МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота

на тему

Розроблення алгоритмів та моделей технологій кібербезпеки

Рівень вищої освіти другий (магістерський)

Спеціальності 122 Комп'ютерні науки

Осітня програм Комп'ютерні науки

Виконав: студент групи МгІТ-1-22
Унгур`ян С.Д.
Науковий керівник:
к.т.н., доц. Астістова Т.І.

Рецензент:
к.т.н., доц. МельникГ.В.

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерні науки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри КН

_____Володимир ЩЕРБАНЬ

«_____»_____2023_____року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ СТУДЕНТА

Унгур`яну Сергію Дмитровичу

1.Тема роботи: Розроблення алгоритмів та моделей технологій кібербезпеки, науковий керівник роботи: Астістова Тетяна Іванівна, к.т.н., доц., затверджені наказом закладу вищої освіти від 12.09.2023 року, № 210-уч.

2. Строк подання студентом роботи 12.11. 2023р.

3. Вихідні дані до роботи: Розробка кафедри комп'ютерних наук

4. Зміст кваліфікаційної роботи (перелік питань, які потрібно розробити)

РОЗДІЛ 1. Теоретичні аспекти предметної області; РОЗДІЛ 2.Алгоритми та методи захисту інформації; РОЗДІЛ 3. Реалізація розробки системи. Додатоки.

5. Дата видачі завдання: 08. 2023р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів	Примітка про виконання
1	Вступ	20.08.2023	
2	Розділ 1. Теоретичні аспекти предметної області	10.09.2023	
3	Розділ 2. Алгоритми та методи захисту інформації	5.10.2023	
4	Розділ 3. Реалізація розробки системи	25.10.2023	
5	Висновки	28.10.2023	
6	Оформлення кваліфікаційної роботи (чистовий варіант)	31.10.2023	
7	Подача кваліфікаційної роботи науковому керівнику для відгуку	01.11.2023	
8	Подача кваліфікаційної роботи на кафедру для рецензування	06.11.2023	
9	Перевірка кваліфікаційної роботи на наявність ознак плагіату	07.11.2023	
10	Подання кваліфікаційної роботи на затвердження завідувачу кафедри	08.11.2023	

Студент _____ Сергій УНГУР`ЯН

Науковий керівник роботи _____ Тетяна АСТІСТОВА

АНОТАЦІЯ

Унгур`яну С. Д. Розроблення алгоритмів та моделей технологій кібербезпеки.

Кваліфікаційна робота за спеціальністю 122 - «Комп'ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

В магістерській кваліфікаційній роботі проведено аналіз технологій в галузі кібербезпеки; моделей інформаційної безпеки; наведені приклади типів атак; розглянуто методи та принципи шифрування; розглянуто алгоритми шифрування на прикладах алгоритму Г. Файстеля, алгоритму DES, алгоритму безпеки BCrypt, алгоритм шифрування Blowfish та шифрування в системі *RSA*. Розглянуто криптосистему з відкритим ключем Ель-Гамалю та цифрового підпис Ель-Гамалю .

В роботі була розроблена система аутентифікації даних користувача на основі клієнт - серверної технології. Ця підсистема може використовуватись як в особистому використанні, так і в інформаційно-телекомунікаційної системи .

Розроблена база даних, інтерфейс користувача, діаграма прецидентів сценарію. Розроблена програмна реалізація алгоритму Ель-Гамаля (AES) на мові програмування Python та програма на основі алгоритму Ель-Гамаля для управління пристроєм та шифрування/дешифрування повідомлень.

Використання алгоритму AES та універсального коду дозволяють встановлювати справжність автора даних у разі виникнення сумнівів щодо авторства та забезпечення конфіденційності.

Ключові слова: Python, алгоритми, клієнт–серверні технології, автентифікація, шифрування, цифровий підпис Ель-Гамалю, RSA, BCrypt, DES, Blowfish.

ANNOTATION

Unguryanu S. D. Development of algorithms and models of cyber security technologies.

Qualification work on specialty 122 - "Computer science". - Kyiv National University of Technology and Design, Kyiv, 2023.

In the master's qualification work, an analysis of technologies in the field of cyber security was carried out; information security models; examples of attack types are given; methods and principles of encryption are considered; encryption algorithms are considered using the examples of H. Feistel's algorithm, DES algorithm, BCrypt security algorithm, Blowfish encryption algorithm and RSA encryption. The El-Gamal public-key cryptosystem and the El-Gamal digital signature are considered.

The work developed a user data authentication system based on client-server technology. This subsystem can be used both in personal use and in the information and telecommunication system. Developed database, user interface, script precedent diagram. Developed software implementation of El-Gamal algorithm (AES) in Python programming language and software based on El-Gamal algorithm for device control and message encryption/decryption.

The use of the AES algorithm and the universal code allow you to establish the authenticity of the author of the data in case of doubts about the authorship and ensure confidentiality.

Keywords: Python, algorithms, client-server technologies, authentication, encryption, El-Gamal digital signature, RSA, BCrypt, DES, Blowf

ЗМІСТ

ВСТУП

Актуальність теми. Інформаційно-телекомунікаційна система, доступ до якої здійснюється з використанням комп'ютерних та інших технічних пристроїв, призначена для передачі інформації. Передавати відеозаписи, аудіозаписи, документи, графіку можна великою кількістю способів, тому можливості інформаційно-телекомунікаційних систем практично необмежені. Способи передачі інформації, висока точність і швидкість передачі даних можлива завдяки високому рівню розвитку інформаційно-телекомунікаційних систем. Розвиток телекомунікацій характеризується переходом від кількісного зростання до якісного, а саме, до надання широкої номенклатури безпечних послуг телекомунікацій.

У сучасних умовах кількість електронних даних, яка зберігається, обробляється і передається, безперервно зростає. Це призводить до забезпечення комплексного захисту даних користувачів: забезпечення конфіденційності (отримання даних третіми особами), цілісності (несанкціоновані зміни інформації) даних; а також підтвердження автентичності.

Інформаційна безпека та життєдіяльність суспільства залежать від надійності, стабільного функціонування телекомунікаційних мереж . В зв’язку з постійним ростом продуктивності обчислювальних засобів необхідно постійно удосконалювати існуючі криптографічні схеми для збереження необхідного рівня стійкості. На сьогодні існує проблема аувентифікації даних в інформаційно-телекомунікаційних системах.

Найпростіший спосіб вирішення даної проблеми – це використання ключів більшої довжини, що призводить до зростання часу роботи алгоритму аутентифікації..

Проблема автентифікації даних користувачів в інформаційно-телекомунікаційних системах є одним з ключових аспектів інформаційної

безпеки.

Мета дослідження. Дослідження алгоритмів та моделей технологій кібербезпеки .

Завдання дослідження. Завданням роботи було розробити варіант системи автентифікації даних користувачів.

Об'єкт дослідження. Об’єктом даного дослідження є порівняння існуючих алгоритмів різних видів шифрування, існуючих моделі технологій кібербезпеки, архітектури мереж наступних поколінь.

Предмет дослідження. Алгоритми технології криптографії на прикладі цифрових підписів та розподілу ключів.

Методи дослідження. Теоретичною основою при вирішенні проблеми є розгляд методів криптографії та криптоаналізу, шифрування,видів атак, найбільш поширених алгоритмів блокового шифрування
опис найбільш поширених алгоритмів блокового шифрування

Наукова новизна. Розробка універсального коду, який можна використати для більшості мов як програмування, так і скріптових в концепції електронно-цифрового підпису на основі алгоритму AES**.**

Практичне значення отриманих результатів. Використання алгоритму AES та універсального коду дозволяють встановлювати справжність автора даних у разі виникнення сумнівів щодо авторства та забезпечення конфіденційності.

Результати роботи були опубліковані в наступних виступах та статтях:

1. Астістова Т. І. Дослідження алгоритмів та моделей технологій кібербезпеки / Т. І. Астістова, С. Д. Унгур'ян // Інформаційні технології в науці, виробництві та підприємництві : збірник наукових праць молодих вчених, аспірантів, магістрів кафедри комп’ютерних наук та технологій / за заг. наук. ред. В. Ю. Щербаня. – Київ : ТОВ "Фастбінд Україна", 2023. – С. 112-115.

https://er.knutd.edu.ua/handle/123456789/24123

РОЗДІЛ 1. ТЕОРЕТИЧНІ АСПЕКТИ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Постановка задачі.

Найціннішим ресурсом завжди була інформація, оскільки завдяки інформації можна отримати різні види даних (внутрішню ситуацію в країні, розташування ворожих військ, місце знаходження ресурсів , і так далі). Інформація такого виду завжди була таємною і всі її хотіли отримати та щось змінити. Якщо інформацією не обмінюватись, то і користі від неї не буде. Це і є слабким місцем в секретності інформації.

Це питання одним із перших вирішив Юрій Цезар. Він почав використовувати шифр підстановки для приватного листування, який з часом був названий в його честь.

На даний момент безпека інформації стосується не тільки державної, військової справи але і цивільної, оскільки ми живемо в світі, який живе в Інтернеті, і вся інформація є широко доступною.

Це стосується особистої інформації, банківських рахунків фізичних і юридичних осіб і так далі. Відповідно і доступ до частини інформації повинен бути обмежений, оскільки отримання її третім особами, може принести шкоду її власникові.

Для забезпечення захисту такого виду інформації її шифрують, щоб отримати доступ до неї зможе лише той хто має права доступу.

Інформаційна безпека та життєдіяльність суспільства залежать від надійності, стабільного функціонування телекомунікаційних мереж . В зв’язку з постійним ростом продуктивності обчислювальних засобів необхідно постійно удосконалювати існуючі криптографічні схеми для збереження необхідного рівня стійкості. На сьогодні існує проблема автентифікації даних в інформаційно-телекомунікаційних системах.

Велика кількість алгоритмів та протоколі, які базуються на різних математичних моделях та криптографічних схемах, використовується для автентіфікації даних.

Автентифікацією даних, це встановлення достовірності даних на основі інформації, що міститься в отриманих по мережі даних.

Шифрування та автентифікація мають різні мети:

1. Шифрування даних - забезпечення захисту від несанкціонованого ознайомлення з цією інформацією, то кінцевою метою
2. Автентифікація інформації - забезпечення захисту учасників інформаційного обміну від нав'язування хибної інформації.

Існує доволі багато алгоритмів шифрування різної категорії. На даний момент одним із найбезпечніших алгоритмів шифрування є алгоритм AES. Алгоритм AES використовується як основний для роботи спецслужб США.

1.2. Основні поняття криптографічного захисту інформації

Технології криптографії – це застосування операцій перетворення даних у простому вигляді для того, щоб забезпечити шифрування (Encryption) їх таємним кодом.

До технологій криптографії відносяться: цифрові підписи, шифрування, розподіл ключів. Розглянемо основні поняття , які використовують у криптографії:

- Шифр - це об’єднання декількох алгоритмів шифрування, що дають криптографічні перетворення, що повертають множину зашифрованих даних на множину відкритих даних і навпаки.
- Ключ - це є параметр криптографічного алгоритму. Ключ дає змогу задати вибір перетворення з декількох відкритих даних у зашифровані дані. Секретність алгоритму шифрування базується на ключі, а не як сам алгоритм шифрування.
- Шифрування - це перетворення даних завдяки певному алгоритму для отримання шифрованого тексту (подальшим приховання даних).

- Розшифрування - перетворення зашифрованих даних у придатний для читання.
- Дешифрування - процес несанкціонованого перетворення інформації з зашифрованих даних (ключ який потрібен для розшифрування є невідомим).
- Шифротекст - це зашифрована інформація в результаті використання шифру.
- Біграмічний шифр - це шифр групи з двох букв.

Алгоритми шифрування (методи криптографії) поділяють на два основних типи:

1. Криптографія симетричного ключа - це алгоритми шифрування в яких використовують один ключ для шифрування і дешифрування інформації . Сторони повинні тримати ключ у таємниці та домовлятися про це. Головною проблемою є розподіл ключів. Система розподілу ключів (Key exchange) повинна встановити сеанси ключа, або ключ менеджменту інформаційним обміном для безпечного з'єднання.

2. Криптографія асиметричного ключа - це алгоритми шифрування з двома ключами: один ключ для шифрування, другий - для розшифрування. Лише власник приватного таємного(закритого) ключа може розшифрувати зашифрований текст.

Для шифрування даних у алгоритмах з асиметричним ключем використовують відкритий ключ. Даний ключ не обов'язково тримати в таємниці і він може бути публічним.

Закритий ключ - це ключ який використовується для дешифрування даних у алгоритмах з асиметричним ключем. Даний ключ обов'язково тримати в таємниці і він повинен бути приватним.

Прикладом практичної реалізації криптографії відкритого ключа є цифрові підписи. Їх використовують у поєднанні з з цифровим сертифікатом.

1.3. Аналіз розвитку шифрування.

Раніше вважалося, що процес перетворення інформації в «незрозумілий» набір символів (процес шифрування) і є криптографія. Зворотнім процесом криптографії є процес розшифрування, коли з «незрозумілої» послідовності символів отримуємо початкову інформацію.

Для створення шифру використовують алгоритми для шифрування і розшифрування. В основу всіх шифрів положено алгоритм, який повинен виконувати всю процедуру і секретний параметр (ключ). Якщо ключ статичний, шифр втрачає всю свою силу.

Останнім часом до задач криптографії почали входити: цифровий підпис, методи перевірки цілісності повідомлення, інтерактивні підтвердження, технології безпечного спілкування, ідентифікувати одержувача і відправника.

В часи, коли криптографія почала зароджуватися, більшість шифрів базувалася на методі перестановки символів. Цей метод захисту інформації виявився дуже слабким. Таким шифром був шифр Юрія Цезаря, який він використовував для спілкування з генералами під час військових дій. Алгоритм шифру був дуже простим: кожна літера тексту відповідно до алфавіту зміщалась на 3 позиції. Цей алгоритм є подібним на EXCESS-3 в булівській алгебрі.

Головною ціллю шифрування – це збереження спілкування в таємниці. Більшість шифрів видають статичну інформацію і це дає можливість зламати алгоритм шифрування і розшифрування. Арабський вчений Аль-Кінд відкрив частотний аналіз в 9 столітті. На сьогодні такі шифри збереглися у вигляді головоломок. В 1467 був винайдений поліалфавітний шифр, коли різні частини повідомлення шифрували різними шифрами. Цей вид шифру назвали шифр Відеженера. Алгоритм цього методу працює наступним чином: за літерою ключового слова визначають, яку літеру потрібно підставляти.

Світовий досвід роботи із секретною інформацією довів, що

недоцільно зберігати у секреті функції шифрування та дешифрування, або алгоритм перетворення повідомлень. Доцільно зберігається лише ключ, а алгоритм перетворення повідомлень є загальновідомим. Оскільки ключ зловмиснику невідомий, він практично не може отримати відкритий текст повідомлення. Це твердження сформував Огюстом Керкгофсом у 1883 році.

Для перетворення повідомлення у вигляд, коли його зміст стане недоступним зловмиснику, можна використовувати окремий алгоритм. Зберігання алгоритму в секреті протягом тривалого часу є малоймовірним, тому краще використати певний параметр алгоритму.

Знання цього параметра дозволяє повернутися від зашифрованого тексту до відкритого, а його відсутність робить таке повернення практично неможливим. Цей параметр називають ключем. На рис.1.2 він показаний як аргумент ***K*** функцій шифрування E та дешифрування D.

Принцип Керкгоффса - надійність схеми шифрування не залежать від секретності алгоритмів шифрування та дешифрування, а має залежати лише від секретності ключа. Змінювати алгоритми складно. Вони вбудовуються в апаратне або програмне забезпечення, яке важко піддається оновленню.

Насправді той самий алгоритм використовується протягом досить багато часу. Простий ключ важко зберігати у секреті, а забезпечити ж секретність цілого алгоритму складніше та дорожче. Кожен учасник системи використовує той самий алгоритм. Зловмиснику достатньо лише отримати алгоритм від одного з них, а знайти одного незахищеного користувача серед мільйонів не так уже й важко.

Існує ще один аргумент на користь відкритості алгоритму. Досить зробити маленьку помилку – і створений криптографічний алгоритм виявиться зовсім непридатним. Якщо алгоритм закритий, ніхто не виявить цієї помилки, поки зловмисник не здійснить напад на систему і не скористається цією помилкою.

Створено було багато інструментів для допомоги в шифруванні.

Спратанцями використовувався перестановочний шифр, а в середньовіччя було винайдено дірковий шифр, найвідомішим з них був Енігма. Він використовувався до кінця Другої Світової війни з Німеччиною. Поява цифрових комп'ютерів дозволила отримувати ускладненні шифри.
Алгоритми працювали з бітами інформації, а не з цілими літерами. Саме поява комп'ютерів дозволило підняти рівень криптоаналізу. Сучасні алгоритми шифрування випереджають криптоаналіз, а це дає можливість безпечно передавати інформацію.

Академічне дослідження криптографії в 1970-х роках породило відкритий стандарт DES, публікацію алгоритму ДіффіХелмана та алгоритму RSA. Безпеку цих алгоритмів надають складні математичні обчислення розкладу цілих чисел, дискретними логарифмами.

Зростання обчислювальної потужності сучасних комп'ютерів, з'являється можливість перебрати всі можливі ключі, тобто провести атаку грубої сили . Це спонукає постійно збільшувати мінімальну довжину ключа необхідну для шифрування.

1.4 Типи атак .

Під атакою розуміють спробу отримати таємну інформацію несанкціонованим шляхом. Існує безліч типів атак, кожен з яких характеризується певними вихідними умовами

Криптографічні атаки здійснюється завдяки криптоаналізу. На основі криптоаналізу можна отримати певній рівень доступу до зашифрованої інформації,. Спочатку користувач використовує ключ і шифр завдяки яким перетворює текст у шифротесті. Наступним кроком це передача по не захищеному каналі зашифрованого тесту. І в подальшому інша сторона розшифровує інформацію завдяки ключу який у нього є для розшифрування.

Розглянемо ситуацію, коли користувачі А і В спілкуються, обмінюючись повідомленнями, які передаються за допомогою деякого каналу зв'язку. Канал спілкування прослуховується зловмисником, і кожне повідомлення, яке відправляється від А до В або від В до А, потрапляє і до зловмисника.

Ситуація з урахуванням можливості витоку інформації представлена - рис.1.1. Щоб зловмисник не зміг зрозуміти перехоплене повідомлення, застосовується схема шифрування.

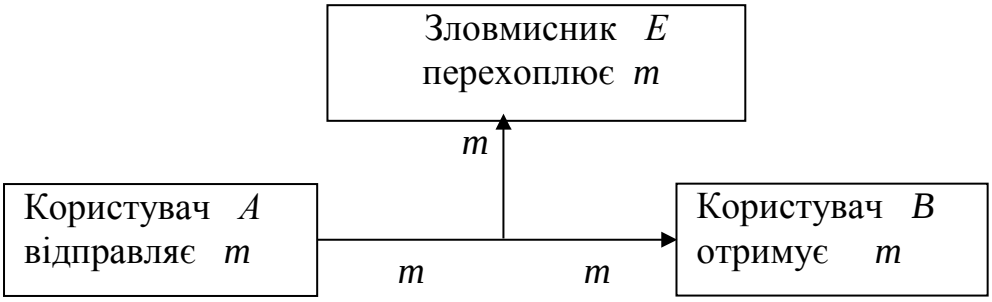

Рисунок 1.1.- Схема передачі даних *x* по відкритому каналу.

Шифрування - це зміна форми повідомлення з метою запобігання витоку інформації та із забезпеченням можливості повернення до вихідної форми повідомлення. Схема використання шифрування показано на рис. 1.2

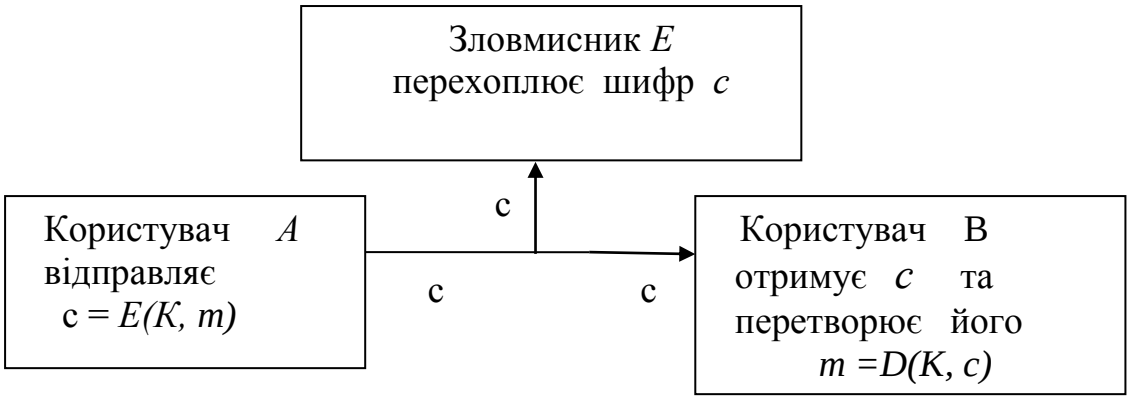

Рисунок 1.2.- Схема шифрування

Згідно з наведеною схемою, перед відправкою, повідомлення перетворюється на шифротекст ***c*** за допомогою функції шифрування *E(K, m)*. Крім вихідного повідомлення або відкритого тексту, параметр цієї функції є деяке число K, яке називається ключем. При тому самому повідомленні різним значенням ключа відповідають різні шифротексти. Для того, щоб від шифротексту перейти до відкритого тексту на приймальній стороні, застосовується функція дешифрування *D(K, c).*

Таким чином, щоб отримати відкритий текст треба знати ключ. Про застосування загального секретного (закритого) ключа користувачі А і В повинні домовитись заздалегідь або скористатися надійним каналом зв'язку для передачі ключа від одного користувача до іншого.

Подана схема шифрування може застосовуватися як обміну повідомленнями, але й зберігання даних. Зберігання інформації можна як передачу повідомлення над просторі, а часі, коли *А* і *У* одне й те обличчя.

Задача криптоаналітики дешифрувати інформацію і спробувати вивести ключ на основі якого можна буде розшифрувати всю подальшу інформацію. Сьогодні усі алгоритми є відкритими і публічними. Це припущення має назву принципом Керкгоффза.

1.4.1. Приклади моделей атак.

1. Атака на основі шифротексту (COA). Зловмисник має шифротекст і не має відкритого тексту. Модель атаки є найбільш популярною, але і найбільш слабкою. Зловмиснику, щоб провести атаку, потрібно знати на якій мові написаний відкритий тест та місце знаходження в потоці інформації.

2. Атака грубої сили – базується на повному переборі усіх можливих ключів. Дану атаку використовують для дослідження ефективності, порівняно з іншими атаками. Жоден алгоритм шифрування не в змозі встояти перед цією атакою. Щоб порівняти чи правильний ключ був підібраний, потрібно мати достатньо інформації про відкритий тест. Рекомендується мінімум N бітів інформації.

3. Атака з відкритим текстом (KPA). Зловмисник знає і відкритий, і зашифрований текст. Мета такої атаки – знайти ключ. Насправді існує багато ситуацій, коли зловмисник може отримати відкритий текст повідомлення. Зловмиснику потрібен фрагмент з зашифрованою інформацією і сама шифрограма. Саме даний спосіб був використаний в успішній операції Enigma.

4. Атака на основі підібраного відкритого тексту (CPA) - Криптоаналітику потрібно декілька текстів, які будуть шифруватися. Він

досліджує всі можливі стани та визначає закономірності даного алгоритму.

5. Атака на основі підібраного шифротексту (CCA) - Аналітику потрібно мати довільний зашифрований текст і доступ до розшифрованого тексту. Для цього йому потрібно мати доступ до повідомлень одержувача та до каналу зв'язку. Для кожного підібраного відкритого тексту зловмисник одержує відповідний шифрований текст, а для кожного шифрованого - відповідний відкритий текст.

6. Атака моделі відкритого ключа - Зловмисник знає і відкритий, і зашифрований текст. Мета такої атаки – знайти ключ.

7. Атаки, що ґрунтуються на парадоксі завдання про дні народження. Парадокс завдання про дні народження: якщо в кімнаті перебувають 23 особи, ймовірність того, що у двох із них збігається день народження приблизно 1/2. Це напрочуд дуже велика.

$$P \approx 1/2$$

У випадку, якщо елемент може приймати N різних значень, очікувати першу колізію можна після вибору приблизно $\sqrt{N}$ значень.

$$\sqrt{365} \approx 19.$$

Розглянемо процес послідовного обчислення $f(x_1), f(x_2), ..., f(x_k)$, де x_i - номер обчислення значення функції f, а значення функції f рівномірно

розподілені на множині {1, 2, ... , N }. Визначимо ймовірність того, що серед набутих значень усі значення різні

$$P\,[f(x_1) \neq f(x_2)] = (N - 1) / N.$$

Імовірність того, що значення $f(x_1), f(x_2), f(x_3)$ різні - $((N - 1) / N)\,(N - 2/ N.$

Аналогічно для випадку, коли всі k значень функції різні, отримуємо

$$P = ((N - 1) / N)\,((N - 2) / N) \ldots ((N - (k\text{-}1)) / N.)$$

При досить великому N та відносно малому x ($lim_{N\to\infty}(1 + 1/N)^N = e$

$$(1 + x / N)^N \approx e^x,$$

або

$$(1 + x / N) \approx e^{x/N}.$$

Тому

$$P \approx e^{-1/N}\, e^{-2/N} \ldots e^{-(k-1)/N} = e^{k(k-1)/2N}.$$

Наявність хоча б однієї колізії - протилежної події, і її ймовірність визначається як

$$P=1 - e^{k(k-1)/2N}.$$

Якщо позначити цю ймовірність як ε, отримуємо

$$e^{k(k-1)/2N} \approx 1 - \varepsilon,$$

або

$$k^2 - k \approx 2N(1/(1\varepsilon)).$$

Якщо $\varepsilon = 1/2$,

$$\underline{k} \approx \sqrt{2N\ ln\ (1/(1-\varepsilon))}.$$

$$k \approx 1{,}1774\sqrt{N} \approx \sqrt{N}.$$

Отже, при розгляді N-бітних чисел для очікування колізії доцільно перевіряти 2N/2 значень. Цей тип атаки заснований на тому факті, що однакові значення їх називають колізіями (collisions), випадкової величини в послідовності випробувань з'являються набагато частіше, ніж можна було б очікувати.

1.5. Модель інформаційної безпеки.

Інформаційна безпека включає 3 ключові властивості інформації, які відтворюють модель КЦД - конфіденційність, цілісність і доступність. CIA model: confidentiality, integrity and availability).

Напрямки інформаційної безпеки умовно можна відобразити на прикладі трьох вимірів куба::

- мета безпеки;
- стан інформації ;
- захисні заходи.

Кожен з цих пунктів має взаємозв'язок факторів, які обов'язково потрібно брати до уваги. Кожен з пунктів: Storage, Processing, Transmission, Technology, Policy and practices, Human factors, включає великау область знань.

Повна модель напрямків інформаційної беєпеки, під назвою «Куб МакКамбера», була розроблена у 1991 році.

Цей Куб складається з наступної інформації:

- зберігання, передача та обробка інформації;
- властивості інформації;
- характеристики інформації;
- інформаційна безпека (КЦД),
- підготовка і навчання, політики і практики, технологій контрзаходів безпеки.

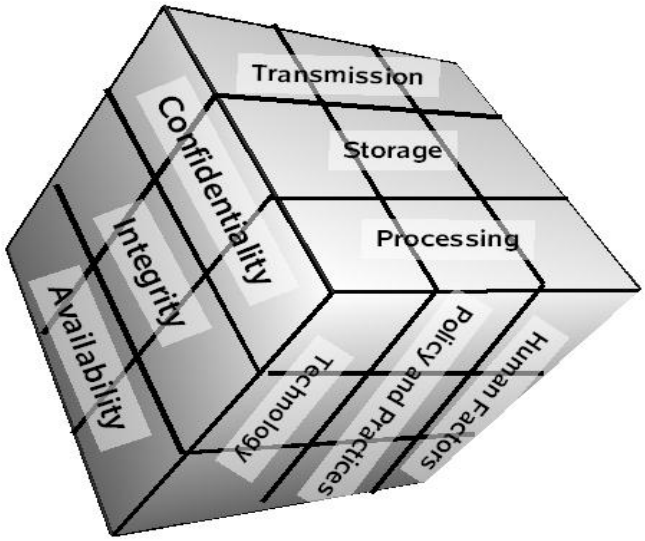

Рисунок 1.3. -Куб МакКамбер

Маконахі (Масопас'ю) в свою чергу створив модель куба «Сервіси безпеки» , де кількість значень цього виміру збільшилася і стала складатися з 3 * 5 * 3 = 45 осередків.

Модель куба Маккей-бера і Маконахі була покладені в основу еталонної моделі забезпечення збереження інформації і безпеки (RMIAS).

Складові моделі технологій кібербезпеки:

- Cryptography;
- Access control;
- System integrity;
- Audit and Monitoring;
- Management.

Однією з ефективних технологій захисту інформаційних ресурсів є застосування криптографічних методів

Криптографія (від грец. Κρυπτός - прихований і γράφω - пишу) - наука про методи забезпечення конфіденційності (неможливості прочитання інформації стороннім), цілісності даних (неможливості непомітного зміни інформації), автентифікація (перевірки справжності авторства), а також неможливості відмови від авторства

Висновки до першого розділу.

В даному розділі розглянуть теоретичні аспекти та основні поняття криптографічного захисту інформації.

Розглянуто та проаналізовано таки види атак: COA (на основі шифротексту); KPA (атака з відкритим текстом) ; CCA(на основі підібраного шифротексту (CCA). Алгоритм атаки, що ґрунтуються на парадоксі завдання про дні народження, заснований на факті, що однакові значення випадкової величини (collisions) з'являються набагато частіше, ніж ми очікуємо.

Модель інформаційної безпеки, складовими якої є 3 ключові властивості інформації: конфіденційність, цілісність і доступність, була покладена в основу еталонної моделі забезпечення збереження інформації і безпеки (RMIAS).

В зв’язку з постійним ростом продуктивності обчислювальних засобів необхідно постійно удосконалювати існуючі криптографічні схеми для збереження необхідного рівня стійкості. На сьогодні існує проблема автентифікації даних в інформаційно-телекомунікаційних системах.

РОЗДІЛ 2. АЛГОРИТМИ ТА МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ

2.1. Види алгоритмів шифрування.

Одним з розділів криптографії є наука про шифрування. Відомо, що шифрування є зміною форми повідомлення для запобігання витоку інформації та можливістю повернення до вихідної форми повідомлення.

Перед відправкою, повідомлення перетворюється на шифротекст . Для цього використовується функція шифрування. Параметр цієї функції є деяке число K, яке називається ключем. Різні шифротексти відповідають різним значенням ключа. Для переходу до відкритого тексту на приймальній стороні, застосовується функція дешифрування .

Задача криптоаналітики дешифрувати інформацію і спробувати вивести ключ, на основі якого можна буде розшифрувати всю подальшу інформацію.

Алгоритми для шифрування бувають двох видів: для симетричних шифрів та асиметричних.

Симетричні алгоритми можна поділити на дві групи:

1. Блокові шифри, - за один крок (одна ітерація) обробляється група елементів даних (наприклад, 64 біта).

2. Потокові шифри,- за кожен кроку обробляється один елемент даних (один байт інформації будь то літера чи біт).

Блоковий шифр на кожному етапі обробляє групу бітів відкритого тексту, яка називається блоком.

Найпоширенішими алгоритмами блокового шифрування є алгоритми DES та AES, а асиметричними системам шифрування алгоритм RSA, алгоритм Ель-Гамалю.

2.2. Режими застосування блокових алгоритмів шифрування.

1. Блоковий шифр дозволяє шифрування тільки одного блоку даних встановленої довжини. Для роботи з блоками різних довжин , дані потрібно розбити на окремі блоки довжиною, яку встановлює даний шифр Зазвичай, останній блок треба доповнити до відповідної довжини підхожим доповненням. Режими дій описує процес шифрування кожного з цих блоків і

звичайно використовують рандомізацію , засновану на додатковому значені на вході (ініціалізаційний вектор), з ціллю зробити це безпечно. довжина блоку в 64 біта є достатньою для забезпечення високої криптостійкості алгоритму

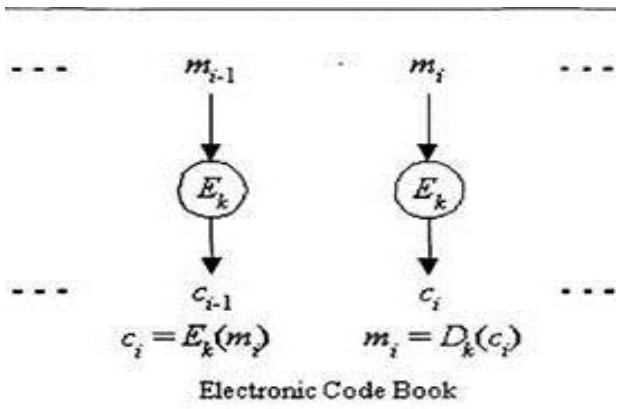

Рисунок 2.1-Шифрування блоків різної довжини

2. ECB (Electronic CodeBook mode)

Найбільш простим і очевидним режимом застосування алгоритму шифрування є шифрування блоків відкритого тексту кожного окремо. Такий режим називається ECB (Electronic CodeBook mode - "режим електронної шифрувальної книги"). Недоліком є той, що однаковим блокам відкритого тексту відповідають однакові блоки зашифрованого тексту.

Найбільш розповсюджений режим застосування алгоритму шифрування має назву CBC (CipherBlock Chaining - "зчеплення зашифрованих блоків"). В цьому режимі до кожного з блоків відкритого тексту, перед шифруванням, за модулем два додається попередній зашифрований блок. До першого блока відкритого тексту за модулем два додається випадковий блок, вектор ініціалізації, який передається у відкритому вигляді разом із зашифрованим текстом.

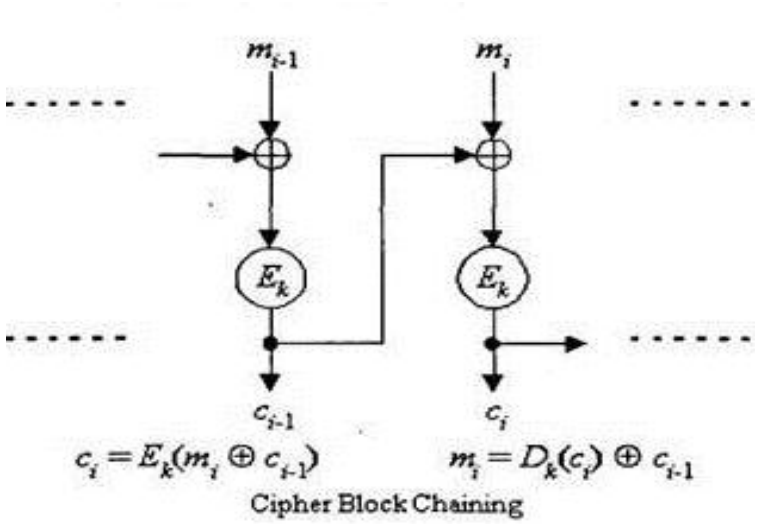

Рисунок 2.2 - Шифрування методом «зчеплення зашифрованих блоків»

3. CFB (Cipher FeedBack)

Третім режимом застосування алгоритму шифрування є CFB (Cipher FeedBack - "шифрування зі зворотним зв'язком"). В цьому алгоритмі кожний блок зашифрованого тексту утворюється шляхом додавання до блока відкритого тексту за модулем два результату шифрування попередньо створеного блока. Вектор ініціалізації використовується в якості нульового зашифрованого блока.

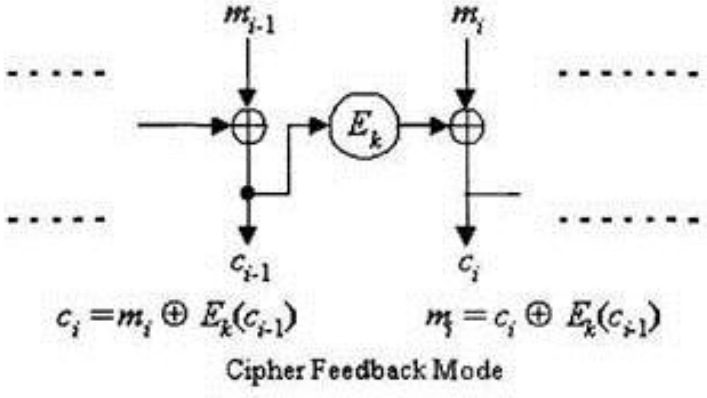

Рисунок 2.3 - Шифрування зі зворотним зв'язком

2.3. Симетричні блокові шифри.

2.3.1 Алгоритм шифрування Г. Файстеля

Алгоритм шифрування, запропонований американським дослідником Г. Файстеля, відноситься до симетричних блокових шифрів.

Шифрування за цим алгоритмом є ітераційною процедурою. Ітерації в цих алгоритмах називають раундами.

Блоковий шифр за один крок - блок, обробляє групу бітів відкритого тексту. Довжина блоку обирається 64, 128 або більше бітів.

Блок тексту розбивається на дві частини з однаковою кількістю бітів. Ліву частину позначимо $l_{0,,}$ а праву r_0. Права частина r_i-$_1$ початкового стану блоку переноситься в ліву частину l_i результату поточного раунду. Права частина r_{i-1} перетворюється функцією F. Значення F залежить від r_{i-1} та ki - ключа раунду. Функція F , після виконання цих дій, буде мати кількість бітів половини блоку. Далі відбувається поразрядне підсумовування по модулю 2 з лівою частиною l_{i-1}. Сума результату раунду утворює праву частину r_i.

Початковим станом блоку для раунду $i + 1$-го, $i = 1, ..., s$ буде саме результат і-го раунду.

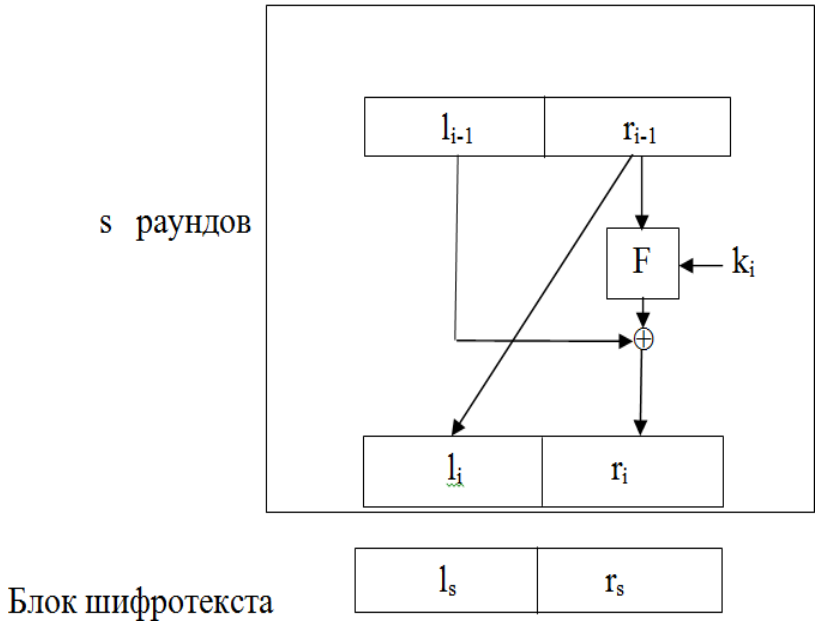

Рисунок 2.4.- Блок-схема одного раунду алгоритму Г. Файстеля.

Особливість представленого алгоритму у тому, що функція раунду має зворотну незалежно від цього, яка функція F застосовується у середині раунду. Кожен ***i***--й раунд шифрування здійснюється згідно з правилами

$$l_i = r_{i-1},\ r_i = l_{i-1} \oplus F(k_i, r_{i-1}).$$

Тому розшифрування відбувається згідно з перетвореннями

$$r_{i-1} = l_i,\ l_{i-1} = r_i \oplus F(k_i, l_i)$$

Наведені формули показують, що для розшифрування треба застосовувати самі ключі раундів, що і при шифруванні, але в зворотному порядку. При цьому алгоритм розшифрування збігається з алгоритмом шифрування, якщо на вхід подати шифр, в якому переставлено ліву та праву половини (r_s, l_s).

Після останнього раунду дешифрування, щоб отримати відкритий текст, треба поміняти місцями ліву та праву частини блоку. Схема роботи алгоритму шифрування та дешифрування складається із чотирьох етапів:

- виконати s раундів шифрування;
- зробити обмін місцями лівої та правої половин блоку;
- виконати знову s раундів шифрування;
- обміняти місцями ліву та праву половини блоку.

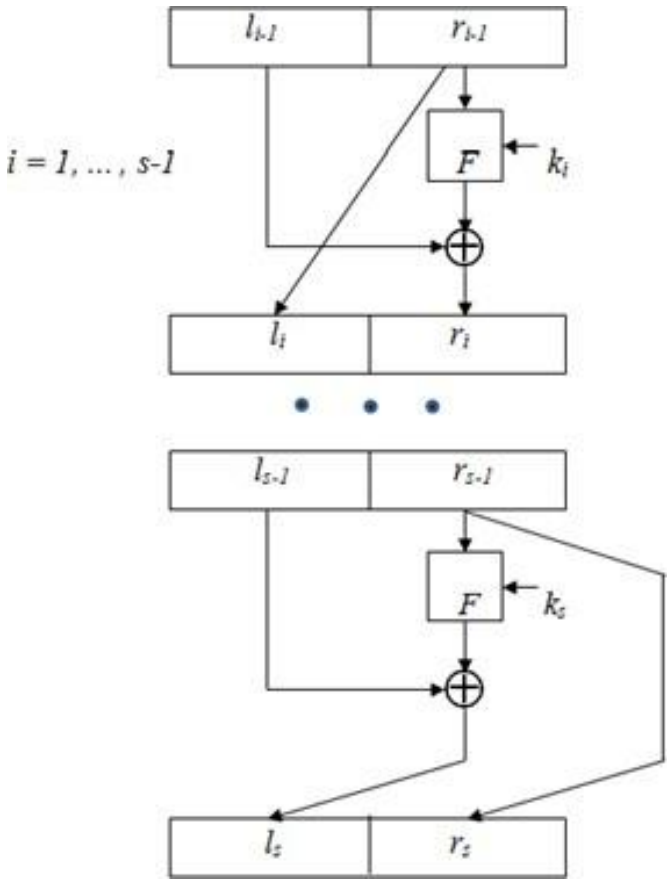

Рисунок 2.5 - Схема шифрування /дешифрування Г. Файстеля.

2.3.2 Алгоритм блокового шифрування *DES*

Алгоритм DES (Data Encryption Standard) складається з 16 раундів, довжина блоку – 64 біта, довжина загального ключа – 56 біт, а довжина кожного ключа раунду – 48 біт. Замість DES використовують 3DES, коли довжина ключа 56 біт може виявитися недостатньою. 3DES є послідовним виконанням трьох алгоритмів DES з різними ключами.

Алгоритм DES шифрування 64-бітного блоку складається з трьох кроків:

1) початкова перестановка бітів блоку згідно з фіксованою таблицею перестановок;

2) виконання 16 раундів алгоритму Файстеля до отриманого блоку;

3) перестановка зворотна до вихідної

Початкова та кінцева перестановки представлені відповідно таблицями. Кожен елемент таблиці дорівнює номеру біта у вхідному блоці (до перестановки). Місце елемента таблиці визначає розташування відповідного біта після перестановки. При цьому елементи першого рядка відповідають розташуванню бітів на місцях з першого до восьмого, другого - з 9-го по шістнадцяте і т.д.

Наприклад, табл. 2.1 вказує, що перше місце результаті перестановки ставиться 58-й біт, але в друге - 50-й.

Таблиця 2.1

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Тому табл.2.2 показує, що у п'ятдесят восьме місце переставляється 1-й біт, але в п'ятдесяте – другий.

Таблиця 2.2

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Тепер розглянемо функцію F, яка отримує два аргументи: r_{i-1} – молодші 32 біти поточного стану блоку та ключ раунду k_i. Функція F послідовно виконує п'ять перетворень: 1) перестановка із розширенням; 2) обчислення суми за модулем 2 результату попереднього перетворення та ключа раунду; 3) розщеплення результату попереднього перетворення -

отримання восьми груп бітів, по 6 біт у кожній; 4) стиск кожної групи до 4-бітного коду; 5) утворення 32-бітного коду на виході *F*.

У першому перетворенні 32-бітний код розтягується до 48 біт і перемішується згідно з таблицею 2.3, зміст елементів якої визначається так само, як у попередніх таблицях.

Таблиця 2.3

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Це допомагає розсіювати зв'язки між вхідними бітами та вихідними, тобто створює лавинний ефект, коли невеликі відмінності між двома вхідними наборами даних призводять до більших відмінностей на виході. Після порозрядного додавання по модулю 2 ключа раунду та отриманого 48-бітного коду (друге перетворення) у сумі послідовно виділяються групи по 6 біт: біти з першого по шостий утворюють першу групу, з сьомого по дванадцятий - другу і т. д. Остання, восьма група утворюється бітами з 43-го до 48-го (третє перетворення). Кожна 6-бітна група перетворюється на 4-бітну за допомогою окремої таблиці, яка називається S-блоком. Усього використовується вісім S-блоків (рис. 2.3). Перший та шостий біти групи визначають номер рядка S-блоку, а біти з другого по п'ятий – номер стовпця. Елемент S-блоку, розташований на перетині певних рядки та стовпця, у двійковому коді дорівнює вихідному 4-бітному значенню (четверте перетворення).

П'яте перетворення здійснюється перестановкою бітів у 32-бітному коді, який утворений послідовним записом виходів S-блоків: молодші чотири біти - вихід S_1, біти з 5-го по 8-й - вихід S_2 і т. д. Старші чотири біти - вихід S_8 . Перестановка здійснюється відповідно до табл. 2.4.

S-блок 1.	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S-блок 2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S-блок 3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S-блок 4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S-блок 5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S-блок 6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
cS-блок 7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S-блок 8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Рисунок. 2.6.- S-блоки

Для визначення алгоритму треба розглянути формування ключів раундів із загального ключа. Загальний ключ представляється 64-бітним числом, у якому кожен восьмий біт є контрольним

Значення контрольних розрядів визначаються таким чином, щоб кількість одиничних бітів у кожному байті була непарною. Зокрема, значення

8-го біта визначається так, щоб кількість бітів дорівнює одиниці, у групі з першого біта по восьму було непарним. Біт 16 відповідає групі з 9-го по 16 біти і т. д

Таблиця 2.4

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Біт 64 відповідає групі з 57 по 64 біти. Перетворення загального ключа на ключі раундів складається з попередньої процедури вилучення контрольних бітів і перестановки інших, і навіть ітераційної процедури, виконуваної кожного раунду. Попередня процедура здійснюється відповідно до табл. 2.5.

Таблиця 2.5

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Результат цієї перестановки називається у літературі *PC-1*. Він ділиться на дві половини по 28 біт: ліва позначається C_0, а права - D_0.

Для кожного *i*-го раунду обчислюються

$$C_i = C_{i-1} <<< p_i ; \quad D_i = D_{i-1} <<< p_i$$

де $A <<< p$ означає циклічний зсув числа A ліворуч на p розрядів; $p_i = 1$ при $i = 1, 2, 3, 9, 16$ та $p_i = 2$ для інших номерів раундів. Отримані C_i *та* D_i

об'єднуються в одне число (C_i – старші біти, D_i – молодші), і це число перетворюється (стиснення та перестановка) відповідно до табл. 2.6.

Таблиця 2.6

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

2.3.3 Алгоритм хешування BCrytp

Односторонній алгоритм шифрування називають хешуванням. BCrypt - це адаптивна функція хешування паролів. Розробниками цього алгоритму були Нільсом Провосом і Девідом Мазьєром . Їх розробка була представлена на USENIX у 1999р. Так як BCrypt є адаптивною функцією, то з часом кількість ітерацій можна збільшити, щоб зробити її повільнішою. Метод дає змогу залишатися стійким до атак грубої сили при збільшенні обчислювальної потужності.

Цей алгоритм відтворює блочне симетричне шифрування зі змінною довжини ключа на основі функції Фейстеля. Алгоритм складається з розширення ключа та шифрування даних. В основі алгоритму шифрування лежить шифр Blowfish. Після хешування практично неможливо відновити початковий вигляд даних. Алгоритм спочатку солить фрагмент тексту, а потім хешує його до стрічки довжиною 60 символів. Діапазон довжини ключа від 32 до 448 біт і блок у 64 біти.

Blowfish представив у 1993 р. Брюс Шнайєр. Він є один з найбільш провідних криптологів. Більшість частин шифрування алгоритму захищені патентом. Blowfish непатентований , є вільною доступною альтернативою серед існуючих алгоритмів шифрування.

Алгоритм Blowfish застосовує 16 раундів для процесу шифрування. Цей алгоритм також ефективно використовується в апаратному забезпеченні VLSI і може бути використаним в програмних додатках. На рисунку 2.4 показана процедура шифрування Blowfish.

Кожен рядок це 32 біти, п’ять масивів підрозділів. Один P-масив із 18 записів та чотири S-поля з 256 записів (S0, S1, S2 та S3).

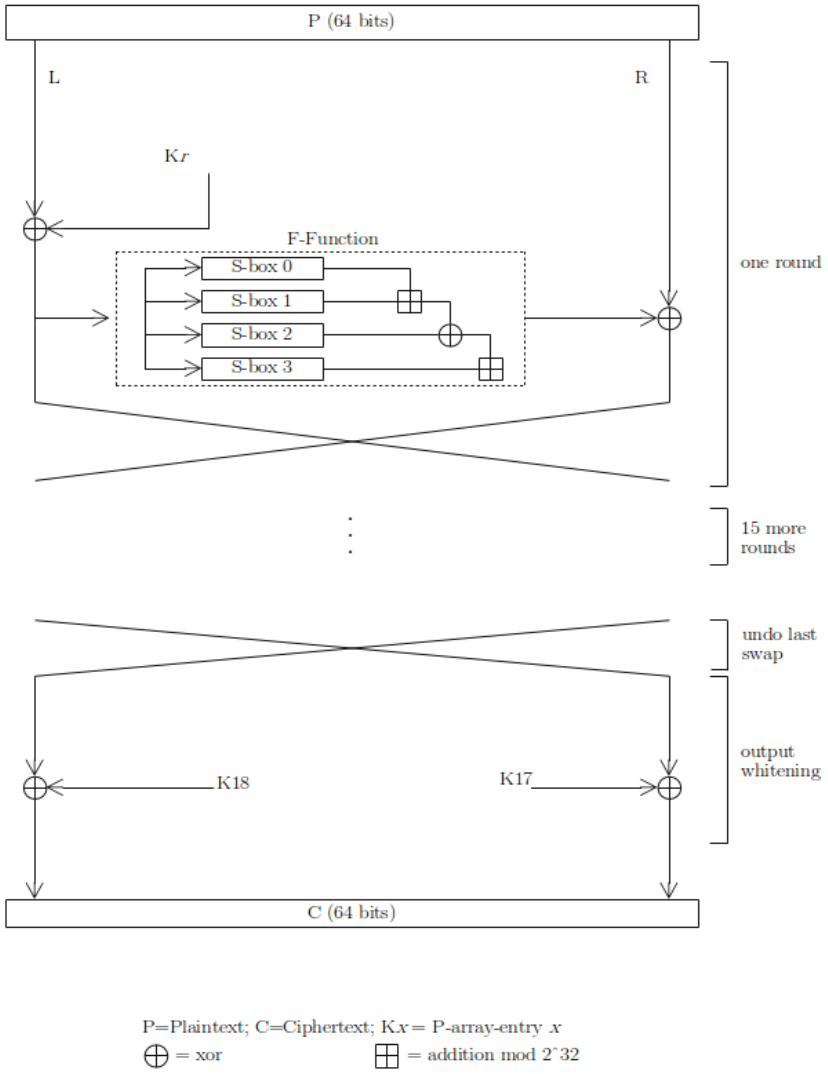

Рисунок 2.7. – Алгоритм шифрування

Кожен раунд *r*, а їх буде 16, r складається з 4 дій:

- XOR - ліва половина (L) даних із введенням r-го P-масиву
- дані XORed (як вхідні дані) для F-функції Blowfish
- XOR вихід функції F з правою половиною (R) дани
- Поміняйте місцями L і R

S-коробки приймають 8-бітний вхід і виробляють 32-бітний вихід. Виходи додаються за модулем 232 та XORed для отримання кінцевого 32-бітного виводу. Дешифрування виконується так само як і шифрування, тільки P1, P2, …, P18 використовуються у протилежному порядку

Після 16-го раунду скасовуємо останній обмін, а XOR L з K18 і R з K17.

BCrypt здатний поєднувати етап налаштування ключа Blowfish із змінною кількістю ітерацій для збільшення тривалості обчислень хешу.

2.3.4. Алгоритм шифрування Blowfish

Криптографічний алгоритм Blowfish реалізовує блочне симетричне шифрування зі змінною довжини ключа. Алгоритм складається з розширення ключа та шифрування даних.

Шифр Blowfish - це швидкий блоковий шифр. Якщо треба зробити зміну ключів і параметрів, то потрібна попередня обробка, яка еквівалентна шифруванню близько 4 кілобайт тексту. Ця зміна ключів корисна для методів хешування паролів.

BCrypt здатний пом'якшити атаки . Перевагою bcrypt є кількість ітерацій, яку можна збільшити, щоб зробити її повільнішою. Це дозволяє bcrypt масштабуватися з обчислювальною потужністю

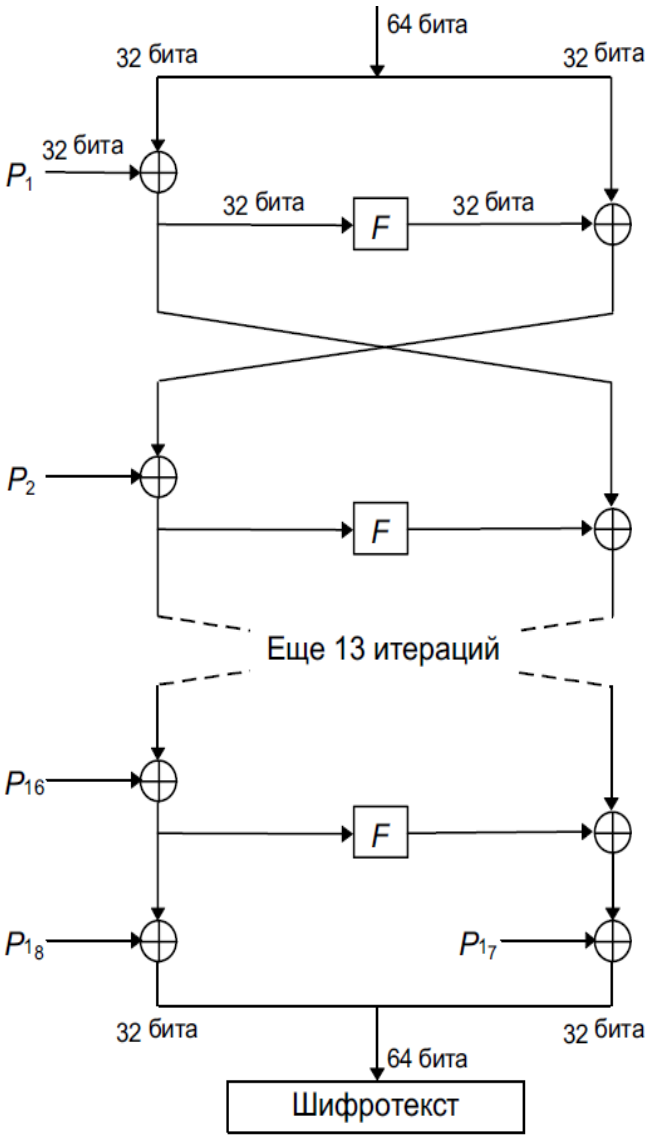

Рисунок 2.8 - Алгоритм блочного ссиметричного шифрування Blowfish

Шифрування складається з простої функції, як виконується послідовно 16 разів. В даному алгоритмі використовується багато підключів, які повинні бути розрахованими до початку шифрування або дешифрування.

P - масив складається з 18 32-бітових підкдючів: P1, P2, …, P18. Кожний із чотирьох S-блоків містить 256 елементів.

$S_{1,0}$, $S_{1,1}$, …, $S_{1,255}$

$S_{2,0}$, $S_{2,1}$, …, $S_{2,255}$

$S_{3,0}$, $S_{3,1}$, …, $S_{3,255}$

$S_{4,0}$, $S_{4,1}$, …, $S_{4,255}$

На вхід подається 64-бітовий елемент даних X.

Спочатку потрібно для шифрування X розбити на дві 32-бітові половини: X_L і $X_{R.,}$ а потім для i = 1 по 16:

$x_L = x_L \oplus P_{18}$

$x_R = F(x_L) \oplus x_R$

Далі потрібно переставити місцями X_L і X_R та зробити об'єднання X_L і X_R.

На рис. 2.9 зображено представлення функції F.

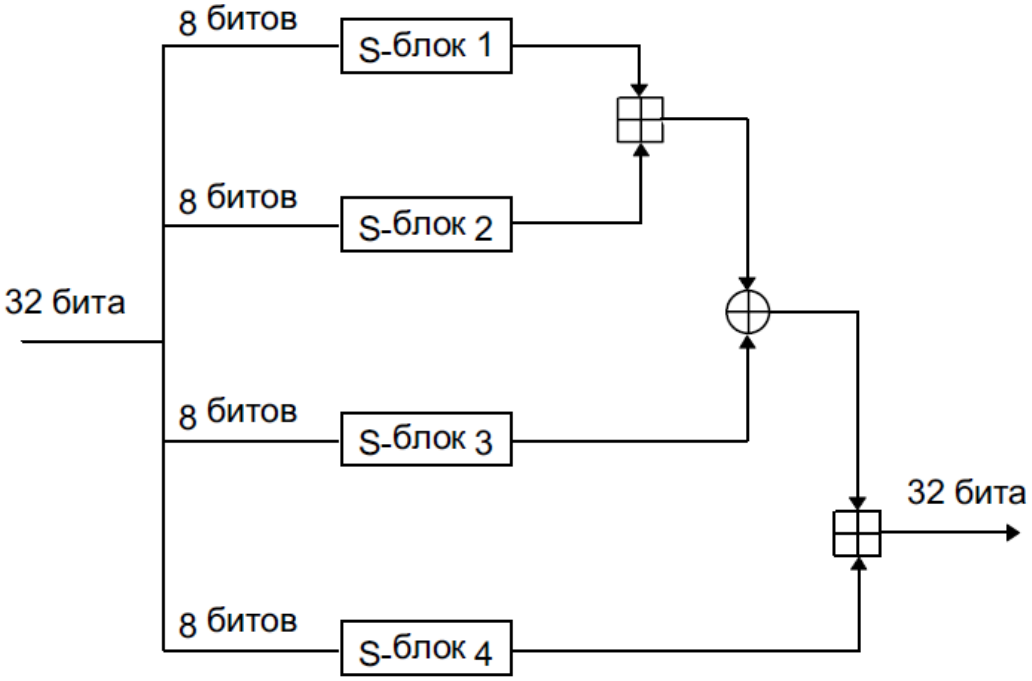

Рисунок 2.9 - Функція F

Далі потрібно розділити X_L на чотири 8-бітових частини (a, b, c, d – дані для S-блоків):

$$F(x_L) = \left(\left(s_{1,a} + s_{2,b} \bmod 2^{32}\right) \oplus s_{3,c}\right) + s_{4,d} \bmod 2^{32}$$

Дешифрування виконується так само як і шифрування, тільки P1, P2, …, P18 використовуються у протилежному порядку.

Blowfish не потребує ліцензії, не патентований. Алгоритм знаходиться у вільному доступі і слугує альтернативою серед існуючих алгоритмів шифрування. Він використовує діапазон довжини ключа від 32 до 448 біт і блок у 64 біти. Алгоритм Blowfish застосовує 16 раундів для процесу

шифрування і використовує діапазон довжини ключа від 32 до 448 біт і блок у 64 біт.

2.4. Асиметричними системам шифрування

2.4.1. Шифрування в системі *RSA*

Представником асиметричного алгоритма шифрування є алгоритм Ривеста-Шаміра-Адлемана (Rivest-Shamir-Adleman - RSA), який був винайдений в 1978 році авторами Rivest, Shamir, Adleman і отримав свою назву по першим літерам прізвищ винахідників RSA.

В алгоритмі RSA використовують односторонню функцію зведення у ступінь за відомим модулем. При відкритих значеннях *n* і *e* для отриманого повідомлення можна обчислити $m^e \pmod n$ - шифр повідомлення. Відновити повідомлення без додаткової інформації практично неможливо. У той же час, якщо відоме розкладання *n* на множники, останнє завдання стає розв'язним.

Модуль системи *N,* це основний параметр алгоритму RSA. По цьому параметру алгоритму проводяться усі обчислення в системи, а $N = R * S$ (R та S – секретні випадкові прості великі числа, зазвичай однакового розміру).

Випадковим чином обирається секретний ключ *k2,* який повинен відповідати умовам: $1 < k2 < F(N)$ *та НОД* $(k2, F(N)) = 1$, де *НОД* – це найбільший спільний дільник. Ключ k1 є відкритим ключем, який обчислюється із співвідношення $(k2 * k1) = 1$ та $mod\ F(N)$. Якщо здобути значення числа *k1,* то зловмисник зможе обчислити і секретний ключ *k2*. Розмір *k1* має досягти дуже великого цисла- до *2048* біт..

Алгоритм RSA складається з наступних етапів:

- генерації ключів;
- шифрування;
- розшифрування;
- розповсюдження ключів.

На рисунку 2.10 представлена схема шифрування за RSA.

Рисунок 2.10 .- Схема шифрування за RSA.

Алгоритм використовує два ключі — відкритий (public) і секретний (private), разом відкритий і відповідний йому секретний ключі утворюють пари ключів (keypair).

Для того, щоб згенерувати пари ключів виконуються такі дії:

1.Вибираються два великі прості числа p і q приблизно 512 біт завдовжки кожне.

2.Обчислюється їх добуток $n = p\ q$

3.Обчислюється функція Ейлера $\varphi(n) = (p-1)(q-1)$

4.Вибирається ціле число **e** таке, що $1 < e < \varphi(n)$ та **e** взаємно просте з $\varphi(n)$

За допомогою розширеного алгоритму Евкліда знаходиться число d таке, що $ed \equiv 1 \pmod{\varphi(n)}$.

Число n - модуль, а числа e і d — відкритою й секретною експонентами (encryption and decryption exponents), відповідно. Пари чисел (n, e) є відкритою частиною ключа, а (n, d) — секретною. Числа p і q можуть бути знищені (після генерації пари ключів), але, в жодному разі, не повинні бути розкриті.

2.5. Системи з відкритим ключем.

2.5.1. Криптосистема Ель-Гамалю

Криптосистема Ель-Гамалю – система з відкритим ключем. Вона використовується як для шифрування, так і для цифрового підпису. Система ґрунтується на складності обчислення дискретного логарифму в кінцевому полі за модулем простого числа. Безліч параметрів системи включає просте число p і ціле число g, таке, що $1 < g < p$ і ступеня числа g породжують всі елементи поля Z_p (g називають примітивним елементом, або початковим коренем поля Z_p).

Користувач A має секретний ключ x і відкритий ключ $y = g^x \bmod p$. Якщо користувач B бажає надіслати повідомлення користувачу A, він вибирає випадкове число k, таке, що $1 < k < p-1$, і обчислює $y_1 = g^k \bmod p$ і $y_2 = y^k \bmod p \oplus m$. Після цього B відправляє пару (y_1, y_2).

Отримавши таке повідомлення, користувач A може обчислити відкритий текст.

$y_1^x \pmod p \oplus y_2 = g^{kx} \pmod p \oplus y^k \pmod p \oplus m = g^{kx} \bmod p \oplus g^{kx} \bmod p \oplus m = m.$

В цьому алгоритмі часто замість операції «Виключающєє - АБО» застосовують множення за модулем p. Це дозволяє зняти обмеження на довжину повідомлення, яка повинна дорівнювати довжині $y^k \bmod p$ при бітовому додаванні по модулю 2.

Алгоритм виглядає наступним чином:

Генерація ключів.

1. Генерується випадкове просте число p довжини n біт.
2. Обирається початковий корінь g (довільний) за модулем p.
3. Випадкове ціле x, повинно відповідати нерівності $1 < x < p-1$.

Це буде секретний ключ.

4. Обчислюється $y = g^x \bmod p$.
5. Відкритим ключем є трійка (p, g, y), а закритим – x.

Шифрування.

1. Обираємо ключ сеансу k, $1 < k < p$.

2. Обчислюються числа $y_1 = g^k \bmod p$ та $y_2 = y^k m \bmod p$.

3. Пара чисел (y_1, y_2) створює шифрований текст.

Дешифрування.

$m = y_2 (y_1^x)^{-1} \bmod p$.

Обґрунтування: $(y_1^x)^{-1} = g^{-kx} \bmod p$, $y_2 (y_1^x)^{-1} = g^{kx} m g^{-kx} \bmod p = m$.

Розглянемо приклад.

Шифрування.

1. Візьмемо $m = 5$.

2. Генерація ключів:

а) Нехай, $p = 11$, $g = 2$ ($g^2 = 4 \bmod 11$, $g^3 = 8 \bmod 11$, $g^4 = 5 \bmod 11$, $g^5 = 10 \bmod 11$, $g^6 = 9 \bmod 11$, $g^7 = 7 \bmod 11$, $g^8 = 3 \bmod 11$, $g^9 = 6 \bmod 11$, $g^{10} = 1 \bmod 11$);

б) оберемо $x = 8$ та обчислимо $y = g^x \bmod p = 28 \bmod 11 = 3$;

в) відкритий ключ (11, 2, 3), закритий – 8.

3. Нехай $k = 9$ (довільне значення) .

4. Обчислюємо $y_1 = g^k \bmod p = 2^9 \bmod 11 = 6$.

5. Обчислюємо $y_2 = y^k m \bmod p = 3^9 5 \bmod 11 = (2^8)^9 5 \bmod 11 = 2_{72} 5 \bmod 11 = 2^{70} + 2\ 5 \bmod 11 = 20 \bmod 11 = 9$.

6. Отримуємо шифрований текст – (6, 9) .

Дешифрування:

$m = y2\ (y1^x)\text{-}1 \bmod p = 9\ (6^8)^{-1} \bmod 11 = 9\ (2^{72})^{-1} \bmod 11 = 9\ (2^2)^{-1} \bmod 11 = 9\ (2^8) \bmod 11 = (9 \times 3) \bmod 11 = 5$.

2.5.2. Цифрова підпис Ель-Гамалю.

Число $1 < x < p - 1$ буде закритим ключем. Відкритий ключ $y = g^x \bmod p$.

Розглянемо спосіб підписання.

Аліса генерує випадкове число $1 < l < p - 1$, таке, що НОД$(l, p - 1) = 1$, і підраховує два числа:

$$r = g^l \bmod p,$$

$$s = l^{-1} (m - xr) \bmod (p - 1),$$

які становлять підпис.

Перевірка підпису зводиться до перевірки рівняння

$$y^r r^s = g^m \pmod p.$$

Обгрунтування : $y^r r^s = g^{xr} (g^l)^{(1/l)(m - xr) \bmod (p-1)} \bmod p = g^{xr} g^{(m - xr) \bmod (p-1)} \pmod p = g^m \pmod p$.

Розглянемо на прикладі.

Якщо обрати $p = 13$, $g = 2$, $x = 7$, то відкритий ключ - $y = g^x \bmod p = 2^7 \bmod 13 = 11$.

Аліса генерує випадкове число $l = 5$.

Для підписання повідомлення $m = 10$ підраховує два числа:

$$r = g^l \bmod p = 2^5 \bmod 13 = 6,$$

$s = l^{-1} (m - xr) \bmod (p - 1) = 5^{-1} (10 - 7 * 6) \bmod 12 = 5 * 4) \bmod 12 = 8.$

Отримуємо пару чисел (6, 8), які і є підписом .

Перевірка підпису зводиться до перевірки рівняння

$$y^r r^s = g^m \pmod p.$$

$$y^r r^s = (11^6 * 6^8) \bmod 13 = (12 * 3) \bmod 13 = 10.$$

$$g^m \pmod p = 2^{10} \bmod 13 = 10.$$

Висновки до другого розділу.

В другому розділі розглядаються функції шифрування та дешифрування, види шифрування , режими застосування блокових алгоритмів шифрування.

В розділі описані базові алгоритми криптосистем та електронно-цифрового підпису, а також аналіз рішень, які існують на даний момент. Представлений алгоритми блокового симетричного Файстеля та DES. Для адаптивна хешування паролів використовують адаптивну функцію BCrypt з шифром Blowfish. В Crypt здатний поєднувати етап налаштування ключа Blowfish із змінною кількістю ітерацій для збільшення тривалості обчислень хешу.

Кожен з цих алгоритмів має свої переваги та недоліки залежно від конкретних вимог та сценаріїв використання. RSA вважається більш стійким до атак на основі факторизації чисел, в той час як Ель-Гамаль може бути вразливим до атак на дискретний логарифм. Шифрування та розшифрування RSA зазвичай є швидшими, ніж Ель-Гамаля, особливо при використанні великих чисел. Однак, Ель-Гамаля може бути швидшим для підписування повідомлень.

Алгоритм Ель-Гамаля має свої унікальні характеристики щодо безпеки, ефективності та використання у телекомунікаційних системах, порівняно з іншими алгоритмами шифрування та електронних підписів.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ РОЗРОБКИ СИСТЕМИ

3.1. Розробка сценарію системи.

На основі проведеного аналізу методів та алгоритмів шифрування та аналізу ключів, було обрано рішення розробити систему автентифікації даних користувача. Ця підсистема може використовуватись як в особистому використанні, так і в інформаційно-телекомунікаційної системи .

Обираємо клієнт - серверну технологію , в якій кожна з частин системи відповідає за свій розділ. Клієнтська частина системи буде відповідати за підписання даних (файлів), а серверна частина – відповідає за перевірку даних.

1. Реалізація серверної частини . На серверах, що будуть використані в роботі системи, потрібно установити відповідне серверне програмне забезпечення. Якщо система буде працювати в обраній інформаційно-телекомунікаційній системі, то не потрібно проводити установку та налаштування. Потрібно буде просто запустити відповідне програмне забезпечення , а далі програма все встановить самостійно. Якщо система буде працювати в іншій системі , то необхідно буде створити базу даних з усіма необхідними таблицями та зі списком користувачів системи та даними про них.

В базу даних сервера, після встановлення ПЗ, записуються дані про користувачів і підсистема чекає, коли клієнти почнуть підключатися та проходити авторизацію.

Оператор сервера може коригувати дані користувачів системи. Інтерфейс програми дає змогу це зробити , обравши відповідний пункт меню: Верхнє меню --> Управління -->обравши необхідний пункт у спадному меню(рис.3.1). Після авторизації користувач створює пару ключів та надсилає на сервер публічний ключ, який він зберігає його у себе та використовує в процесі автентифікації.

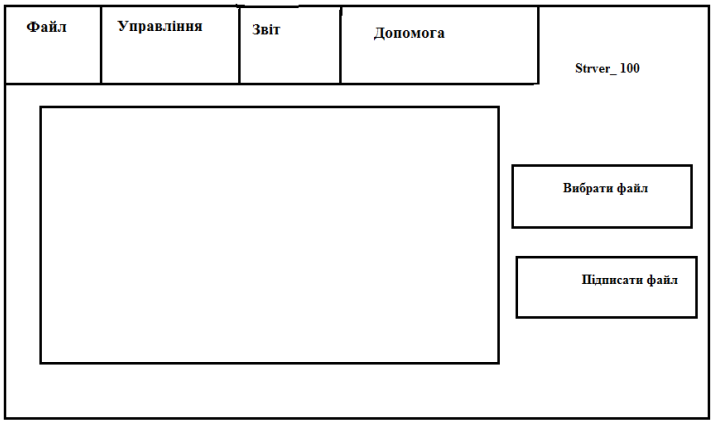

Рисунок 3.1.- Рисунок форми Інтерфейсу

Для ручної перевірки даних, використовуємо пункт меню інтерфейсу : Верхнє меню --> Файли --> Обрати дані -->Перевірити дані. Після перевірки всіх даних, система відправляє звіт користувачу з результатами перевірки

2. Реалізація клієнтської частини.

Клієнтські комп'ютери повинні мати клієнтське ПЗ. Для початку роботи користувачу, при першому запуску, необхідно ввести свої дані для авторизації в підсистемі: логін та пароль рис 3.2 .

Рисунок 3.2 - Форма реєстрації

Форма авторизації однакові для користувача та оператора сервера.

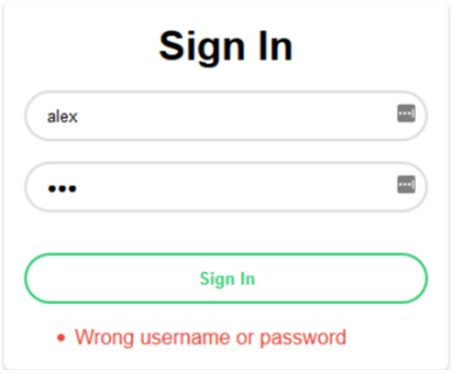

Рисунок 3.3 - Форма авторизації

Після авторизації користувач отримує доступ до інтерфейсу. Інтерфейс користувача дещо схожий на інтерфейс оператора сервера та має всі необхідні функції для успішної взаємодії користувача з підсистемою. Після авторизації користувача в підсистемі необхідно створити пару ключів. Це можна зробити вибравши: Верхнє меню --> Управління --> Створити пару ключів. Підсистема перевірить чи немає у цього користувача існуючої пари ключів, якщо ні – почне працювати алгоритм генерації публічного і приватного ключів. Ключі створюються на рік чи півроку, в залежності від типу користувача в системі.

В будь-який момент можна обновити пару ключів, вибравши варіант в меню: Верхнє меню --> Управління --> Обновити пару ключів. За звичай ключі оновлюють у разі:

- втрати ключа ;
- підозри про заволодіння ключем третьою особою;
- пошкодження ключа ;
- сплив термін дії ключа .

Підсистема автоматично підписує дані, які користувач збирається відправляти на сервер. Якщо необхідно вручну підписати дані, то це можна зробити вибравши: Верхнє меню --> Файли --> Обрати дані, а потім після вибору необхідних даних натиснути Підписати дані. Також для підписання даних можна скористатися кнопками швидкого доступу для

вибору та підписання даних.

3.2. Розробка бази даних

База даних складається з основних таблиць, які використовує сервер та ще додаткових таблиць. Кількість додаткових таблиць залежить від кількості користувачів даного сервера.

3.2.1. Опис прецидентів сценарію.

Далі проводиться опис всіх прецидентів сценарію використання підсистеми у вигляді таблиць.

Опис сценарію використання підсистеми «Авторизація користувача в підсистемі» представлений в таблиці 3.1

Таблиця 3.1 – Авторизація користувача

Назва	Авторизація користувача в підсистемі
ID	1
Опис	Користувач при першому запуску підсистеми повинен пройти процедуру авторизації
Актори	Користувач
Частота використання	Низька
Тригери	Натискання на кнопку авторизації
Передумова	Користувач не авторизований; Наявність на екрані форми авторизації
Постумова	Користувач отримує доступ до підсистеми
Головний курс	1. Після першого запуску підсистеми відображається форма авторизації 2. Користувач водить свій ID та пароль в відповідні поля 3. Користувач натискає кнопку авторизації 4. Перевірка ведених даних 5. Користувач входить в систему
Альтернативний курс	1. Після першого запуску підсистеми відображається форма авторизації 2. Користувач водить свій ID та пароль в відповідні поля 3. Користувач натискає кнопку авторизації 4. Дані введені не відповідають тим, які зберігаються в базі даних 5. Виводиться повідомлення про неправильно введені данні
Винятки	Користувач не правильно ввів данні чи взагалі не має доступу до підсистеми

Опис сценарію використання підсистеми «Управляння користувачами» представлений в таблиці 3.2

Таблиця 3.2- Управляння користувачами

Назва	Управляння користувачами
ID	2
Опис	Оператор серверу може створювати та видаляти користувачів підсистеми
Актори	Оператор
Частота використання	Середня
Тригери	Натискання кнопки для управління користувачами підсистеми
Передумова	У сервера повинен бути користувач, який уже авторизований чи який пройшов процедуру авторизації в підсистемі
Постумова	Доданий чи заблокований користувач
Головний курс	1. Оператор серверу натискає кнопку додати дані про користувача 2. Оператор вводить необхідні дані 3. Програма перевіряє чи є такий користувач в підсистемі 4. Якщо такого користувача немає, то його дані записуються до бази даних. Після чого він є користувачем системи 5. Якщо такий користувач є і ведені дані відрізняються від тих, що зберігаються в базі даних, в оператора сервера запитується, чи дійсно він хоче перезаписати дані користувача
Альтернативний курс	1. Оператор серверу натискає кнопку заблокувати доступ користувача до системи 2. Виводиться список авторизованих користувачів 3. Оператор обирає користувача, якого він хоче видалити 4. Виводиться повідомлення про необхідність підтвердити свою дію 5. Після підтвердження дії, всі дані про користувача будуть видалені з бази даних. Публічні ключі даного користувача також будуть видалені
Винятки	В підсистеми немає користувачів чи користувач

Таблиця 3.3 – Опис сценарію використання підсистеми «Створення публічного та секретного ключа».

Таблиця 3.3- Створення публічного та секретного ключа

Назва	Створення публічного та секретного ключа
ID	3
Опис	Користувач може згенерувати пару ключів для ЕЦП: публічний та секретний ключ
Актори	Користувач

Частота використання	Низька
Тригери	Натискання кнопки для створення ключів для реалізації ЕЦП
Передумова	Користувач повинен бути авторизованим в підсистемі Створена нова пара ключів для ЕЦП
Постумова	Створена нова пара ключів для ЕЦП
Головний курс	1. Клієнт натискає кнопку для генерації ключів для реалізації ЕЦП: публічний та секретний ключ 2. Підсистема генерує ключі 3. На екрані з'являється повідомлення про успішне створення ключів
Альтернативний курс	1. Клієнт натискає кнопку для генерації ключів для реалізації ЕЦП: публічний та секретний ключ 2. У даного клієнта вже створена пара ключів 3. На екрані з'являється повідомлення про наявність пари ключів та пропозицію згенерувати нову пару 4. Користувач обирає варіант подальших дій
Винятки	Користувач вже має пару ключів, і відмовляється від створення нових

Опис сценарію використання підсистеми «Зберігання публічних ключів користувачів» представлений в таблиці 3.4

Таблиця 3.4- Зберігання публічних ключів користувачів

Назва	Зберігання публічних ключів користувачів
ID	4
Опис	Підсистема зберігає публічні ключі користувачів, які використовуються для перевірки підпису
Актори	Оператор
Частота використання	Середня
Тригери	Сигнал від системи про отримання публічних ключів від користувача
Передумова	Отримання ключів від користувача
Постумова	Зберігання публічних ключів користувачів на сервері
Головний курс	1.Отримуємо сигнал від системи, про отримання публічного ключа від сервера 2. Оновлюємо статус користувача в базі даних 3. Переміщуємо публічний ключ в місце, де він буде зберігатися
Альтернативний курс	-
Винятки	-

Опис сценарію використання підсистеми «Підписання файлів» представлений в таблиці 3.5

Таблиця 3.5 Підписання файлів

Назва	Підписання файлів за допомогою секретного ключа
ID	5
Опис	Користувач може підписати свої файли за допомогою секретного ключа
Актори	Користувач
Частота використання	Висока
Тригери	Натискання кнопки підписання файлів; в автоматизованій системі викликається самою системою
Передумова	Користувач повинен мати пару ключів для реалізації електронно-цифрового підпису
Постумова	Документ підписаний
Головний курс	1. Користувач натискає кнопку для підписання ключів 2. Обирає файли, які необхідно підписати 3. Користувач отримує повідомлення про успішне підписання файлів
Альтернативний курс	1. Автоматизована система передає сигнал про необхідність підписати файли, які в подальшому будуть передані на сервер 2. Підсистема підписує файли 3. Файли готові до відправки на сервер
Винятки	Користувач вже має згенеровані пари ключів чи існуючих ключі не дійсні

Опис сценарію використання підсистеми «Перевірка підпису» представлений в таблиці 3.6

Таблиця 3.6 – Перевірка підпису

Назва	Перевірка підпису
ID	6
Опис	Оператор сервера може перевірити справжність отриманих від користувача даних
Актори	Оператор
Частота використання	Висока
Тригери	Натискання кнопки для перевірки даних отриманих від користувача
Передумова	На сервері повинні бути підписані користувачем дані
Постумова	Повідомлення про справжність отриманих даних
Головний курс	1. Оператор сервера натискає кнопку для перевірки даних отриманих від користувача 2. Оператор обирає, які файли він хоче перевірити 3. Програма перевіряє всі обрані файли 4. Запускається сценарій “Вести журналу результатів автентифікації” 5. Якщо файл по якійсь причині не пройшов

	перевірку то, про це повідомляється оператору сервера, який приймає рішення. На час прийняття рішення файли не передаються системі, а поміщаються в карантин.
Альтернативний курс	-
Винятки	На сервері немає файлів користувача

Опис сценарію використання підсистеми «Ведення журналу результатів автентифікації» представлений в таблиці 3.7

Таблиця 3.7 – Ведення журналу результатів автентифікації

Назва	Ведення журналу результатів автентифікації
ID	7
Опис	В підсистемі ведеться журнал даних з результатами автентифікації
Актори	Оператор
Частота використання	Висока
Тригери	викликається підсистемою під час перевірки підпису
Передумова	Наявність підписаних файлів
Постумова	Запис результатів автентифікації в базі даних
Головний курс	Запис результатів автентифікації в базу даних
Альтернативний курс	-
Винятки	-

Опис сценарію використання підсистеми «Контроль за сроком дії ключів» представлений в таблиці 3.8

Таблиця 3.8 – Контроль за терміном дії ключів

Назва	Контроль терміну дії ключа
ID	8
Опис	Підсистема слідкує за терміном дії ключів і при закінчені терміну дії створює нові
Актори	Користувач
Частота використання	Користувач
Тригери	Щоденне спрацювання, по розкладу
Передумова	Користувач повинен мати пару ключів для реалізації електронно-цифрового підпису
Постумова	Уникнення припинення роботи підсистеми

	у разі закінчення терміном дії пари ключів
Головний курс	1. По розкладу, підсистема запускає алгоритм перевірки терміну дії пари ключів 2. Якщо термін дії ключів вийшов, про це повідомляється користувачеві, який приймає рішення, що далі робити
Альтернативний курс	-
Винятки	

3.3. Проєктування бази даних.

Розробимо таблиці полів бази даних.

Таблиця 3.9 - Опис полів таблиці «Список користувачів»

Назва поля	Тип даних	Розмір поля	Опис
name_client id	INT	11	Код користувача (ключове поле)
category_status	TINYINT	4	Інформатор про статус користувача (wait, connect, disconnect)
category_status	TINYINT	4	Інформатор про статус категорії (Увімкнена/Вимкнена)
category_date_ add	DATETIME	-	Дата та час додання категорії
category_date_ modificated	DATETIME	-	Дата та час змінення категорії

Таблиця 3.10 - Опис полів таблиці « Дані користувача»

Назва поля	Тип даних	Розмір поля	Опис
user_id	INT	11	Код користувача (ключове поле)
name_client	VARCHAR	45	Інформатор про ім’я користувача
category_status status	TINYINT	4	Статус користувача (wait, connect, disconnect)
category_status	TINYINT	4	Інформатор про статус категорії (Увімкнена/Вимкнена)
category_ num_in_system	TINYINT	20	Інформатор про номер комп’ютера в системі
category_ key_validly	TINYINT	18	Інформатор про публічний ключ користувача

category_ key_date -	DATETIME		Інформатор про ключі (Дійсний/ не дійсний)
category phone number –	VARCHAR	80	Інформатор про номер телефону користувача
user_email	VARCHAR	80	Електронна пошта
user_name	VARCHAR	45	Ім’я

В рамках поставленої задачі кваліфікаційної роботи, для візуалізації розроблених таблиць відтворюємо зв’язки, які існують повинні бути в базі даних (рис. 3.6).

Далі розробляємо схему бази даних

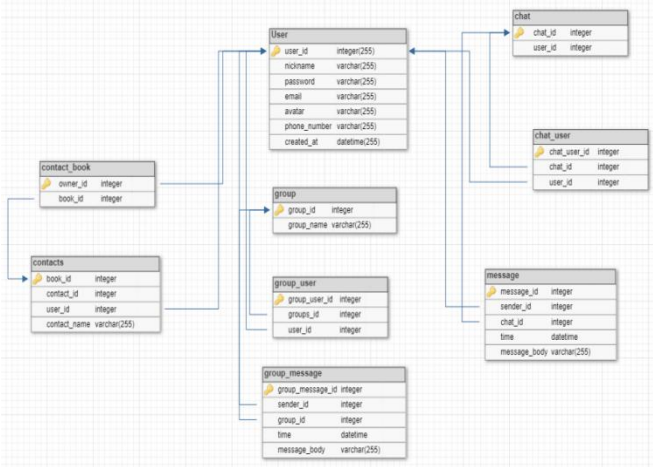

Рисунок 3.6 - Схему бази даних

Далі база даних створюється за допомогою pgAdmin (рис 3.7).

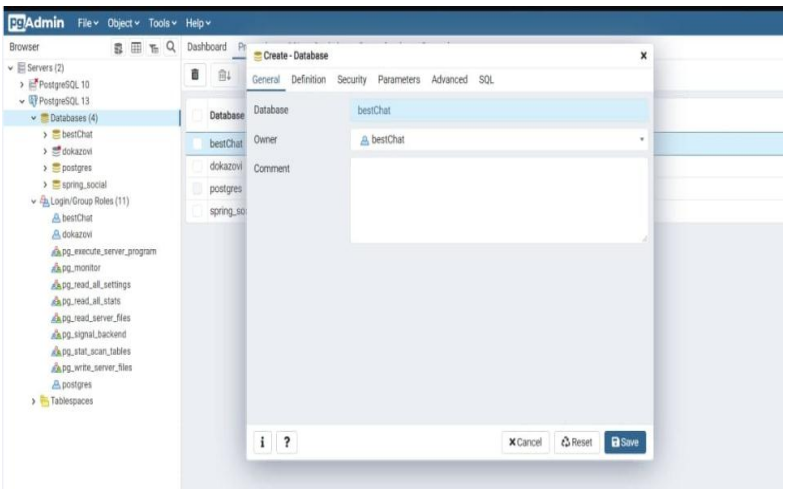

Рисунок 3.7 - Створення бази даних

Для початку потрібно підключити залежності в середовищі розробки IntelliJ IDEA.

```
dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  implementation 'org.flywaydb:flyway-core'
  runtimeOnly 'org.postgresql:postgresql'
  }
```

Після цього потрібно підключити connection driver, прописавши необхідні параметри в application.properties, як зображено на рис. 3.8.

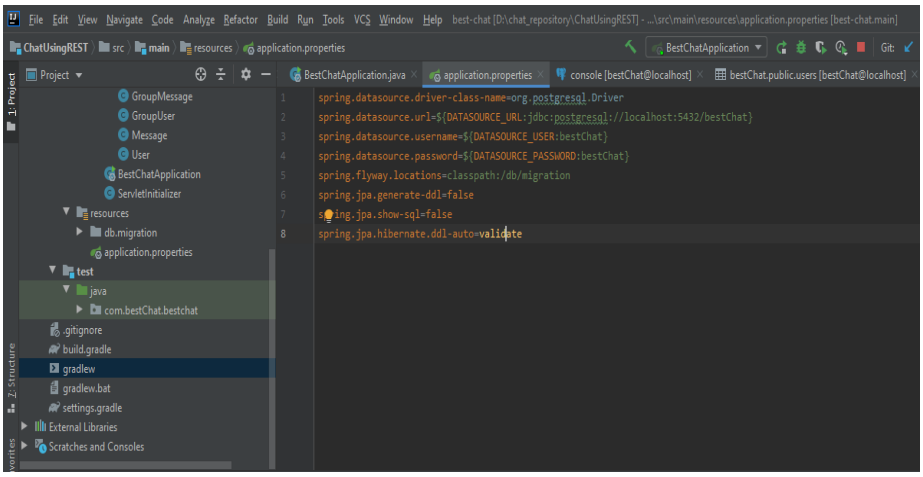

Рис. 3.8 Підключення connection driver

Дані зареєстрованих користувачів в базі даних представлено на рисунку 3.9.

Data Output | Messages | Notifications

	id [PK] integer	email character varying	password character varying	role character varying
1	1	mmmm@gmail.co...	$2b$05$KcYTplcT2QGdHJ5az5KRce89i7JHodt0ldEUNPoLm7o9BYqbMCe...	USER
2	2	user@gmail.com	$2b$05$LD5lqvZVjybl/0/yYrd1oexKxhulqYFGXpb7Bzg9OOiSPY6tCD4N.	USER
3	3	admin@gmail.com	$2b$05$nqz01ZAOHWXb8SbOcR9Toelnj2ASmbRlkeo/3GR5hBbGhfNp4z...	ADMIN
4	4	user1@gmail.com	$2b$05$NC8iPamrJdVQh/Sv3TQTGOWUIZ0EtDNNTzRhf3N8ZgXnOA5QJ...	ADMIN
5	5	user3@gmail.com	$2b$05$T4z4rWomlBQs8ZCCBMsCiuc6Kps8LrdQ6dwzgYA7ziRhoUdnjtN...	ADMIN

Рисунок 3.9 – Дані зареєстрованих користувачів в базі даних

3.4. Програмна реалізація модуля реєстрації та авторизації.

При натисканні кнопки «Авторизація», користувач попадає на сторінку авторизації, де він може зареєструватись, або увійти, якщо обліковий запис вже був створений . Вікно для авторизації оператора сервера та користувача мають однакову форму (рис. 3.10, рис. 3.11). Авторизація здійснюється за допомогою електронної адреси користувача та індивідуального паролю.

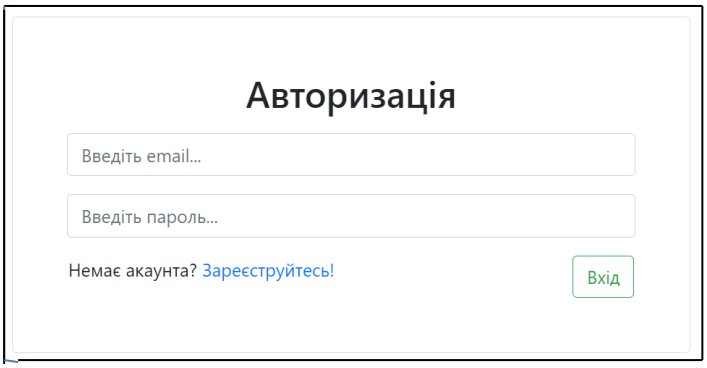

Рисунок 3.10 – Вікно авторизації корисувача.

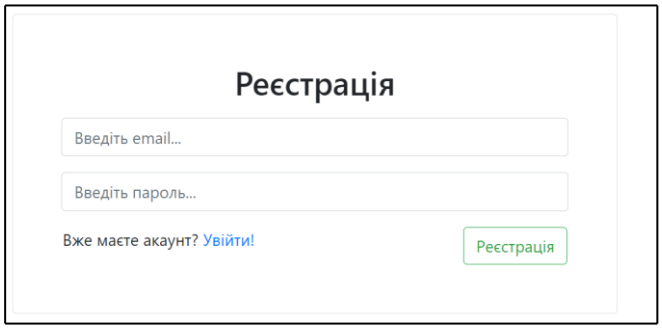

Рисунок 3.11 – Вікно «Реєстрація»

Розробка програмного модуля задачі реєстрації та авторизації

```
const Auth = observer(() => {
    const {user} = useContext(Context)
    const location = useLocation()
    const history = useHistory()
    const isLogin = location.pathname === LOGIN_ROUTE
    const [email, setEmail] = useState('')
    const [password, setPassword] = useState('')
 const click = async () => {
        try {
            let data;
            if (isLogin) {
                data = await login(email, password);
            } else {
                data = await registration(email, password);
            }
            user.setUser(user)
            user.setIsAuth(true)
            history.push(SHOP_ROUTE)
        } catch (e) {
            alert(e.response.data.message)
        }

    }
    return (
        <Container
            className="d-flex justify-content-center align-items-center"
```

```
        style={{height: window.innerHeight - 54}}
      >

<Card style={{width: 600}} className="p-5">
          <h2 className="m-auto">{isLogin ? 'Авторизація' : "Реєстрація"}</h2>
          <Form className="d-flex flex-column">
            <Form.Control
              className="mt-3"
              placeholder="Введіть email..."
              value={email}
              onChange={e => setEmail(e.target.value)}
            />
            <Form.Control
              className="mt-3"
              placeholder="Введіть пароль..."
              value={password}
              onChange={e => setPassword(e.target.value)}
              type="password"
            />
            <Row className="d-flex justify-content-between mt-3 pl-3 pr-3">
              {isLogin ?
                <div>
                  Немає акаунта? <NavLink

to={REGISTRATION_ROUTE}>Зареєструйтесь!</NavLink>
  </div>
                :
 <div>
                  Вже маєте акаунт? <NavLink to={LOGIN_ROUTE}>Увійти!</NavLink>
```

```
                </div>
              }
              <Button
                variant={"outline-success"}
                onClick={click}
              >
                {isLogin ? 'Вхід' : 'Реєстрація'}
              </Button>
            </Row>

          </Form>
        </Card>
      </Container>
    );
});
export default Auth;
```

3.5. Реалізація алгоритма Ель-Гамаля.

3.5.1.Обрання мови програмування .

В цьому алгоритмі використовуються складні математичні операції та операціях дискретного логарифмування, що робить його стійким до обчислювальних атак.

Для розробки програми реалізації алгоритму було обрано мову Python (Hypertext Preprocessor). Мова програмування PHP, це високорівнева мова програмування загального призначення, структурована, об'єктно-орієнтована, імперативна, функціонально орієнтована на поліпшення читабельності коду та продуктивності розробників. Вона має розвинену обчислювальну частину, а наявність в мові досить великих можливостей запису повноцінного набору керуючих конструкцій, дозволяє реалізовувати

алгоритми будь якої складності.

PHP складається з ядра і набору підключаються розширень: для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і ін.

Python підтримує кілька парадигм програмування : структуровану,. Код на Python організований у класи та функції, які можна об'єднати в модулі.

3.5.2. Програмна реалізація алгоритма Ель-Гамаля

Listing program Алгоритм Ель-Гамаля на мові програмування Python

```
 import random
def modexp(base, exp, modulus):
  if modulus == 1:
    return 0
   result = 1
 base = base % modulus
     while exp > 0:
    if exp % 2 == 1:
       result = (result * base) % modulus
       exp = exp >> 1
     base = (base * base) % modulus
   return result
def generate_keypair(p, g, x):
   y = modexp(g, x, p)
 return (p, g, y), x
 def encrypt(pk, plaintext):
  p, g, y = pk
   k = random.randint(1, p - 1)
    a = modexp(g, k, p)
 b = (modexp(y, k, p) * plaintext) % p
  return a, b

def decrypt(sk, ciphertext):
  p, _, _ = sk
```

```
 a, b = ciphertext
  x = sk[1]
 plaintext = (modexp(a, p - 1 - x, p) * b) % p
  return plaintext
 if __name__ == '__main__':
    print("ElGamal Encrypter/Decrypter")
   p = int(input("Enter a prime number (p): "))
     g = int(input("Enter a primitive root (g) modulo p: "))
 x = int(input("Enter a private key (x) less than p: "))     public_key, private_key =
generate_keypair(p, g, x)
  print("Your public key is ", public_key)
  print("Your private key is ", private_key)
message = int(input("Enter a message to encrypt (an integer): "))
 encrypted_msg = encrypt(public_key, message)     print("Your encrypted message is: ")
print(encrypted_msg)
 print("Decrypting message with private key ", private_key, " . . .")
 decrypted_msg = decrypt(private_key, encrypted_msg)
 print("Your decrypted message is:")
 print(decrypted_msg)
```

3.5.3. Програмна реалізація алгоритма Ель-Гамаля для управління пристроєм та шифрування/дешифрування повідомлень на мові програмування Python

Listing program на основі алгоритму Ель-Гамаля для управління пристроєм та шифрування/дешифрування повідомлень.

```
  import random  def modexp(base, exp, modulus):
    if modulus == 1:
    return 0
   result = 1
    base = base % modulus
  while exp > 0:
if exp % 2 == 1:
   result = (result * base) % modulus
  exp = exp >> 1
 base = (base * base) % modulus
  return result  def generate_keypair(p, g):
   x = random.randint(1, p-1)
```

```
 y = modexp(g, x, p)
 return (p, g, y), (p, g, x)
 def encrypt(public_key, plaintext):
   p, g, y = public_key
   k = random.randint(1, p-1)
   a = modexp(g, k, p)
 b = (modexp(y, k, p) * plaintext) % p
 return a, b  def decrypt(private_key, ciphertext):
  p, g, x = private_key
   a, b = ciphertext
  plaintext = (modexp(a, p-1-x, p) * b) % p
return plaintext
  if __name__ == '__main__':
   print("Ель-Гамаль Encrypter/Decrypter")
   p = int(input("Введіть просте число p: "))
 g = int(input("Введіть примітивний корінь g: "))
     public_key, private_key = generate_keypair(p, g)
  print("Ваш публічний ключ: ", public_key)
  print("Ваш приватний ключ: ", private_key)
   message = int(input("Введіть повідомлення для шифрування (ціле число): "))
  ciphertext = encrypt(public_key, message)
  print("Зашифроване повідомлення: ", ciphertext)
   decrypted_message = decrypt(private_key, ciphertext)
   print("Розшифроване повідомлення: ", decrypted_message)
```

ВИСНОВКИ ДО ТРЕТЬОГО РОЗДІЛУ

В розділі представлено розробка сценарію, технологію та опис прецидентів сценарію.

Для реалізації було обираємо клієнт - серверну технологію, в якій кожна з частин системи відповідає за свій розділ.

В системі представлено два типа користувачів:

1. Перший тип – це користувач клієнтської частини підсистеми, який після ідентифікації в системі, може створювати ЕЦП та використовувати його для підпису своїх даних.
2. Другий тип – це оператор серверу, який проводить автентифікацію даних користувача, зберігає результати в базі даних та реагує на дані, які не пройшли автентифікацію.

Для збереження інформації була розроблена база даних. Для задачі роботи з клієнтами розроблено інтерфейсу та програма реалізації модуля реєстрації та авторизації . Було розроблено програму,яка реалізує алгоритм Ель-Гамаля та систему шифровування/ дешифрування по Ель-Гамаля.

Мовою для розробки всіх модулів було обрано мову Python.

ВИСНОВКИ

Кваліфікаційна магістерська робот присвячена питанням розробки алгоритмів та моделей технологій кібербезпеки.

Під політикою кібербезпеки інформації слід розуміти набір законів, правил,обмежень, рекомендацій тощо, які регламентують порядок обробки інформації та спрямовані на захист інформації від певних загроз.

В роботі представлено загальний опис завдань та методів криптографії та криптографічного аналізу, наводиться опис найбільш поширених алгоритмів блокового шифрування DES та Файстеля. В частині, присвяченій асиметричним системам шифрування, розглядається алгоритм RSA та алгоритм Ель-Гамаля. Для хешування паролів використовують адаптивну функцію BCrypt з шифром Blowfish. Робота супроводжується прикладами. В третьому розділі представлена розробка сценарію, технологію та опис прецидентів сценарію.

Для реалізації було обрано клієнт - серверну технологію, в якій кожна з частин системи відповідає за свій розділ. Була розроблена база даних. інтерфейс для роботи клієнтів та програмна реалізація модуля реєстрації та авторизації. Була розроблена програма алгоритму Ель-Гамаля та програма алгоритму шифрування / дешифрування по методу Ель-Гамаля. Мовою програмування було обрано мову Python.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Saltzer J. H., Schroeder M. D. The Protection of Information in Computer Systems [Електронний ресурс]. URL: http://www.acsac.org/secshelf/papers/protection_information.pdf (дата звернення: 30.03.2016).
2. McCumber J. Information Systems Security: A Comprehensive Model // Proceeding of the 14th National Computer Security Conference, NIST, Baltimore, MD, 1991.
3. Pender-Bey G. The Parkerian Hexad: The CIA Expanded [Електронний ресурс]. URL: http://www.zigthis.com/145/parkerianhexad (дата звернення: 30.03.2016).
4. A Model for Information Assurance: An Integrated Approach // W. Maconachy, C. Schou, D. Ragsdale, D. Welch // Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, U.S. Military Academy, NY, 2014.
5. Loeb L. Information assurance powwow [Електронний ресурс]. URL: http://www.ibm.com/developerworks/security/library/s-confnotes2 (дата звернення : 30.03.2016).
6. Yen-Hung Hu. Challenges in building a trustworthy network [Електронний ресурс]. URL: http://dx.doi.org/10.4172/2332-0796.1000111 (дата звернення 30.03.2016).
7. Засоби криптографічного захисту інформації. Бібліотека libcrypto. [Електронний ресурс]. – 2018. – Режим доступу: http://www.cryptocom.ua /docs/cryptopack21-libcrypto.pdf
8. Технології та продукти Microsoft в забезпеченні інформаційної безпеки. [Електронний ресурс]. – 2018. – Режим доступу: http://www.intuit.ua /studies/courses/600/456/in
9. Бернет, С. Криптографія. Официальное руководство RSA Security / С. Бернет, С. Пэйн. –Львів .: 2017. – 384 с.

10. Гайворонський, М.В., Новіков, О.М. Безпека інформаційно-комунікаційних систем. – К.: Видавнича група BHV, 2009. – 478 с.
11. Гончарова Л.Л. Основи захисту інформації в телекомунікаційних та комп'ютерних мережах. / Л.Л. Гончарова, А.Д. Возненко, О.І. Стасюк та ін. – К.: ДЕТУТ, 2013. – 435 с.
12. Вокалюк Т.А. Захист інформації в комп'ютерних системах. / Т.А. Вокалюк. – Житомир: Вид-во ЖДУ, 2013. – 131 с.
13. Tanenbaum A.S. Modern operating systems. / A.S. Tanenbaum, H. Bos. Fourth edition. - by Pearson Education, Inc., Upper Saddle River, New Jersey, 2015.–
14. Гайворонський М.В., Безпека інформаційно- комунікаційних систем. / М.В. Гайворонський , , О.М. Новіков, – К.: Видавнича група BHV, 2019. – 478 с.
15. Бернет, С. Криптографія. RSA Security / С. Бернет, С. Пэйн. –Львів.: ПРО, 2012. – 384 с.
16. Гайворонський М.В., Безпека інформаційно- комунікаційних систем. / М.В. Гайворонський , , О.М. Новіков, – К.: Видавнича група BHV, 2015. – 478 с.
17. Вокалюк Т.А. Захист інформації в комп'ютерних системах. / Т.А. Вокалюк. – Житомир: Вид-во ЖДУ, 2013. – 131 с.
18. Paraska G. Engineering and methodology of modern technology/ Monograph.- Khmelnytsky.– 2012.- 406c
19. Астістова Т.І Розробка автоматизованої системи аналізу текстів «Антиплагіат» / Т.І. Астістова, В.О.Керіб. //Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проектуванн. - К. : КНУТД, 2017. С. 118-121 – ISBN 978-966.

20. Перебийніс В.С.. Математична лінгвістика . Українська мова : енциклопедія./ В. С. Перебийніс /— К. : Українська енциклопедія, 2000. — ISBN 966-7492-07-9.
21. Handbook of Applied Cryptography, Menezes A.J., Oorschot P.C., Vanstone S.A. C. 25—26
22. Бук С. Основи статистичної лінгвістики: Навчально-методичний посібник / Відп. ред. проф. Ф. С. Бацевич./— Львів: Видавничий центр ЛНУ імені Івана Франка, 2008.— 124 с.
23. A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. Handbook of Applied Cryptography. — 1997. — ISBN 0-8493-8523-7.
24. Шнайер Б. Прикладна криптографія. Протоколи, алгоритми, тексти на Си = Applied Cryptography. Protocols, Algorithms and Source Code in C. — Житомир: Вид-во ЖДУ, 2012. — 816 с. — 3000 экз. — ISBN 5-89392-055-4.
25. Астістова Т.І Дослідження та розробка програмного продукту моделі кібербезпеки / Т.І. Астістова, К.Джеладзе. //Інформаційні технології в науці, виробництві та підприємництві: зб. наук. праць молодих вчених, аспірантів, магістрів кафедри інформаційних технологій проєктування - К. : КНУТД, 2019. С. 233-236 – ISBN 978-966.
26. Тарасюк Г.М., Шваб Л.І. Планування діяльності підприємства: Навч. посіб. - К.: Каравела, 2013.- 432 с.
27. Принципи проектування програм [Електронний ресурс] – Режим доступу до ресурсу: https://allref.com.ua/uk/skachaty/Principi_proektuvannya_program
28. Соціальні мережі та месенджери в Україні [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: http://rb.com.ua/uk/blog-uk/omnibus-uk/socialni-merezhi-ta-mesendzheri-v-ukraini/
29. Staticprogramanalysis [Електронний ресурс] // wikipedia.org. – 2014. – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Static_program_analysis.

30. Dynamicprogramanalysis [Електронний ресурс] // wikipedia.org. – 2015. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Dynamic_program_analysis.
31. Грицюк Ю. І., Аналіз вимог до програмного забезпечення. Навчальний посібник/ Ю. І. Грицюк .// Київ,2018, С.425
32. Тестування програмного продукту [Електронний ресурс]. Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L11.htm
33. Basics of BDD in testing, by Alex McPeak [Електронний ресурс]. Режим доступу: https://crossbrowsertesting.com/blog/development/what-is-bdd
34. The Pros and Cons of Behavior-Driven Development, by Emilie Maxie [Електронний ресурс]. Режим доступу:
35.. Керівництво з програмування на C# [Електронний ресурс]
Режим доступу: https://msdn.microsoft.com
36. Апаратне та програмне забезпечення ПК [Електронний ресурс]
- Режим доступу:
http://eprints.zu.edu.ua/18/1/Konspect_modul_1_Windows.pdf.
37.. Створення додатків Windows Forms у Visual Studio з C #
[Електронний ресурс] - Режим доступу: https://docs.microsoft.
38. Рад Б. Я. Архітектура інформаційних систем / Б. Я. Рад, А. І.Водяхо, В. А. Дубенецький, В. В. Цехановскій. –Львів.: Академія, 2012. 220с
39. Рой філдінг Архітектурні стилі та дизайн мережевих архітектур програмного забезпечення [Електронний ресурс]. – Режим доступу: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
40. Харченко О. Г. Експерта система проектування архітектури програмного забезпечення / О. Г. Харченко, І. О. Боднарчук, В. В. Яцишин // Комп'ютерні технології друкарства. – № 29. – 2013. – с. 10-26.
41.. Клієнт-серверна архітектура [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Клієнт-серверна_архітектура.

42. Береза А. М. Основи створення інформаційних систем. Навч. посібник. — 2- е вид., перероб. і доп. — К.: КНЕУ, 2011. — 214 с.
43. Бази данних і мова SQL [Електронний ресурс] Режим доступу: https://function-x.ua /sql_join.html.
44. DataSets, DataTables, and DataViews [Електронний ресурс] Режим доступу: https://docs.microsoft.com/enus/ dotnet/framework/data/adonet/dataset-