МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук
«_____»_____ 20__ р.

На правах рукопису

ДИПЛОМНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Клієнт-серверний веб-додаток для обміну зашифрованими повідомленнями

Виконавець: М.Р. Вдовиченко
Керівник: д.т.н., професор С.В. Толюпа
Нормоконтролер: д.т.н., професор С.В. Толюпа

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Вдовиченко Миколи Романовича

1. Тема: *Програмний модуль захисту електронної пошти від небажаних надсилань* затверджена наказом в.о. ректора від «26» квітня 2021 р. № 652/ст.
2. Термін виконання: з 10.05.2021 р. по 20.06.2021 р.
3. Вихідні дані: проаналізувати існуючі системи обміну зашифрованими повідомленнями; розробити програмний модуль для обміну зашифрованими повідомленнями.
4. Зміст пояснювальної записки: аналіз предметної області; аналіз вимог до системи обміну зашифрованими повідомленнями, проєктування та розробка веб-додатку.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1.	Уточнення постановки задачі	03.02.2021	*Виконано*
2.	Аналіз літературних джерел	05.02.2021 – 10.03.2021	*Виконано*
3.	Обґрунтування вибору рішення	11.03.2021	*Виконано*
4.	Збір інформації	12.03.2021 – 14.04.2021	*Виконано*
5.	Дослідження програмних методів, патернів та протоколів для реалізації системи обміну зашифрованими повідомленнями	15.04.2021 – 25.04.2021	*Виконано*
6.	Розробка методики, алгоритму та програмного забезпечення обміну зашифрованими повідомленнями	26.04.2021 – 20.05.2021	*Виконано*
7.	Оформлення і друк пояснювальної записки	21.05.2021 – 31.05.2021	*Виконано*
8.	Оформлення презентації	01.06.2021	*Виконано*
9.	Перевірка на антиплагіат	04.06.2021	*Виконано*
10.	Отримання рецензій від рецензента	08.06.2021	*Виконано*
11.	Підготовка до захисту	16.06.2021	*Виконано*

Здобувач вищої освіти _____ М.Р. Вдовиченко
(підпис, дата)

Керівник дипломної роботи _____ С.В. Толюпа
(підпис, дата)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи «Клієнт-серверний веб-додаток для обміну зашифрованими повідомленнями» складається зі вступу, переліку прийнятих скорочень, двох розділів, загальних висновків, списку використаних джерел, додатків і має 45 сторінок, 16 рисунків, 3 таблиці. Список використаних джерел містить 8 найменуваннь і займає 1 сторінку. Загальний обсяг роботи 45 сторінок.

Метою дипломної роботи є створення програмної реалізації системи обміну зашифрованими повідомленнями.

В роботі проаналізувано існуючі загрози при спілкуванні двох користувачів в інтернеті та методи запобігання викрадення інформаціїї.

В роботі розроблено систему обміну зашифрованими повідомленнями на сервлетній технології, що дало змогу безпечно обмінюватися інформацією між користувачами.

Розроблене програмне забезпечення відноситься до галузі інформаційної безпеки і може бути використане для секретного спілкування в інтернеті.

Ключові слова: шифрування, повідомлення, система, сервлет, фільтр.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ЧБ	Чат бот
СМОП	Система миттєвого обміну повідомленнями
API	Application Programing Interface
CI	Continuous Integration
ППІ	Прикладний програмний інтерфейс
DNSBL	Domain Name System-based blackhole list
DNS	Domain Name System
IP	Internet Protocol

ВСТУП

Що стосується обміну миттєвими повідомленнями, кожен має свої уподобання. Більшість людей використовують на своїх мобільних телефонах такі програми, як WhatsApp та Google Hangouts. Інші люди використовують Threema або Signal, щоб скористатися функціями підвищення приватності. Ще одна група людей користується Telegram, який для зручності продає частину своєї безпеки. На персональних комп'ютерах різноманітність включає деякі програми для мобільних телефонів, а також такі програми, як ICQ та IRC. Список програм обміну повідомленнями неймовірно довгий, а також список внутрішніх систем, які використовуються за лаштунками. Це означає, що зазвичай між користувачами різних додатків не може бути спілкування. Кожен повинен використовувати безліч програм обміну повідомленнями, щоб мати можливість спілкуватися з усіма своїми друзями, родиною та колегами. Той факт, що фактично не існує стандарту для обміну миттєвими повідомленнями, є досить дивовижним, враховуючи, що для довших повідомлень існує лише один помітний стандарт: електронна пошта. Це має ту перевагу, що кожен може використовувати програму, яка йому найбільше подобається, для надсилання та отримання електронної пошти, але все ще може обмінюватися повідомленнями з усіма іншими. Те саме стосується служби коротких повідомлень (SMS). Кожен мобільний телефон може обмінюватися SMS-повідомленнями з усіма іншими мобільними телефонами, незалежно від постачальника або програми, яка використовується для складання та читання повідомлень.

Незважаючи на те, що вони вважаються стандартними, ці системи мають недоліки, що робить їх поганим вибором для обміну миттєвими повідомленнями. Це правда, що різні програми обміну повідомленнями мають свій унікальний набір функцій, і кожен обирає собі бажану програму відповідно до того, яка з цих функцій є для них найбільш важливою. Але для користувачів

було б набагато зручніше, якби за всіма програмами обміну миттєвими повідомленнями існував єдиний протокол, щоб їх можна було використовувати як взаємозамінні. Далі

аналізуються найпопулярніші системи обміну повідомленнями, щоб виділити їх сильні та слабкі сторони. Сюди входять системи, які зазвичай не використовуються для обміну миттєвими повідомленнями, але мають риси, які можуть бути корисними в системі обміну миттєвими повідомленнями.

Потім, у розділі 2, встановлюється набір мінімальних вимог, які повинні виконувати рішення для обміну миттєвими повідомленнями, щоб зробити його життєздатним конкурентом у сучасному середовищі обміну повідомленнями. Крім того, встановлюється набір необов’язкових, які не є абсолютно необхідними, але можуть зробити протокол більш захищеним у порівнянні з іншими системами обміну повідомленнями та функції, які можуть бути втрачені користувачами. Після визначення мінімальних вимог пропонується протокол, який відповідає цим мінімальним вимогам і включає також багато необов’язкових функцій. Цей протокол використовує існуючі технології, коли це можливо, і максимально спрощений.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Роль криптографії при обміні зашифрованими повідомленнями

Знову і знову можна почути про кібератаки на великі компанії. Часто ці компанії мають великий бюджет на IT-безпеку і вкладають значні кошти у заходи безпеки. Але загальновідомо, що ланцюг настільки міцний, наскільки найслабша його ланка. Зловмиснику набагато простіше знайти найслабшу ланку, ніж компанії, яка захищає весь ланцюг. Американський банк JPMorgan Chase - нещодавня справа, яка викликала світову сенсацію - за повідомленням BBC News, злочинці змогли отримати дані про адресу 76 мільйонів приватних клієнтів та семи мільйонів бізнес-клієнтів. Ретроспективно задається питанням, чи не можна було цього уникнути. У випадку з JPMorgan зловмиснику вдалося отримати права адміністратора через недіючу двофакторну автентифікацію і, як наслідок, зміг скористатися різними даними користувача. Імовірно, дані були скопійовані безпосередньо з бази даних. Швидше за все веб-сервер та веб-додаток були добре захищені, але дані були доступні в незашифрованому вигляді в базі даних і їх можна було прочитати безпосередньо з неї за допомогою відповідних привілеїв. Ця проблема є досить поширеною, оскільки часто вважається законним зберігати дані в незашифрованому вигляді, коли базу даних не видно зовні, а вектор атаки, таким чином, вважається малим. Навряд чи шифрування бази даних запобігло б набігу хакера, навряд чи може відповісти сторонній. Але те, про що можна говорити, це наслідки, які може спричинити подальше обладнання з функціональними можливостями шифрування.

Рівень стійкості потрібно було б розширити для шифрування даних для програми, а в подальшому дані мали б бути зашифровані. Шифрування може також спричинити перепроектування схеми бази даних, оскільки зашифрований висновок - це двійкові дані. Складні зміни в схемі бази даних та простої під час шифрування, ймовірно, стримують компанії від вибору цього рішення. Натомість вони можуть двічі подумати про цей крок і шукати кращого шляху. Чи не можна мінімізувати вплив такого втручання? Існує маловідомий криптографічний додаток, яке, серед інших можливостей, може допомогти у цьому сценарії. Він отримав назву шифрування, що зберігає формат. Шифрування, що зберігає формат, як випливає з назви, дозволяє зберегти формат шифрування. Це означає, що зашифрований текст виглядає як оригінальний відкритий текст. У наведеному вище сценарії це мало б ту перевагу, що в схему бази даних потрібно внести майже ніяких змін і що шифрування можна було інтегрувати майже на льоту.

Шифрування збереження формату (ШЗФ) - це тема, яка належить до важливої галузі інформатики, IT-безпеки, а точніше до криптографічного підрозділу IT-security. У наступних параграфах дається короткий вступ до криптографії та пояснюэться кілька основних термінів у цій галузі, які важливі для розуміння методів ШЗФ. Як правило, криптографія описує науку, яка намагається захистити інформаційні системи від несанкціонованого читання та зміни. Криптографія будується разом із криптоаналізом, наукою про криптологію.

На відміну від криптографії, де інженери намагаються розробити безпечні алгоритми, наприклад для шифрування криптоаналіз - це наука, коли намагаються зламати такі алгоритми. Криптографія в основному використовувалася як синонім шифрування, але сьогодні вона охоплює набагато ширший спектр методів захисту. Відповідно до вікіпедії, чотири

основні цілі криптографії для захисту інформаційних цінностей можна визначити наступним чином.

Конфіденційність повідомлення (або конфіденційність) - тільки уповноважений одержувач повинен мати змогу витягати вміст повідомлення із зашифрованої форми . Це результат кроків, щоб приховати, зупинити або затримати вільний доступ до зашифрованої інформації. Цілісність повідомлення. Одержувач повинен мати можливість визначити, чи було повідомлення змінено.

Аутентифікація відправника. Одержувач повинен мати можливість підтвердити особу відправника, походження або шлях, який він пройшов (або комбінації) із повідомлення, щоб перевірити претензії емітента або перевірити очікування одержувача.

Відмова від відправника: випромінювач не повинен мати можливість заперечувати, що він надіслав повідомлення.

Шифрування збереження формату - це техніка, яка охоплює лише першу мету криптографії, конфіденційність. Останнє досягається за допомогою шифрування, що означає, що воно змінює дані таким чином, що лише люди, які володіють певним ключем, можуть розшифрувати та прочитати інформацію. Інші захисні механізми, як, наприклад, забезпечення цілісності даних, не є частиною ШЗФ і повинні вирішуватися окремою процедурою, яка не буде розглянута в бакалаврській роботі. Щоб мати змогу зрозуміти роботу та методи шифрування, що зберігають формат, розраховуємо мати базові знання симетричного шифрування , включаючи блокові шифри та режим їх роботи, а також знайомство з детермінованими шифрування.

Коли людей запитують про їх поведінку щодо криптографії в цілому, як, наприклад, якщо вони підписують електронні листи або зашифровують жорсткий диск, більшість із них, напевно, відповідають, що вони навіть не знають значення цих слів, або їх просто не хвилює їхні дані, оскільки їм нема

чого приховувати чи втрачати. Але таке ставлення зовсім не добре, адже сьогодні, як ніколи раніше, Інтернет-компанії, соціальні мережі, спецслужби або зловмисні хакери намагаються отримати всю особисту інформацію, яку вони можуть отримати, навіть від невинних і не підозрілих громадян. Згодом ці дані використовуються в рекламних цілях, для їх продажу третім особам або для створення детального профілю користувача. У діловому або фінансовому середовищі ризик скомпрометованих даних ще вищий через величезну грошову вигоду, яку можуть досягти зловмисники. Для захисту конфіденційності та персональних даних має

бути здоровим глуздом використовувати криптографію у повсякденних додатках, таких як Інтернет (SSL / TLS), eBanking, шифрування / підписання електронної пошти, технології VPN, шифрування файлів або комплектних жорстких дисків.

Шифрування збереження формату (ШЗФ) - одна із специфічних технологій досягнення шифрування. Коротше кажучи, шифрування збереження формату - це, як випливає з назви, - шифрування, де вихідні дані мають той самий формат, що і вхідні дані. З поширеними технологіями шифрування, як, наприклад, AES, вихід є нечитабельним бітовим рядком. Коли шифрується швейцарський номер телефону 032 321 61 11 за допомогою AES, отримується щось на зразок C2AC 2A8D 5910 53BE 4F80 FD24 1C53 47F6. Якщо використовується шифр ШЗФ, результат може бути приблизно таким, як 043 453 23 92, знову дійсний швейцарський номер телефону.

1.2. Шифрування збереження формату.

Для пояснення ШЗФ часто використовується приклад шифрування номера кредитної картки. Але чому формат шифрування є проблемою? Чи не могли б ми використовувати зашифрований номер кредитної картки як бітовий

рядок. Банку належить банкомат, і банкомат взаємодіє з банківським сервером. Зв'язок здійснюється через зовнішню мережу постачальника фінансових послуг, з якою також пов'язані інші банки. Тому зв'язок повинен бути у заданому форматі, а номер кредитної картки не повинен шифруватися, щоб забезпечити можливість маршрутизації.

Що може зробити банк, щоб захиститися від постачальника фінансових послуг? Тут може зіграти ШЗФ. За допомогою ШЗФ тепер можна зашифрувати номер кредитної картки, а зашифрований вихід - це знову номер кредитної картки. Тому постачальник фінансових послуг не може відрізнити зашифрований від незашифрованого. Але в цьому випадку шифрування цілого номера неможливе. Номер кредитної картки складається з шестизначного ідентифікаційного номера емітента, який ідентифікує організацію-емітент, індивідуального ідентифікатора рахунку із змінною довжиною до 12 цифр та контрольної суми, обчисленої за алгоритмом Люна. Коли шифрується ціле число, маршрутизація буде вже неможливою. Але банку не потрібно шифрувати її ідентифікаційний номер емітента, який і так є загальнодоступним. Він може зашифрувати лише її індивідуальний ідентифікатор рахунку в ШЗФ, додати до нього ідентифікаційний номер емітента і, нарешті, обчислити контрольну суму на ньому. Наприклад, число “371449 63539843 1” може привести до зашифрованого “371449 37194735 9”. Можно бачити, що номер кредитної картки дійсний і може бути направлений постачальником послуг фінансових послуг, але конфіденційна інформація недоступна.

В основному є дві причини, чому потрібно зберегти формат відкритого тексту. З одного боку, оскільки існує обмеження на систему, подібно до прикладу з номером кредитної картки, який було видно. У цьому випадку мета полягає в тому, щоб обдурити комп’ютерну програму, повіривши, що на даних не застосовується шифрування. З іншого боку, існують звичаї, мета яких полягає в тому, щоб обдурити людину, повіривши, що не існує шифрування

даних. У наступному неповному списку наведено компіляцію деяких програм, де шифрування збереження формату вже використовується або де це може бути серйозною альтернативою існуючим методам шифрування. Подальше шифрування даних у базі даних без необхідності зміни структур даних, відповідно типів даних атрибутів.

ШЗФ дозволяє шифрувати базу даних, яка повинна працювати цілодобово та без вихідних, оскільки для бази даних насправді нічого не змінюється у форматі або структурі. За допомогою шифрування змінюються лише дані. Єдине, що може змінитися для бази даних, - це додавання атрибута, який визначає, чи запис уже зашифрований чи ні. Але цей крок залежить від того, як реалізована функція ШЗФ. Спадкові системи, які повинні бути захищені, але на яких ніхто не хоче нічого змінювати. Наприклад, тому, що в компанії більше не знають про цю систему або вплив прямих змін на неї не ясний.

Зовнішні або сторонні програми, які очікують певного формату введення, наприклад, для адреси або номера телефону. Людина не хоче надавати реальні та, можливо, особисті дані додатку, але хоче мати можливість відтворювати реальну інформацію на основі фіктивних даних.

Для шифрування даних, щодо яких сторонні сторони, такі як хмарний провайдер, не повинні бачити, що вони насправді зашифровані. У цьому випадку подальше використання персональних даних може бути виключено. Тести додатків, де слід використовувати реальні дані в базі даних, щоб мати справжній сценарій тестування з правильним форматом, логічним значенням та розумними взаємозв’язками. Іноді неможливо використовувати продуктивні дані, оскільки вони є конфіденційними. За допомогою ШЗФ ці дані можуть бути зашифровані для отримання тестових даних.

Шифрування жорстких дисків, де метою є зашифрування жорсткого диска таким чином, щоб можна було підключити його до операційної системи,

не отримуючи помилок форматування. Жорсткий диск розділений на так звані дискові сектори, і для успішного прив'язки його до операційної системи ці сектори повинні бути розпізнані та пов'язані з відповідним номером сектору. Це означає, що має бути забезпечено шифрування, яке нічого не змінює на структурі секторів жорсткого диска і залишається інформацією заголовка, а також їх розмір, який традиційно становить 512 байт для звичайних жорстких дисків. Для більшості з цих областей застосування комерційних продуктів із застосуванням технологій ШЗФ вже існує.

Однією з можливостей класифікувати схеми ШЗФ є їх диференціація відповідно до розміру простору повідомлень. Простір повідомлень - це область, в якій лежить відкритий текст і зашифрований текст. Порядок простору повідомлень визначає кількість можливих простих відповідно шифртекстів. Залежно від кількості використаних бітів, ці схеми можна класифікувати на схеми крихітних просторів, малих просторів та великих просторів. Межа між схемами крихітного простору та малого простору не встановлена. Це може варіюватися залежно від програми та вимог. Також можна не розрізняти і оперувати лише схемами малих просторів. Перевага диференціації полягає в безпеці схеми малого простору, що можна довести математично. Невідома атака на схеми малого простору з крихітними доменами (наприклад, день тижня), але вони, як правило, мають гірші або відсутні доказові гарантії безпеки.

Перевагою недиференціації є зменшення складності та схильності до помилок, оскільки використовується менше криптографічних схем. Наступним моментом є ефективність схем крихітного простору, яка може бути меншою. Складність часу - O (N). Для шифрування / дешифрування необхідно пройти всі елементи простору повідомлень. Щоб компенсувати це, всі значення можна було попередньо обчислити і записати в таблицю, щоб їх не потрібно було обчислювати для кожного виклику шифрування. Проте межа між схемами

малого та великого космосу є фіксованою. Це визначається розміром блоку використовуваного блок-шифру. З найбільш часто використовуваним стандартом шифрування AES він становить 128 біт.

Шифрування рангу. Що розглядалося дотепер, це схеми ШЗФ, які дозволяють зашифрувати номер у заздалегідь визначеному діапазоні чисел з гарантією, що зашифрований номер також буде значенням у цьому діапазоні. Але ці схеми насправді мають багато обмежень. Наприклад, якщо використовується її як малий простір ШЗФ, можно шифрувати числа від нуля до довільної довжини, але не можна зашифрувати від’ємні числа або визначити діапазон чисел, який не починається з нуля або 128 біт. Іншими словами, неможливо охопити дуже прості випадки використання, наприклад, числом в діапазоні 500-1000 та шифруванням з метою отримати інше число в цьому діапазоні як зашифрований текст. Окрім цих обмежень, до цього часу говорилося лише про шифрування номерів за схемами ШЗФ. Але чи не було б цікаво мати можливість зашифрувати будь-який рядок у певному форматі та забезпечити, щоб зашифрований рядок також був у тому ж форматі? Цей тип питань турбував також багатьох дослідників комп'ютерної безпеки, і тому був розроблений новий підхід, який називається шифруванням рангу.

Основна ідея такого підходу полягає в тому, що не власне відкритий текст

шифрується, як і всі схеми, які ми вже бачили, але це так званий "ранг", який робить його чітко ідентифікованим у певному просторі повідомлень . Після цього

ранг, який є просто простим числом, може бути зашифрований з базовою схемою, такою як FFX або EME2. Використовуючи одну з цих схем та визначаючи максимальне значення простору повідомлень як найбільший можливий результат шифрування, ми можемо забезпечити, щоб зашифрований номер також був дійсним рангом у цьому просторі повідомлень. Після

шифрування отримується ще один ранг і, позбавивши цього рангу, отримується зашифрований текст, який є просто значенням, присвоєним зашифрованому рангу. Основна ідея цього такого підходу, очевидно, відносно проста. Важливою частиною цього підходу є вже згаданий простір повідомлень. Як і ключ, або твік, простір повідомлень завжди повинен бути однаковим для шифрування та дешифрування значення.

1.3. Існуючі системи обміну зашифрованими повідомленнями.

Перш ніж починати розробляти протокол обміну миттєвими повідомленнями, життєво важливо поглянути на різні системи обміну повідомленнями, які вже існують. У цьому розділі перелічено кілька найбільш широко використовуваних систем обміну повідомленнями. Усі ці системи розглядаються з метою висвітлення їх сильних і слабких сторін. Якщо вони присутні, обговорюються також інші цікаві характеристики. У листопаді 2014 року Electronic Frontier Foundation (EFF) створив систему показників безпечних повідомлень для порівняння систем обміну миттєвими повідомленнями. З того часу EFF вносив періодичні зміни, щоб підтримувати оновлену систему показників. Він складається з повного переліку майже 40 систем обміну миттєвими повідомленнями, які порівнюються згідно з набором із семи вимог.

Ці вимоги сформульовані за такими питаннями:

- чи зашифровано ваше спілкування під час транзиту? Чи зашифровано ваше спілкування ключем, до якого постачальник не має доступу?
- чи можете ви самостійно підтвердити особу свого кореспондента?
- чи захищені попередні зв’язки у разі викрадення ваших ключів?
- чи відкритий код для незалежного перегляду?
- чи добре задокументовано криптодизайн?
- чи проводився незалежний аудит безпеки?

Чим більше на ці запитання можна відповісти так для системи обміну миттєвими повідомленнями, тим краще. Зауважте, що остання вимога, незалежно від того, чи був проект захисту дійсно перевірений незалежною системою захисту, залежить від популярності системи обміну миттєвими повідомленнями, а не лише від самого проекту захисту. Якщо система популярна, то, швидше за все, її перевірять незалежні сторони. Таблиця показників безпечних повідомлень дає хороший і систематичний огляд різних систем обміну миттєвими повідомленнями. Однак вимоги сформульовані як дуже конкретні запитання, що стосуються приватного життя.

Потрібно також розглянути питання щодо розгортання та зручності використання. На ці запитання нелегко відповісти так чи ні, що робить пряме порівняння систем досить складним. Тому в цій дипломній роботі системи обміну повідомленнями розглядаються індивідуально, лише за кількома прямими порівняннями. Тим не менше, вимоги, які використовує карта показників, служать хорошою основою для оцінки, орієнтованої на конфіденційність. Зауважте, що багато систем обміну повідомленнями, про які йдеться в цій главі, використовуються не для обміну миттєвими повідомленнями, а для інших форм обміну повідомленнями. Деякі з них є навіть не повними системами обміну повідомленнями, а скоріше протоколами, які можна використовувати як розширення для інших систем обміну повідомленнями.

WhatsApp, є, мабуть, найпопулярнішою системою обміну миттєвими повідомленнями з базою користувачів понад 900 мільйонів активних користувачів щомісяця. Станом на квітень 2016 року WhatsApp використовує наскрізне шифрування (шифрування E2E) для всіх комунікацій, включаючи медіафайли та голосові дзвінки. З моменту цієї зміни WhatsApp набирає шість

із семи балів на балі показників безпечних повідомлень, бракуючи лише балу за закрите джерело. Хоча загалом добре, що WhatsApp використовує шифрування E2E, виникають деякі запитання, чому Facebook це дозволяє. Бізнес-модель Facebook покладається на показ своїх користувачів стрічкою новин, наповненою вмістом, який користувачі насправді хочуть бачити, з рекламою, яку Facebook хоче, щоб користувачі бачили. Подібний канал новин вимагає від Facebook знати багато про своїх користувачів та про те, як вони пов’язані з іншими користувачами.

Деякі люди стверджують, що Facebook дозволяє WhatsApp використовувати шифрування E2E, оскільки вони все ще можуть зібрати достатньо інформації про своїх користувачів, переглядаючи метадані.

Ще один важливий момент, про який слід пам’ятати, - це те, що самі повідомлення - не єдиний спосіб, яким WhatsApp може збирати інформацію про своїх користувачів. Наприклад, знайомство з друзями вимагає завантаження особистих контактів на сервери WhatsApp. Крім того, використання вбудованої функції резервного копіювання WhatsApp може завантажити базу даних повідомлень на сторонні хмарні сервери. Хоча резервні копії зашифровані, вони стають більш доступними для зловмисників. Для того, щоб користуватися WhatsApp, користувачі повинні створити обліковий запис, прив'язаний до їх номера телефону.

Створення облікового запису в основному приховується від користувача, за винятком кроку підтвердження, де наданий номер телефону підтверджується SMS-повідомленням. Користувач не може мати кілька облікових записів WhatsApp, не маючи декількох телефонних номерів, але можна мати кілька телефонних номерів, пов’язаних з одним обліковим записом. Крім того, WhatsApp не можна одночасно використовувати на кількох пристроях з одним і тим самим обліковим записом. Незважаючи на те, що WhatsApp має веб-клієнт,

його не можна використовувати самостійно; його потрібно активувати, скануючи за допомогою коду швидкого реагування (QR-код). Під час використання веб-клієнта повідомлення продовжують надсилати та отримувати по телефону у фоновому режимі.

Facebook Messenger - ще один великий конкурент у галузі обміну миттєвими повідомленнями, який налічує близько 800 мільйонів. Це також закрите джерело і не має шифрування E2E. Його перевага полягає в тому, що він не пов’язаний з номером телефону, а скоріше з обліковим записом Facebook, тому немає необхідності активувати веб-клієнт, скануючи QR-код.

Hangouts Google, раніше відомий як Google Talk, - це програма обміну миттєвими повідомленнями, яка попередньо встановлюється на всіх телефонах Android. Оскільки більшість користувачів Android пов’язують свій обліковий запис Google, їх можна зв’язати через Hangouts. Однак деякі з цих користувачів можуть не використовувати його регулярно. Надійні дані про активну базу користувачів Google Hangouts недоступні.

Подібно до Facebook Messenger, Hangouts пов’язаний з обліковим записом Google, а не з номером телефону. Але він також не пропонує шифрування E2E (повідомлення шифруються лише під час передачі), а вихідний код є власницьким. Крім того, у Hangouts є деякі проблеми з юзабіліті. Не існує простого способу пересилання повідомлень і немає функції пошуку.

Hangouts доступний майже для всіх операційних систем тим чи іншим чином. На мобільних телефонах і планшетах можна встановлювати власні програми. Клієнт Hangouts для настільних комп’ютерів доступний як розширення Google Chrome (Google Chrome можна встановити на всіх основних платформах). Як варіант, існує веб-клієнт, де Hangouts можна використовувати в браузері без встановлення додаткового програмного забезпечення. Зверніть увагу, що функція "Без запису" Hangouts не пов'язана з протоколом OTR (Off-

the-Record), про який йдеться нижче, і вона не захищає повідомлення будь-яким додатковим способом, за винятком того, що повідомлення видаляються з пристрою після закриття розмови.

iMessage від Apple раніше був дуже популярний серед користувачів iPhone, оскільки дозволяв користувачам iPhone надсилати повідомлення іншим користувачам iPhone через мобільні дані замість SMS. Коли повідомлення надсилається комусь, хто не використовує iPhone, iMessage просто повертається до SMS. IMessage від Apple досі постачається з iPhone та іншими продуктами Apple, але його можна використовувати лише на продуктах Apple. Це, швидше за все, не зміниться, оскільки основними продуктами Apple є пристрої, а не програмне забезпечення.

Електронна пошта - одна з найдавніших та найбільш широко використовуваних форм електронних повідомлень. Хоча це не система обміну миттєвими повідомленнями, вона настільки популярна та універсальна, що на неї варто придивитися детальніше. Найбільш примітна з його характеристик - популярність. Щоразу, коли ми бачимо успішний продукт у світі технологій, його миттєво копіюють інші організації, які пропонують додаткові вдосконалення. Це призводить до фрагментації бази користувачів.

У галузі обміну миттєвими повідомленнями ця фрагментація є дуже помітною і є мотивацією для цієї тези. Однак електронна пошта не зазнала такої фрагментації своєї бази користувачів. Існує велика різноманітність програмного забезпечення сервера електронної пошти, програмного забезпечення клієнта електронної пошти та програмного забезпечення веб-клієнта, але протокол, а точніше протоколи, використовують переважно однаково. Перевагою є те, що користувачі можуть вільно обирати провайдера для розміщення своєї електронної пошти та вільно вибирати клієнтське програмне забезпечення для надсилання та отримання своїх електронних повідомлень.

Усі вони можуть надсилати повідомлення людям, використовуючи різні провайдери та різне програмне забезпечення. Оскільки більшість користувачів комп'ютерів або смартфонів мають один або кілька облікових записів електронної пошти, надсилати повідомлення електронної пошти сім'ї, друзям та колегам легко, не перемикаючись між програмами.

Крім того, облікові записи електронної пошти, як правило, не пов'язані з номером телефону чи обліковим записом у соціальній мережі. Облікові записи електронної пошти зазвичай можна створювати анонімно та безкоштовно. Потім користувач може одночасно входити в обліковий запис із будь-якого пристрою через поштовий клієнт або веб-інтерфейс. Однак електронна пошта далеко не ідеальна. У ній відсутнє вбудоване шифрування: повідомлення може читати

кожен, хто обробляє їх по дорозі до місця призначення (вони іноді шифруються під час передачі між клієнтом та сервером). Повідомлення можуть бути легко підроблені, щоб надходити від іншого відправника, і вони містять багато метаданих. Існує кілька інструментів, якими можна зашифрувати тіло повідомлень електронної пошти. Два з них, безпечне / багатоцільове розширення пошти в Інтернеті (S / MIME) та досить хороша конфіденційність (PGP), розглядаються нижче. Однак ці інструменти, як правило, дуже громіздкі у налаштуванні і працюють лише в тому випадку, якщо одержувач створив їх для своєї електронної пошти. Але навіть тоді шифрується лише тіло повідомлення та вкладення.

Безпечне / багатоцільове розширення пошти в Інтернеті (S / MIME) S / MIME7 - це схема шифрування та підписання із відкритим ключем, яка покладається на централізований центр сертифікації (CA) для обробки ключів шифрування. Ключі складаються з сертифікатів X.509, того самого типу сертифікату, який використовується для зашифрованого веб-трафіку TLS. Це не автономна система обміну повідомленнями, а швидше схема шифрування, яка

може використовуватися в поєднанні з електронною поштою для шифрування основного тексту та вкладень повідомлень.

Налаштування та підтримка такого CA є досить складною, тому S / MIME в основному використовується більшими корпораціями чи іншими установами. Фізичні особи рідко стикаються з проблемою створення такого CA для особистого користування. Крім того, для обміну повідомленнями, зашифрованими за допомогою S / MIME, обидві сторони повинні бути зареєстровані в центрі сертифікації, з якої вони отримують свої ключі шифрування. Використання S / MIME вимагає певної довіри до центру сертифікації для відповідального управління ключами. Сертифікати X.509 мають ієрархічну деревоподібну структуру, де ключовими кодами користувачів є листи. Кореневий та посередницькі ключі можуть бути використані для підробки повідомлень, що надходять від користувачів, які проходять автентифікацію за допомогою ключів, що є листи в одній гілці. Якщо кореневий або посередницький ключі просочились, несанкціонований власник ключів міг підробляти повідомлення, виконувати атаки та довільно видавати ключі, яким буде довіряти кожен. Що стосується TLS, це вже неодноразово траплялось із серйозними CA, як Google Internet Authority G2. Зверніть увагу, що такий вид депонування ключів може бути корисним для робочого середовища. Зміст електронного листа, надісланого працівниками компанії, може мати великий вплив на репутацію компанії, і часто перевіряється з метою забезпечення якості. У деяких випадках електронна пошта працівника може бути використана як доказ у судових спорах.

Pretty Good Privacy (PGP) - це програма для шифрування та підписання вмісту за допомогою схеми шифрування із відкритим ключем, яка не базується на централізованому ЦС. Натомість PGP покладається на децентралізовану мережу довіри для автентифікації ключів. Зверніть увагу, що OpenPGP Alliance підтримує відкриту версію протоколу під назвою OpenPGP9. PGP - це не

автономна система обміну повідомленнями, а скоріше схема шифрування, яку можна використовувати практично з будь-якою іншою системою обміну повідомленнями для шифрування та підпису основного тексту та вкладень повідомлення.

Метадані таких повідомлень зазвичай залишаються незашифрованими. Здебільшого PGP використовується для шифрування та підпису електронних листів. Для автоматизації цього завдання доступні плагіни для різних поштових клієнтів (наприклад, Enigmail для Thunderbird). Існує навіть розширення браузера Mailvelope, яке дозволяє шифрувати та розшифровувати електронні листи за

допомогою таких популярних веб-клієнтів, як GMail. PGP існує деякий час і зарекомендував себе як дуже безпечний. У своєму витоку секретів Агенції національної безпеки у 2013 році викривач Едвард Сноуден покладався на зашифроване електронне повідомлення PGP для спілкування з журналістом Гленном Грінвальдом. На відміну від S / MIME, PGP в основному використовується в приватних середовищах, оскільки це не надто практично для робочого середовища. На жаль, PGP не був прийнятий широкою громадськістю; він в основному використовується невеликою частиною дуже підкованої та

орієнтованої на конфіденційність групи людей. Багато в чому це пов’язано з тим, що налаштовувати та використовувати дуже громіздко. Крім того, система вимагає багато роботи від користувачів, щоб їхній відкритий ключ був підписаний іншими людьми.

Telegram - це хмарний месенджер, орієнтований на конфіденційність, створений російськими братами Миколою та Павлом Дуровими. Інтерфейс прикладного програмування (API) Telegram добре задокументований та доступний в Інтернеті. Це дозволяє користувачам створювати власні клієнти, які надсилають та отримують повідомлення через Telegram. Telegram розрізняє

два режими обміну повідомленнями. У стандартному режимі повідомлення шифруються лише під час передачі та синхронізуються на всіх пристроях. У режимі таємного чату повідомлення обмінюються лише між двома пристроями (не всі клієнти підтримують секретний чат) і шифруються перед виходом з пристрою. Секретний чат пропонує кілька додаткових функцій, таких як самознищення повідомлень віддалене видалення повідомлень, сповіщень, якщо зроблено скріншот знімка, і неможливо переслати повідомлення з секретних чатів. Усі ці функції вимагають певної форми дії клієнта партнера чату. Якщо партнер чату використовує клієнта, який не підтримує ці функції, вони не працюватимуть. Це проблематично, оскільки ці особливості створюють враження безпеки, хоча їх легко обійти. Деякі джерела стверджують, що функції не працюють належним чином за звичайних обставин: видалені повідомлення з секретного чату нібито залишаються в базі даних повідомлень пристрою. Тим не менше, таємний чат пропонує також інші функції, які є надійними та їх практично неможливо обійти. Двома з них є пряма таємниця та автентифікація позасмугового ключа.

Висновки до першого розділу

Сьогодні використовується багато систем обміну повідомленнями, особливо систем обміну миттєвими повідомленнями. На жаль, користувачі стикаються з компромісом між безпекою та зручністю користування. Старіші системи обміну повідомленнями, такі як електронна пошта та SMS, як правило, дуже небезпечні, оскільки вони не мають жодного вбудованого кодування.

Необов’язкові протоколи шифрування, які можуть бути нанесені шарами зверху, зазвичай обмежуються шифруванням тіла повідомлення і залишають безліч зашифрованих метаданих. Однак деякі з цих старих систем дуже широко

прийняті, і люди можуть обмінюватися повідомленнями незалежно від постачальника послуг обміну повідомленнями, яким вони користуються.

З іншого боку, новіші системи обміну повідомленнями, такі як Threema і Signal, мають вже вбудоване сильне шифрування. Але користувачі можуть обмінюватися повідомленнями лише з людьми, які використовують одного постачальника, і зазвичай змушені користуватися власними клієнтами обміну повідомленнями.

РОЗДІЛ 2. АНАЛІЗ ВИМОГ ДО СИСТЕМИ ОБМІНУ ЗАШИФРОВАНИМИ ПОВІДОМЛЕННЯМИ

2.1. Функціональні вимоги

Функціональні вимоги - це опис послуги, яку має пропонувати програмне забезпечення. Він описує програмну систему або її компонент. Функція - це не що інше, як вхідні дані в програмну систему, її поведінку та результати. Це може бути обчислення, обробка даних, бізнес-процес, взаємодія з користувачем або будь-яка інша специфічна функціональність, яка визначає, яку функцію система може виконувати. Функціональні вимоги також називаються функціональними специфікаціями.

У програмній інженерії та системній інженерії функціональні вимоги можуть варіюватися від абстрактного твердження високого рівня про необхідність відправника до детальних математичних специфікацій функціональних вимог. Функціональні вимоги до програмного забезпечення допомагають визначити передбачувану поведінку системи.

Функціональні вимоги повинні включати такі речі:

- деталі операцій, що проводяться на кожному екрані;
- логіка обробки даних повинна бути введена в систему;
- вони повинні мати описи системних звітів або інших результатів;
- повна інформація про робочі процеси, що виконуються системою;
- у них має бути чітко визначено, кому дозволено створювати / змінювати / видаляти дані в системі;
- яким чином система буде відповідати застосовним нормативним вимогам та потребам відповідності, слід зафіксувати у функціональному документі.

Плюси / переваги створення типового документа про функціональні вимоги:

- допомагає перевірити, чи надає програма всі функціональні можливості, згадані у функціональних вимогах цієї програми;

- документ про функціональні вимоги допомагає визначити функціональність системи або однієї з її підсистем;

- функціональні вимоги разом з аналізом вимог допомагають визначити відсутні вимоги. Вони допомагають чітко визначити очікуваний системний сервіс та поведінку;

- помилки, виявлені на етапі збору функціональних вимог, є найдешевшими для виправлення.

- підтримує цілі, завдання чи дії користувачів.

Найпоширеніші типи функціональних вимог:

- обробка транзакцій;
- правила ведення бізнесу;
- вимоги до сертифікації;
- вимоги до звітності;
- адміністративні функції;
- рівні авторизації;
- відстеження аудиту;
- зовнішні інтерфейси;
- історичне управління даними;
- правові та нормативні вимоги.

В таблиці 2.1. зібрані всі функціональні вимоги до клієнт-серверного веб-додатку для обміну зашифрованими повідомленнями.

Таблиця 2.1

Номер функціональної вимоги	Опис функціональної вимоги
ФВ1	Веб-додаток має містити в собі

	функцію авторизації.
ФВ2	Веб-додаток має містити в собі функцію реєстрації.
ФВ3	Для авторизації необхідно вводити пошту та пароль.
ФВ4	Для авторизації необхідно вводити пошту, пароль та підтвердження паролю.
ФВ5	Авторизований користувач може відправити на сервер зашифроване повідомлення.
ФВ6	Авторизований користувач повинен мати змогу вказати папку отримувача зафифрованого повідомлення.
ФВ7	Система не повинна відправляти повідомлення без указання адреси отримувача.
ФВ8	При відправці повідомлення, система повинна запам’ятовувати ідентифікатор відправника
ФВ9	Веб-додаток повинен бути максимально простим для використання.
ФВ10	Веб-додаток повинен бути візуалізовний в більшості світлими кольорами.

2.2. Нефункціональні вимоги

Нефункціональні вимоги визначають атрибут якості програмної системи. Вони оцінюють програмну систему на основі оперативності, зручності використання, безпеки, портативності та інших нефункціональних стандартів, які мають вирішальне значення для успіху програмної системи. Приклад нефункціональної вимоги "як швидко завантажується веб-сайт?" Невиконання нефункціональних вимог може призвести до систем, які не задовольняють потреби користувачів.

Нефункціональні вимоги дозволяють накладати обмеження або обмеження на конструкцію системи в різних гнучких відставаннях. Наприклад, сайт повинен завантажуватися за 3 секунди, коли кількість одночасних користувачів> 10000. Опис нефункціональних вимог є настільки ж важливим, як і функціональна вимога.

Типи нефункціональних вимог:

1. Вимоги юзабіліті.
2. Вимоги до справності.
3. Вимоги керованості.
4. Вимоги відновлення.
5. Вимоги безпеки.
6. Вимоги цілісності даних.
7. Вимоги до ємності.
8. Вимоги доступності.
9. Вимоги до масштабованості.
10. Вимоги сумісності.
11. Вимоги надійності.
12. Вимоги ремонтопридатності.
13. Нормативні вимоги.

14. Вимоги до середовища.

В таблиці 2.2. зібрані всі функціональні вимоги до клієнт-серверного веб-додатку для обміну зашифрованими повідомленнями.

Таблиця 2.2

Номер нефункціональної вимоги	Опис нефункціональної вимоги
НФ1	Мовою розробки повинна бути Java Enterprise Edition.
НФ2	Використати мікросервісну архітектуру для системи.
НФ3	Візуалізація повинна бути у вигляді JSTL сторінок.
НФ4	Реалізувати мікросервіс Web
НФ5	Реалізувати мікросервіс MessageConverter.
НФ6	Реалізувати мікросервіс Encryptor.
НФ7	Реалізувати мікросервіс SftpCommunicator.
НФ9	База даних – PostgreSQL.
НФ10	Зберігати зашифровані повідомлення на sftp-сервері.
НФ11	Всі дані про віправника або отримувача зберігати в базі даних.

Висновки до другого розділу

Технічне завдання є невід’ємною частиною майьутнього проєкту, це буде співвідносити бажання головного замовника та можливості програмістів, це збереже час та усуне усі можливі конфлікти. Зрештою, технічне завдання охоплює коло задач, які необхідно виконати в рамках розробки системи обміну зашифрованими повідомленнями.

Для того, щоб розробники веб-додатків могли повністю реалізувати ідею головного замовника, необхідно максимально детально її проектувати, пояснити своє представлення того, яким має бути кінцевий варіант. Відповідно, технічне завдання на розробку системи обміну зашифрованими повідомленнями повинно містити:

- технічні задачі;
- функціональні вимоги;
- нефункціональні вимоги;

В ідеалі всі деталі функціонування та тип системи повинні бути вказані в робочому звіті - тоді суперечності будуть мінімальними, а результат відповідатиме вимогам та очікуванням головного замовника.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РОЗРОБКА ВЕБ-ДОДАТКУ

3.1. Проектування клієнт среверної архітектури

"Мікросервіси" - ще один новий термін на переповнених вулицях архітектури програмного забезпечення. Хоча наше природне прагнення полягає в тому, щоб пропускати подібні речі з презирливим поглядом, ця частина термінології описує стиль програмних систем, який ми вважаємо все більш привабливим. За останні кілька років ми бачили, що багато проектів використовують цей стиль, і досі результати були позитивними, настільки, що для багатьох наших колег це стає стилем за замовчуванням для побудови корпоративних додатків. На жаль, проте, є не так багато інформації, яка б визначала, що таке мікросервіс і як це зробити.

Коротше кажучи, архітектурний стиль мікросервісу - це підхід до розробки єдиного додатка як набору невеликих сервісів, кожен з яких працює у своєму власному процесі та взаємодіє з легкими механізмами, часто API ресурсів HTTP. Ці служби побудовані на комерційних можливостях і можуть бути розгорнуті незалежно від повністю автоматизованого механізму розгортання. Існує мінімум централізованого управління цими службами, яке може бути написане різними мовами програмування та використовувати різні технології зберігання даних.

Мій посібник з мікросервісів містить посилання на найкращі статті, відео, книги та подкасти про мікросервіси.

Для початку пояснення стилю мікросервісу корисно порівняти його з монолітним стилем: монолітним додатком, побудованим як єдине ціле. Підприємницькі програми часто будуються з трьох основних частин:користувацький інтерфейс на стороні клієнта (що складається із сторінок HTML та javascript, що працюють у браузері на машині користувача), база даних (що складається з багатьох таблиць, вставлених у загальне та, як правило, реляційне управління базами даних система), а також серверна програма. Серверна програма обробляє HTTP-запити, виконує логіку домену,

отримує та оновлює дані з бази даних, а також вибирає та заповнює подання HTML для надсилання у браузер. Цей серверний додаток являє собою моноліт - єдиний логічний виконуваний файл. Будь-які зміни в системі передбачають створення та розгортання нової версії серверної програми.

Такий монолітний сервер є природним способом підійти до побудови такої системи. Вся ваша логіка для обробки запиту працює в одному процесі, що дозволяє використовувати основні функції вашої мови для розподілу програми на класи, функції та простори імен. З певною обережністю ви можете запустити та протестувати додаток на ноутбуці розробника та скористатися конвеєром розгортання, щоб забезпечити належне тестування та впровадження змін у виробництво. Ви можете горизонтально масштабувати моноліт, запускаючи багато екземплярів за балансиром навантаження.

Монолітні програми можуть бути успішними, але все частіше люди відчувають розчарування з ними - тим більше, що все більше програм розгортається в хмарі. Цикли змін пов’язані між собою - зміна, внесена до невеликої частини програми, вимагає перебудови і розгортання всього моноліту. З часом часто важко зберегти хорошу модульну структуру, ускладнюючи збереження змін, які повинні стосуватися лише одного модуля в цьому модулі. Масштабування вимагає масштабування всієї програми, а не її частин, що потребують більших ресурсів.

Клієнт-серверний веб-додаток для обміну зашифрованими повідомленнями проектується на основі мікросервісної архітектури.

Далі, на діаграмі взаємодії компонентів показані всі зв'язки відокремлених модулів програми.

Було використано даних підхід, для того щоб в майбутньому легко можно було замінити або перевикористати окремий мікросервіс.

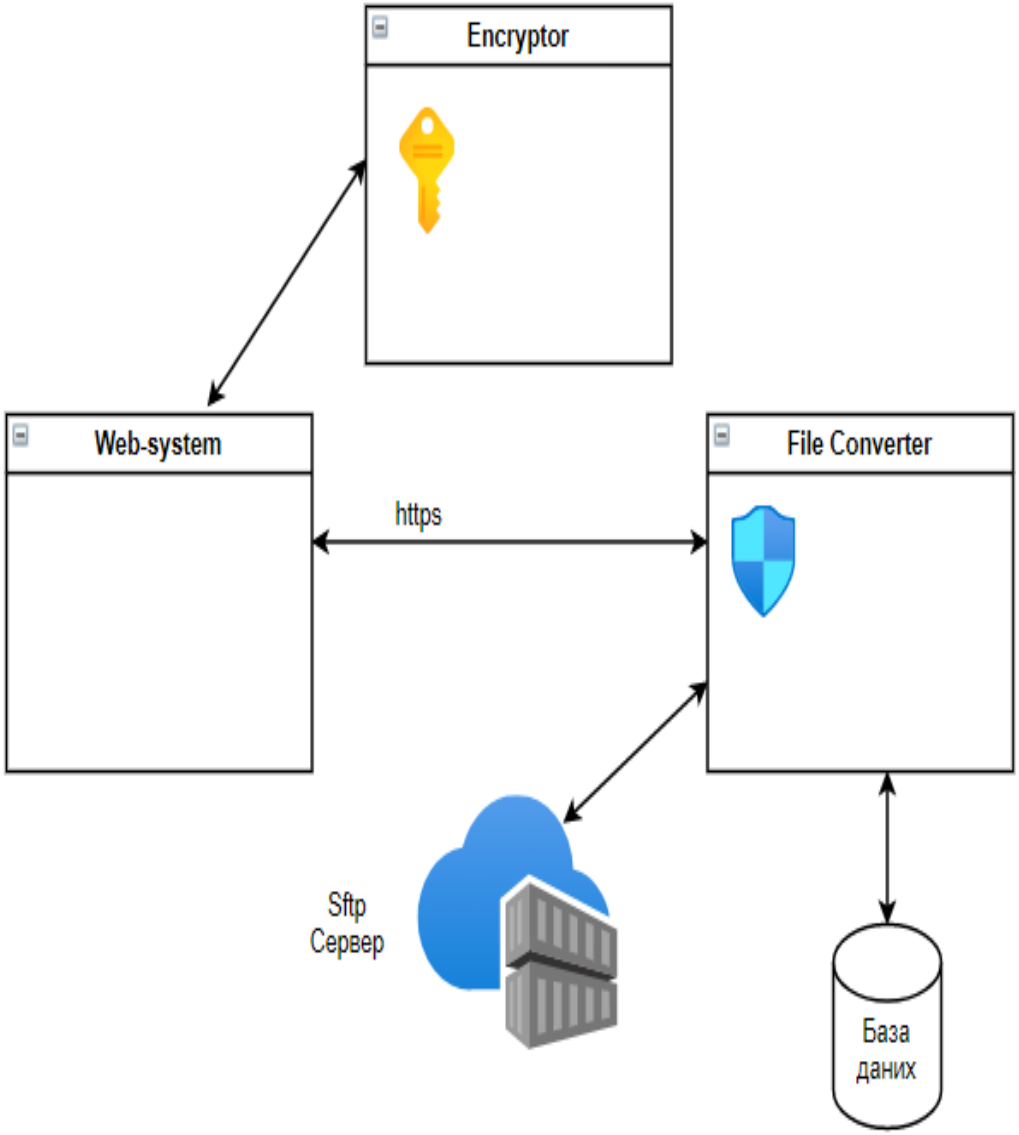

Рис. 3.1. Мікросервісна архітектура веб-додатку для обміну зашифрованими повідомленнями.

Таблиця 3.1

Компонент	Функція
Web-system	Слугує для відображення візуалізації html-сторінок функціоналу для користувача.
Encryptor	Даний мікросервіс призначений для

	шифрування та дефишрування приватних повідомлень користувачів.
FileConverter	Обробляє запити користувача на зберігання приватних повідомлень та відправляє їх до sftp серверу.
База даних	Зберігає інформацію про відправників та отримувачів повідомлень та метадані користувачів.
Sftp сервер	Сервер для зберігання зашифрованих повідомлень.

3.2. Вибір програмного середовища

Maven. Maven - це потужний інструмент управління проектами, який базується на POM (об'єктна модель проекту). Він використовується для побудови проектів, залежностей та документації. Це спрощує процес побудови, як ANT. Але він набагато просунутий, ніж АНТ.

Коротше кажучи, ми можемо сказати, що maven - це інструмент, який можна використовувати для побудови та управління будь-яким проектом на основі Java. maven полегшує повсякденну роботу розробників Java і, як правило, допомагає зрозуміти будь-який проект на основі Java.

Maven виконує багато корисних завдань, таких як

1. Створення проекту за допомогою maven.

2. Додавання бібіліотек та інші залежності проекту, використовуючи допомогу maven.

3. Maven надає інформацію про проект (журнал, список залежностей, звіти про модульні випробування тощо)

4. Maven дуже корисний для проекту під час оновлення центрального сховища JAR та інших залежностей.

5. За допомогою Maven можна вбудувати будь-яку кількість проектів у типи виводу, такі як JAR, WAR тощо, не виконуючи сценаріїв.

6. За допомогою maven можна легко інтегрувати проект із системою керування джерелами (наприклад, Subversion або Git).

Основні концепції Maven:

1. Файли POM: Файли об’єктної моделі проекту (POM) - це XML-файл, що містить інформацію, пов’язану з проектом та інформацію про конфігурацію, таку як залежності, вихідний каталог, плагін, цілі тощо, що використовуються Maven для побудови проекту. Коли вам слід виконати команду maven, надається maven файл POM для виконання команд. Maven читає файл pom.xml, щоб виконати його конфігурацію та операції.

2. Залежності та сховища: Залежності - це зовнішні бібліотеки Java, необхідні для проекту, а сховища - каталоги упакованих файлів JAR. Локальне сховище - це просто каталог на жорсткому диску вашої машини. Якщо залежності не знайдені в локальному сховищі Maven, Maven завантажує їх із центрального сховища Maven і поміщає у ваше локальне сховище.

3. Життєві цикли побудови, фази та цілі: Життєвий цикл побудови складається з послідовності фаз побудови, а кожна фаза побудови складається з послідовності цілей. Команда Maven - це назва життєвого циклу, фази або цілі збірки. Якщо запитується життєвий цикл, виконуючи команду maven, виконуються також усі фази побудови в цьому життєвому циклі. Якщо вимагається виконання фази збірки, виконуються також усі фази збірки до неї у визначеній послідовності.

4. Побудова профілів: Необхідно створити набір значень конфігурації, що дозволяє будувати проект, використовуючи різні конфігурації. Наприклад, може знадобитися побудувати проект для локального комп’ютера, для розробки та тестування. Щоб увімкнути різні збірки, можна додавати різні профілі збірки

до файлів POM за допомогою елементів його профілів і запускаються різними способами.

5. Створення плагінів: Плагіни побудови використовуються для досягнення певної мети. Можна додати плагін до файлу POM. Maven має деякі стандартні плагіни, якими можна скористатися, а також можна реалізувати свій власний на Java.

MySQL. MySQL - це система управління реляційними базами даних (RDBMS) із відкритим кодом, що підтримується Oracle, заснована на мові структурованих запитів (SQL). MySQL працює практично на всіх платформах, включаючи Linux, UNIX та Windows. Хоча його можна використовувати в широкому діапазоні програм, MySQL найчастіше асоціюється з веб-додатками та публікацією в Інтернеті.

MySQL є важливим компонентом корпоративного стеку з відкритим кодом, який називається LAMP. LAMP - це платформа веб-розробки, яка використовує Linux як операційну систему, Apache як веб-сервер, MySQL як реляційну систему управління базами даних та PHP як об'єктно-орієнтовану мову сценаріїв. (Іноді замість PHP використовується Perl або Python.)

Спочатку задуманий шведською компанією MySQL AB, MySQL був придбаний Sun Microsystems в 2008 році, а потім Oracle, коли він придбав Sun у 2010 році. Розробники можуть використовувати MySQL під загальною публічною ліцензією GNU (GPL), але підприємства повинні отримати комерційну ліцензію від Oracle.

Сьогодні MySQL є СУБД, що стоїть за багатьма провідними веб-сайтами у світі та незліченними корпоративними та споживчими веб-додатками, включаючи Facebook, Twitter та YouTube.

MySQL базується на моделі клієнт-сервер. Ядром MySQL є сервер MySQL, який обробляє всі інструкції бази даних (або команди). Сервер MySQL доступний як окрема програма для використання в мережевому середовищі клієнт-сервер та як бібліотека, яка може бути вбудована (або пов'язана) в окремі програми.

MySQL працює разом із декількома утилітами, які підтримують адміністрування баз даних MySQL. Команди надсилаються на MySQLServer через клієнт MySQL, який встановлений на комп'ютері.

Спочатку MySQL був розроблений для швидкої обробки великих баз даних. Хоча MySQL зазвичай встановлюється лише на одній машині, він може надсилати базу даних у різні місця, оскільки користувачі можуть отримати до неї доступ через різні клієнтські інтерфейси MySQL. Ці інтерфейси надсилають на сервер оператори SQL, а потім відображають результати.

3.3. Розробка системи обміну зашифрованими повідмленнями

Web-system. Даний модуль реалізовано за допомогою фреймворку Spring Boot. Далі представлений лістинг конфігураційного файлу для налаштування та запуску веб-додатку.

```
<properties>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
<maven.compiler.source>1.7</maven.compiler.source>
<maven.compiler.target>1.7</maven.compiler.target>
</properties>

<dependencies>
<dependency>
<groupId>mysql</groupId>
<artifactId>mysql-connector-java</artifactId>
<version>5.1.6</version>
</dependency>

<!-- Tests -->
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.12</version>
<scope>test</scope>
</dependency>
<dependency>
```

```
<groupId>org.mockito</groupId>
<artifactId>mockito-core</artifactId>
<version>2.23.4</version>
<scope>test</scope>
</dependency>

<!-- Spring -->
<dependency>
<groupId>javax.servlet</groupId>
<artifactId>javax.servlet-api</artifactId>
<version>3.1.0</version>
<scope>provided</scope>
</dependency>

<dependency>
<groupId>log4j</groupId>
<artifactId>log4j</artifactId>
<version>1.2.17</version>
</dependency>

<dependency>
<groupId>commons-dbcp</groupId>
<artifactId>commons-dbcp</artifactId>
<version>1.4</version>
</dependency>

<dependency>
<groupId>jstl</groupId>
<artifactId>jstl</artifactId>
<version>1.2</version>
</dependency>
<dependency>
<groupId>taglibs</groupId>
<artifactId>standard</artifactId>
<version>1.1.2</version>
</dependency>
```

Тут необхідні бібліотеки для розгортання веб-додатку на сервері. Також можем бачити jstl, безпосередньо для розгортання front-end частини.

JSTL підтримує загальні структурні завдання, такі як ітерація та умови, теги для маніпулювання документами HTML, теги інтернаціоналізації та теги SQL. Він також забезпечує основу для інтеграції існуючих спеціальних тегів з тегами JSTL.

Далі показаний приклад реалізації сторінки входу в систему.

```
<article class="card-body mx-auto" style="max-width: 400px;">
<h4 class="card-title mt-3 text-center"><fmt:message
key="login.login"/></h4>
<form class="form-signin"
action="${pageContext.request.contextPath}/login" method="post">

<input type="username"  name="username" class="form-control"
placeholder="Username" required>
<input type="password" name="password" class="form-control"
placeholder="Password" required>
<button class="btn btn-lg btn-primary btn-block btn-signin"
type="submit"><fmt:message key="login.button.login"/></button>
<p class="text-center"><fmt:message key="login.link.to.registration"/><a
href="${pageContext.request.contextPath}/registration"><u><fmt:message
key="login.button.registration"/></u></a></p>
</form>
</article>
```

File Converter в першу чергу конвертує файли повідомлень, та відправляє до sftp серверу. Далі наведений лістинг налаштування конфігурації з’єднання з sftp-сервером.

Spring Integration забезпечує підтримку операцій передачі файлів через SFTP.

Протокол безпечної передачі файлів (SFTP) - це мережевий протокол, який дозволяє передавати файли між двома комп'ютерами в Інтернеті через будь-який надійний потік.

Протокол SFTP вимагає захищеного каналу, такого як SSH, і видимість ідентифікації клієнта протягом сеансу SFTP.

Spring Integration підтримує надсилання та отримання файлів через SFTP, надаючи три кінцеві точки на стороні клієнта: адаптер вхідного каналу, адаптер вихідного каналу та вихідний шлюз. Він також забезпечує зручну конфігурацію простору імен для визначення цих клієнтських компонентів.

Мені потрібно включити цю залежність у свій проект:

```
<dependency>
<groupId>org.springframework.integration</groupId>
<artifactId>spring-integration-sftp</artifactId>
<version>5.5.0</version>
</dependency>
```

Далі програмуємо налаштування сессії та підключення до серверу безпосередньо.

```
@Bean
public SessionFactory<LsEntry> sftpSessionFactory() {
   DefaultSftpSessionFactory factory = new DefaultSftpSessionFactory(true);
   factory.setHost("localhost");
   factory.setPort(2343);
   factory.setUser("vdovichenko");
   factory.setPassword("1223d32");
   factory.setAllowUnknownKeys(true);
   factory.setTestSession(true);
   return new CachingSessionFactory<LsEntry>(factory);
}

@Bean
@ServiceActivator(inputChannel = "sftpChannel")
public MessageHandler handler() {
   return new MessageHandler() {

      @Override
      public void handleMessage(Message<?> message) throws
MessagingException {
         System.out.println(message.getPayload());
      }

   };
}
```

Модель зашифрованого повідомлення. Для передачі інформації про відправника, отримувача та зашифрованого повідомлення необхідно створити dto клас модель.

```
import java.time.LocalDateTime;

public class MessageDto {

    private Long id;
    private String encryptedBody;
    private String addressSender;
    private String addressGetter;
    private String sftpFolder;
    private String status;
    private LocalDateTime time;
```

1. Встановлюємо ідентифікатор повідомлення.
2. Далі описуємо поле для зберігання зашифрованого повідомлення.
3. Наступним йде адреса відправника.
4. Далі вказана адреса отримувача.
5. Вказуємо назву папки на sftp куди покласти повідомлення.
6. Зберігаємо статус повідомлення.

Висновки до третього розділу

У даному розділі було розроблено програму для обміну зашифрованими повідомленнями. Наведено повний опис всіх етапів розробки з додаванням ілюстрацій до кожного кроку розробки. Для розробки було обрано мову програмування Java, конфігуратор проєктів maven та систему управління базами даних MySQL. Налаштовано проєкт за допомогою maven та MySQL.

ВИСНОВОК

У дипломній роботі розглянуті існуючі протоколи та реалізації розподілених систем обміну повідомленнями.

Розроблено програму для обміну зашифрованими повідомленнями. Наведено повний опис всіх етапів розробки з додаванням ілюстрацій до кожного кроку розробки. Для розробки було обрано мову програмування Java, конфігуратор проєктів maven та систему управління базами даних MySQL. Налаштовано проєкт за допомогою maven та MySQL.

Запропонована архітектура системи обміну зашифрованими повідомленнями, що характеризується високою автономністю клієнтів, можливістю відкритого шифрування повідомлень без участі сервера та полегшеним масштабуванням серверів.

Розроблений протокол взаємодії клієнтської та серверної частин.

Розроблена архітектура серверного та клієнтського ПЗ.

Результати роботи можуть бути застосовані для побудови розподіленого сервісу обміну повідомленнями з високою надійністю. Розроблене програмне забезпечення відноситься до галузі інформаційної безпеки і може бути використане для секретного спілкування в інтернеті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Web-програмування [Веб-сайт] Режим доступу: http://webstudio2u.net/ua/programming/96-cms.html
2. Мастер-класс по цифровому звуку [Веб-сайт] Режим доступу: https://compress.ru/article.aspx?id=11797
3. Биков В.Ю. Моделі організаційних систем відкритої освіти : монографія / В.Ю. Биков. – К. : Атіка, 2009. – 682 с
4. Носенко Ю.Г. Еволюція хмарних обчислень як актуального засобу навчання / Носенко Ю.Г. // Інформатика та інформаційні технології в навчальних закладах. – № 5. – 2015. – С. 16-21.
5. Носенко Ю.Г. Хмарні технології у просторі відкритої освіти / Юлія Носенко // Моделювання й інтеграція сервісів хмаро орієнтованого навчального середовища : монографія; за заг. ред. С.Г. Литвинової. – К. : ЦП «Компринт», 2015. – С. 24-34.
6. ISO/IEC 17788:2014(E). Information technology – Cloud computing – Overview and vocabulary : International Standard. – Switzerland : ISO/IEC, 2014. – 14 p.
7. Хилайер, С. Программирование Active Server Pages [Текст] / С. Хилайер, Д. Мизик. — М. : Русская редакція.
8. Спейнауэр, С. Справочник Web-мастера [Текст] / С. Спейнауэр, В. Куэрсиа. — К. : BHV, 1997. — 368 с.