НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«ТЕХНІЧНІ АСПЕКТИ БЕЗПЕЧНОЇ ПЕРЕДАЧІ ІНФОРМАЦІЇ В КІБЕРПРОСТОРІ»

Виконав: здобувач 4 курсу

Рівень бакалавр

Галузь знань 12 «Інформаційні технології»,

спеціальність 125 «Кібербезпека»

Біруля Олександр Вячеславович

Керівник: к.фіз.-мат.н., доцент

Черєвко Євген Володимирович

Рецензент: к.т.н., доцент

Кухаренко Сергій Вікторович

Одеса – 2025

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»
Факультет **кібербезпеки та інформаційних технологій**
Кафедра **кібербезпеки**
Галузь знань: **12 Інформаційні технології**
Спеціальність: **125 Кібербезпека**
Назва освітньої програми: **Кібербезпека**

ЗАТВЕРДЖУЮ
Завідувач кафедри кібербезпеки
професор С.А. Горбаченко
_____ «____ »_____ 20__ року

З А В Д А Н Н Я
на кваліфікаційну (бакалаврську) роботу
здобувачу Бірулі Олександру Вячеславовичу

1. Тема роботи: «Технічні аспекти безпечної передачі інформації в кіберпросторі»
Керівник роботи: к.фіз.-мат.н., доцент Черєвко Євген Володимирович
затверджені наказом НУ ОЮА від 18 березня 2025 року № 548-6
2. Строк подання здобувачем роботи 19 травня 2025 року
3. Дата видачі завдання 16 вересня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка

Здобувач _____
(підпис) (прізвище та ініціали)

Керівник роботи _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Технічні аспекти безпечної передачі інформації в кіберпросторі» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 Кібербезпека, освітньою програмою: Кібербезпека, містить 20 рисунків, 3 таблиці, 5 додатки, 20 джерел за переліком використаних посилань. Обсяг роботи – 47 сторінки, з яких основний текст займає 33 сторінки.

Метою дослідження є аналіз технічних засобів захисту інформації під час її передавання в мережі, виявлення вразливостей і створення прикладних рішень на основі криптографічних алгоритмів.

У роботі розглянуто загрози, пов’язані з передачею даних (атаки типу «людина посередині», фішинг, шкідливе ПЗ), а також технічні методи захисту: шифрування, хешування, цифрові підписи, VPN і міжмережеві екрани. Окрему увагу приділено міжнародним стандартам, зокрема ISO/IEC 27001 та FIPS 140-3.

Практична частина містить реалізацію вдосконаленого шифру Цезаря з динамічним ключем та системи авторизації з хешуванням паролів і сіллю. Результати можуть бути використані в локальних системах захисту інформації, навчальних проєктах і як база для подальших досліджень.

КІБЕРПРОСТІР, ШИФРУВАННЯ, ХЕШУВАННЯ, КРИПТОГРАФІЯ, АВТОРИЗАЦІЯ, БЕЗПЕКА

ABSTRACT

Qualification work on the topic "Technical aspects of secure information transmission in cyberspace" for obtaining the first (bachelor's) level of higher education in the specialty 125 Cybersecurity, educational program: Cybersecurity, contains 20 figures, 3 tables, 5 appendices, 20 sources according to the list of references used. The volume of the work is 47 pages, of which the main text takes up 33 pages.

The purpose of the study is to analyze technical means of protecting information during its transmission over the network, identify vulnerabilities and create applied solutions based on cryptographic algorithms.

The work considers threats associated with data transmission (man-in-the-middle attacks, phishing, malware), as well as technical protection methods: encryption, hashing, digital signatures, VPN and firewalls. Special attention is paid to international standards, in particular ISO/IEC 27001 and FIPS 140-3.

The practical part includes the implementation of an improved Caesar cipher with a dynamic key and an authorization system with password hashing and salt. The results can be used in local information protection systems, educational projects and as a basis for further research.

CYBERSPACE, ENCRYPTION, HASHING, CRYPTOGRAPHY, AUTHORIZATION, SECURITY

ЗМІСТ

ВСТУП 6
1. КІБЕРПРОСТІР ТА ІНТЕРНЕТ З’ЄДНАННЯ 8
1.1 Загрози в кіберпросторі 8
1.2 З’єднання користувача з сервером 9
1.3 Інтернет провайдери 11
2. ТЕОРЕТИЧНІ АСПЕКТИ ЗАХИСТУ ІНФОРМАЦІЇ 14
2.1 Поняття захисту інформації 14
2.2 Історія створення криптографічних методів захисту інформації 15
2.3 Огляд шифру Цезаря та шифру Віженера 18
2.4 Міжнародні стандарти у сфері захисту інформації 20
2.5 Мережеві екрани 22
2.6 Сучасні методи захисту переданої інформації 24
3. РОЗРОБКА ПРОГРАМ ШИФРУВАННЯ ТА АВТЕНТИФІКАЦІЇ 29
3.1 Удосконалення шифру «Цезаря» 29
3.2 Програма «Для реєстрації та входу» 32
ВИСНОВКИ 37
ПЕРЕЛІК ПОСИЛАНЬ 39
ДОДАТОК А 41
ДОДАТОК Б 42
ДОДАТОК В 43
ДОДАТОК Г 44
ДОДАТОК Д 45

ВСТУП

У сучасному світі інформаційні технології стали фундаментом функціонування всіх сфер суспільного життя від державного управління до приватного бізнесу. Усе частіше критично важливі рішення ґрунтуються на швидкому доступі до достовірної та захищеної інформації. Водночас активне поширення мережевих технологій та глобального інтернету створює нові виклики в контексті інформаційної безпеки. З кожним роком зростає кількість кіберінцидентів, пов’язаних з несанкціонованим доступом до інформації, її викраденням, підміною або знищенням. Це зумовлює необхідність безперервного вдосконалення методів захисту, особливо в частині технічних аспектів безпечної передачі даних у кіберпросторі.

Об’єктом дослідження в цій роботі є процес захисту інформації в інформаційно-комунікаційних системах.

Предметом дослідження є технічні засоби, методи і технології, що забезпечують безпечну передачу інформації в кіберпросторі, зокрема криптографічні протоколи, засоби автентифікації, шифрування, хешування та інші інструменти інформаційної безпеки.

Актуальність теми полягає у зростаючому значенні інформації як стратегічного ресурсу та одночасному збільшенні кількості загроз, які можуть призвести до її компрометації. У зв’язку з цим технічний захист інформації (ТЗІ) є не лише запорукою ефективного функціонування організацій, а й ключовим елементом національної безпеки будь-якої держави.

Метою дослідження є проведення комплексного аналізу сучасних технічних методів захисту інформації під час її передавання в цифровому середовищі, виявлення їх переваг і обмежень, а також розроблення практичних рішень щодо підвищення стійкості до кіберзагроз.

Завдання роботи: проаналізувати загрози, що виникають під час передавання інформації в кіберпросторі, вивчити основні технічні засоби захисту, включаючи мережеві екрани, VPN, криптографічні протоколи, дослідити алгоритми симетричного й асиметричного шифрування, хешування

та цифрового підпису, розробити прикладні програмні засоби шифрування та автентифікації.

Структура роботи включає теоретичний аналіз ключових понять, огляд історичного розвитку криптографії, оцінку сучасних стандартів і практик захисту інформації, а також практичну реалізацію окремих технічних рішень.

1 КІБЕРПРОСТІР ТА ІНТЕРНЕТ З'ЄДНАННЯ

1.1 Загрози в кіберпросторі

Перед тим як розповідати про захист інформації у кіберпросторі, потрібно розібратися що таке кіберпростір, та що може загрожувати інформації в ньому.

Поняття кіберпростору можна знайти в законі України «Про основні засади забезпечення кібербезпеки України».

«Кіберпростір – це середовище (віртуальний простір), яке надає можливості для здійснення комунікацій та/або реалізації суспільних відносин, утворене в результаті функціонування сумісних (з'єднаних) комунікаційних систем та забезпечення електронних комунікацій з використанням мережі Інтернет та/або інших глобальних мереж передачі даних» [1].

Тепер, коли із кіберпростором все зрозуміло, потрібно розібратися що може становити загрозу для інформації яка в ньому знаходиться. Інформація може підлягати ризику під час передачі каналами зв'язку. Якщо канал зв'язку недостатньо захищений, то інформацію можна перехопити або заблокувати цей канал, в залежності від мети злочинця. Це особливо актуально коли користувач використовує публічну Wi-Fi мережу.

Окрім цього не варто забувати що всі ми люди. Людина може легко пошкодити інформацію в наслідок своєї неуважності.

Тож можна зробити висновок, що інформація в кіберпросторі може бути ушкоджена внаслідок недбалого зберігання, недостатнього рівня захисту каналів зв'язку або через людський фактор.

Окрім технічних недоліків захисту каналів зв'язку та людського фактору, варто розглянути ще кілька серйозних загроз, які часто використовують зловмисники.

Одна з таких – атака "людина посередині" (Man-in-the-Middle). Суть цієї атаки в тому, що зловмисник непомітно стає між користувачем і сервером, через що весь трафік проходить через нього. Він може просто спостерігати за трафіком або ж змінювати передану інформацію. Це особливо небезпечно, коли

не використовується шифрування (наприклад, сайт працює через HTTP, а не HTTPS). Навіть якщо шифрування є, але зламаний або підроблений сертифікат, це може дозволити злочинцю втрутитися в обмін повідомленнями. Зараз такі атаки можна реалізувати навіть через фальшиві точки доступу Wi-Fi, які видаються за публічні мережі [15].

Ще одна велика загроза – фішинг та соціальна інженерія. В цьому випадку сам користувач добровільно передає дані, навіть не підозрюючи цього. Наприклад, він отримує повідомлення від «банку» чи «державної установи» з проханням підтвердити свої дані. Перейшовши за посиланням, користувач опиняється на підробленому сайті й вводить там свій логін, пароль або навіть дані картки. Такі методи працюють, бо на них розраховано - що людина поспішить, не зверне уваги або не розуміється на кібербезпеці. Найгірше те, що жодна система шифрування не врятує, якщо користувач сам передав свої конфіденційні дані [16].

Також поширеною загрозою є шкідливе програмне забезпечення. Це можуть бути віруси, трояни, програми-шпигуни, банери чи навіть програми-здирники, які шифрують файли користувача і вимагають гроші за розшифрування. Часто шкідливе ПЗ маскується під звичайні програми або прикріплюється до файлів у листах. Варто пам’ятати, що навіть одна така програма може дати зловмиснику повний доступ до особистих даних, банківської інформації або навіть камери та мікрофона пристрою. Наприклад, у 2017 році шкідливе ПЗ WannaCry паралізувало роботу сотень компаній по всьому світу, шифруючи дані та вимагаючи викуп [17].

1.2З’єднання користувача з сервером

Обмін інформацією між користувачем та сервером завжди проходить в декілька етапів (Див. Рисунок 1.1).

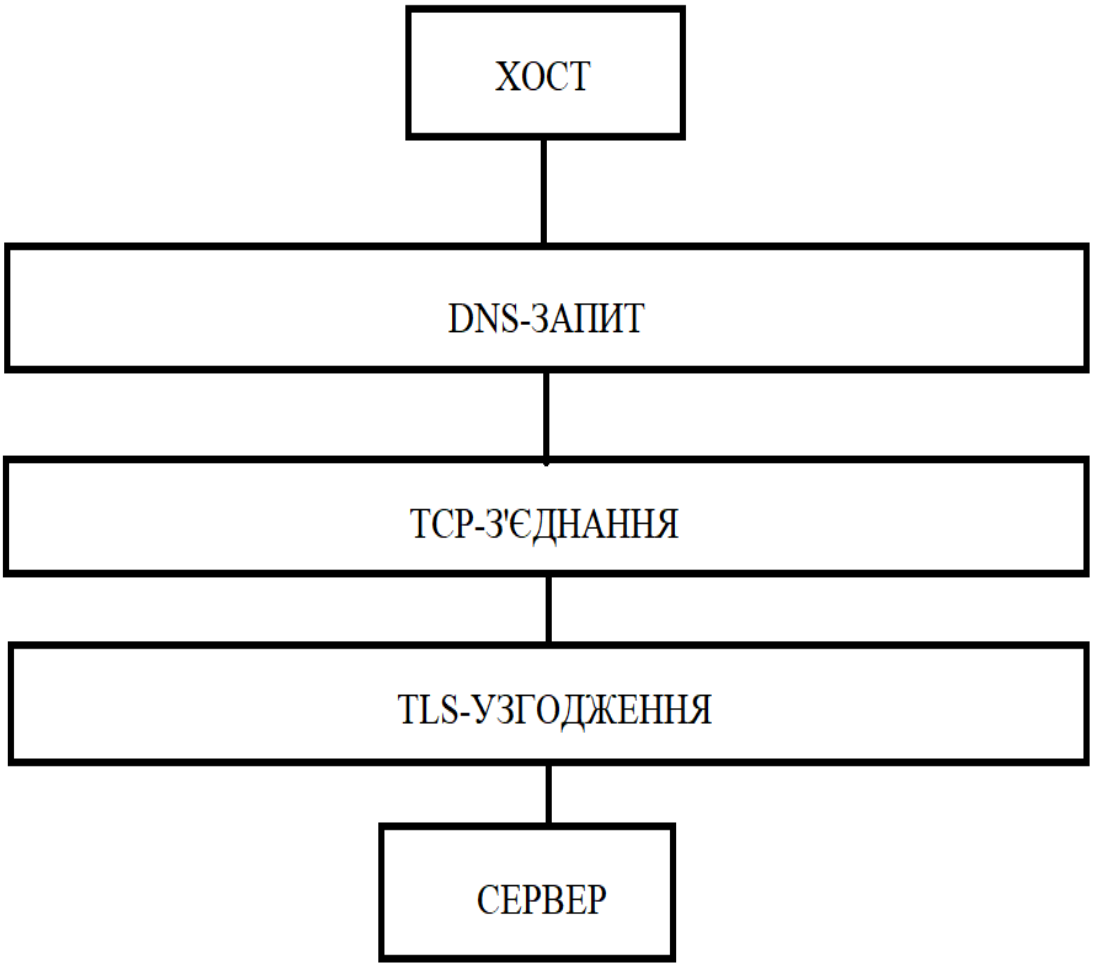

Рисунок 1.1 – Процес з’єднання між хостом та сервером

Проблеми починаються з першого ж етапу. З початку треба знайти IP-адресу сервера. Щоб це зробити хост робить DNS-запит. Проблема в тому, що DNS-запити зазвичай не шифруються. Це створює уразливість, яка полягає в тому, що будь хто, хто зможе перехопити трафік, може дізнатися з якими серверами відбувається спілкування а також підміняти IP-адреси серверів.

DNS дозволяє браузеру знайти потрібний веб-сайт за доменним ім'ям.

DNS-сервер зберігає IP-адреси веб-сайтів. DNS-сервер надає браузеру адресу веб-сайту, коли ви вводите доменне ім'я, а також кешує DNS-записи домену.

DNS-сервер дуже схожий на файл hosts.txt, але працює на набагато більшому рівні.

Далі між хостом та сервером встановлюється TCP-з’єднання. TCP працює, з’єднуючи кожен кінець і обіцяючи, що кожен пакет буде надісланий

так, як він має бути. На початку для встановлення з’єднання надсилаються та приймаються спеціальні пакети. Пакети обмінюються по каналу зв’язку, і кожен пакет перевіряється, щоб дізнатися, чи дійшов він до одержувача. Якщо доставка не вдалася, пакет відправляється на наступну спробу. Варто зауважити, що під час TCP-з’єднання дані теж не шифруються, це може бути використаним для збору метаданих.

Після того, як було отримано IP-адресу та встановлено з’єднання починається етап TLS-узгодження. Під час цього користувач обмінюється з сервером сертифікатами і між ними створюється захищений канал обміну даними. Найчастіше для захисту каналу використовується HTTPS.

HTTPs – це вдосконалення HTTP, яке забезпечує безпеку транзакцій. Ваші дані передаються по HTTP у вигляді простого тексту. Якщо хтось може перехопити мережевий трафік, він може отримати доступ до конфіденційної інформації. HTTPs дозволяє шифрувати дані так само, як це робить HTTP.

У комп'ютерних технологіях HTTPs (HyperText Transfer Protocol Secure) описує захищену версію стандартного протоколу HTTP. Безпека встановлюється, коли HTTP використовується разом з криптографічним протоколом TLS.

1.3 Інтернет провайдери

Існують компанії, які називаються інтернет-провайдерами, що надають доступ до Інтернету приватним особам, підприємствам та організаціям. Вони відіграють важливу роль у сучасному цифровому світі, оскільки пропонують безліч послуг для користування глобальною мережею. Деякі провайдери надають послуги хостингу та зберігання даних. Це означає, що вони надають своїм клієнтам серверні ресурси, щоб ті могли зберігати свої веб-сайти, додатки та онлайн-сервіси. Деякі користувачі потребують віртуального хостингу, а інші віддають перевагу фізичному хостингу.

Існує декілька типів інтернет-провайдерів, які відрізняються за своїми послугами та технологіями:

- компанії, які забезпечують високошвидкісне підключення через кабельні мережі, DSL або оптоволоконні лінії;
- мобільні оператори, які надають доступ до інтернету через мережі 4G, 5G;
- супутникові провайдери, які забезпечують доступ до глобальної мережі через супутникові антени [13].

Під час спілкування між користувачем та сервером увесь трафік проходить через інтернет-провайдера. Сам по собі провайдер не є загрозою, але якщо інформація не шифрується це може мати потенційні ризики.

Провайдер може бачити на які сайти заходить користувач, який контент він дивиться. Може бачити вміст файлів і повідомлень, обсяг переданих даних. Деякі дані можуть зберігатися на серверах провайдера.

Отже теоретичною загрозою є те, що провайдер може робити з отриманими даними все що завгодно. Також не можна забувати що сервер провайдера може бути зламаний і тоді зловмисники отримають доступ до всієї інформації яка на ньому зберігається. Окрім цього провайдер може блокувати деякі сайти чи сервіси що може становити загрозу доступності інформації.

Таблиця 1 – Порівняння звичайного підключення з VPN та TOR

Характеристика	Звичайне підключення	VPN	TOR
Шифрування трафіку	Немає	Повне	Багатошарове
Анонімність IP	Відсутня – видно справжню IP-адресу	IP замінюється на IP сервера VPN	IP постійно змінюється
Швидкість	Висока	Трохи нижча за звичайне підключення	Повільна

Продовження Таблиці 1

Характеристика	Звичайне підключення	VPN	TOR
Захист від провайдера	Видно весь трафік	Видно лише з’єднання з VPN-сервером	Не видно зміст трафіку, лише підключення
Шифрування трафіку	Немає	Повне	Багатошарове
Анонімність IP	Відсутня – видно справжню IP-адресу	IP замінюється на IP сервера VPN	IP постійно змінюється
Швидкість	Висока	Трохи нижча за звичайне підключення	Повільна
Захист від провайдера	Видно весь трафік	Видно лише з’єднання з VPN-сервером	Не видно зміст трафіку, лише підключення
Захист від атак посередника	Мінімальний	Високий	Високий
Доступ до заблокованих сайтів	Немає	Є	Є
Захист метаданих	Відсутній	Частковий	Частковий
Простота використання	Проста	Середня	Складна
Цільове використання	Повсякденне користування	Безпечний доступ до інтернету, обходи цензури	Максимальна анонімність, обходи цензури
Можливі недоліки	Слабкий захист	Може зберігати логи	Дуже повільний

2 ТЕОРЕТИЧНІ АСПЕКТИ ЗАХИСТУ ІНФОРМАЦІЇ

2.1 Поняття захисту інформації

Будь яка інформація має три основні властивості які підлягають захисту: конфіденційність, цілісність і доступність (Див. Рисунок 2.1).

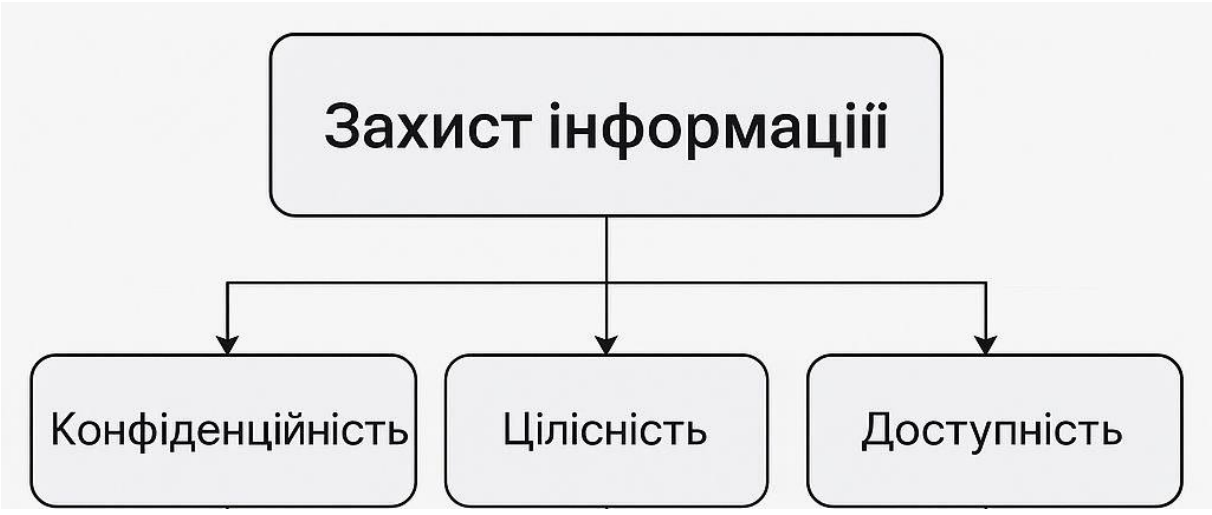

Рисунок 2.1 – Властивості інформації які захищаються

Метою захисту інформації є запобігання незаконному ознайомленню з інформацією, її зміні або обмеженню доступу до неї. Будь-яка форма інформації, включаючи дані і програми з комп'ютерних систем, протиправні дії щодо власника, користувача та інших осіб, можуть призвести до небезпечних втрат у науці, техніці, управлінні, економіці та підприємницькій діяльності, і все це підлягає захисту.

Захист інформації передбачає визначення можливих каналів її витоку, оцінювання важливості самої інформації та розроблення заходів для попередження її витоку й викрадення. Визначення потенційної цінності інформації необхідне, аби забезпечити захист найважливіших даних, витік яких може завдати найбільших збитків.

Найрізноманітніші технічні, а особливо електронні засоби виробничої та трудової діяльності, засоби зв’язку і допоміжні засоби становлять небезпеку виникнення каналів витоку інформації.

Досвід показує, що ефективним може бути лише комплексний захист інформації – поєднання законодавчих, морально-етичних, фізичних, адміністративних, технічних, криптографічних та програмних заходів [9].

У зв’язку з розвитком цифрових технологій постійно з’являються нові загрози, які можуть спричинити витік або втрату важливих даних. Саме тому захист інформації – це не одноразовий захід, а безперервний процес, який потребує регулярного оновлення методів і засобів безпеки. Важливо також враховувати людський фактор, адже більшість витоків і порушень стається через помилки або недбалість користувачів. Ефективний захист можливий лише тоді, коли всі учасники процесу розуміють свою відповідальність і дотримуються встановлених правил.

2.2 Історія створення криптографічних методів захисту інформації

Відомо, що принаймні деякі стародавні культури покладалися на примітивну криптографію для захисту, і більшість ранніх суспільств, схоже, пробували її. І стародавні єгиптяни, і месопотамці використовували підстановку символів, яка є основою криптографії, у своєму письмі. Вперше цей вид криптографії був знайдений у гробниці Хнумхотепа II, єгипетського вельможі, який жив понад 3900 років тому.

Причина заміни символів у написах Хнумхотепа полягала в тому, щоб зробити письмо більш привабливим з мовної точки зору. Близько 3500 років тому криптографія вперше була використана для захисту формули глазурі, написаної на глиняних табличках писцем, який працював у стародавній Месопотамії.

У пізні античні часи криптографія використовувалася для захисту військової інформації, і вона використовується для подібних цілей і сьогодні. Спартанці моделювали свою криптографію на вірі в те, що без відповідного розміру циліндра одержувач не зможе прочитати повідомлення. Закодовані повідомлення використовували шпигуни в Стародавній Індії вже у 2 столітті до нашої ери.

Ймовірно, що римські технології досягли вершини витонченості в криптографії, яка була відома ще в античні часи. Одним з відомих прикладів римської криптографії є шифр Цезаря, який переміщував літери в зашифрованому повідомленні на позиції на кілька пробілів нижче в латинському алфавіті. Якщо одержувач розумів, як працює гра і скільки існує комбінацій, він міг відкрити повідомлення.

Криптографія набула значного розвитку в Середньовіччі, хоча шифр Цезаря залишався типовим способом надсилання та отримання повідомлень. Процес зламу кодів і шифрів, який називається криптоаналізом, з'явився на основі простих методів криптографії. Близько 800 року н.е. арабський математик Аль-Кінді винайшов частотний аналіз, який означав, що шифри заміни можна легко розшифрувати. Завдяки цьому методу люди могли розгадувати зашифровані повідомлення крок за кроком, що призвело до збільшення використання і зростання криптографії.

У 1465 році Леоне Альберті винайшов поліалфавітний шифр, який вирішив проблему частотного аналізу Аль-Кінді. Поліалфавітний шифр кодує повідомлення, змінюючи і використовуючи кілька різних алфавітів. Алфавіт, в якому повідомлення з'являється у формі, називається першим алфавітом, а алфавіт закодованого повідомлення – наступним. Поліалфавітні шифри, що використовуються разом з традиційними шифрами заміни, роблять інформацію набагато більш захищеною. Якщо хтось не може прочитати початковий алфавіт, використання частотного аналізу не допоможе.

Двійкове кодування, одна з основних нових систем кодування, була створена ерудитом сером Френсісом Беконом у 1623 році в епоху Відродження.

З плином часу сфера криптографії дуже швидко розвивалася. У 1790-х роках Томас Джефферсон представив ідею криптографії, але вона, можливо, не була зрозуміла в той час. Це був його винахід – шифрувальне колесо з 36 кільцями з літерами на обертових колесах, які можна було обертати для виконання складного кодування. Ця ідея була настільки досконалою, що керувала американською військовою криптографією аж до Другої світової війни.

Під час Другої світової війни використовувався відомий приклад аналогової криптографії – машина «Енігма». Так само, як і шифрувальне колесо, ця машина «Енігма» використовувала колеса, що оберталися, щоб зашифрувати повідомлення так, щоб ніхто не зміг його прочитати без іншої машини «Енігма». Користувачі ранніх комп'ютерів врешті-решт змогли розгадати шифр «Енігма». Перемога союзників значною мірою залежала від їхньої здатності розшифровувати повідомлення «Енігми».

Завдяки комп’ютерам шифрування покращилося порівняно з усім, що використовувалося в минулому. Більшість сучасних пристроїв і комп’ютерів зараз покладаються на 128-бітове математичне шифрування, яке є дуже безпечним у порівнянні зі старими або середньовічними методами.

Раніше основна увага була направлена на фізичні загрози. Зараз більшість злочинців проводять атака дистанційно. При цьому вони можуть проводити як звичайні атаки, так і багатовекторні. При багатовекторній атаці існує одразу декілька загроз. Вона може включати в себе соціальну інженерію, шкідливе програмне забезпечення а також вразливості програмного забезпечення. Окрім цього традиційні методи шифрування стали менш ефективними. Це сталося через те, що зараз можливо використовувати квантові обчислення. Через це потрібно розробляти нові алгоритми шифрування. Щоб більш ефективно протидіяти цим загрозам, країни активно співпрацюють між собою. Після 1990 року вчені почали розробляти новий тип криптографії під назвою квантова криптографія, щоб ще більше підвищити безпеку сучасного шифрування. За останні 4000 років криптографія значно розвинулася і, як очікується, продовжить рухатися вперед. Якщо приватні дані все ще потребують захисту, використання криптографії буде продовжувати розвиватися. Криптовалюти блокчейн використовують одні з найсучасніших криптографічних інструментів на сьогоднішній день. Вони належать до традицій того, як люди жили в минулому [8].

2.3 Огляд шифру Цезаря та шифру Віженера

Шифр Цезаря – це простий, але ефективний спосіб приховати інформацію. Так, його легко зламати, але він є одним з початкових у вивченні криптографії. З його допомогою людина може легко зрозуміти у чому полягає суть криптографії.

Перед тим як щось зашифрувати, треба мати якийсь алфавіт, з яким ми будемо працювати. В цьому випадку використовується український алфавіт без літер "Є", "І", "Ї" та інших – він виглядає так: АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЬЫЭЮЯ.

Текст, який ми шифруємо, спочатку переводиться у верхній регістр – тобто всі літери стають великими. Далі починається головне. Кожна літера з тексту перевіряється. Якщо вона є в алфавіті, то ми дізнаємось її позицію (номер). Потім до цієї позиції додається ключ, і ми отримуємо нову позицію. За цією новою позицією ми беремо літеру з алфавіту і додаємо її до нового, зашифрованого тексту. Якщо символ не входить в алфавіт, він просто додається без змін. В результаті текст "ОДЕСА" перетворюється на "СЗИФГ" (Див. Рисунок 2.2).

```
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
Зашифрований текст: СЗИФГ
PS C:\Users\38095>
```

Рисунок 2.2 – Результат роботи шифру «Цезаря»

Після того як зашифроване повідомлення було передано, його треба розшифрувати. Для цього використовують код розшифрування. Все працює майже так само, тільки у зворотному напрямку. Маємо зашифрований текст, і той самий алфавіт. Для кожної літери ми знову шукаємо її позицію в алфавіті. Але цього разу ми не додаємо ключ, а віднімаємо. Отримуємо нову позицію, знаходимо відповідну літеру і додаємо її до розшифрованого рядка. Символи,

яких немає в алфавіті, знову лишаються без змін. У результаті з рядка "СЗИФГ" знову отримуємо "ОДЕСА" (Див. Рисунок 2.3).

```
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
Розшифрований текст: ОДЕСА
PS C:\Users\38095>
```

Рисунок 2.3 – Результат дешифрування шифру «Цезаря»

Шифр Віженера – це як Цезар, тільки трохи складніше. Тут для кожної літери в повідомленні використовується свій зсув, який береться із ключа. Тобто, якщо в Цезаря один ключ, то тут багато, і вони повторюються. В ньому використовують той самий алфавіт: АБВГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ.

Повідомлення і ключ перетворюються у верхній регістр, щоб усе було однаково. Далі потрібно зробити так, щоб ключ був тієї ж довжини, що і повідомлення. Просто копіюємо його багато разів і, якщо треба, додаємо ще кілька букв з початку. Так виходить "повторений ключ". Тепер кожну літеру шифруємо.

Дивимось, яка вона за номером в алфавіті. Беремо літеру з ключа на тому ж місці і дивимось її номер. Складаємо ці два номери. Якщо вийшло більше, ніж довжина алфавіту – починаємо з початку. Додаємо нову літеру до зашифрованого повідомлення.

У результаті, з тексту "ОДЕСА" і ключа "КІБЕРБЕЗПЕКА", отримуємо щось типу: "ЮЛЄЦР" (Див. Рисунок 2.4).

```
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
Зашифроване повідомлення: ЮЛЄЦР
PS C:\Users\38095>
```

Рисунок 2.4 – Результат роботи шифру «Віженера»

А тепер розшифрування. Все те саме, тільки замість додавання віднімаємо. Повторюємо ключ як і в шифруванні. Для кожної літери дивимось її номер в алфавіті. Віднімаємо номер відповідної літери з ключа. Якщо виходить менше нуля теж працюємо по модулю. Знаходимо букву і додаємо до розшифрованого тексту.

Так з "ЮЛЄЦР" і ключа "КІБЕРБЕЗПЕКА" виходить знову "ОДЕСА" (Див. Рисунок 2.5).

```
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
Розшифроване повідомлення: ОДЕСА
PS C:\Users\38095>
```

Рисунок 2.5 – Результат дешифрування шифру «Віженера»

2.4 Міжнародні стандарти у сфері захисту інформації

Почнемо з найбільш відомих – стандартів, розроблених Міжнародною організацією зі стандартизації (ISO) спільно з Міжнародною електротехнічною комісією (IEC).

ISO/IEC 27001 – це основний документ, який описує, як правильно будувати систему управління інформаційною безпекою. У ньому йдеться не лише про захист серверів чи баз даних, а й про безпечну передачу даних між користувачами або сервісами. Наприклад, стандарт передбачає, що чутлива інформація має передаватися через зашифровані канали, з автентифікацією одержувача [3].

Його доповнення – ISO/IEC 27002 – подає конкретні приклади заходів захисту. Зокрема, це рекомендації щодо використання паролів, налаштування мережевого обладнання, обмеження доступу до даних [4].

Ще один важливий документ – ISO/IEC 27033, що більш глибоко занурюється у питання захисту комп’ютерних мереж. Тут докладно описано, як захищати інформацію під час її передавання – від використання VPN до шифрування IP-пакетів [5].

Надійний захист інформації неможливий без криптографії. У США цю сферу регулює набір стандартів FIPS, що публікується Національним інститутом стандартів і технологій (NIST). Особливо важливим є FIPS 140-3, який встановлює вимоги до криптомодулів, тобто спеціального програмного або апаратного забезпечення для шифрування.

Окрім того, практично всі сучасні браузери, банківські системи та хмарні сервіси використовують протоколи TLS (Transport Layer Security), описані в документах RFC. Наприклад, RFC 8446 визначає сучасну версію TLS 1.3, яка значно підвищила рівень захисту даних у порівнянні з попередніми версіями [18].

Якщо говорити про стандарти, що регулюють передачу інформації через Wi-Fi, то тут слід згадати серію IEEE 802.11. У цих документах описані правила функціонування бездротових мереж, включаючи протоколи шифрування – зокрема WPA2 та WPA3. Вони запобігають перехопленню трафіку та несанкціонованому доступу до мережі.

Ще один вагомий механізм – IPSec. Це набір протоколів, які забезпечують шифрування та автентифікацію IP-трафіку. Його часто використовують у віртуальних приватних мережах (VPN), особливо в корпоративному середовищі.

У країнах Європейського Союзу діє загальний регламент із захисту даних – GDPR. Він не лише визначає, які дані вважаються персональними, а й вимагає від компаній вживати технічних заходів для їхнього захисту під час передавання. Наприклад, шифрування та псевдонімізація вважаються базовими інструментами.

В Україні функціонує Закон «Про захист інформації в інформаційно-комунікаційних системах», який зобов’язує державні та приватні організації використовувати сертифіковані засоби захисту інформації. До них входять криптографічні засоби, системи контролю доступу, а також механізми моніторингу безпеки мереж [2].

Також в Україні діє національний стандарт ДСТУ ISO/IEC 27001, який практично повністю відповідає міжнародному аналогу й часто є умовою для

участі організацій у державних тендерах або співпраці з іноземними партнерами.

Наявність стандартів не є самоціллю. Їхнє головне завдання – створити єдиний підхід до забезпечення безпеки, якого дотримуються усі – від малих компаній до державних установ. Стандарти дозволяють не тільки мінімізувати ризики втрати або перехоплення даних, а й формують довіру між сторонами, що обмінюються інформацією.

Організації, які впроваджують ці стандарти, часто проходять сертифікацію. Це дозволяє не лише поліпшити репутацію, а й відкрити доступ до нових ринків та партнерств.

2.5 Мережеві екрани

Терміни брандмауер, брандмауер і фаєрвол означають одне й те саме. Єдина відмінність полягає в тому, що ці слова написані двома різними мовами.

Брандмауер поєднує в собі апаратне і програмне забезпечення для перевірки трафіку, що надходить ззовні і всередині мережі. Основне завдання брандмауера – переглядати трафік і захищати дані.

Хакери вигадують нові способи крадіжки нашої приватної інформації, а віруси все частіше зустрічаються в Інтернеті. Фірми шукають вдосконалені системи, щоб уникнути кібератак. Крім того, все більше людей звертаються до брандмауерів.

За мить кожен комп'ютер, що використовує всесвітню мережу, буде заражений вірусами і стане непридатним для використання, якщо на ньому не буде брандмауера. У великій компанії ризик для локальної мережі ще вищий. Вірус швидко заражає всі комп'ютери в компанії і заважає кожному співробітнику виконувати свою роботу. Втрата даних може призвести до того, що компанія втратить дохід.

Побудова корпоративної мережі завжди вимагає додавання брандмауера. Важливою метою криптографії є запобігання витоку конфіденційної інформації.

Крім запобігання, брандмауер буде вести журнал, в якому будуть відображатися всі незвичайні події у вашій мережі.

Звичайні функції брандмауера включають в себе:

- Блокування операцій, які можуть отримати доступ до обмежених або незахищених частин системи.
- Заборона будь-кому бачити конфіденційну інформацію та гарантування того, що захищені дані не передаються;
- Обмеження того, хто може підключати пристрої та отримувати доступ до мережевих портів;
- Спостереження за тим, що відбувається в мережі, та збереження результатів для статистики;
- Сповіщення користувачів про підозри щодо будь-яких загроз або мережевих атак.

Сучасні брандмауери не обмежуються простим блокуванням трафіку. Вони здатні аналізувати вміст мережевих пакетів і виявляти підозрілу активність ще до того, як вона завдасть шкоди. У багатьох випадках такі системи працюють разом з антивірусами або іншими інструментами захисту, створюючи багаторівневу систему безпеки. Це особливо важливо для компаній, де витік інформації може спричинити значні фінансові та репутаційні втрати. Брандмауер стає не просто бар’єром, а активним учасником виявлення та протидії загрозам.

Принцип дії мережевого екрану зображено на рисунку (Див. Рисунок 2.6).

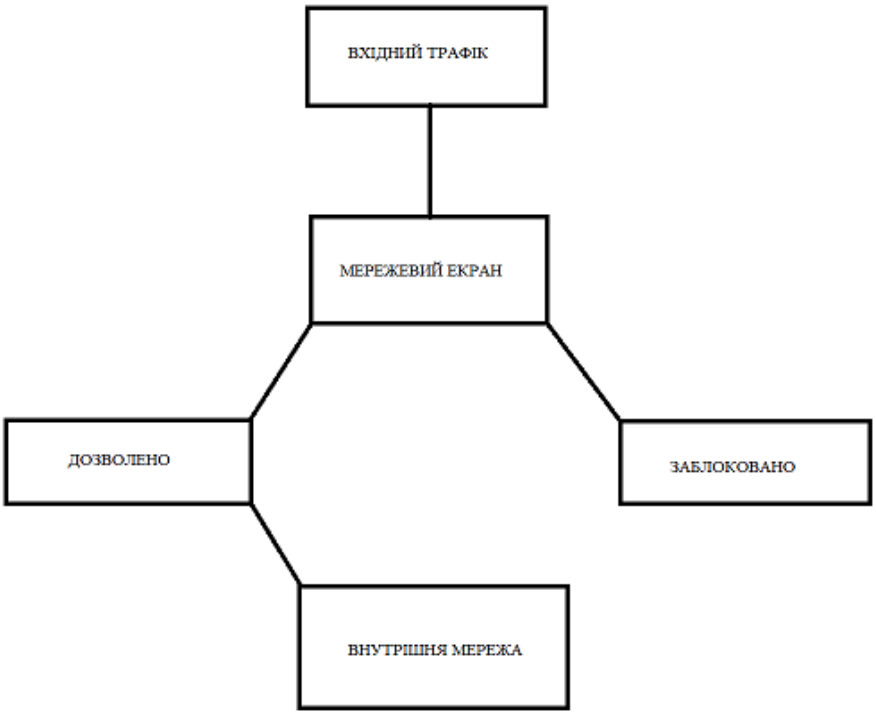

Рисунок 2.6 – Схема роботи мережевого екрану

2.6 Сучасні методи захисту переданої інформації

Криптографія використовується для захисту інформації при передачі по незахищеним каналам зв’язку. В криптографії використовують різні підходи до шифрування інформації.

В симетричному шифруванні використовується лише один ключ. Завдяки цьому воно є більш швидким. Це дозволяє ефективно захищати велику кількість даних.

Симетричне шифрування складається з декількох етапів. З початку генерується ключ фіксованої довжини. Потім починається процес шифрування повідомлення. Текст шифрується за допомогою секретного ключа та крипто-алгоритму. Після цього він передається каналами зв’язку і отримувач дешифрує його за допомогою ключа, який використовувався при шифруванні.

Одними з перших алгоритмів шифрування були DES та 3DES. Вони використовували ключ довжиною лише 56 біт. Різниця між ними полягала в тому, що 3DES три рази послідовно шифрував та дешифрував інформацію. Але

через таку малу довжину ключа цей алгоритм шифрування неефективний і зараз майже ніде не використовується.

Також одним з прикладів симетричних алгоритмів шифрування є AES. Він може використовувати ключі довжиною 128, 192 та 256 біт. Це робить його стійким до крипто-аналітичних атак. Найчастіше його використовують для захисту конфіденційної інформації.

Ще одним прикладом симетричних алгоритмів є ChaCha20. Він більш продуктивний ніж AES.Тому його використовують в мобільних пристроях та VPN.

Симетричне шифрування використовують в різних сферах. В VPN воно використовується у протоколах (OpenVPN, IPSec) які захищають інформацію при передачі. Також його використовують для шифрування даних у мережах Wi-Fi. Програма VeraCrypt використовує його для того, щоб захистити дані які зберігаються на дисках.

Основним недоліком симетричного шифрування є те, що відправник і отримувач мають обмінюватись ключами. Це може стати потенційною загрозою. Якщо зловмисник отримає доступ до ключа то зможе отримати доступ до всіх даних.

На відміну від симетричного шифрування, при асиметричному обмінюватися ключами не потрібно. Асиметричні алгоритми шифрування використовують одразу два ключа. Відкритий ключ використовується для шифрування інформації, а закритий для дешифрування. Через це асиметричне шифрування є більш надійним. Однак у асиметричного шифрування теж є недолік. Через складність обчислювання, процес передачі даних уповільнюється. Це робить його незручним при використанні з великим обсягом даних.

Як і симетричне шифрування, асиметричне проходить в декілька етапів. Генеруються ключі, в кожного користувача є свій закритий ключ без якого розшифрувати повідомлення не вийде. Потім, користувач який відправляє повідомлення, шифрує його із використанням відкритого ключа що є в отримувача. Повідомлення каналом зв’язку доходить до отримувача і той

розшифровує його використовуючи свій закритий ключ. Таким чином навіть якщо повідомлення попаде в чужі руки, розшифрувати його невдасться. Також можна користуватися спільним закритим ключем. Це можливо якщо використовувати алгоритм Діффі-Геллмана, але це потребує додаткових засобів захисту.

Гарними прикладами асиметричних шифрів є RSA та ECC. Шифр RSA працює з більш великими ключами. Довжина ключа може сягати 4096 біт. Він часто використовується в електронних підписах та протоколах безпечної передачі даних. Прикладом таких протоколів є HTTPS. Шифр ECC використовує більш малі ключі порівняно з RSA. Це робить його більш швидким, тому він частіше використовується в мобільних пристроях.

Таблиця 2 – Порівняння симетричного та асиметричного шифрування

Властивість	Симетричне шифрування	Асиметричне шифрування
Кількість ключів	Один	Два
Швидкість	Висока	Низька
Безпека	Висока	Висока
Сфери застосування	VPN, Wi-Fi, дискове шифрування	Електронний підпис, автентифікація, протоколи безпечної передачі даних
Проблеми	Обмін ключами між відправником та отримувачем	Низька швидкість передачі даних

Після того як інформація була передана, її треба перевірити на цілісність. Для цього використовують хешування (Див. Рисунок 2.7).

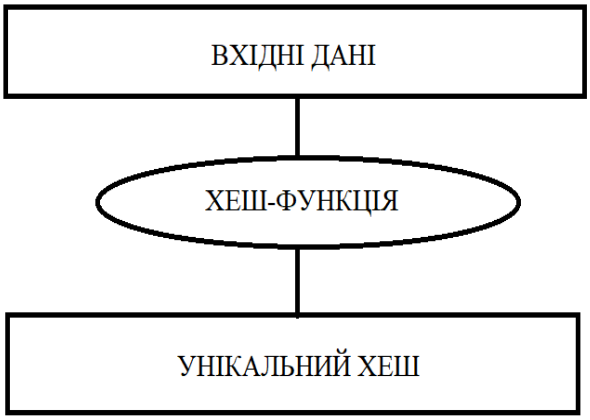

Рисунок 2.7 – Етапи хешування

В процесі хешування утворюється такий хеш, який неможливо отримати хешуванням інших даних. Якщо дані було змінено то і хеш зміниться також, тому хешування є надійним способом перевірки інформації на цілісність. Хеші також можуть бути різного розміру. Наприклад MD5 має довжину хешу 128 біт, SHA-1 – 160 біт. Вони не дуже надійні. Для порівняння SHA-2 та SHA-3 можуть мати довжину хешу до 512 біт, відповідно вони більш надійні ніж MD5 та SHA1. Зараз SHA-2 часто використовують в блокчейні, цифрових підписах та протоколах безпечної передачі даних.

Таблиця 3 – Порівняння довжини хешу в хеш-функціях

Назва хеш-функції	Довжина хешу
MD5	128 біт
SHA-1	160 біт
SHA-2	256 біт, 512 біт
SHA-3	224 біт, 256 біт, 384 біт, 512 біт

Електронні підписи поєднують в собі хешування та асиметричне шифрування. Для створення електронного підпису з початку роблять хешування потрібного документу. Потім отриманий хеш шифрують використовуючи закритий ключ користувача, який підписує документ. Після цього цей підпис можна додати до документу. Для того, щоб перевірити підпис,

треба порівняти хеш документу з хешем, який вийде після розшифрування підпису. Вони мають співпадати. Для шифрування електронного підпису найчастіше використовують RSA. Використовуючи хеш-функцію можна побачити, чи було змінено дані, а електронний підпис використовують для перевірки автентичності.

Для того щоб захистити дані при передачі по мережі використовують спеціальні протоколи. Під час з’єднання між клієнтом та сервером працює протокол TLS. Його використовують у веб-браузерах, VPN та в електронній пошті. Цей протокол призначений для шифрування даних та автентифікації користувачів (клієнти, сервери). Він поєднує симетричне та асиметричне шифрування. За рахунок цього він може гарантувати захист від атак «людини посередині». Таким чином він захищає конфіденційність користувачів.

Протокол IPSec призначений для захисту IP-трафіку. В нього є два режими роботи, транспортний та тунельний. Транспортний шифрує лише частину пакету, а тунельний шифрує весь пакет. Його використовують в корпоративних мережах а також під час VPN-з’єднання.

Для адміністрування серверів та безпечної передачі даних використовують протокол SSH. В ньому поєднується асиметричне шифрування, симетричне шифрування та хешування.

3 РОЗРОБКА ПРОГРАМ ШИФРУВАННЯ ТА АВТЕНТИФІКАЦІЇ

3.1 Удосконалення шифру «Цезаря»

Шифр Цезаря – один із найдавніших криптографічних алгоритмів, що належить до групи шифрів прямої заміни. Він полягає в тому, що кожна літера повідомлення зміщується в алфавіті на фіксовану кількість позицій. Незважаючи на свою простоту, цей метод має історичне значення і демонструє базовий принцип симетричного шифрування. Проте в сучасних умовах такий шифр легко піддається зламу методом повного перебору (brute force), оскільки кількість можливих ключів дуже мала (наприклад, 32 варіанти в українському алфавіті).

Щоб підвищити стійкість до зламу, було вирішено вдосконалити цей шифр шляхом генерації випадкового зсуву для кожного символу, тобто зробити ключ змінним по всій довжині повідомлення. Це дозволяє частково імітувати поведінку складніших симетричних алгоритмів, зберігаючи при цьому легкість реалізації.

Суть удосконалення полягає в тому, що в звичайному шифрі «Цезаря» кожна літера зсувається на фіксовану кількість позицій, а в удосконаленому кожна з літер зсувається на випадкову кількість позицій. Це робить його більш стійким до зламу.

Спочатку береться повідомлення яке треба зашифрувати та генерується випадковий ключ довжина якого дорівнює довжині повідомлення. Робиться це за допомоги наступної програми:

```
import random
def generate_random_key_for_message(message):
    alphabet = 'АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ'
    alphabet += alphabet.lower()
    key_digits = []
    for char in message:
        if char in alphabet:
            digit = random.randint(1, 9)  # 0 не
використовуємо, бо не дає зсуву
            key_digits.append(str(digit))
    return ''.join(key_digits)
message = 'Я люблю своє місто'
```

```
random_key = generate_random_key_for_message(message)
print("Згенерований ключ:", random_key)
```

В результаті роботи цієї програми отримаємо випадковий ключ (Див. Рисунок 3.1).

```
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
Згенерований ключ: 957474957642182
PS C:\Users\38095>
```

Рисунок 3.1 – Випадковий ключ для удосконаленого шифру «Цезаря»

Коли випадковий ключ згенеровано можна приступати до самого процесу шифрування. Як було сказано раніше кожна з літер зсувається на випадкову кількість позицій. Ця випадкова кількість позицій залежить від того з якою цифрою ключа співпадає літера. Результат шифрування (Див. Рисунок 3.2).

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
Зашифрований текст: Б рєдтб уєчі укучч
PS C:\Users\38095>
```

Рисунок 3.2 – Результат шифрування удосконаленим шифром «Цезаря»

Реалізується це за допомогою такої програми:

```
def caesar_advanced_encrypt(text, key):
    alphabet = 'АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ'
    alphabet_lower = alphabet.lower()
    key_digits = [int(d) for d in str(key)]
    encrypted_text = ''
    key_index = 0
    for char in text:
        if char.upper() in alphabet:
            is_upper = char.isupper()
            alph = alphabet if is_upper else
alphabet_lower
```

```
            index = alph.find(char)
            shift = key_digits[key_index %
len(key_digits)]
            new_index = (index + shift) % len(alph)
            encrypted_text += alph[new_index]
            key_index += 1
        else:
            encrypted_text += char  # залишаємо пробіли
та розділові знаки без змін
    return encrypted_text
text = 'Я люблю своє місто'
key = 259473
encrypted = caesar_advanced_encrypt(text, key)
print("Зашифрований текст:", encrypted)
```

Коли зашифроване повідомлення дійде до отримувача він зможе розшифрувати його без проблем лише якщо знатиме ключ шифрування. Результат розшифрування (Див. Рисунок 3.3)

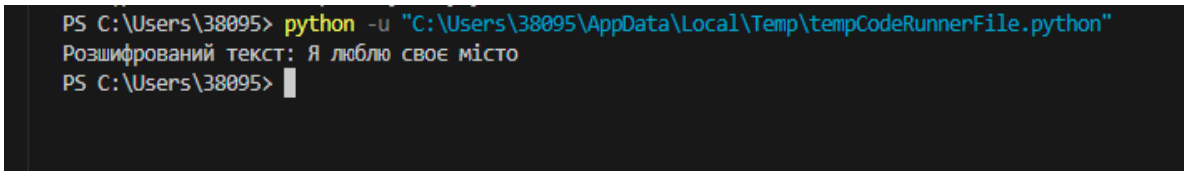

Рисунок 3.3 – Результат роботи програми з розшифрування удосконаленого шифру «Цезаря»

Розшифровується воно наступною програмою:

```
def caesar_advanced_decrypt(text, key):
    alphabet = 'АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ'
    alphabet_lower = alphabet.lower()
    key_digits = [int(d) for d in str(key)]
    decrypted_text = ''
    key_index = 0
    for char in text:
        if char.upper() in alphabet:
            is_upper = char.isupper()
            alph = alphabet if is_upper else
alphabet_lower
            index = alph.find(char)
            shift = key_digits[key_index %
len(key_digits)]
            new_index = (index - shift) % len(alph)
```

```
            decrypted_text += alph[new_index]
            key_index += 1
        else:
            decrypted_text += char
    return decrypted_text
encrypted_text = 'Б рєдтб уєчі укучч'
key = 259473
decrypted = caesar_advanced_decrypt(encrypted_text, key)
print("Розшифрований текст:", decrypted)
```

Програма, реалізована в межах цього підрозділу, вирішує проблему низької криптостійкості шифру Цезаря. За рахунок використання масиву зсувів (тобто змінного ключа), вона ускладнює аналіз частот і значно підвищує час, необхідний для зламу методом перебору. Цей підхід зберігає основну перевагу симетричних алгоритмів – швидкість шифрування та дешифрування, адже симетричні шифри, на відміну від асиметричних, менш ресурсоємні та не вимагають великої обчислювальної потужності для обробки тексту [19].

Таким чином, хоча запропонований варіант не може повністю конкурувати із сучасними стандартами шифрування, він чудово підходить для демонстрації ключових принципів криптографії та дає змогу наочно побачити, як навіть невелике ускладнення алгоритму може підвищити рівень захисту даних.

3.2 Програма «Для реєстрації та входу»

Зберігання паролів у відкритому вигляді – одна з найгрубіших і водночас найпоширеніших помилок у програмуванні. Навіть якщо база даних потрапляє до рук зловмисника, добре реалізоване хешування дозволяє зберегти конфіденційність користувацьких даних. Проте звичайне хешування (наприклад, за допомогою SHA-256) вже недостатнє – у мережі існують цілі таблиці готових хешів (так звані rainbow tables), які дозволяють миттєво підібрати відповідність.

Щоб запобігти цьому, у сучасній практиці використовується сіль (salt) – випадкова послідовність байтів, яка додається до пароля перед хешуванням. Це

робить кожен хеш унікальним, навіть якщо декілька користувачів мають однаковий пароль. У цьому підпункті реалізовано програму, що виконує реєстрацію та авторизацію користувачів із використанням сіллю, хешуванням та збереженням інформації в базі даних.

Програма дозволяє користувачам створювати акаунти, перевіряти правильність введених паролів та здійснювати входи в систему. Вона використовує паролі з сіллю та збереження даних у базі даних SQLite. Для кожного користувача генерується унікальна сіль, яка додається до пароля перед його хешуванням за допомогою алгоритму SHA-1. Дані про користувачів зберігаються в базі даних, де є поля для імені користувача, хешу пароля, солі та дати реєстрації.

При запуску програма просить обрати одну з дій (Див. Рисунок 3.4).

```
PS C:\Users\38095> python -u "C:\Users\38095\AppData\Local\Temp\tempCodeRunnerFile.python"
База даних та таблиця створено (якщо їх не було).
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії:
```

Рисунок 3.4 – Перший запуск програми «Для реєстрації та входу»

Оскільки це перший запуск, одразу увійти не вдасться. Якщо спробувати увійти одразу то програма скаже що пароль не вірний та поверне користувача в меню вибору дії (Див. Рисунок 3.5).

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії: 1
Введіть ім'я користувача: Олександр
Введіть пароль: 7912
Невірний пароль.
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії:
```

Рисунок 3.5 – Невдала спроба входу

Тож спочатку треба буде зареєструватися. Звернути увагу варто на те, що програма не дозволяє користувачам мати однакове ім’я (Див. Рисунок 3.6)

тому, якщо ім’я зайняте потрібно буде вигадати щось нове. Я буду використовувати ім’я «ОДЕСА» та пароль «1397». Після того як нове ім’я та пароль були вигадані можна приступати до реєстрації. Процес реєстрації (Див. Рисунок 3.7).

```
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії: 2
Введіть ім'я користувача: Олександр
Користувач з ім'ям Олександр вже існує!
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії:
```

Рисунок 3.6 – Спроба реєстрації під вже існуючим ім’ям

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії: 2
Введіть ім'я користувача: ОДЕСА
Введіть пароль: 1397
Користувач ОДЕСА зареєстрований!
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії:
```

Рисунок 3.7 – Процес реєстрації користувача

Після того як користувача було зареєстровано, він може увійти без проблем, а після входу програма пропанує користувачу вийти з акаунту (Див. Рисунок 3.8).

```
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії: 1
Введіть ім'я користувача: ОДЕСА
Введіть пароль: 1397
Вхід до системи успішний! Добро пожаловать, ОДЕСА.
Бажаєте вийти з акаунту? (так/ні):
```

Рисунок 3.8 – Вхід до акаунту

Окрім цього в програмі ще є функція «Застарілий пароль». Для перевірки працездатності цієї функції код було трохи змінено (Див. Рисунок 3.9-3.10).

```
if datetime.now() - registration_date > timedelta(seconds=300000000000):
    print(f"Ваш пароль застарів! Необхідно змінити пароль.")
else:
    print(f"Вхід до системи успішний! Добро пожаловать, {username}.")
```

Рисунок 3.9 – Код до змін

```
if datetime.now() - registration_date > timedelta(seconds=30):  # Змінили для тестування
    print(f"Ваш пароль застарів! Необхідно змінити пароль.")
else:
    print(f"Вхід до системи успішний! Добро пожаловать, {username}.")
```

Рисунок 3.10 – Код після змін

Для перевірки працездатності програму треба перезапустити тому обираємо варіант (так-3), програма успішно вимикається (Див. Рисунок 3.11). Одразу після вимкнення, запускаємо її знову, вже з зміненим кодом. На цей раз при спробі увійти в акаунт, вона каже що код застарілий (Див. Рисунок 3.12).

```
Бажаєте вийти з акаунту? (так/ні): так
Вихід з акаунту.
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії: 3
До побачення!
PS C:\Users\38095>
```

Рисунок 3.11 – Вихід з акаунту

```
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії: 1
Введіть ім'я користувача: ОДЕСА
Введіть пароль: 1397
Ваш пароль застарів! Необхідно змінити пароль.
Виберіть дію:
1. Вхід
2. Реєстрація
3. Вихід
Введіть номер дії:
```

Рисунок 3.12 – Перевірка працездатності функції «Застарілий пароль»

Основна мета програми – демонстрація на практиці, як правильно зберігати паролі, аби мінімізувати ризик компрометації даних. Проблема, яку вона вирішує, полягає у відсутності безпечного механізму автентифікації в багатьох простих вебпроєктах та додатках. Програма реалізує алгоритм, який генерує унікальну сіль для кожного користувача, додає її до пароля, а потім хешує комбінацію та зберігає її в базі. Під час входу порівнюється новий хеш із тим, що зберігається, що виключає необхідність зберігати сам пароль.

Це рішення базується на сучасних практиках безпеки, описаних, зокрема, у рекомендаціях OWASP [20]. Воно дозволяє захистити облікові записи навіть у разі витоку бази даних, і тому може бути легко масштабоване для використання у справжніх проєктах.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи на тему «Технічні аспекти безпечної передачі інформації в кіберпросторі» було здійснено комплексне дослідження ключових методів захисту інформації, що передається по цифрових каналах зв’язку. Робота поєднує теоретичний аналіз сучасних технологій інформаційної безпеки та практичну реалізацію власних програмних рішень, що дозволяють підвищити стійкість до кіберзагроз.

У першому розділі було охарактеризовано структуру кіберпростору, виявлено основні джерела загроз при передачі даних, а також проаналізовано вразливості, пов’язані з протоколами з'єднання, DNS-запитами, діями інтернет-провайдерів та людським фактором. Зроблено висновок, що навіть базові мережеві процеси, без належного шифрування та автентифікації, можуть бути використані зловмисниками для втручання в передавання інформації.

У другому розділі досліджено теоретичні основи технічного захисту інформації. Проаналізовано еволюцію криптографічних методів – від історичних шифрів до сучасних алгоритмів, таких як AES, RSA, ECC, а також принципи використання хеш-функцій і цифрових підписів.

Особливу увагу приділено сучасним міжнародним стандартам (ISO/IEC, FIPS, GDPR) та їхньому значенню для побудови комплексної системи інформаційної безпеки.

У третьому розділі представлено реалізацію двох програмних рішень: удосконаленого шифру Цезаря з динамічним ключем та системи автентифікації користувачів із застосуванням хешування паролів з сіллю та збереженням даних у базі.

У результаті дослідження було підтверджено, що ефективний захист інформації в кіберпросторі потребує комплексного підходу, який об’єднує технічні, програмні та організаційні заходи. Тільки постійне вдосконалення засобів захисту, орієнтоване на сучасні виклики та загрози, може гарантувати

конфіденційність, цілісність і доступність інформації в умовах стрімкого розвитку цифрових технологій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Закон України «Про основні засади забезпечення кібербезпеки України» від 5 жовт. 2017р. Дата оновлення 20.04.2025р. № 2163-VIII URL: https://zakon.rada.gov.ua/laws/show/2163-19

2. Закон України «Про захист інформації в інформаційно-комунікаційних системах» від 5 лип. 1994р. Дата оновлення 20.04.2025р. № 80/94-ВР URL: https://zakon.rada.gov.ua/laws/show/80/94-вр

3. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection – Information security management systems – Requirements URL: https://www.iso.org/standard/82875.html

4. ISO/IEC 27002:2022. Information security, cybersecurity and privacy protection – Code of practice for information security controls URL: https://www.iso.org/standard/75652.html

5. ISO/IEC 27033. Information technology – Security techniques – Network security URL: https://www.iso.org/standard/44375.html

6. FIPS PUB 140-3. Security Requirements for Cryptographic Modules / National Institute of Standards and Technology (NIST), USA, 2019 URL: https://csrc.nist.gov/publications/detail/fips/140/3/final

7. General Data Protection Regulation (GDPR) URL: https://gdpr.eu

8. Binance Academy. The history of cryptography URL: https://academy.binance.com/uk-UA/articles/history-of-cryptography

9. Захист інформації // Енциклопедія сучасної України URL: https://esu.com.ua/article-15872

10. Як працює DNS / HostIQ URL: https://hostiq.ua/blog/ukr/how-does-dns-work/

11. HTTP та HTTPS: в чому різниця? / HostIQ URL: https://hostiq.ua/wiki/ukr/http-https/

12. Протокол TCP/IP або як працює інтернет для початківців / QALight URL: https://qalight.ua/baza-znaniy/protokol-tcp-ip-abo-yak-praczyuє-internet-dlya-pochatkivcziv/

13. Хто такі інтернет-провайдери і які функції вони виконують / VSN.ua URL: https://vsn.ua/news/hto-taki-internet-provayderi-i-yaki-funktsiyi-voni-vikonuyut-49822

14. Що таке міжмережевий екран і навіщо він потрібен / Stack Systems URL: https://stack-systems.com.ua/blogs/shtcho-take-mizmerezevyj-ekran-i-navishtcho-vin-potriben

15. Kaspersky. Атака «людина посередині»: як не стати жертвою? URL: https://www.kaspersky.ua/resource-center/definitions/man-in-the-middle-attack

16. CERT-UA. Що таке фішинг та як йому запобігти? URL: https://cert.gov.ua/article/1159865

17. Кіберполіція України. Що таке вірус-здирник WannaCry та як від нього захиститися URL: https://cyberpolice.gov.ua/news/wannacry-virus-ransomware/

18. RFC 8446. The Transport Layer Security (TLS) Protocol Version 1.3 [Електронний ресурс] / E. Rescorla ; Internet Engineering Task Force (IETF). – Серп. 2018. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc8446

19. Енциклопедія сучасної України. Шифр Цезаря URL: https://esu.com.ua/search_articles.php?id=46062

20. OWASP Foundation. Password Storage Cheat Sheet URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

Додаток А. Шифрування тексту шифром Цезаря

```
def caesar_cipher(text, key):
    alphabet = 'АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'
    text = text.upper()
    encrypted_text = ''
    for char in text:
        if char in alphabet:
            index = alphabet.index(char)
            new_index = (index + key) % len(alphabet)
            encrypted_text += alphabet[new_index]
        else:
            encrypted_text += char
    return encrypted_text
open_text = "ОДЕСА"
key = 3
encrypted = caesar_cipher(open_text, key)
print("Зашифрований текст:", encrypted)
```

Додаток Б. Шифрування тексту шифром Віженера

```
def vigenere_encrypt(message, key,
alphabet="АБВГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ"):
    encrypted_message = []
    alphabet_length = len(alphabet)
    # Перетворюємо повідомлення і ключ у верхній регістр
    message = message.upper()
    key = key.upper()
    repeated_key = (key * (len(message) // len(key))) +
key[:len(message) % len(key)]
    for i in range(len(message)):
        message_index = alphabet.index(message[i])
        key_index = alphabet.index(repeated_key[i])
        # Шифруємо символ за формулою Віженера
        encrypted_index = (message_index + key_index) %
alphabet_length
        encrypted_message.append(alphabet[encrypted_index])
    return ''.join(encrypted_message)
message = "ОДЕСА"
key = "КІБЕРБЕЗПЕКА"
encrypted_message = vigenere_encrypt(message, key)
print(f"Зашифроване повідомлення: {encrypted_message}")
```

Додаток В. Розшифрування шифру Цезаря

```
def caesar_decipher(cipher_text, key):
    alphabet = 'АБВГДЕЖЗИКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'
    cipher_text = cipher_text.upper()
    decrypted_text = ''
    for char in cipher_text:
        if char in alphabet:
            index = alphabet.index(char)
            new_index = (index - key) % len(alphabet)
            decrypted_text += alphabet[new_index]
        else:
            decrypted_text += char
    return decrypted_text
cipher_text = ""
key = 3
decrypted = caesar_decipher(cipher_text, key)
print("Розшифрований текст:", decrypted)
```

Додаток Г. Розшифрування шифру Віженера

```
def vigenere_decrypt(encrypted_message, key,
alphabet="АБВГДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ"):
    decrypted_message = []
    alphabet_length = len(alphabet)
    # Перетворюємо зашифроване повідомлення і ключ у верхній
регістр
    encrypted_message = encrypted_message.upper()
    key = key.upper()
    # Повторюємо ключ до довжини зашифрованого повідомлення
    repeated_key = (key * (len(encrypted_message) // len(key))) +
key[:len(encrypted_message) % len(key)]
    for i in range(len(encrypted_message)):
        # Отримуємо індекси символів у алфавіті
        encrypted_index = alphabet.index(encrypted_message[i])
        key_index = alphabet.index(repeated_key[i])
        # Дешифруємо символ за формулою Віженера
        decrypted_index = (encrypted_index - key_index) %
alphabet_length
        decrypted_message.append(alphabet[decrypted_index])
    return ''.join(decrypted_message)
encrypted_message = ""
key = "КІБЕРБЕЗПЕКА"
decrypted_message = vigenere_decrypt(encrypted_message, key)
print(f"Розшифроване повідомлення: {decrypted_message}")
```

Додаток Д. Програма «Для реєстрації та входу»

```
import hashlib
import os
import sqlite3
from datetime import datetime, timedelta
# Функція для хешування пароля з сіллю
def hash_password(password, salt):
    password_salt = password + salt
    return hashlib.sha1(password_salt.encode()).hexdigest()
# Створення нової бази даних та таблиці для користувачів
def create_database():
    try:
        conn = sqlite3.connect('users.db')
        c = conn.cursor()
        c.execute('''CREATE TABLE IF NOT EXISTS users (
                        username TEXT PRIMARY KEY,
                        password_hash TEXT,
                        salt TEXT,
                        registration_date TEXT
                    )''')
        conn.commit()
        conn.close()
        print("База даних та таблиця створено (якщо їх не було).")
    except Exception as e:
        print(f"Помилка при створенні бази даних: {e}")
# Перевірка наявності користувача
def user_exists(username):
    conn = sqlite3.connect('users.db')
    c = conn.cursor()
    c.execute("SELECT 1 FROM users WHERE username=?", (username,))
    exists = c.fetchone()
    conn.close()
    return exists is not None
# Реєстрація нового користувача
def register_user(username, password):
    if user_exists(username):
        print(f"Користувач з ім'ям {username} вже існує!")
        return
    try:
        salt = os.urandom(16).hex()  # Генерація випадкової солі
        password_hash = hash_password(password, salt)  # Хешуємо
пароль з сіллю
        registration_date = datetime.now().strftime('%Y-%m-%d
%H:%M:%S')
        conn = sqlite3.connect('users.db')
        c = conn.cursor()
        c.execute("INSERT INTO users (username, password_hash,
salt, registration_date) VALUES (?, ?, ?, ?)",
                  (username, password_hash, salt,
registration_date))
        conn.commit()
        conn.close()
```

```
        print(f"Користувач {username} зареєстрований!")
    except Exception as e:
        print(f"Помилка при реєстрації користувача: {e}")
# Перевірка правильності введеного пароля
def verify_password(username, password):
    conn = sqlite3.connect('users.db')
    c = conn.cursor()
    c.execute("SELECT password_hash, salt, registration_date FROM
users WHERE username=?", (username,))
    user = c.fetchone()
    conn.close()

    if user:
        password_hash, salt, registration_date = user
        if hash_password(password, salt) == password_hash:
            registration_date =
datetime.strptime(registration_date, '%Y-%m-%d %H:%M:%S')
            if datetime.now() - registration_date >
timedelta(seconds=30):  # Змінили для тестування
                print(f"Ваш пароль застарів! Необхідно змінити
пароль.")
            else:
                print(f"Вхід до системи успішний! Добро
пожаловать, {username}.")
                return True
        else:
            print("Невірний пароль.")
    else:
        print("Користувач не знайдений.")
    return False
# Головна функція для взаємодії з користувачем
def main():
    create_database()  # Створення бази даних (виконується лише
один раз)

    while True:
        action = input("Виберіть дію:\n1. Вхід\n2. Реєстрація\n3.
Вихід\nВведіть номер дії: ")
        if action == '1':  # Вхід
            username = input("Введіть ім'я користувача: ")
            password = input("Введіть пароль: ")
            if verify_password(username, password):
                # Якщо пароль правильний та не застарілий
                exit_account = input("Бажаєте вийти з акаунту?
(так/ні): ").strip().lower()
                if exit_account == "так":
                    print("Вихід з акаунту.")
                    continue  # Повертаємось до головного меню
                else:
                    print(f"Вітаємо, {username}! Ви увійшли в
систему.")
                    continue  # Якщо не хочете вийти, залишаємося
в системі
        elif action == '2':  # Реєстрація
            username = input("Введіть ім'я користувача: ")
```

```
            if user_exists(username):
                print(f"Користувач з ім'ям {username} вже існує!")
            else:
                password = input("Введіть пароль: ")
                register_user(username, password)
                continue  # Після реєстрації автоматично
повертається до меню
        elif action == '3':  # Вихід
            print("До побачення!")
            break
        else:
            print("Невірний вибір. Спробуйте ще раз.")
# Запуск програми
if __name__ == "__main__":
    main()
```