання JavaScript дозволяє створювати універсальні та швидкореагуючі застосунки, але важливо враховувати його обмеження та особливості при розробці великих та складних систем.

3.3 Клієнтська сторона застосунку (ReactJS)

Клієнтська сторона застосунку використовує фреймворк ReactJS для створення ефективних та інтерактивних користувацьких інтерфейсів. ReactJS є відкритим віджетним фреймворком, розробленим Facebook, який дозволяє розробникам легко створювати складні інтерфейси з використанням компонентного підходу.

3.3.1 Основні характеристики ReactJS

- Компонентний підхід. ReactJS базується на концепції компонентів, які можна перевикористовувати та легко обслуговувати. Кожен компонент відповідає за свою частину інтерфейсу, спрощуючи розробку та підтримку.
- Віртуальний DOM. React використовує віртуальний DOM для ефективного оновлення та маніпулювання DOM-елементами. Це забезпечує оптимізацію продуктивності застосунку.
- Односторінкові додатки (SPA). React дозволяє легко створювати односторінкові додатки, що покращує швидкість та зручність взаємодії з користувачем.
- Декларативний синтаксис. Декларативний підхід робить код React більш простим для розуміння та обслуговування. Розробники описують, як має виглядати інтерфейс, і React самостійно вирішує, як його побудувати.
- Спільнота та екосистема. React має велику та активну спільноту розробників, а також широку екосистему додаткових бібліотек і інструментів для полегшення розробки.

3.3.2 Використання ReactJS в клієнтській частині

- Створення користувацьких інтерфейсів. ReactJS використовується для розробки зручних та динамічних користувацьких інтерфейсів, що взаємодіють з користувачем без перезавантаження сторінки.
- Управління станом. React дозволяє ефективно управляти станом компонентів, що спрощує роботу зі змінами даних та оновленням інтерфейсу.
- Взаємодія з сервером. Використання React дозволяє легко взаємодіяти з сервером через AJAX-запити або використання бібліотек, таких як Axios.

- Маршрутизація. React використовується для створення системи маршрутизації в SPA, що дозволяє користувачеві легко переміщатися між різними сторінками додатку.

3.3.3 Переваги використання ReactJS

- Ефективність та оптимізація. Використання віртуального DOM та ефективного оновлення дозволяє підтримувати високу продуктивність.
- Спрощена розробка. Компонентний підхід спрощує структуру коду та полегшує розробку.
- Активна спільнота та підтримка. Наявність великої спільноти розробників та регулярні оновлення дозволяють тримати застосунок у відповідності до сучасних стандартів.

3.3.4 Недоліки використання ReactJS

- Великий розмір бібліотеки. Деякі проекти можуть стикатися з проблемою великого обсягу завантаження через розмір бібліотеки React.
- Брак уніфікації. В React-екосистемі є багато різних бібліотек та інструментів, що може призводити до неоднорідності у виборі та підтримці різних рішень.

ReactJS використовується для розробки клієнтської сторони застосунків, де його потужність управління інтерфейсом, станом та компонентністю дозволяє забезпечити високоякісний та динамічний користувацький досвід.

3.4 Серверна сторона застосунку (NestJS)

Серверна сторона застосунку використовує фреймворк NestJS для створення ефективних та масштабованих серверів, які забезпечують обробку запитів та взаємодію з базою даних. NestJS — це фреймворк для створення серверних додатків на основі Node.js, який поєднує принципи об'єктно-орієнтованого програмування, функціонального програмування та функціонального реактивного програмування.

3.4.1 Основні характеристики NestJS

- Модульність та компоненти. NestJS побудований на концепції модульності та компонентів, що спрощує структуру коду та робить його легко розширюваним та підтримуваним.
- Dependency injection. Використання Dependency Injection (DI) дозволяє краще керувати залежностями та забезпечує гнучкість у виборі та заміні компонентів.
- Типізація та TypeScript. NestJS написаний на мові TypeScript, що дозволяє використовувати статичну типізацію та підвищує читабельність коду.
- Middleware та pipes. Використання middleware та pipes дозволяє забезпечувати обробку запитів на різних етапах обробки.
- ORM та база даних. NestJS легко інтегрується з різними ORM (наприклад, TypeORM) та базами даних, спрощуючи роботу з персистентними даними.

3.4.2 Використання NestJS на сервері

- Створення API. NestJS використовується для розробки API, що дозволяє клієнтам взаємодіяти з сервером та отримувати необхідні дані.
- Маршрутизація та контролери. Використання маршрутизації та контролерів дозволяє керувати тим, як сервер обробляє різні типи запитів.
- Робота з базою даних. NestJS використовується для взаємодії з базами даних, забезпечуючи доступ до даних та їх обробку.
- WebSocket та реальний час. NestJS підтримує WebSocket та може використовуватися для реалізації функцій в режимі реального часу.

3.4.3 Переваги використання NestJS

- Модульність та компонентна архітектура. Спрощена структура коду дозволяє розробникам легше розуміти та підтримувати систему.
- Типізація та TypeScript. Використання TypeScript полегшує виявлення та усунення помилок, а також підвищує ефективність розробки.
- Інтеграція з ORM та базами даних. Зручна інтеграція з ORM дозволяє легко взаємодіяти з базами даних та забезпечити персистентність.

3.4.4 Недоліки використання NestJS

- Великий інтерфейс та навантаження. Для менших проектів або мікросервісів може здаватися зайвою складністю.

NestJS використовується для побудови ефективних та модульних серверних сторін застосунків, де його компонентна архітектура, типізація та інші характеристики дозволяють створювати надійні та масштабовані серверні додатки.

3.5 База Даних Застосунку (PostgreSQL)

PostgreSQL - реляційна база даних з відкритим вихідним кодом, яка підтримує ряд передових функцій та розширень. Основні характеристики PostgreSQL включають:

- Реляційна Структура. Використовує модель даних реляційної бази даних, що включає таблиці, зв'язки та SQL для операцій з даними.
- Підтримка SQL. PostgreSQL повністю сумісний з мовою SQL, що полегшує взаємодію та виконання операцій з даними.
- Транзакції та ACID. Забезпечує підтримку транзакцій та гарантує дотримання принципів ACID (Atomicity, Consistency, Isolation, Durability).

- Розширені Функції. Підтримує ряд розширених функцій, таких як власні типи даних, функціональні індекси, тригери та інше.
- Гнучкість та Розширюваність. PostgreSQL дозволяє визначати власні типи даних та функції, що робить його гнучким та розширюваним.

3.5.1 Порівняння з NoSQL Базами Даних

- Сховище даних. PostgreSQL - реляційна база даних, в той час як NoSQL бази даних (наприклад, MongoDB, Cassandra) можуть бути документ-орієнтованими, ключ-значеннями, колоночними тощо.
- Схема даних. В PostgreSQL потрібно визначити структуру даних (схему) перед їх використанням. NoSQL бази даних дозволяють гнучку схему (документи можуть мати різні поля).
- Транзакції та ACID. PostgreSQL гарантує ACID-властивості для транзакцій. У NoSQL базах даних може відбутися узгодженість за кількома або жодною з ACID-властивостей.
- Гнучкість та швидкодія. NoSQL бази даних можуть бути більш гнучкими та швидкими у випадках, коли необхідно обробляти великі обсяги даних з різною структурою.
- Запити та мова запитів. PostgreSQL використовує SQL для запитів. У NoSQL можуть використовуватися різні мови запитів (наприклад, MongoDB використовує JavaScript-подібну мову запитів).

Висновки

Вибір між PostgreSQL та NoSQL базами даних залежить від конкретних потреб застосунку. PostgreSQL підходить для ситуацій, коли важлива структура даних, ACID-транзакції та повна сумісність з SQL. NoSQL бази даних гнучкі та швидкі, але можуть бути менш консистентними та вимагати особливого підходу до сховища даних.

3.6 Схема даних

Схема даних для застосунку включає в себе 4 нормалізовані таблиці даних:

- users
- chats
- messages
- user_chats

3.6.1 Таблиця "users"

Таблиця "users" містить інформацію про користувачів месенджера. Кожен користувач має унікальний ідентифікатор (`user_id`), нікнейм (`nickname`), пароль (`password`), токен для аутентифікації (`refresh_token`), публічний ключ (`public_key`) та часові мітки для визначення часу створення та останнього оновлення запису.

Поля

- `user_id` (`uuid`). Унікальний ідентифікатор користувача, генерується автоматично за допомогою `uuid_generate_v4()`.
- `nickname` (`varchar(20)`). Нікнейм користувача, обмежений 20 символами.
- `password` (`text`). Хеш паролю користувача.
- `token_valid_from` (`timestamp with time zone`). Час початку дії токена користувача.
- `refresh_token` (`varchar`). Токен для оновлення аутентифікації користувача.
- `public_key` (`varchar`). Публічний ключ користувача.
- `created_at` (`timestamp`). Час створення запису.
- `updated_at` (`timestamp`). Час останнього оновлення запису.

Ключі

- `PK_user_id` (`Primary Key`). Унікальний ідентифікатор користувача є первинним ключем таблиці.

3.6.2 Таблиця "chats"

Таблиця "chats" зберігає інформацію про чати в месенджері. Кожен чат має унікальний ідентифікатор (chat_id) та часові мітки для визначення часу створення та останнього оновлення запису.

Поля

- chat_id (uuid). Унікальний ідентифікатор чату, генерується автоматично за допомогою uuid_generate_v4().
- created_at (timestamp). Час створення запису.
- updated_at (timestamp). Час останнього оновлення запису.

Ключі

- PK_chat_id (Primary Key). Унікальний ідентифікатор чату є первинним ключем таблиці.

3.6.3 Таблиця "messages"

Таблиця "messages" містить дані про повідомлення у чатах месенджера. Кожне повідомлення має унікальний ідентифікатор (message_id), текстовий контент (payload), час створення запису, а також посилання на користувача-відправника та чат, до якого воно відноситься.

Поля

- message_id (uuid). Унікальний ідентифікатор повідомлення, генерується автоматично за допомогою uuid_generate_v4().
- payload (varchar). Текстовий контент повідомлення.
- created_at (timestamp). Час створення запису.
- senderUserId (uuid). Зовнішній ключ, що посилається на ідентифікатор користувача-відправника в таблиці "users".
- chatChatId (uuid). Зовнішній ключ, що посилається на ідентифікатор чату в таблиці "chats".

Ключі

- PK_message_id (Primary Key). Унікальний ідентифікатор повідомлення є первинним ключем таблиці.
- FK_sender_id (Foreign Key). Зовнішній ключ для зв'язку з ідентифікатором користувача-відправника.
- FK_chat_id (Foreign Key). Зовнішній ключ для зв'язку з ідентифікатором чату.

3.6.4 Таблиця "users_chats"

Таблиця "users_chats" встановлює зв'язок багато до багатьох між користувачами і чатами. Кожен запис містить посилання на користувача та чат, до якого він належить.

Поля

- usersUserId (uuid). Зовнішній ключ, що посиляється на ідентифікатор користувача в таблиці "users".
- chatsChatId (uuid). Зовнішній ключ, що посиляється на ідентифікатор чату в таблиці "chats".

Ключі

- PK_user_chat_id (Primary Key). Первинний ключ, що об'єднує ідентифікатор користувача та чату.
- FK_user_id (Foreign Key). Зовнішній ключ для зв'язку з ідентифікатором користувача.

- FK_chat_id (Foreign Key) Зовнішній ключ для зв'язку з ідентифікатором чату.

Ці таблиці спільно працюють для забезпечення зберігання та взаємодії даних користувачів, чатів та повідомлень у месенджері.

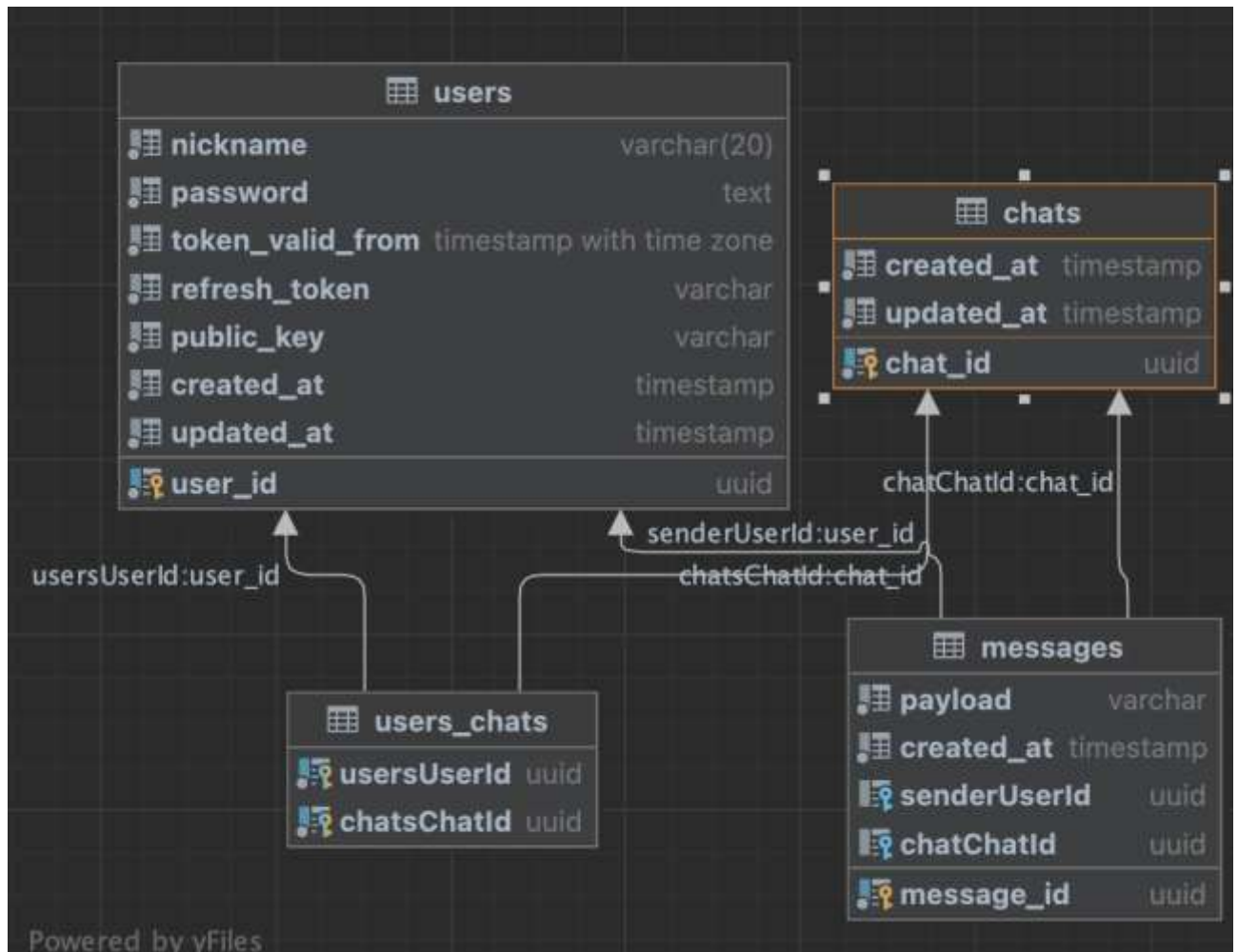


Рисунок 3.1 - Реляційна схема бази даних

Розділ 3.7 HTTP протокол

HTTP (Hypertext Transfer Protocol) є основним протоколом передачі даних у світі Всесвітньої павутини. У цьому розділі розглядається роль HTTP у функціонуванні месенджерів та важливі аспекти взаємодії клієнта і сервера через цей протокол.

3.7.1 Основні характеристики HTTP

- HTTP використовується для передачі гіпертекстової інформації і базується на клієнт-серверній моделі.
- Протокол визначає способи і формати обміну даними між клієнтом та сервером.
- Однією з ключових особливостей є те, що HTTP є безстанційним, що означає, що кожен запит трактується незалежно від інших.

3.7.2 Методи HTTP запитів

HTTP використовує різні методи запитів, такі як GET, POST, PUT, DELETE та інші. У месенджерах, наприклад, метод GET може використовуватися для отримання повідомлень, а метод POST - для відправлення нового повідомлення на сервер.

3.7.3 Статус-коди HTTP

Статус-коди вказують на результат виконання HTTP-запиту. У месенджерах вони можуть вказувати на успішну доставку повідомлення (наприклад, статус 200 OK) або повідомляти про помилку (наприклад, статус 404 Not Found).

Одні з найчастіше використовуваних статус-кодів:

- 200 OK. Запит був успішно оброблений. Це основний статус-код, який вказує на успішне завершення запиту.
- 201 Created. Ресурс був успішно створений відповідно до запиту.
- 204 No Content. Запит був успішно оброблений, але відсутня інформація для відправлення в тілі відповіді.
- 400 Bad Request. Сервер не може обробити запит через помилку в запиті клієнта.
- 401 Unauthorized. Клієнт не має дозволу доступу до ресурсу або потребує аутентифікації.
- 403 Forbidden. Клієнт має аутентифікацію, але не має дозволу доступу до ресурсу.

- 404 Not Found. Ресурс не знайдено на сервері.
- 500 Internal Server Error. Загальна помилка сервера, яка не була передбачена.
- 502 Bad Gateway. Сервер, який виконує роль шлюзу або проксі, отримав неприпустимий відповідь від вищестоячого сервера.
- 503 Service Unavailable. Сервер тимчасово не готовий обробляти запити. Це може бути через перевантаження або тимчасові проблеми з обслуговуванням.
- 504 Gateway Timeout. Сервер, який виконує роль шлюзу або проксі, не отримав вчасно відповідь від сервера.

Ці статус-коди допомагають взаємодіяти між клієнтом і сервером, повідомляючи про різні аспекти результату виконання HTTP-запиту.

3.7.4 Заголовки HTTP

Заголовки додають метадані до HTTP-запиту або відповіді. Вони можуть містити інформацію про тип вмісту, дату відправлення, кешування тощо. Використання правильних заголовків може покращити ефективність і безпеку взаємодії.

3.7.5 Сесії та куки

HTTP є безстанційним протоколом, але для збереження стану між запитами використовуються сесії та куки. У контексті месенджера це може бути використано для зберігання даних автентифікації та інших параметрів сесії.

3.7.6 Захист від атак

Однією з важливих аспектів використання HTTP є захист від різних видів атак, таких як перехоплення даних, атаки типу "man-in-the-middle" та інші. Розділ розглядає методи і засоби забезпечення безпеки взаємодії клієнта і сервера.

3.8 Зв'язок між клієнтом і сервером на основі GraphQL

В сучасних месенджерах, де швидкість передачі даних та ефективність комунікації важливі для забезпечення високоякісного користувацького досвіду, використання GraphQL стає актуальним засобом для забезпечення ефективного зв'язку між клієнтом та сервером. У цьому розділі розглядається впровадження GraphQL в архітектурі месенджера та його вплив на спілкування між клієнтом і сервером.

3.8.1 Основні принципи GraphQL

GraphQL - це мова запитів, яка дозволяє клієнтам отримувати лише ті дані, які їм потрібні, і забезпечує гнучкий та ефективний спосіб отримання інформації з сервера. Порівняно з традиційним REST API, GraphQL дозволяє здійснювати запити, вказуючи точно ті дані, які потрібно отримати, що зменшує обсяг передачі даних та покращує продуктивність.

3.8.2 Структура запитів та відповідей GraphQL

В месенджерах на основі GraphQL, взаємодія між клієнтом і сервером відбувається за допомогою структурованих запитів та відповідей. Клієнти формують запити, описуючи структуру даних, які їм потрібні, і сервер відповідає зазначеними даними у форматі, який відповідає структурі запиту.

3.8.3 Переваги використання GraphQL в месенджерах

Впровадження GraphQL в архітектурі месенджера має кілька вагомих переваг. Однією з головних переваг є зменшення кількості запитів, які клієнт повинен робити для отримання потрібної інформації, що сприяє зниженню трафіку і покращенню продуктивності месенджера.

3.8.4 Реалізація механізму підписки (Subscriptions) в GraphQL

Додатково, у месенджері на основі GraphQL може бути використаний механізм підписок для отримання реального часу сповіщень про події. Це дозволяє клієнтам отримувати оновлення без необхідності постійного опитування сервера, що робить взаємодію більш ефективною та реактивною.

3.8.5 Виклики до сервера та оптимізація ресурсів

Однією з ключових аспектів використання GraphQL є можливість групування декількох операцій в один запит, що дозволяє економити ресурси сервера та зменшувати кількість мережевого трафіку.

3.9 Запуск застосунку використовуючи docker і docker compose

Docker — це платформа для розробки, доставки та запуску застосунків в контейнерах. Docker Compose — інструмент для визначення та запуску багатоконтейнерних Docker застосунків. Використання Docker і Docker Compose спрощує розгортання та управління інфраструктурою.

В даному випадку нам потрібні контейнери для:

- клієнтської частини застосунку
- серверної частини застосунку
- бази даних.

В такому випадку `docker-compose.yml` файл буде мати такий вигляд:

```
version: '3'
services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile
```

```
ports:
  - "4000:4000"
depends_on:
  - db
env_file:
  - ./backend/.env
environment:
  POSTGRES_HOST: db
  POSTGRES_PORT: 5432
  POSTGRES_DB: myappdb
  POSTGRES_USER: postgres
  POSTGRES_PASSWORD: password
  JWT_SECRET: qweqweqwe
  PORT: 4000
networks:
  - myapp-network
profiles:
  - prod
frontend:
  build:
    context: ./frontend
    dockerfile: Dockerfile
  ports:
    - "80:3000"
  depends_on:
    - backend
  environment:
    REACT_APP_GRAPHQL_HOST: http://localhost:4000/graphql
  networks:
    - myapp-network
  profiles:
    - prod
db:
  image: postgres:16
  ports:
    - "5432:5432"
  environment:
    POSTGRES_DB: myappdb
    POSTGRES_USER: postgres
    POSTGRES_PASSWORD: password
  networks:
    - myapp-network
  profiles:
    - db
    - prod
networks:
```

myapp-network:
driver: bridge

Висновки

У цьому розділі дипломної роботи було представлено огляд застосунку, який реалізує асиметричне шифрування. Детально розглянуті основні функції додатку, включаючи сторінку реєстрації, сторінку логіну та головну сторінку додатку.

Спеціально виокремлено процес завантаження приватного ключа в програму, що є важливим етапом для забезпечення безпеки комунікації. Також описано створення нового чату та відправку повідомлень користувачеві. Ці елементи додатку ілюструють практичну реалізацію асиметричного шифрування в реальному середовищі.

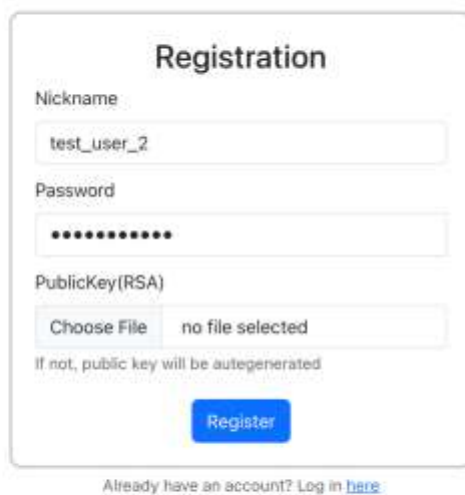
Аналіз функціоналу додатку вказує на його здатність взаємодіяти з асиметричними алгоритмами, а також на зручність та ефективність використання для кінцевого користувача. Подана інформація є важливою для оцінки практичної вартості розробленого застосунку з точки зору застосування асиметричного шифрування у реальних умовах.

Отже, результати огляду застосунку підтверджують можливість успішної реалізації асиметричного шифрування в контексті вказаного додатку, що робить його важливим інструментом для забезпечення конфіденційності та безпеки комунікаційних процесів.

4. ОГЛЯД ЗАСТОСУНКУ, ЯКИЙ РЕАЛІЗОВУЄ АСИМЕТРИЧНЕ ШИФРУВАННЯ

4.1 Сторінка реєстрації

По шляху /register була реалізована можливість додати користувача



The image shows a web form titled "Registration". It contains three input fields: "Nickname" with the value "test_user_2", "Password" with masked characters, and "PublicKey(RSA)" with a file selection interface showing "Choose File" and "no file selected". Below the RSA field is a note: "If not, public key will be autogenerated". A blue "Register" button is at the bottom. At the very bottom, there is a link: "Already have an account? Log in [here](#)".

Рисунок 4.1 - Сторінка реєстрації

Для додавання користувача потрібно створити логін і пароль. Додатково можна завантажити свій власний публічний RSA ключ, якщо ключ був згенерований раніше. В випадку відсутності публічного ключа, він буде згенерований і завантажений в базу даних. При натисканні на кнопку на сервері створюється користувач і починається завантаження приватного ключа на ваш комп'ютер. Приватний ключ зберігається тільки на вашому комп'ютері.

4.2 Сторінка логіну

За шляхом /login ми можемо увійти в свій обліковий запис, ввівши логін та пароль, заданий раніше



Рисунок 4.2 - Сторінка логіну

4.3 Головна сторінка додатку

На головній сторінці додатку є поля для:

- завантаження приватного ключа для розшифрування вхідних повідомлень
- можливість створити новий чат. Для створення нового чату вам потрібно знати логін іншого аккаунту
- виходу з облікового запису
- головне поле для чату

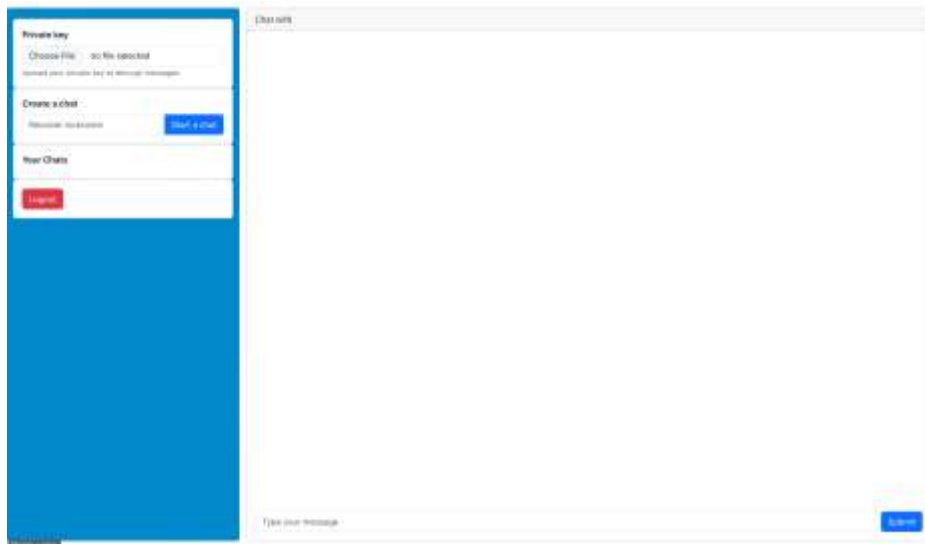


Рисунок 4.3 - Головна сторінка застосунку

4.3.1 Завантаження приватного ключа в програму

Для використання програми перш за все нам потрібно завантажити приватний ключ в програму. Це можна зробити в відповідному вікні зліва зверху.

При натисканні на кнопку у нас з'явиться вікно завантаження файлу:

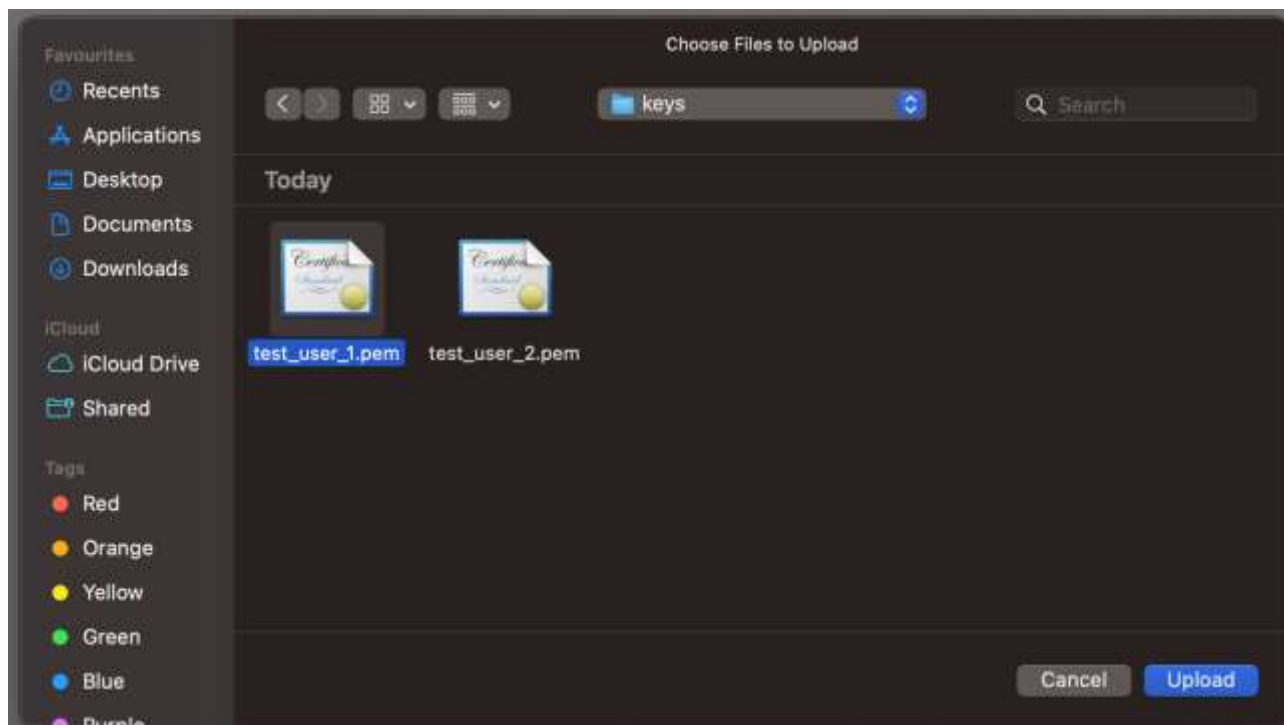


Рисунок 4.4 - Вивантаження ключа в додаток

Вибираємо потрібний файл і завантажуюємо його в програму

4.3.2 Створення нового чату

Для створення нового чату потрібно ввести логін облікового запису користувача в відповідній формі. Після натискання на кнопку у нас з'явиться чат з потрібним користувачем



The image shows a web interface for creating a chat. At the top, there is a section titled "Create a chat". Inside this section, there is a text input field with the value "test_user_1" and a blue button labeled "Start a chat". Below this section, there is another section titled "Your Chats". Inside this section, there is a list item that says "Chat with test_user_1".

Рисунок 4.5 - Форма створення чату

4.3.3 Відправка повідомлення користувачеві

Для надсилання повідомлення потрібно ввести текст повідомлення в відповідному полі чату.

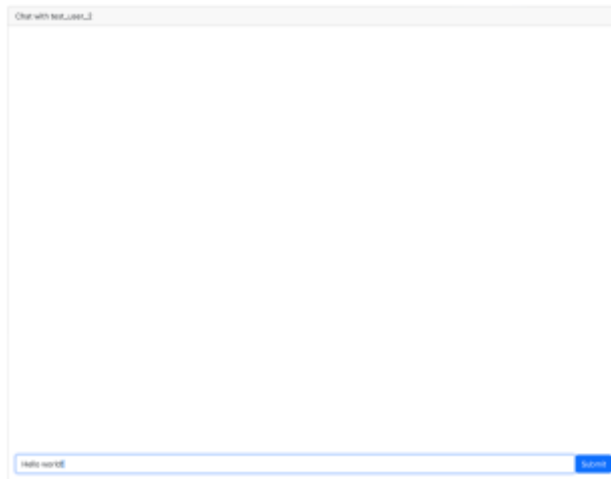


Рисунок 4.6 - Сторінка листування

Після відправлення повідомлення ми побачимо зашифроване повідомлення в самому чаті

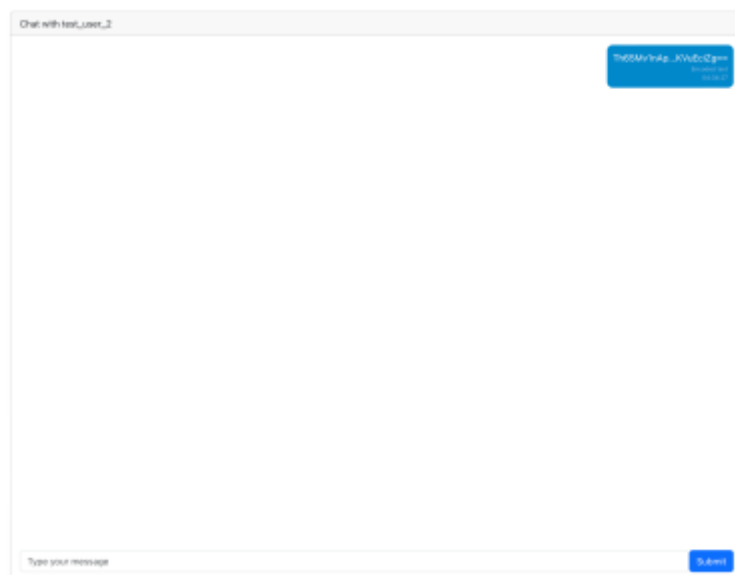


Рисунок 4.7 - Сторінка листування з надісланим повідомленням

Оскільки тільки отримувач має фізичну можливість розшифрувати повідомлення, то навіть відправник після відправки повідомлення не має можливості його прочитати після надсилання.

Після надсилання повідомлення, воно з'явилося в чаті отримувача в розшифрованому варіанті



Рисунок 4.8 - Сторінка листування зі сторони отримувача

Висновки для четвертого розділу

В цьому розділі ми розглянули застосунок, який реалізовує в собі надсилання повідомлень, зашифрованих асиметричними ключами. Особливість цього шифрування полягає в тому, що тільки отримувач має змогу розшифрувати отримане повідомлення, оскільки приватний ключ не зберігається ніде, крім його комп'ютера і, відповідно, ніхто не має можливості скористатися ним.

ВИСНОВКИ

Протягом написання дипломної роботи розглянуто актуальну проблему забезпечення безпеки передачі інформації в комп'ютерних мережах, використовуючи асиметричні алгоритми шифрування. Застосування асиметричних методів шифрування в сучасних умовах набуває особливої важливості через поширеність комп'ютерних технологій та зростання кількості загроз кібербезпеці.

У роботі детально розглянуто принципи роботи асиметричних алгоритмів шифрування, таких як RSA, їхня стійкість до атак, а також практичні аспекти їх впровадження в комп'ютерних мережах. Особлива увага приділена обміну ключами, який є важливим етапом в процесі взаємодії сторін.

Досліджено можливості застосування асиметричного шифрування для захисту конфіденційності та цілісності інформації в різних типах мереж, включаючи Інтернет, корпоративні мережі та хмарні середовища. Виявлено переваги та недоліки застосування асиметричного шифрування в залежності від контексту його використання.

Результати експериментальних досліджень та аналізу свідчать про ефективність асиметричних алгоритмів шифрування у забезпеченні безпеки інформаційного обміну. Однак важливо враховувати витрати на обчислювальні ресурси при виборі конкретного алгоритму та його параметрів.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

Наукові статті та журнали

1. Diffie, W., & Hellman, M. E. (1976). New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6), 644–654.
2. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2), 120–126.
3. Boneh, D., & Shoup, V. (2000). A Graduate Course in Applied Cryptography.
4. Elliptic Curve Cryptography (ECC): A Survey. (2005). *International Journal of Information Security*, 4(3), 213–238.

Книги

5. Stinson, D. R. (2005). *Cryptography: Theory and Practice*. CRC Press.
6. Schneier, B. (1996). *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. Wiley.
7. Paar, C., & Pelzl, J. (2009). *Understanding Cryptography: A Textbook for Students and Practitioners*. Springer.
8. Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). *Handbook of Applied Cryptography*. CRC Press.

Наукові конференції та збірники

9. *Advances in Cryptology – EUROCRYPT*.

10. International Conference on the Theory and Application of Cryptographic Techniques – CRYPTO.
11. Annual International Cryptology Conference – ASIACRYPT.
12. IEEE Symposium on Security and Privacy.

Офіційні документи та стандарти

13. FIPS PUB 197. (2001). Advanced Encryption Standard (AES).
14. RFC 4880. (2007). OpenPGP Message Format.
15. RFC 5246. (2008). The Transport Layer Security (TLS) Protocol Version 1.2.

Інтернет-ресурси та віртуальні бібліотеки

16. National Institute of Standards and Technology (NIST) - Computer Security Resource Center.
17. Electronic Frontier Foundation (EFF) - Surveillance Self-Defense.
18. Cryptography Stack Exchange.
19. IACR ePrint Archive.

Дослідження та реалізації алгоритмів

20. RSA Laboratories.
21. OpenSSL Project.
22. Microsoft Research – Cryptography.
23. Elliptic Curve Cryptography: Mathematics and Implementation.

Освітні платформи та відео-курси

24. Coursera – Cryptography I (by Stanford University).
25. edX – Introduction to Cryptography (by Microsoft).
26. Khan Academy – Cryptography.
27. YouTube – Numberphile: RSA Encryption.

Статті та ресурси про застосування в комп'ютерних мережах

28. Diffie-Hellman Key Exchange – Explanation and Implementation.
29. Digital Signatures in Networking.
30. Practical Use Cases of Asymmetric Cryptography in Network Security.

Аналіз та заходи захисту

31. Koblitz, N. (1998). Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177), 203–209.
32. Boneh, D., & Shacham, H. (2008). Group signatures with verifier-local revocation. *ACM Transactions on Information and System Security (TISSEC)*, 9(3), 225–244.
33. Quantum Computers and Cryptography: Challenges and Solutions.

Статті про проблеми та вразливості

34. Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. *Advances in Cryptology—CRYPTO '98*, 1–12.
35. Menezes, A. J., & Vanstone, S. A. (1997). Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 43(1), 224–226.

Статті про використання в протоколах та системах

36. Rivest, R. L., & Shamir, A. (1987). How to Expose an Eavesdropper. *Communications of the ACM*, 30(5), 416–422.
37. Rivest, R. L., & Shamir, A. (1985). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. MIT/LCS/TM-176.

Документи та ресурси про стандарти інтернету

38. IETF RFC 5246 – The Transport Layer Security (TLS) Protocol Version 1.2.
39. IETF RFC 8446 – The Transport Layer Security (TLS) Protocol Version 1.3.
40. IETF RFC 4880 – OpenPGP Message Format.

Дослідження проблем збереження ключів

41. Canetti, R., & Dakdouk, G. (2005). A Summation Scheme Secure Against Adaptive Chosen-Ciphertext Attack. *Advances in Cryptology - CRYPTO 2005*, 403–419.
42. Bellare, M., & Namprempre, C. (2000). Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. *Advances in Cryptology - EUROCRYPT 2000*, 531–545.

Статті про сучасні тенденції

43. Post-Quantum Cryptography: NIST's Ongoing Standardization Efforts.

44. Homomorphic Encryption: A Survey.

Розробка та впровадження алгоритмів

45. Key Management in Cryptography – Best Practices.

46. Integrating Asymmetric Cryptography into Network Security Protocols.

47. Implementation Challenges and Solutions in Asymmetric Cryptography.

Актуальні проекти та новітні розробки

48. Microsoft SEAL – A Homomorphic Encryption Library.

ДОДАТОК А ТЕЗИ

УДК 004.414

Глеб Владислав Юрійович, здобувач вищої освіти
КПІ ім. Ігоря Сікорського, Україна

Науковий керівник: Тарасенко-Клятченко Оксана Володимирівна, кандидат технічних наук,
доцент кафедри системного програмування та спеціалізованих комп'ютерних систем
КПІ ім. Ігоря Сікорського, Україна

Передача інформації, захищеної за допомогою асиметричних алгоритмів шифрування(RSA та ін.)

Анотація. Доповідь присвячена розгляду застосування асиметричних алгоритмів шифрування, зокрема RSA, для передачі інформації в безпечних комунікаційних системах. У доповіді буде проаналізовано принципи роботи асиметричних шифрів, їхню основну концепцію та математичні засади. Особлива увага буде приділена практичним застосуванням RSA для забезпечення конфіденційності та цілісності інформації під час її передачі через відкриті мережі. Доповідь також розгляне можливі ризики та виклики, пов'язані з використанням асиметричних алгоритмів, та запропонує рекомендації щодо забезпечення ефективного та безпечного використання цих методів шифрування в різних сферах інформаційної безпеки.

Ключові слова. Асиметричне шифрування, шифрування, асиметричні алгоритми шифрування

Abstract. The presentation is dedicated to the exploration of the application of asymmetric encryption algorithms, particularly RSA, for secure information transmission in communication systems. The presentation will analyze the working principles of asymmetric ciphers, their core concepts, and mathematical foundations. Special attention will be given to the practical implementations of RSA to ensure the confidentiality and integrity of information during transmission over open networks. The presentation will also examine potential risks and challenges associated with the use of asymmetric algorithms, offering recommendations for ensuring effective and secure utilization of these encryption methods in various realms of information security.

Key words: Asymmetric encryption, encryption, asymmetric encryption algorithms

Вступ. Захист конфіденційної інформації є найважливішою задачею в сучасному цифровому світі. Асиметричні алгоритми шифрування, такі як RSA (Rivest-Shamir-Adleman), надають високий рівень безпеки у передачі інформації через мережі. У даному рефераті розглянемо застосування цих алгоритмів для захисту передачі конфіденційної інформації.

Основна частина. Асиметричні алгоритми шифрування використовують пари ключів: публічний і приватний. Публічний ключ використовується для шифрування інформації, а приватний — для розшифрування. Така схема гарантує високий рівень безпеки, оскільки навіть володар публічного ключа не може розшифрувати дані без відповідного приватного ключа. Найпопулярніші алгоритми асиметричного шифрування включають RSA (Rivest–Shamir–Adleman), ECC (Elliptic Curve Cryptography), та DSA (Digital Signature Algorithm).

RSA (Rivest–Shamir–Adleman):

- **Опис:** Розроблений в 1977 році, RSA використовує математичні властивості великих простих чисел для генерації пари ключів. Публічний ключ використовується для шифрування, а приватний – для розшифрування.
- **Використання:** Шифрування та цифровий підпис. Застосовується в HTTPS для захищеного обміну даними в інтернеті, електронних підписах, електронній пошті.

ECC (Elliptic Curve Cryptography):

- **Опис:** Використовує математичні властивості еліптичних кривих для генерації ключів. Забезпечує високий рівень безпеки при меншому обсязі ключів порівняно з традиційними алгоритмами.
- **Використання:** Шифрування, цифровий підпис, генерація ключів для інших алгоритмів шифрування. Застосовується в багатьох сучасних криптографічних протоколах.

DSA (Digital Signature Algorithm):

- **Опис:** Розроблений для генерації цифрових підписів. Використовується у комбінації з іншими алгоритмами для забезпечення аутентифікації та цифрового підпису.
- **Використання:** Головним чином використовується для цифрового підпису, особливо в протоколах інтернет-послуг та електронної комерції.

Кожен з цих алгоритмів володіє своїми унікальними перевагами і використовується у різних сценаріях, забезпечуючи безпеку комунікації в цифровому середовищі. Їхнє використання допомагає забезпечити конфіденційність, цілісність та аутентифікацію даних в онлайн-середовищі.

Висновки. За останні десятиліття потужність комп'ютерів збільшилась в тисячі разів. Сьогодні в кожної людини в кармані є комп'ютер який ще кілька десятиліть назад міг займати цілу кімнату. В зв'язку з цим багато алгоритмів можуть бути зламані звичайним перебором. На протидію цим потужностям були винайдені асиметричні алгоритми шифрування. На даний момент ці алгоритми використовуються в генерації ключів доступу для держустанов, https протокол використовує асиметричні алгоритми для передачі даних в браузер і з браузера. Ці алгоритми мають високі ступені захисту, що дозволяє їх використання в будь яких програмах, для яких безпека є одним з основних критеріїв.

Список інформаційних джерел

Електронні ресурси	1. Rivest R., Shamir A., Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. 1978. URL:https://web.williams.edu/Mathematics/lg5/302/RSA.pdf 2. Hankerson D., Menezes A., Vanstone S. Guide to Elliptic Curve Cryptography. Springer, 2004. URL:http://tomlr.free.fr/Math%E9matiques/Math%20Complete/Cryptography/Guide%20to%20Elliptic%20Curve%20Cryptography%20-%20D.%20Hankerson,%20A.%20Menezes,%20S.%20Vanstone.pdf 3. Stallings W. Cryptography and Network Security: Principles and Practice. 6th ed. Pearson, 2016. URL:https://uomustansiriyah.edu.iq/media/lectures/6/6_2017_03_17!10_56_57_PM.pdf 4. Antoniewicz B. Elliptic Curve Cryptography Explained. 2018. URL:https://hal.science/hal-01914807/document
---------------------------	--

ДОДАТОК Б

ФРАГМЕНТИ КОДУ ЗІ СТОРОНИ КЛІЄНТА

Реєстрація і створення ключів зі сторони клієнта

```
const submit = async (payload: any) => {  
  const keyPair = forge.pki.rsa.generateKeyPair({ bits: 2048 });  
  
  const privateKey = forge.pki.privateKeyToPem(keyPair.privateKey);  
  
  let publicKey = forge.pki.publicKeyToPem(keyPair.publicKey);  
  
  if (fileData) {  
    publicKey = fileData;  
  }  
  
  if (fileData?.includes('ssh-rsa')) {  
    publicKey = fileData.split(' ')[1];  
  }  
  
  try {  
    const { data } = await register({ variables: { ...payload, public_key: publicKey }  
  });  
  if (data?.register.access_token) {  
    localStorage.setItem('access_token', data?.register.access_token);  
  }  
}
```

```

    }

    if (privateKey) {
        downloadBinaryStringAsFile(privateKey, 'privateKey.pem');
    }
} catch (e: any) {
    setResponseError(JSON.parse(e?.message).message.message);
}
};

```

ДОДАТОК В

ФРАГМЕНТИ КОДУ ЗІ СТОРОНИ СЕРВЕРУ

Реєстрація зі сторони серверу

```

@CommandHandler(RegisterAction)
export class RegisterHandler implements ICommandHandler<RegisterAction> {
    constructor(private readonly userService: UsersService, private readonly
authService: AuthService) {}

    async execute(userInfo: RegisterAction): Promise<RegisterResponse> {
        const hashedPassword = await this.authService.hashPassword(userInfo.password);

        const duplicated = await this.userService.find({ where: { nickname:
userInfo.nickname } });

        if (duplicated) {
            throw ErrorHandler(HttpStatus.BAD_REQUEST,
ERROR_MESSAGES.ALREADY_EXISTS("This User"));
        }

        const refreshToken = this.authService.createRefreshToken();

        const user = await this.userService.createUser({
            ...userInfo,

```

```

    password: hashedPassword,
    refresh_token: refreshToken,
    token_valid_from: new Date(new Date().getTime() - 1),
  });

  const token = await this.authService.sign({ user_id: user.user_id });

  return { access_token: token, refresh_token: refreshToken };
}
}

```

Авторизація зі сторони серверу

```

@CommandHandler(LoginAction)
export class LoginHandler implements ICommandHandler<LoginAction> {
  constructor(private readonly userService: UsersService, private readonly
    authService: AuthService) {}

  async execute(userInfo: LoginAction): Promise<LoginResponse> {
    const user = await this.userService.find({
      where: { nickname: userInfo.nickname },
    });
    const isValid = this.authService.comparePasswords(userInfo.password,
    user.password);
    if (!isValid) {
      throw ErrorHandler(HttpStatus.FORBIDDEN,
    ERROR_MESSAGES.WRONG_PASSWORD);
    }
    const { access_token, refresh_token } = await
    this.authService.generateTokens(user.user_id);
    await this.userService.update({ user_id: user.user_id, refresh_token });
    return { access_token, refresh_token };
  }
}

```

ДОДАТОК Г

Вигляд завантаженого приватного ключа

-----BEGIN RSA PRIVATE KEY-----

MIIEowIBAAKCAQEAmMhm8K6nvWk4OyURE4l7+7so/yJq3LrWRDQRgP6wXf
cdG0Zp
6udv1yELKnoole8UGFOSEnBmOydmYEG1uiRXDpqM2ii0ZrCoq1zTlvkT0r8E4i+
8
KREDl/hIB3TPbenOHs95o0mb3FI35EK8aOjfVy+GHZJTN1GlKKSbTdt+Jk02wG
Am
DqTSBssXUkrGpSGmtstUQS04WlkgSl8DmBhW/fO/MSdYH/yQQsNiA4bvm5cV
Kin0
VDcGKv+yHkGoRlcKgj4mUWp/R4tFOOwhuXDxNEy2hni+5vACuaJjDhfs6Faoi1e
7
SXH6IQ67hPcTdFEG1CY83/ztQSeE2oC2YExetwIDAQABAoIBAQCWTpoPcRRv
IMQE
U+wfl2NVfYTQtDCDpYBEqyEJUgd6+jD1c2migzwhXk3Y5UNmdkXDxPvxDglV
auaq
DhtV3liVQnEN83u/h074/6JsRrU84VZRm/q2QBnT+ZwLUdz8WkfbQgZ+IV2n/G8
m
Lo2jfG61HHkKQLmDs9wTRxXPHABYdpq+y5VmLvbFlb3vJ8Oz373j75srsbRVP
HIZ
gIZl9AAWUNMVBVYFQkgeZpn5CilTrlJksk9dgyR+BGHKBbDCuCjXLqgjawp81
pUi

NenjQHUmPpocA5MmR8b5MmLj2bqFHoDn8DhuJnnBpeZ832WnSRplYSDmPzk1
/PIO
P1YkVMFJAoGBAM7e+k+1MTcYKH7AH+fFZGGGz9EyS0/ALM4IqGHe1qB+7r
bXgcwq
rh+gZcvr9ZUa/3xJiLNZfzZnw9hKJne+1kKKdEHRq1vuzUvIXCkUXcn9gyW0XH
W2
ANqDxbhbBDWW7IZj1fBc0pD5RmKkssZKlRGzF7tscb2aGEj9TakN4bCTAoGBA
L0R
DgY2DIO3UdPBkJyvyRRd95m9lx2X2u+d80k06tRdfy824AQ/QJvvJBBQexjk1Wb
0
tcX14kSQKEnyYS3R3QzRzfgSzGt82pH7/s7p9EFA+5G1K6i+GEZe41dGFFqXQZI
D
rCYaJChsX/koywDqCTOWb3YBvGIX5sENAMiX5sPNAoGACRWlrGZqmN7luH
g7bnQu
qlG/QpKQ2Bm6XfnnWxTsG/Q3C3R/ULX3G5Bxp/jeugOLgKvCAw5FFeCMqyzm
b4/D
3PwzishoXAsqnM9ZrLwmOj6VycwX21t4wh/TSPtGPgw4qtYriPiqjo9D1gUhjK+t
yC8zNZnXjNn9gwcMqAXiCV0CgYB8bM7n84YE4Q4u58I9+8m/Z5/dQ9JBnYBuF
WrY
kMTCgc2Dyv9ejCn+9UCso+p/FYoDuaQu5YTFToHVOHgP6X8qQqDkU6rISmCll
ku4
7GrczuFAoRRcds4O8b933hzjCYwTheszIB0PpVL77Ia80UjA9I3iAX5yp/wqrF4J
pT4vOQKBgHKrIDs2YkhIQqqwVCIqrf30ABHwqC6s4Wss9wvx+Qh+XaO5h9ns1
UAS
/Gw1H7KgEm9z/OmhutuhChZ8H1dYelJkNeWJcrc4g1c2ZNw+hnmyw0Vi7Pf4p9b
8
1lHGBPQfRBIAdAyaECA11jDVjOAcDQ5ovlXK3eBu9UwhLDp08ixr
-----END RSA PRIVATE KEY-----