МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

ПРИКЛАДНА КРИПТОЛОГІЯ

навчальний посібник

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ ДЕРЖАВНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

Практикум з прикладної криптології

Навчальний посібник з дисципліни **«Прикладна криптологія»**
для здобувачів освітнього ступеня «бакалавр» спеціальності
125 «Кібербезпека та захист інформації» усіх форм навчання

Черкаси 2023

УДК 003.26:004.056.55 (07)
М 54

Затверджено вченою радою ЧДТУ,
наказ №182/04 від 26.06.2023р.
згідно з рішенням методичної комісії факультету електронних технологій, автотранспорту та машинобудування,
протокол № 4 від 24.04.2023 р.

Упорядники: В.В.Палагін., д.т.н., професор, О.А.Палагіна, к.т.н., доцент, Івченко О.В., к.т.н., доцент.

Рецензенти: Є.А.Мачуський, д.т.н., професор, заслужений діяч науки і техніки України, професор кафедри інформаційної безпеки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського»;
С.О.Гнатюк, д.т.н., професор, декан факультету комп'ютерних наук та технологій Національного авіаційного університету;
О.В.Криворучко, д.т.н., професор, завідувач кафедри інженерії програмного забезпечення та кібербезпеки Державного торговельно-економічного університету.

М 54	Практикум з прикладної криптології: Навчальний посібник з дисципліни «Прикладна криптологія» для здобувачів першого (бакалаврського) рівня вищої освіти зі спеціальності 125 «Кібербезпека та захист інформації» усіх форм навчання [Електронний ресурс] / [Автори В.В.Палагін, О.А.Палагіна, О.В.Івченко] – ; М-во освіти і науки України, Черкас. держ. технол. ун-т. – Черкаси: ЧДТУ, РВЦ, 2023. - Назва з титульного екрана.

Видання містить практикум з прикладної криптології та знайомить студентів з основними принципами криптографічних перетворень, зокрема з побудовою базових шифрів підстановки й перестановки, блокових та складених шифрів, потокових шифрів, принципами побудови симетричних та асиметричних криптографічних систем шифрування, принципами побудови хеш-функцій, методами та механізми автентифікації в криптосистемах, застосуванню цифрових підписів та їх властивостей, різновидами стеганографічних систем, які використовуються для забезпечення конфіденційності інформації в інформаційно-телекомунікаційних системах. Наведено теоретичні відомості, порядок виконання практикуму, перелік рекомендованої літератури, контрольних питань, які допоможуть студенту в якісному засвоєнні матеріалу.

Навчальне електронне видання мережного використовування
Практикум з прикладної криптології: Навчальний посібник з дисципліни «Прикладна криптологія» для здобувачів першого (бакалаврського) рівня вищої освіти зі спеціальності 125 «Кібербезпека та захист інформації»

Упорядники: Палагін Володимир Васильович
Палагіна Олена Анатоліївна
Івченко Олександр віталійович
В авторській редакції.

ЗМІСТ

ПЕРЕДМОВА

Всеосяжний розвиток інформаційних технологій, впровадження і застосування широкого спектру різноманітних додатків, обмін даними по різноманітним каналах звязку, інтенсивне використання хмарних сервісів, прогресуюче інтегрування штучного інтелекту в різноманітні сфери життя людей – далеко не повний перелік тих ризиків, з якими може зустрітися кожен з нас при передачі та збереженні даних.

Переваги подання та передачі даних у цифровому вигляді (легкість відновлення, висока потенційна завадостійкість, перспективи використання універсальних апаратних і програмних рішень) можуть бути перекреслені з легкістю, з якою можливі їх викрадення та модифікація. Тому в усьому світі назріло питання розробки методів та засобів захисту інформації організаційного, методологічного й технічного характеру, серед них – методи криптографії та стеганографії**.**

Криптографія, як наука про математичні методи та алгоритми забезпечення конфіденційності, цілісності і автентичності інформації, розвинулась з практичної потреби передавати важливі відомості найнадійнішим чином. Дослідженням вразливих місць таких алгоритмів та розробкою методів зламу зашифрованих повідомлень займається криптоаналіз. Ці два наукових напрями тісно пов’язаних між собою, активно конкурують і разом складають науку, яка називається криптологія.

Навчальний посібник «Практикум з прикладної криптології» спрямований на ознайомлення студентів із загальними принципами побудови систем криптографічного захисту даних шляхом використанням сучасних симетричних та асиметричних алгоритмів шифрування, застосування стеганографічних систем приховування інформації. Структуру та зміст навчального посібника визначено, виходячи зі змісту програми навчальної дисципліни «Прикладна криптологія» та її змістовних модулів.

Метою вивчення навчальної дисципліни є отримання базових знань і практичних навичок з прикладної криптології, основ стеганографії, розуміння побудови засобів криптографічного захисту інформації.

Навчальний посібник містить набір різноманітних практикумів, які висвітлюють принципи побудови класичних методів шифрування, реалізацію симетричних та асиметричних криптосистем, блокових та потокових шифрів, реалізацію функцій хешування, алгоритмі обміну ключами, створення цифрових підписів, наводяться основи реалізації стеганографічного приховування інформації. Кожен практикум складається з теоретичної та практичної частин, містить контрольні питання та задачі, перелік додаткової самостійної роботи, що сприятиме кращому засвоєнню матеріалу.

Навчальний посібник призначений для здобувачів вищої освіти в галузі 12 «Інформаційні технології» за спеціальністю 125 «Кібербезпека та захист інформації» та осіб, що самостійно вивчають дисципліну і цікавляться питаннями криптографічного захисту інформації.

ВСТУП

Одним із сучасних завдань національної безпеки є захист інформації (*Data protection*), що означає застосування різноманітних методів та засобів для забезпечення цілісності, конфіденційності і доступності інформації незалежно від загроз природного або штучного характеру, що можуть завдати шкоди власникам і користувачам цієї інформації [1-4].

Обробка інформації в системі включає в себе виконання різних операцій за допомогою технічних і програмних засобів, таких як збирання, введення, запис, перетворення, зчитування, зберігання, видалення, реєстрація, приймання, отримання та передавання даних.

Важливо забезпечити надійний захист інформації від несанкціонованого та неконтрольованого доступу, недозволених змін, знищення, копіювання та поширення, щоб зберегти основні характеристики її захищеності [5]:

- *конфіденційності інформації;*
- *цілісності інформації;*
- *достовірності інформації;*
- *оперативності доступу до інформації;*
- *юридичної значущості інформації, наданої у вигляді електронного документа;*
- *невідстежуваності дій клієнта.*

Надамо деякі визначення наведених термінів.

Конфіденційність означає, що інформація має бути доступна лише обмеженій групі користувачів інформаційно-технічних систем (ІТС). Конфіденційні дані - це дані, які потребують захисту. Ці дані повинні бути відомі тільки авторизованим суб'єктам системи, які пройшли перевірку.

Цілісність інформації або програмного забезпечення відноситься до їх здатності зберігати структуру та зміст під час передачі та зберігання. *Цілісність даних* означає, що дані не були незаконно змінені, підмінені або знищені. Розглядаючи передачу інформації через мережу, можна зрозуміти, що кожне повідомлення відноситься до певного класу за своїм змістом. Завдяки цьому, кінцеве повідомлення залишається з таким самим змістовим наповненням незалежно від зміни форми його представлення в електронному вигляді. *Модифікація даних* включає в себе зміну їх змісту та порушення цілісності, включаючи часткове вилучення.

Достовірність означає, наскільки інформація точно відображає об'єкт, що є її джерелом, або той об'єкт, від якого отримана ця інформація.

Оперативність визначається як можливість швидкого доступу кінцевого користувача до інформації або певного інформаційного ресурсу залежно від його поточних потреб.

Юридична значущість – властивість документа мати юридичну силу. З метою забезпечення підтвердження юридичної значущості переданого повідомлення, суб'єкти домовляються про універсальне визнання певних

атрибутів інформації, що підтверджують її юридичну значущість. Ця характеристика інформації особливо важлива в системах електронних платежів, де проводиться операція з переказом грошових коштів. Враховуючи сказане, можна сформулювати певні вимоги до атрибутів інформації, що підтверджують її юридичну значущість. Інформацію потрібно так сформувати, щоб було чітко зрозуміло, що лише відправник, якому належить цей платіжний документ, міг його створити з формально-правової точки зору.

Невідстежуваність - це здатність здійснювати певні дії в ІТС, що не помітні для інших об'єктів. Актуальність цієї вимоги стала очевидною завдяки появі нових концепцій, таких як електронні гроші й Інтернет-банкінг. Наприклад, для авторизації доступу до електронної платіжної системи користувач мусить надати певні відомості, які його ідентифікують. Швидкий розвиток таких систем може призвести до реальної небезпеки, такої, що всі платіжні операції будуть контролюватися, створюючи умови для загального спостереження за користувачами ІТС.

Для забезпечення безпеки при передачі інформації з однієї системи до іншої, необхідно вживати ряд заходів, які включають процеси шифрування. Ці процеси виконуються за допомогою технічного та криптографічного захисту інформації.

Криптографічний захист інформації реалізується за допомогою програмних, програмно-апаратних та апаратних засобів, які перетворюють інформацію з використанням спеціальних (ключових) даних. Це зроблено з метою приховування або відновлення інформації, підтвердження її справжності, цілісності, авторства та інших аспектів безпеки.

Розробка і впровадження криптографічного захисту інформації грунтується на відповідних стандартах, які затверджені і є обовязковими для їх реалізації. Наведемо перелік деяких стандартів та технічних специфікацій, дозволених для реалізації в засобах криптографічного захисту інформації згідно наказу Адміністрації Державної служби спеціального зв'язку та захисту інформації України від 27 жовтня 2020 року № 687 [6].

I. Стандарти, що визначають вимоги до блокових шифрів (block ciphers)

1. ДСТУ 7624:2014 "Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення" (КАЛИНА)
2. ДСТУ ISO/IEC 18033-3:2015 (ISO/IEC 18033-3:2010, IDT) "Інформаційні технології. Методи захисту. Алгоритми шифрування. Частина 3. Блокові шифри".
3. ДСТУ ISO/IEC 10116:2019 (ISO/IEC 10116:2017, IDT) "Інформаційні технології. Методи захисту. Режими роботи n-бітних блокових шифрів".

II. Стандарти, що визначають вимоги до потокових шифрів (stream ciphers)

1. ДСТУ 8845:2019 "Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного потокового перетворення". «СТРУМОК»

2. ДСТУ ISO/IEC 18033-4:2015 (ISO/IEC 18033-4:2011, IDT) "Інформаційні технології. Методи захисту. Алгоритми шифрування. Частина 4. Потокові шифри"

III. Стандарти, що визначають вимоги до асиметричних криптографічних алгоритмів та методів (asymmetric algorithms and techniques)

ДСТУ 4145-2002 "Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння".
2. ДСТУ ISO/IEC 9796-2:2015 (ISO/IEC 9796-2:2010, IDT) "Інформаційні технології. Методи захисту. Схеми цифрового підпису, які забезпечують відновлення повідомлення. Частина 2. Механізми, що ґрунтуються на факторизації цілих чисел".
3. ДСТУ ISO/IEC 9796-3:2015 (ISO/IEC9796-3:2006, IDT) "Інформаційні технології. Методи захисту. Схеми цифрового підпису, які забезпечують відновлення повідомлення. Частина 3. Механізми, що ґрунтуються на дискретному логарифмі".
4. ДСТУ ISO/IEC 14888-2:2015 (ISO/IEC 14888-2:2008, IDT) "Інформаційні технології. Методи захисту. Цифрові підписи з доповненням. Частина 2. Механізми, що ґрунтуються на факторизації цілих чисел".
5. ДСТУ ISO/IEC 14888-3:2019 (ISO/IEC 14888-3:2018, IDT) "Інформаційні технології. Методи захисту. Цифрові підписи з доповненням. Частина 3. Механізми, що ґрунтуються на дискретному логарифмуванні".
6. ДСТУ ISO/IEC 15946-5:2019 (ISO/IEC 15946-5:2017) "Інформаційні технології. Методи захисту. Криптографічні методи на основі еліптичних кривих. Частина 5. Генерування еліптичних кривих".
7. ДСТУ ISO/IEC 18033-2:2015 (ISO/IEC 18033-2:2006, IDT) "Інформаційні технології. Методи захисту. Алгоритми шифрування. Частина 2. Асиметричні шифри".

Запропонований Практикум з прикладної криптології побудований на основі затверджених стандартів по криптографічному захисту інформації та знайомить студентів з математичними основами криптографічних перетворень, основними принципами побудови криптографічних алгоритмів з наданням конкретних прикладів, зокрема з класичними методами шифрування, побудовою блокових та складених шифрів, потокових шифрів, принципами побудови симетричних та асиметричних криптографічних систем шифрування, методами та механізми автентифікації в криптосистемах, побудовою цифрових підписів, різновидами стеганографічних систем та їх практичною реалізацією.

В посібнику наведено теоретичні відомості, порядок виконання та вимоги до оформлення практикумів, перелік контрольних питань та рекомендованої літератури, які допоможуть студентам в якісному засвоєнні матеріалу з дисципліни «Прикладна криптологія».

ПРАКТИКУМ № 1
Шифри простої моноалфавітної підстановки (заміни). Шифр Цезаря, квадрат Полібія

Мета роботи. Ознайомитися з теоретичними положеннями шифрів підстановки (заміни), на практичних прикладах продемонструвати створення ключа шифру заміни, провести зашифрування відкритого і розшифрування шифрованого повідомлення.

1.1 Основні поняття та визначення

Надамо деякі визначення деяких термінів, які є широко вживаними при реалізації процедур захисту інформаційних ресурсів [5, 7-20].

Криптологія (англ. – «*cryptology*», грецьк. *«таємний» та «слово, вчення»*) - наука, складовими якої є ***криптографія, криптоаналіз*** та, за деякими визначеннями, стеганографія. Вивчає методи побудови та аналізу систем захисту інформаційних ресурсів, основаних на математичних перетвореннях даних з використанням секретних параметрів. Криптологія поєднує у собі два взаємозалежні напрями: криптографію та криптоаналіз.

Криптографія (англ. – «*cryptography*», грецьк. *«таємний» та «писання», «тайнопис»*) - це наука, що займається вивченням та розробленням методів, способів та засобів перетворення інформації у вигляд, який ускладнює чи унеможливлює несанкціоновані дії з нею. Це напрям у криптології, що вивчає основні закономірності, протиріччя, методи, системи та засоби забезпечення конфіденційності, цілісності, дійсності, доступності та спостережливості інформації та ресурсів тощо, ґрунтуючись на криптографічних перетвореннях. Криптографія базується на методах (алгоритмах) шифрування та дешифрування.

Криптоаналіз (англ. – «*cryptanalysis*», грецьк. *«таємний» та «аналіз»*) - наука, що займається вивченням та розробленням методів, способів та засобів розкриття шифрів. Вивчає основні закономірності, протиріччя, методи та засоби аналізу криптографічних систем, ґрунтуючись на їх вхідних та вихідних даних, що здійснюється з метою визначення спеціальних (ключових) даних, які можуть бути використані для порушення конфіденційності, цілісності, неспростовності інформації та ресурсів.

Шифрування даних – це процес, що складається із *зашифрування* та *розшифрування (дешифрування)* даних:

Зашифрування (англ. - *encryption*) - це процес криптографічного перетворення даних, за допомогою якого відкритий текст перетворюється на зашифрований з метою захисту від несанкціонованого доступу.

Розшифрування/Дешифрування (англ. - *decryption*) - це процес, зворотний розшифруванню.

Алгоритм шифрування (англ. – «*encryption algorithm*») - це алгоритм, згідно з яким здійснюється спеціальне криптографічне перетворення інформації (*криптографічний алгоритм або криптоалгоритм*).

Стеганографія (англ. – «*stegonography*», грецк. — «*прихований + пишу*») - набір засобів і методів приховування факту передавання повідомлення. Стеганографія — тайнопис, при якому повідомлення, закодоване таким чином, що не виглядає як повідомлення, на відміну від криптографії. Таким чином непосвячена людина принципово не може розшифрувати повідомлення, бо не знає про факт його існування.

Криптосистема – це система криптографічного перетворення даних, що містить у собі п’ять компонентів: множину відкритих текстів, множину шифротекстів, множину ключів, сімейство зашифровуючих та розшифровуючих перетворень. Фахівець, який займається розробкою криптосистем називається ***криптографом.***

Ключ — параметр криптографічної системи, який використовується для зашифрування і/або дешифрування повідомлення при шифруванні; накладення та перевірки коду автентифікації повідомлень або електронного цифрового підпису.

Криптостійкість – це властивість криптосистеми протидіяти атакам супротивника, спрямованим на отримання секретного ключа або відкритого повідомлення. Стійкість криптосистеми визначається її здатністю протидіяти усім можливим атакам.

Абсолютно стійкі системи - доказ існування абсолютно стійких алгоритмів шифрування було виконано Клодом Шенноном і опубліковано в роботі «Теорія зв'язку в секретних системах». Там же визначено вимоги до такого роду систем:

- ключ генерується для кожного повідомлення (кожен ключ використовується один раз);
- ключ статистично надійний (тобто ймовірності появи кожного з можливих символів рівні, символи в ключовий послідовності незалежні і випадкові);
- довжина ключа дорівнює або більше довжини повідомлення.

Домовленості або певні правила поведінки функціонування системи та обробки даних для досягнення певної мети будемо називати протоколами. **К*риптографічними протоколами*** будемо називати ті, в яких учасники для досягнення певної мети використовують криптографічні перетворення інформації.

Криптостійкість часто вимірюється кількістю операцій, необхідних для перебору всіх можливих ключів, або інтервалом часу, необхідного для зламу. Вона оцінюється у процесі проведення криптографічного аналізу. Фахівець який займається розробкою методів криптоаналізу називається ***криптоаналітиком.*** Синонімами є терміни ***зловмисник, порушник, супротивник.***

Стійкість криптосистем не повинна залежати від того, якими обчислювальними можливостями володіє криптоаналітик. Практичне застосування систем, що задовольняють вимогам абсолютної стійкості, обмежене міркуваннями вартості і зручності користування.

Незаперечність причетності до авторства - це поняття, зворотне поняттю відмови від авторства, тобто заперечення причетності до створення або передавання якого-небудь документа чи повідомлення. У свою чергу, *незаперечність причетності до одержання документа або повідомлення* - поняття, зворотне поняттю відмови від причетності до одержання будь-якого документа чи повідомлення.

Важливим поняттям криптографії є ***криптоаналітична атака*** (англ. – «*cryptoanalytic attack*») - це загальна назва методу, за допомогою якого криптоаналітик намагається зламати криптосистему. Моделі атаки або типи атаки у криптоаналізі є класифікацією криптографічних атак, що визначають вид доступу, який криптоаналітик має до системи, що піддається атаці, при спробі «зламати» зашифроване повідомлення (також відоме як шифротекст). Чим більший доступ може отримати криптоаналітик, тим більше корисної інформації може бути вилучено та використано для порушення роботи системи.

Основні принципи криптоаналізу сформульовані наступним чином:

1. Принцип Керкгоффза. Тільки супротивник може судити про криптостійкість системи.

Приховання інформації тільки на основі факту невідомості зловмисникові методу або методів, закладених в основу приховання, на сьогоднішній день є малоефективним. Ще в 1883 р. фламандський криптограф А. Керхгофс (A. Kerckhoffs) указував на той факт, що система захисту інформації повинна виконувати покладені на неї функції навіть при **повній інформованості противника** про її структуру та алгоритм функціонування

Принцип Керкгоффза - правило, згідно з яким стійкість криптографічного алгоритму не має залежати від архітектури алгоритму, а має залежати **тільки від ключів**. Іншими словами, при оцінці надійності шифрування необхідно вважати, що супротивник знає все про систему шифрування, що використовується, крім ключів

Стійкість шифру або криптосистеми повинна визначатися тільки секретністю ключа. Це пояснюється тим, що криптосистема, яка являє собою сукупність апаратних і програмних засобів, яку можна змінити тільки при значних затратах часу і засобів, тоді як ключ змінюється дуже легко.

2. Принцип Керкхоффа-Шеннона. Супротивник знає використовувану систему з точністю до ключової інформації.

Класифікацію атак можливо представити наступним чином.

Атака на основі шифротексту. Криптоаналітик має тільки шифртексти декількох повідомлень, причому усі вони зашифровані з використанням того самого алгоритму шифрування. Робота криптоаналітика полягає в тому, щоб розкрити вихідні тексти, або, ще краще, обчислити ключ, використаний для шифрування цих повідомлень, для того, щоб розшифрувати й інші повідомлення, зашифровані цим ключем.

Атака на основі відкритого тексту. Криптоаналітик має доступ не тільки до шифротекстів декількох повідомлень, але також до відкритих текстів цих повідомлень. Його робота полягає в пошуку ключа, використовуваного при шифруванні цих повідомлень, або алгоритму розшифрування будь-яких нових повідомлень, зашифрованих тим же самим ключем.

Атака на основі підібраного відкритого тексту. Криптоаналітик не тільки має доступ до шифротекстів і зв'язаним з ними відкритим текстам декількох повідомлень, але і може за бажанням обирати відкриті тексти, що потім одержує в зашифрованому виді. Такий криптоаналіз виходить більш могутнім у порівнянні з криптоаналітиком за відомим відкритим текстом, тому що криптоаналітик може вибрати для шифрування такі блоки відкритого тексту, що дадуть більше інформації про ключ. Робота криптоаналітика складається в пошуку ключа, використаного для шифрування повідомлень, або алгоритму расшифрування нових повідомлень, зашифрованих тим же ключем.

Існують і інші, менш розповсюджені, криптоаналітичні атаки.

Під ***вразливістю*** (англ. – «vulnerability») інформаційної системи розуміється будь-яка її характеристика, використання якої порушником може призвести до реалізації загрози.

Загрозою *(*англ. – «*threat»)* комп'ютерним даним (інформаційній системі) називається потенційно можлива подія або процес, що може прямо чи опосередковано нанести збитки даним чи іншим ресурсам системи шляхом розкриття, модифікації або руйнування даних, відмовлення в обслуговуванні тощо. Типовими видами загроз є:

- загрози цілісності даних;
- загрози конфіденційності;
- загрози доступності даних.

1.2. Класифікація методів шифрування повідомлень

Історія розвитку методів шифрування є цікавою і різноманітною, що обумовлює її різносторонню класифікацію [21]. Криптографічні системи та шифри класифікуються за різними ознаками, наприклад, наведеними на рис.1.1. [11].

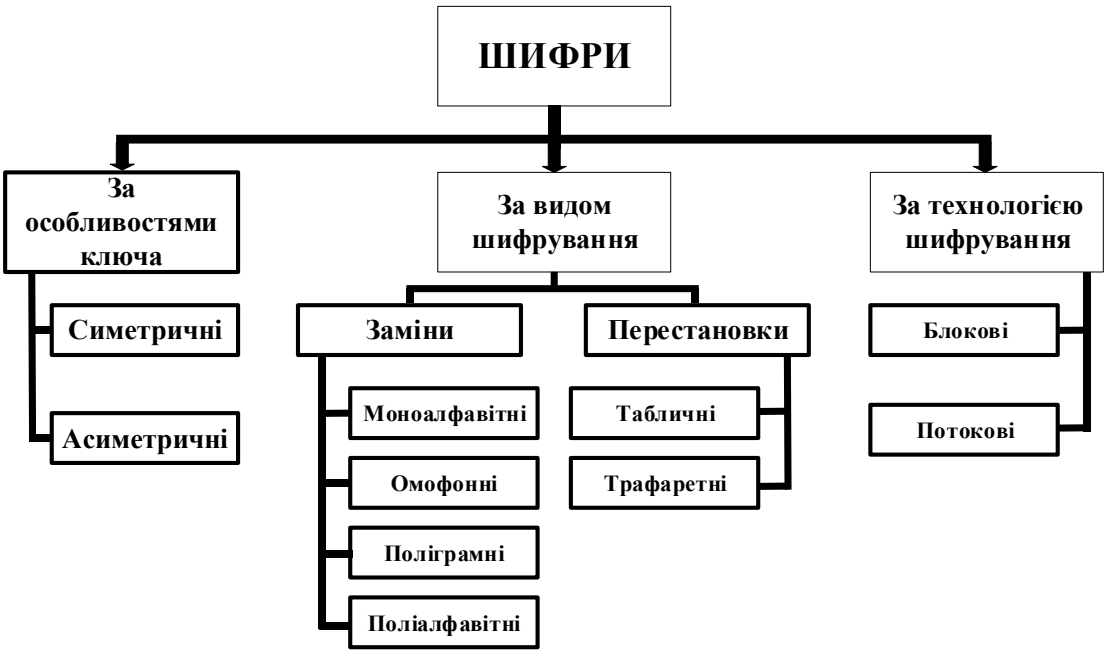


Рис. 1.1. Класифікація криптосистем

Симетричні криптосистеми (із закритим ключем, одноключові) – криптосистема, у якій один і той самий алгоритм, а також один і той самий ключ використовується для шифрування та дешифрування повідомлень (рис. 1.2).

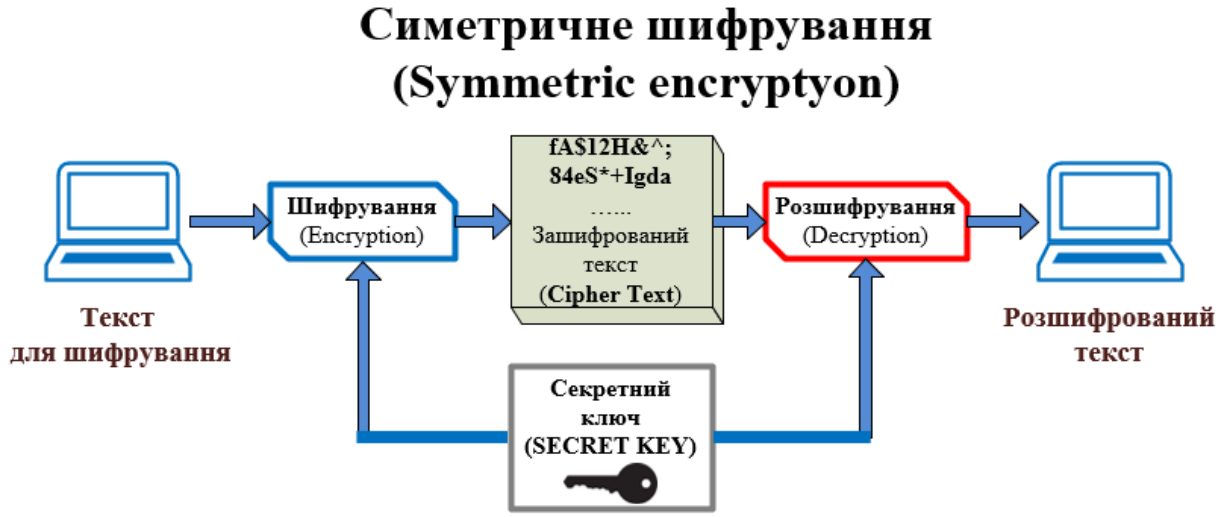


Рис. 1.2. Схема роботи симетричної криптосистеми

Асиметричні криптосистеми (із відкритим ключем, двоключові) – криптосистема, у якій використовуються два ключі – відкритий (публічний) і закритий (секретний), які математично пов'язані один з одним. Повідомлення зашифровується за допомогою відкритого ключа, що доступний усім бажаючим, а розшифровується за допомогою закритого ключа, відомого тільки одержувачу (рис. 1.3).

Головна відмінність асиметричних криптосистем полягає у тому, що навіть той, хто за допомогою відкритого ключа зашифрував повідомлення, не зможе його самостійно розшифрувати без секретного ключа. Тому ці системи називаються *асиметричними*, або *системами з відкритим ключем*.

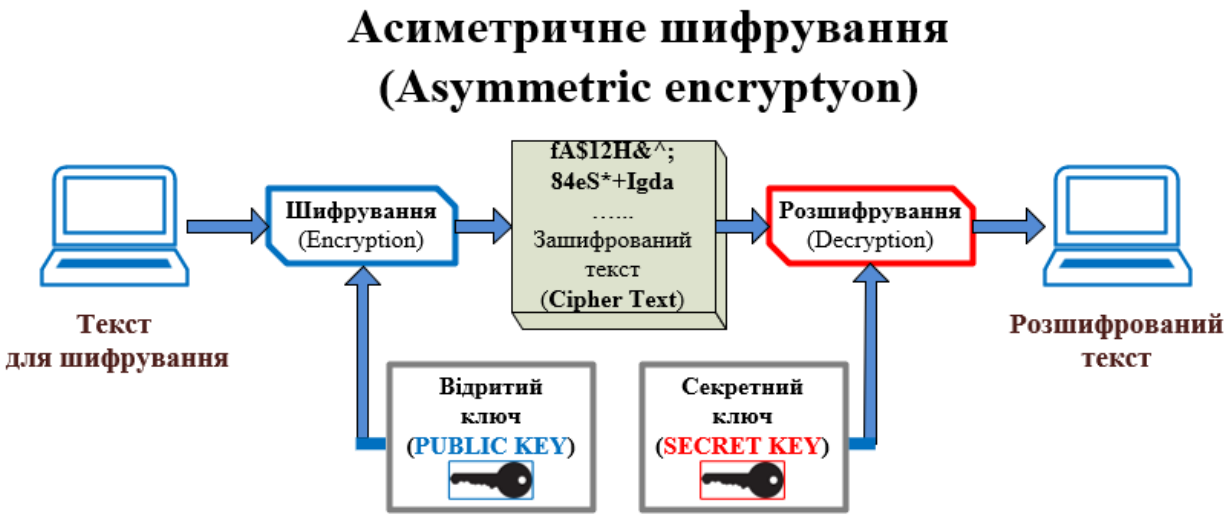


Рис. 1.3. Схема роботи асиметричної криптосистеми

Гібридними прийнято називати криптографічні системи, що поєднують обидва типи криптографічних систем, у них, як правило, текст повідомлення зашифровується з використанням симетричної криптографічної системи, а секретний ключ, використаної симетричною криптографічною системою, зашифровується з використанням асиметричної криптографічної системи [5].

Класична або одноключова криптографія спирається на використання симетричних алгоритмів шифрування, у яких зашифрування й розшифрування відрізняються тільки порядком виконання та напрямком деяких кроків. Ці алгоритми використовують той самий секретний елемент (ключ), і друга дія (розшифрування) є простим оберненням першої (зашифрування). Тому, зазвичай кожний з учасників обміну може як зашифрувати, так і розшифрувати повідомлення. Схематичну структуру такої системи показано на рис. 1.4. [5].

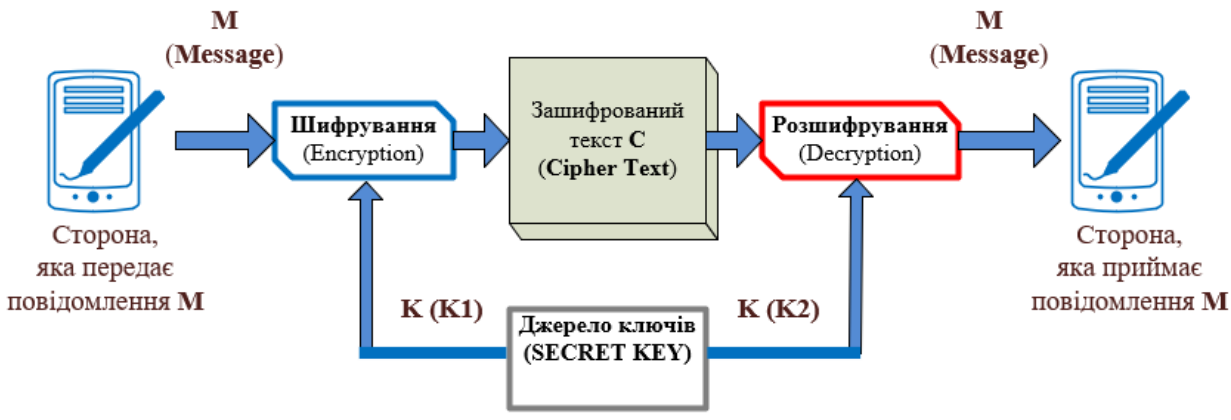


Рис. 1.4. Узагальнена схема криптографічного перетворення

На передавальній стороні є джерело повідомлень і джерело ключів. Джерело ключів вибирає конкретний ключ ***K*** (*Key*) серед усіх можливих ключів даної системи. Цей ключ ***K*** передається деяким способом приймаючій стороні, причому передбачається, що його не можна перехопити, наприклад, ключ передається спеціальним кур'єром або по секретному/захищеному каналу зв’язку. Тому симетричне шифрування називається також шифруванням із закритим ключем. Джерело повідомлень формує деяке повідомлення ***M*** *(Message, Plain Text)*, яке потім зашифровується з використанням вибраного ключа. Внаслідок процедури зашифрування виходить зашифроване повідомлення ***C*** (*Cryptotext, Cipher Text* - зване також криптограмою). Далі криптограма ***C*** передається по каналу зв'язку. Оскільки канал зв'язку є відкритим, незахищеним, наприклад, радіоканал або комп'ютерна мережа, то передане повідомлення може бути перехоплене противником. На приймаючій стороні криптограму ***C*** за допомогою ключа розшифровують і отримують вхідне повідомлення ***M***.

Через велику надмірність природних мов безпосередньо в зашифроване повідомлення надзвичайно складно внести осмислену зміну, тому класична криптографія забезпечує також захист від нав'язування помилкових даних. Якщо ж природної надмірності виявляється недостатньо для надійного захисту повідомлення від модифікації, надмірність може бути штучно збільшена шляхом додавання до повідомлення спеціальної контрольної комбінації, званої ***імітовставкою*.**

Відомі різні методи шифрування [5], де часто використовуються алгоритми *перестановки, підстановки, а також комбіновані методи* (рис. 1.5).

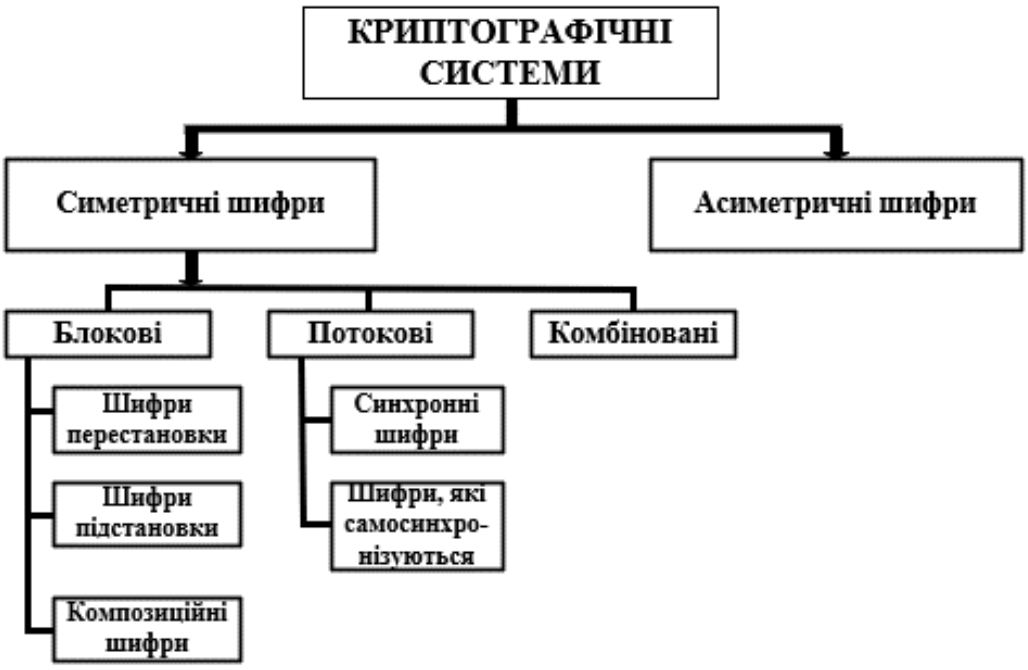


Рис.1.5. Класифікація методів шифрування повідомлень

За типом обробки вхідної інформаційної послідовності криптографічні системи поділяються на *потокові*, у яких перетворюються всі повідомлення одразу, і *блокові*, у яких повідомлення обробляються у вигляді блоків певної довжини.

Блоковий шифр — різновид симетричного шифру. Особливістю блочного шифру є обробка блоку декількох байт за одну ітерацію.

Потоковий шифр — група симетричних шифрів, які шифрують кожен символ відкритого тексту незалежно від інших символів.

Блокові шифри поділяються на шифри перестановки та підстановки.

Шифр перестановки – це шифр, у якому символи повідомлення переставляються місцями безпосередньо у відкритому тексті за певним правилом, що залежить від ключа. Найчастіше перестановка виконується за допомогою *таблиці*, комірки якої спочатку заповнюють відкритим текстом в деякому порядку, а потім шифротекст зчитують відповідно до заздалегідь визначеного алгоритму. У методах перестановки символи вхідного тексту міняються місцями один з одним за певним правилом.

Шифр підстановки (заміни) – це шифр, у якому кожен символ відкритого тексту у шифротексті замінюється іншим символом. Найчастіше виділяють такі типи шифрів підстановки:

- проста підстановка, або моноалфавітна заміна – це шифр, в якому кожен символ відкритого тексту замінюється відповідним символом шифротексту, при чому, конкретній літері відкритого повідомлення відповідає єдина, завжди одна і та сама, літера шифротексту;
- однозвучний шифр підстановки схожий на простий шифр підстановки за винятком того, що один символ відкритого тексту символ відкритого тексту замінюється на один з декількох можливих символів шифротексту;
- поліграмний шифр підстановки – це шифр, який блоки символів шифрує по групах, наприклад, біграма – це група з двох символів, триграма – з трьох символів і т.д.;
- поліалфавітна підстановка складається з декількох простих шифрів підстановки, тобто одна і та сама літера відкритого тексту може бути замінена кожен раз по різному.

По суті, шифри перестановки й підстановки є ***цеглинками***, з яких будуються різні інші більш стійкі шифри.

З метою підвищення надійності зашифрування текст, зашифрований за допомогою одного методу, може бути ще раз зашифрований за допомогою іншого методу. У такому випадку виходить ***комбінований*** або ***композиційний*** **шифр**.

Застосовувані на практиці в теперішній час *блокові* або *потокові* симетричні шифри також належать до комбінованих, оскільки в них використовується декілька операцій для зашифрування повідомлення.

Основна відмінність сучасної від докомп'ютерної криптографії полягає у тому, що раніше криптографічні алгоритми оперували символами природних мов, наприклад, літерами англійської чи української мови. Ці літери переставлялися або замінювалися іншими за певним правилом. У сучасних криптографічних алгоритмах використовуються операції над двійковими знаками, тобто над нулями й одиницями. У даний час основними операціями під час шифрування також є перестановка або підстановка, причому для підвищення надійності шифрування ці операції застосовуються разом (комбінуються) і багато разів циклічно повторюються.

Ідея, що лежить в основі *складених*, або *композиційних, шифрів*, полягає в побудові криптостійкої системи шляхом багаторазового застосування відносно простих криптографічних перетворень, які К. Шеннон запропонував використовувати як перетворення *підстановки* (*substitution*) і *транспозиції* (*permutation*). Багаторазове використання цих перетворень дозволяє забезпечити дві властивості, які повинні бути притаманні стійким шифрам: *розсіювання* (*diffusion*) і *перемішування* (*confusion*).

Розсіювання передбачає поширення впливу одного знака відкритого тексту, а також одного знака ключа на значну кількість знаків зашифрованого повідомлення. Наявність у шифрі цієї властивості, з одного боку, дозволяє приховувати статистичну залежність між знаками відкритого тексту, інакше кажучи, перерозподілити надмірність вхідного мови за допомогою поширення її на весь текст, а з іншого — не дозволяє відновити невідомий ключ частинами. Наприклад, звичайна перестановка символів дозволяє приховати частоти появи біграм, триграм і т.д.

Мета ***перемішування*** — зробити якомога складнішою залежність між ключем і зашифрованим повідомленням. Криптографічний аналітик на основі статистичного аналізу перемішаного тексту не повинен отримати будь-яку кількість інформації про використаний ключ. Зазвичай перемішування здійснюється за допомогою підстановки. Застосування розсіювання й перемішування окремо не забезпечує необхідну стійкість, стійка криптографічна система виходить тільки внаслідок їх спільного використання.

1.3. Шифри підстановки (заміни)

Традиційні шифри з симетричним ключем можна розділити на дві великі категорії: шифри *підстановки та шифри перестановки*. У шифрі підстановки замінюється один символ у зашифрованому тексті на інший символ; у шифрі перестановки — міняються місцями позиції символів у початковому тексті

Шифр підстановки замінює один символ іншим. Якщо символи в початковому тексті — символи алфавіту, то одна літера замінюється іншою. Наприклад, літера ***A*** *замінюється на літеру* ***M*** за певним правилом і т.д. Якщо використовуються символи або цифри, то вони також підлягають заміні, наприклад, 5 замінюється на 2, а 6 на 9 і т.д.

Шифри підстановки можуть бути розбиті на дві категорії: *моноалфавітної* (*одноалфавітної*) й *багатоалфавітної* підстановки.

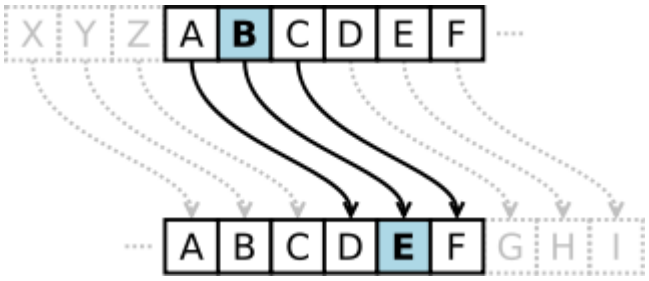
Для шифрів підстановки довжина алфавіту відкритого тексту (даних) (*nM*) і довжина алфавіту зашифрованого тексту (даних) (*nC*) однакова, тобто *nM* = *nC*.

При використанні цих шифрів літери алфавіту зручно ототожнювати з їх порядковими номерами.

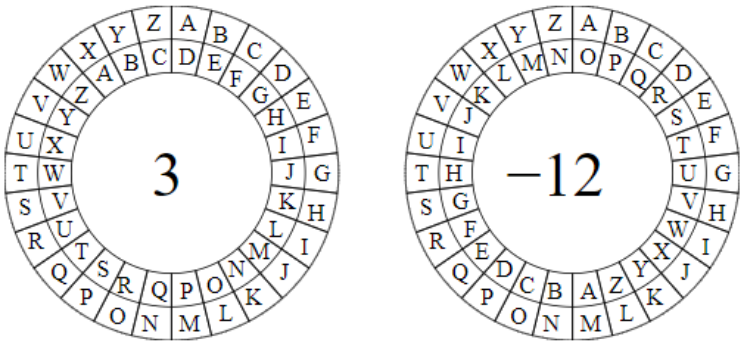
Адитивні моноалфавітні шифри підстановки

Найпростіший моноалфавітний шифр підстановки — *адитивний шифр*, його іноді називають шифром зсуву, а іноді — ***шифром Цезаря.***

Шифр Цезаря — симетричний моноалфавітнийалгоритм шифрування, в якому кожна буква відкритого тексту заміняється на ту, що віддалена від неї в алфавіті на сталу кількість позицій. Римський імператор Юлій Цезар використовував для приватного листування шифр зсуву з ключем 3 — замість літери A підставляв D, замість B — E і так далі. Іншими словами, відбувається циклічний зсув алфавіту на n літер (наприклад n=3) ліворуч (рис.1.6.):



а)



б) в)

Рис.1.6. Зсув літер англійської абетки на 3 позиції ліворуч (а), (б) та праворуч на 12 позицій (в) (https://uk.wikipedia.org/wiki/Шифр_Цезаря)

При зашифруванні літера відкритого тексту замінюється на літеру, що знаходиться під нею в нижній строчці.

Приклад 1.1. Наприклад: «***РИМ***» перетвориться на слово «***УЙП***» (рис.1.7). Ключем в шифрі Цезаря є величина зсуву нижнього рядка.

А Б В Г Ґ Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я

Г Ґ Д Е Є Ж З И І Ї Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ь Ю Я А Б В

Рис.1.7. Зсув літер української абетки на три позиції ліворуч

Часто для зручності використання шифру Цезаря використовують два диски різного діаметру з намальованими на краях дисків літерами (рис. 1.8).

Рис.1.8. Диск для реалізації шифру Цезаря (https://uk.wikipedia.org/wiki/Файл:Alberti_Ciper.jpg)

Спочатку диски повертають так, щоб напроти кожної букви алфавіту зовнішнього диска знаходилася та сама буква алфавіту внутрішнього диска. Якщо тепер повернути внутрішній диск на декілька символів, то ми отримаємо відповідність між символами зовнішнього диска і внутрішнього – шифр Цезаря. Цей диск можна використовувати як для зашифрування, так і для розшифрування повідомлень.

Термін “адитивний шифр” краще характеризує математичні властоивості ***шифра Цезаря***. Припустимо, що початковий текст складається з маленьких літер (від *a* до *z*), а зашифрований текст складається з великих літер (від *A* до *Z*). Щоб забезпечити застосування математичних операцій до вхідного та зашифрованого текстів, надамо кожній літері числове значення, як це показано на рис. 1.9 для англійської (26 літер), а на рис. 1.10 — для української мови (33 літери).

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Рис. 1.9. Подання літер вхідного й зашифрованого тексту для англійської мови потужністю *Z26*

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

Рис. 1.10. Подання літер вхідного й зашифрованого тексту для української мови потужністю *Z33*

На рис. 1.9 до кожного символу (верхній регістр) поставлено у відповідність ціле число з Z_{26}. Ключ засекречування між відправником та одержувачем — також ціле число в Z_n. Алгоритм зашифрування додає ключ до символу вхідного тексту; алгоритм розшифрування віднімає ключ із

символу зашифрованого тексту. Усі операції проводяться в Z_n. Рис. 1.11 показує процес шифрування й розшифрування повідомлень [5].

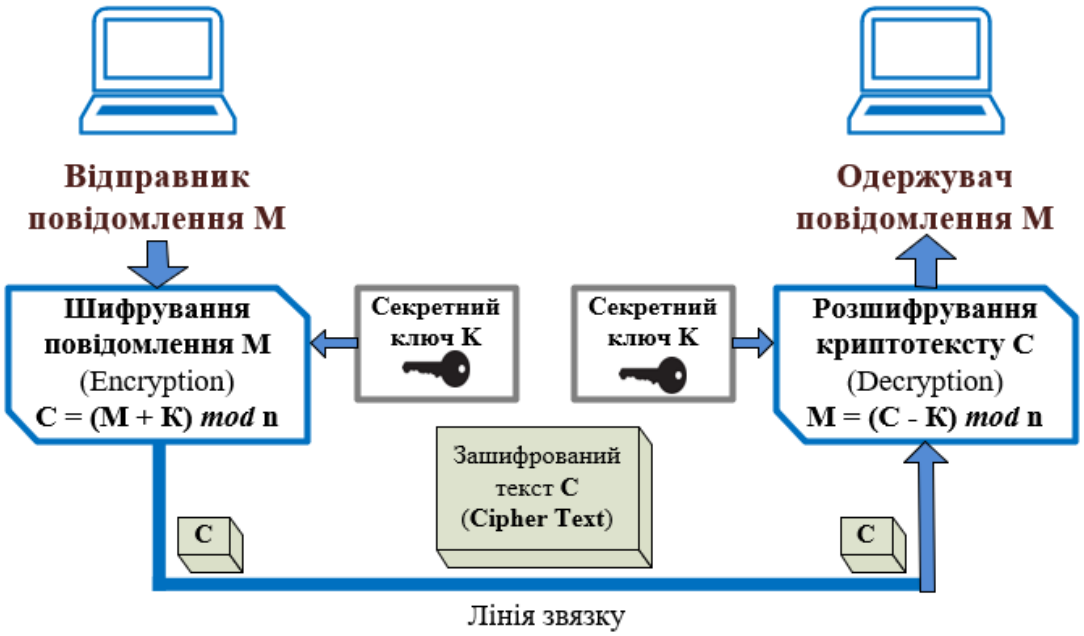


Рис. 1.11. Зашифрування й розшифрування повідомлень за допомогою адитивного шифру підстановки.

Зашифрування текстових повідомлень за допомогою адитивного шифру підстановки здійснюється за допомогою виразу:

$$C = (M + K) \bmod n,$$

де n — довжина алфавіту (для англійського алфавіту $n = 26$); M і C — вхідний і зашифрований текст відповідно; K — ключ зашифрування (розшифрування), а розшифрування

$$M = (C - K) \bmod n.$$

Можна легко показати, що зашифрування й розшифрування є інверсними один одному, тому що створений одержувачем (M') вхідний текст, той самий, що й передано відправником (M):

$$M' = (C - K) \bmod n = (M + K - K) \bmod n = M.$$

Подальший розвиток шифру Цезаря є очевидним: нижній рядок може бути записаний з випадковим порядком літер. Такий шифр носить назву ***шифру простої заміни***. Ключем такого шифру є порядок розташування літер в нижньому рядку, так звана «таблиця заміни». Якщо в шифрі Цезаря існує тільки 33 варіанта ключів, то в шифрі простої заміни їх вже 33! (33 факторіал).

Шифр Цезаря є прикладом однієї з перших систем шифрування. Шифр Цезаря не дуже складний, при шифруванні повідомлень з декількох слів відразу буде зрозумілим, скільки слів містив вихідний текст. Також можна отримати деяку інформацію з аналізу повторів букв в зашифрованому повідомленні.

Приклад 1.2. В зашифрованому слові «**У Ґ Р Г Р Є В**» одна з літер «**Р**» повторюється двічі. Для розшифрування повідомлення «**У Ґ Р Г Р Є В**»

необхідно знати тільки сам алгоритм шифрування. Будь-яка людина, що знає спосіб шифрування, легко може розшифрувати секретне повідомлення. Таким чином, ключем в даному методі є сам алгоритм.

Яким чином можна вдосконалити шифр Цезаря? Можна було б спробувати розширити алфавіт з 33 до 36 символів і більше за рахунок включення розділових знаків і пробілів. Це збільшення алфавіту замаскувало б довжину кожного окремого слова. У криптографії прийнято вважати, що противник може знати використаний алгоритм шифрування, характер переданих повідомлень і перехоплений шифротекст, але не знає секретний ключ. Це називається **принципом Керкгоффза**.

Принцип Керкгоффза (англ. Kerckhoffs's principle) - правило, згідно з яким стійкість криптографічного алгоритму не має залежати від архітектури алгоритму, а має залежати тільки від ключів. Іншими словами, при оцінці надійності шифрування необхідно вважати, що супротивник знає все про систему шифрування, що використовується, **крім ключів**.

Яким же чином може діяти зловмисник, щоб дізнатися зміст повідомлення?

Приклад 1.2. Нехай, наприклад, перехоплено секретне повідомлення:

«**op ovd hyl fvb**».

Припустимо, що також відомо про використання англійської абетки та шифру Цезаря. Зловмиснику відомо, що ключ (параметр зсуву ***n***) може набувати значень від 0 до 25. Намагаючись знайти значення секретного ключа, ми будемо проводити атаку по шифротексту. Розглянемо спосіб послідовного перебору всіх можливих ключів (це так званий метод "**грубої сили**"). Запишемо на 26 рядках всі варіанти, які виходять зрушенням кожної літери на 0, 2, 3, ..., 25 позицій відповідно. Цю операцію можна проводити вручну, а можна запрограмувати і побачити наступний результат (табл.1.1):

Табл.1.1.

Результат повного перебору ключів при застосуванні шифра Цезаря

Key #0: op ovd hyl fvb	Key #13: bc biq uly sio
Key #1: no nuc gxk eua	Key #14: ab ahp tkx rhn
Key #2: mn mtb fwj dtz	Key #15: za zgo sjw qgm
Key #3: lm lsa evi csy	Key #16: yz yfn riv pfl
Key #4: kl krz duh brx	Key #17: xy xem qhu oek
Key #5: jk jqy ctg aqw	Key #18: wx wdl pgt ndj
Key #6: ij ipx bsf zpv	Key #19: vw vck ofs mci
Key #7: hi how are you	Key #20: uv ubj ner lbh
Key #8: gh gnv zqd xnt	Key #21: tu tai mdq kag
Key #9: fg fmu ypc wms	Key #22: st szh lcp jzf
Key #10: ef elt xob vlr	Key #23: rs ryg kbo iye
Key #11: de dks wna ukq	Key #24: qr qxf jan hxd
Key #12: cd cjr vmz tjp	Key #25: pq pwe izm gwc

Як ми бачимо, серед всіх можливих варіантів повного перебору для ключа під номером key=7 ми отримуємо текст, який можемо прочитати:

«**hi how are you**».

Існують і інші методи взаму шифру, наприклад частотний аналіз, який грунтується на статистичних властивостях появи літер у відповідному алфавіті.

Шифр Цезаря з ключовим словом.

Шифр Цезаря з ключовим словом є одно алфавітною системою підстановки. Особливістю цієї системи є використання ключового слова для зсуву та зміни порядку символів в алфавіті підстановки. Виберемо деяке число *K*, від 0 до 25, і слово або коротку фразу в якості ключового слова. Бажано, щоб всі букви ключового слова були різними.

Приклад 1.3. Нехай слово DIPLOMAT буде ключовим і число *K=5* [12]. Ключове слово записується під літерами алфавіту, починаючи з букви, числове значення якої збігається з обраним числом *K* (рис. 1.12):

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	**B**	**C**	**D**	**E**	**F**	**Q**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**0**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
					D	***I***	***P***	***L***	***O***	***M***	***A***	***T***													

Рис. 1.12. Представлення англійської абетки та ключового слова, зсунутого на 5 позицій

Решта букв алфавіту підстановки записують після ключового слова в алфавітному порядку (рис.1.13).

					5																				
A	**B**	**C**	**D**	**E**	**F**	**Q**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**0**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
V	**W**	**X**	**Y**	**Z**	***D***	***I***	***P***	***L***	***O***	***M***	***A***	***T***	**B**	**C**	**E**	**F**	**G**	**H**	**J**	**K**	**N**	**Q**	**R**	**S**	**U**

Рис. 1.13. Представлення англійської абетки зсунутих літер для їх співтавлення при шифруванні

Тепер для кожної букви довільного повідомлення маємо підстановку. Фраза «*i remember that september*» буде зашифровано як «*lgztztwzgjpvjhzejztwzg*».

Вимога про відмінність всіх букв ключового слова не є обов'язкова. Можна просто записати ключове слово (або фразу) без повторення однакових букв.

Одна з відомих модифікацій шифру простої заміни – **квадрат Полібія**. Укриптографії квадрат Полібія (англ. «Polybius square»), також відомий як **шахова дошка Полібія** — оригінальний код простої заміни, одна з найдавніших систем кодування, запропонована Полібієм (грецький історик, полководець, державний діяч, III століття до н.е.). Цей спосіб кодування спочатку застосовувався для грецької абетки, але потім поширився на інші мови.

Візьмемо алфавіт з 32 літери (без літери **Г**). Виберемо ключ – будь-яке слово, в якому немає однакових літер. Запишемо його в перші клітинки квадрата розміром, наприклад, 4×8. В останні клітинки запишемо алфавіт за віднятком тих літер, що зустрічаються в ключі. Для зашифрування літери повідомлення замінюються на літери, що стоять під ними в квадраті Полібія.

Приклад 1.4. Нехай ключ – «**КИЇВ**», повідомлення – «**ІСТОРІЯ КРИПТОГРАФІЇ – ЦЕ ІСТОРІЯ ЛЮДСТВА**». Тоді на рис. 1.12 представлено квадрат Полібія для ключа «**КИЇВ**».

К	**И**	**Ї**	**В**
А	Б	Г	Д
Е	Є	Ж	З
І	Й	Л	М
Н	О	П	Р
С	Т	У	Ф
Х	Ц	Ч	Ш
Щ	Ь	Ю	Я

Рис.1.14. Квадрат Полібія для ключа «**КИЇВ**»

Зашифроване повідомлення набуде наступного виду:

«НХЦТФНВ АФБУЦЕЖФЕШНГ – ЫНХЦТФНВ ПЇЗХЦДЕ».

Для розшифрування букви шифротексту замінюються на ті літери, що стоять **НАД** ними в квадраті Полібія.

Також можуть бути різні модифікації даного шифру, наприклад побудова таблиці 8 на 8 і шифрування парами літер, які будуть вказувати на рядок і стовпець відповідної літери тексту, який шифрується.

Для цього створюється квадрат, наприклад розміром 8x8, і заповнимо його наступним чином (рис.1.13).

N/N	a	b	c	d	e	f	g	h
a	a	b	c	d	e	f	g	h
b	i	j	k	l	m	n	o	p
c	q	r	s	t	u	v	w	x
d	y	z	A	B	C	D	E	F
e	G	H	I	J	K	L	M	N
f	O	P	Q	R	S	T	U	V
g	W	X	Y	Z	0	1	2	3
h	4	5	6	7	8	9		,

Рис.1.15. Квадрат Полібія розміром 8x8.

Координати кожного символу вказані на краях таблиці. Спочатку йде координата стовпця, а потім рядка).

Приклад 1.5. Нехай для прикладу потрібно зашифрувати символ «**A».** Для нього стовпець буде відповідати як «**c**», а рядок як «**d**». Отже результатом шифрування буде пара «**cd**».

Приклад 1.6. Слово «**Hello»** зашифрується як «**beeadbdbgb».** Зашифроване слово текст «**eegdcg**», згідно даним правилам шифрування, відповідає оригінальному тексту «**KEY**».

1.4 Підготовка до завдання

Ознайомитися з теоретичними положеннями задач інформаційної безпеки стосовно захищеності даних, задач інформаційної безпеки стосовно працездатності інформаційно-телекомунікаційних систем, дій, що забезпечують рішення задач інформаційної безпеки.

Ознайомитися з поняттям криптографічних методів захисту, що забезпечують рішення задач інформаційної безпеки.

Ознайомитися з базовими шифрами підстановки, методами їх побудови та реалізації.

1.5 Практичне завдання

1. Записати своє прізвище, ім’я та по-батькові (ПІП) та представити відповідність літери у цифровій формі, згідно рис.1.10. Наприклад, ШЕВЧЕНКО -→ «28 06 02 27 06 17 14 18».
2. Для шифрування тексту використати моноалфавітний шифр Цезаря із самостійно обраним ключем n, 0 =<n =<32.
3. Обрати ключове слово та для свого ПІП провести шифрування, використовуючи квадрат Полібія, як це продемонстровано на рис.1.12.
4. Виконати програмну реалізацію функцій шифрування і дешифрування по п.1 і п.2.
5. Надати аналіз отриманим результатам.

1.6 Зміст протоколу роботи

Протокол до виконаного практикуму оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

1.7 Контрольні питання для самоперевірки

1. Які основні властивості захищеності інформації?
2. Дати визначення властивостей інформації: конфіденційність; цілісність; достовірність, юридична значущість інформації, невідстежуваність.

3. Дати визначення наступним термінам: шифр; ключ; зашифрування, розшифрування, криптоалгоритм, криптографічна система, криптографія, криптоаналіз, стеганографія, криптостійкість.
4. Дати визначення криптографічного протоколу.
5. Які вимоги повинні виконуватися для побудови абсолютно криптостійких систем (по Шенону)?
6. Дати пояснення сутності розсіювання даних у процесі їх шифрування.
7. Що є метою перемішування даних у процесі їх шифрування?
8. Що таке криптографічна атака та які типи криптографічних атак існують?
9. Дати характеристику атаки на зашифрований текст. Пояснити сутність атаки “грубої сили”.
10. Дати характеристику атаки на відкритий текст, підібраний текст. Пояснити сутність статистичної атаки, атаки за зразком.
11. Що називають загорозою і які їх типові різновиди.
12. Охарактеризуйте клисифікацію криптосистем за видами шифрування.
11. Пояснити загальну схему симетричного та асиметричного шифрування. В чому полягає основна їх відмінність?
12. Дайте визначення шифрів підстановки (заміни). Сформулювати загальні принципи для методів шифрування підстановкою.
15. Охарактеризуйте сутність блокових та потокових шифрів та їх прикладне застосування.
16. Які шифри називають композиційними?
15. Пояснити сутність адитивного шифру підстановки.
16. Пояснити сутність мультиплікативного шифру підстановки.
17. Пояснити сутність шифру Цезаря, квадрата Полібія.
18. У чому полягає основна криптографічна слабкість шифрів простої моноалфавітної заміни?
19. В чому полягає Принцип Керкгоффза?
20. Що називають потужністю ключового простору? Чому дорівнює потужність ключового простору для шифра Цезаря?

1.8. Задачі до Практикуму №1

1. Розшифруйте наступні повідомлення, зашифровані шифром Цезаря, і визначте ключ n, 0 =<n =<32, якщо відомо, що вихідні повідомлення складені з літер українського алфавіту, представленого на рис.1.10:
   «У Ґ Р Г Р Є В».
1. Розшифрувати шифротекст «**Ґ З У Г**», при зашифруванні якого використовували мультиплікативний шифр з ключем key =5 та українську абетку (33 літери, нумерацію літер розпочати з «0»).
2. Використовуючи мультиплікативний шифр, зашифрувати повідомлення українською мовою “УКРАЇНА” з ключем K = 7. (33 літери, нумерацію літер розпочати з «0»). Провести дешефрування отриманого повідомлення.
3. Використовуючи афінний шифр для значень ключів key1 = 5 і key2 = 7 зашифрувати повідомлення українською мовою “ВОЛЯ” (33 літери,

нумерацію літер розпочати з «0»). Провести дешефрування отриманого повідомлення.

4. Визначити ключі Цезаря, якщо відомі наступні пари відкритий текст – шифротекст якщо відомо, що вихідні повідомлення складені з ліер українського алфавіту, представленого на рис.1.10:
   а) АПЕЛЬСИН – ПДХБМЄЩГ
   б) МАНДАРИН – СДТИДХЛТ
5. Розшифруйте наступні повідомлення, зашифровані шифром Цезаря, і визначте ключ n, 0 =<n =<32, якщо відомо, що вихідні повідомлення складені з літер українського алфавіту, представленого на рис.1.10:
   а) УЧЦЬШЛУЮ
   б) ИБЗСЕЧ

1.9 Завдання до самостійної роботи

1. Дати визначення мультиплікативно оберненого числа за заданим модулем.
2. Умова існування мультиплікативно оберненого числа. Навести алгоритм знаходження такого числа.
3. Призначення та застосування розширеного алгоритму Евкліда.
4. Пояснити сутність афінного шифру підстановки, навести приклад.
5. Пояснити сутність автоключового шифру підстановки, навести приклад.
6. У чому суть методу частотного криптоаналізу? Обмеження при його застосуванні.
7. Яка літера найчастіше зустрічається у текстах українською (англійською) мовою?

ПРАКТИКУМ № 2
Шифри простої багатоалфавітної підстановки (заміни). Шифр Плейфера, Віженера

Мета роботи. Ознайомитися з теоретичними положеннями шифрів багатоалфавітної підстановки (заміни), на практичних прикладах продемонструвати створення ключа шифрів, провести зашифрування відкритого і розшифрування шифрованого повідомлення.

2.1. Основні поняття та визначення

Багатоалфавітний (поліалфавітний) шифр — це будь-який шифр, заснований на підстановці з використанням кількох алфавітів підстановки. У поліалфавітних шифрах заміни літери відкритого тексту шифруються по-різному залежно від їх розміщення в тексті. Між кожною літерою та її замінниками існує не однозначна відповідність, а зв'язок «один до багатьох». Наприклад, літера «**a**» може бути зашифровано як «**d**» на початку тексту, але як «**n**» у середині.

Багатоалфавітні шифри мають перевагу приховування частоти літер основної мови. Тому зловмисник не може використовувати частотний аналіз для атаки на зашифрований текст.

Першим багатоалфавітним шифром був шифр Альберті, який був представлений Леоном Баттістою Альберті в 1467 році - італійським вченим, письменником. Він використовував випадковий алфавіт для шифрування відкритого тексту, але в різних точках він міг змінитися на інший змішаний алфавіт, позначаючи зміну великими літерами. Для шифрування використовувався так званий «Диск Альберті», щоб показати, як літери відкритого тексту пов'язані з літерами шифрованого тексту. У цьому шифрі кожен символ зашифрованого тексту базується як на відповідному символі відкритого тексту, так і на позиції символу відкритого тексту в повідомленні. Це досягається використанням кількох ключів, а не лише одного ключа. Це означає, що ключ має бути потоком підключів, у якому кожен підключ так чи інакше залежить від позиції символу відкритого тексту, якому потрібен підключ для шифрування.

Іншими словами, потрібно мати s ключовий потік K = (K1, K2, K3 ...), у якому Ki використовується для шифрування i-го символу у відкритому тексті, щоб створити i-й символ у зашифрованому тексті. Найвідоміший і найпростіший такий алгоритм визначається як шифр Віженера.

2.2. Шифр або квадрат Плейфера

Шифр Плейфера або квадрат Плейфера — ручна симетрична техніка шифрування, в якій вперше використано заміну біграм [5, 11]. Шифр у 1854 році винайшов англійський фізик Чарльз Вітстон, але він отримав ім'я лорда Лайона Плейфера, який просував використання цієї системи у державній службі. Цей метод передбачає шифрування пар символів (біграм) замість одиночних символів, як у шифрі підстановки й у більш складних системах

шифрування Віженера. Через це шифр Плейфера стійкіший до злому методом частотного аналізу. Знаходження закономірності розподілу 26 х 26 = 676 можливих біграм потребує суттєво більших зусиль і обсягу зашифрованого тексту, ніж для 26 монограм (літер латинського алфавіту).

Для побудови шифру Плейфера створюється матриця літер 5 × 5 (для латинського алфавіту, або матриці 6 × 6 для кириличного алфавіту) – (рис.2.1.).

A	**B**	**C**	**D**	**E**
F	**G**	**H**	**I/J**	**K**
L	**M**	**N**	**O**	**P**
Q	**R**	**S**	**T**	**U**
V	**W**	**X**	**Y**	**Z**

А	**Б**	**В**	**Г**	**Ґ**	**Д**
Е	**Є**	**Ж**	**З**	**И**	**І**
Ї	**Й**	**К**	**Л**	**М**	**Н**
О	**П**	**Р**	**С**	**Т**	**У**
Ф	**Х**	**Ц**	**Ч**	**Ш**	**Щ**
Ь	**.**	**Ю**	**,**	**Я**	**—**

Рис.2.1. Створення матриць з набором літер абетки

Ключ засекречування в цьому шифрі зроблений із 25 літер латинського алфавіту і розміщений на початку у матриці 5 × 5 без повторів літер, або з 33 літер для кириличного алфавіту в матриці 6 × 6, також без повторів літер.

Для створення матриці шифрування й використання шифру достатньо запам’ятати ключове слово й чотири прості правила.

Щоб скласти ключову матрицю, у першу чергу потрібно *заповнити порожні комірки матриці літерами ключового слова* (не записуючи повторювані символи).

Потім *заповнити порожні комірки матриці*, які залишилися вільними символами алфавіту, що не зустрічаються в ключовому слові, по порядку (в англійських текстах зазвичай опускається символ “q”, щоб зменшити алфавіт, в інших версіях “I” і “J” об’єднуються в одну комірку). Ключове слово може бути записано у верхньому рядку матриці зліва направо, або по спіралі з лівого верхнього кута до центру. Ключове слово, доповнене алфавітом, складає матрицю 5 × 5 і є ключем шифру. За допомогою різних домовленостей про розміщення літер у матриці можна створити багато різних ключів засекречування.

Для того, щоб зашифрувати повідомлення необхідно розбити його на біграми (групи з двох символів). Наприклад “**hello world**” стає “**he ll ow or ld**”, і відшукати ці біграми у таблиці [5].

Два символи біграми відповідають кутам прямокутника в ключовій матриці. Визначаємо положення кутів цього прямокутника відносно один одного. Потім, керуючись наступними чотирма правилами, зашифровуємо пари символів вхідного тексту:

1. Якщо *два символи біграми збігаються*, додаємо після першого символа *фіктивний символ* “**x**”, зашифровуємо нову пару символів і

продовжуємо. У деяких варіантах шифру Плейфера замість “x” використовується “q” або “_”.

2. Якщо символи біграми вхідного тексту зустрічаються *в одному рядку*, то ці символи замінюються на символи, розташовані в найближчих *стовпцях справа* від відповідних символів. Якщо символ є останнім у рядку, то він замінюється на перший символ цього самого рядка.

3. Якщо символи біграми вхідного тексту зустрічаються в *одному стовпці*, то вони перетворюються на символи того самого стовпця, що знаходяться безпосередньо *під ними*. Якщо символ є нижнім у стовпці, то він замінюється на перший символ цього самого стовпця.

4. Якщо символи біграми вхідного тексту знаходяться в різних *стовпцях і різних рядках*, то вони замінюються на символи, що знаходяться в тих самих рядках, але *відповідають іншим кутам прямокутника* в ключовій матриці.

Наприклад потрібно зашифрувати біграму «**OR»**. Розглянемо чотири випадки, які можуть зустрітися при шифруванні і представлені на рис.2.2.

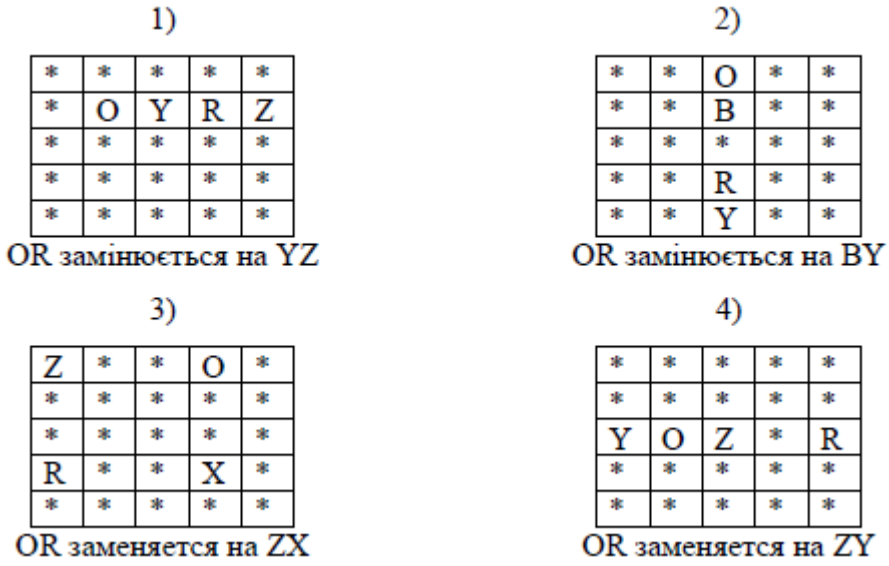


Рис.2.2. Приклад заміни біграми «**OR»** в залежності від розташування в таблиці.

Для розшифрування зашифрованого повідомлення необхідно використовувати інверсію цих чотирьох правил, відкидаючи символи “x” (або “q”), якщо вони не мають сенсу у вхідному повідомленні.

Шифр Плейфера відповідає критеріям для багатоалфавітного шифру. Ключ — потік підключів, у якому вони створюються по два одночасно. У шифрі Плейфера потік ключів і потік шифру — такі самі. Це означає, що вищезазначені правила можна представити як правила для створення потоку ключів. Алгоритм шифрування бере пару символів з вхідного тексту та створює пару підключів, згідно вищезазначених правил. Можна сказати, що потік ключів залежить від позиції символу у початковому тексті. Ця залежність від позиції має різну інтерпретацію: підключ для кожного символу вхідного тексту залежить від наступного або попереднього. Отже,

розглядаючи шифр Плейфера, можна сказати, що зашифрований текст — це фактично потік ключів.

Нехай вхідний текст: $M = m_1, m_2, m_3, \ldots$.

Зашифрований текст: $C = c_1, c_2, c_3, \ldots$

Ключ: $K = [(k_1,k_2), (k_3,k_4), \ldots]$.

Шифрування текстових повідомлень за допомогою шифру Плейфера здійснюється за виразом

$$c_i = k_i,$$

а розшифрування

$$mi = ki.$$

Приклад 2.1. Зашифрувати повідомлення «***Hide the gold in the tree stump***» для ключового слова «**playfair example**» [11] .

Розв'язання. Побудуємо матрицю літер з врахуванням наданого ключового слова, яке буде вписане першим без повторів літер (рис.2.3):

P	L	A	Y	F
I	R	E	X	M
B	C	D	G	H
J	K	N	O	S
T	U	V	W	Z

Рис.2.3. Побудова квадрата Плейфера з наданим ключовим словом

З оригінального повідомлення утворимо послідовність біграм: ***HI DE TH EG OL DI NT HE TR EX ES TU MP***. Згідно наведених правил заміни, утвориться наступний набір біграм:

HI → BM
DE → ND
TH →ZB
EG →XD
OL →KY
DI →BE
NT →JV
HE →DM
TR →UI
EX →XM
ES → MN
TU →UV
MP →IF

Таким чином повідомлення «***Hide the gold in the tree stump***» перетвориться на «***BMNDZBXDKYBEJVDMUIXMMNUVIF***».

Цей приклад показує, що шифр Плейфера — фактично багатоалфавітний шифр: однакові символи зашифровані різними символами.

Для використання кириличного алфавіту необхідно збільшити розмір матриці до 6×6. Використовуються 32літери алфавіту та додатково чотири символи: “.” (крапка); “,” (кома); “–” (тире); “_” (знак підкреслення).

Очевидно, атака “грубої сили” на шифр Плейфера є надто складною, адже розмір домену — n! (n факторіал). Крім того, зашифрований текст приховує частоту появи окремих символів. Однак частота появи двосимвольних комбінацій (біаграм) зберігається до деякого ступеня через вставки наповнювача так, що криптографічний аналітик для знаходження ключа може використовувати тільки атаку на зашифрований текст, засновану на випробуванні частоти появи біграм.

2.3. Шифр підстановки Віженера

Ще одним цікавим видом багатоалфавітного шифру підстановки є шифр Віженера, створений Блезом де Віженером, французьким дипломатом і криптографом, алхіміком і астрологом. Його іменем названо шифр Віженера, який насправді винайшов Джованні Баттіста Белласо, але цей шифр був помилково приписаний у 19 столітті саме Віженеру. Шифр Віженера є класичним представником поліалфавітних шифрів, в яких ПОЗИЦІЯ букви у відкритому тексті впливає на те, за яким саме ПРАВИЛОМ ця буква буде змінена.

Відкритий текст і **криптотекст** записують в одному й тому ж алфавіті. Для довільних букв **X** та **Y** цього алфавіту їх сумою **X** + **Y** будемо вважати результат циклічного зсуву букви **X** вправо в алфавіті на кількість позицій, що дорівнює номеру букви **Y** в алфавіті. При цьому нумерація букв алфавіту починається з нуля (рис.2.4).

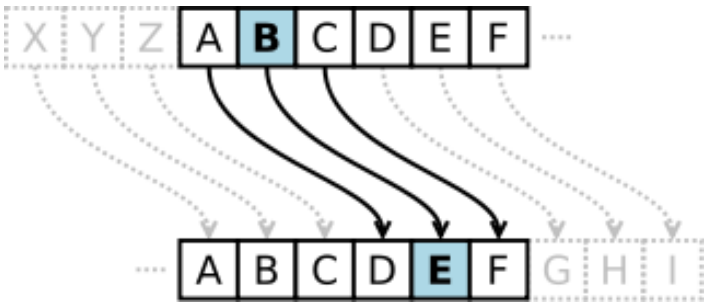


Рис.2.4. Зсув літери «B» (**X**) на кількість позицій, визначених значенням літери «**Y**», яка є елементом ключа (Key) (*https://uk.wikipedia.org/wiki/Шифр_Цезаря*).

Шифр **Віженера** застосовується до повідомлення, записаного в рядок без пропусків між словами та розділових знаків. Ключем є слово у тому ж алфавіті. Якщо **ключ коротший** за повідомлення, то його **записують багато разів підряд**, доки не вийде рядок такої ж довжини. Перед шифруванням потрібно вилучити з тексту розділові знаки та пробіли (рис.2.5.)

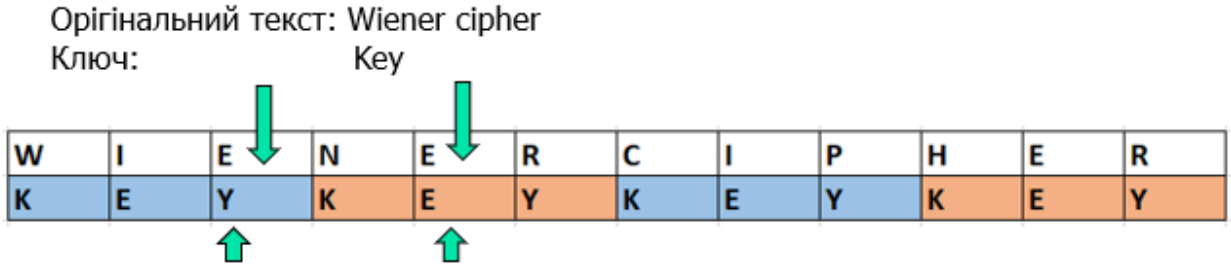


Рис.2.5. Багаторазове повторення ключа відносного повідомлення

Шифр Віженера використовує різну стратегію створення потоку ключів. Потік ключів — повторення початкового потоку ключа засекречування довжини n.

Шифр може бути описаний таким чином: (k_1, k_2, ..., k_n) — секретний ключ, узгоджений відправником та одержувачем. Нехай вхідний текст:

M = m1, m2, m3, ...

Зашифрований текст:

C = c1, c2, c3, ...

Потік ключів:

K = k1, k2, k3, ...

Зашифрування текстових повідомлень за допомогою шифру Віженера здійснюється за допомогою виразу

ci = (mi + ki)mod n,

а розшифрування

mi = (ci – ki)mod n.

Однією з важливих відмінностей між шифром Віженера та іншими розглянутими багатоалфавітними шифрами є те, що потік ключів шифру Віженера не залежить від символів вхідного тексту: він залежить тільки від позиції символу в початковому тексті. Іншими словами, потік ключів може бути створений без знання суті вхідного тексту [5].

На відміну від шифру простої заміни при використанні шифру Віженера ОДНАКОВИМ БУКВАМ у відкритому тексті можуть відповідати РІЗНІ БУКВИ у криптотексті. Це значно ускладнює частотний криптоаналіз. Шифр Віженера кілька століть вважався надійним, поки у минулому столітті не виявлено, що цей шифр все ж піддається частотному методу.

Приклад 2.2. Зашифрувати повідомлення «**She is listening**», використовуючи ключове слово із шести символів «**PASCAL**».

Розв'язання. Для переведення літер в їх числові позначення скористаємося рис.1.9. Початковий потік ключів для слова «**PASCAL**»— це (15, 0, 18, 2, 0, 11). Потоком ключів є повторення цього початкового потоку ключів (стільки разів, скільки потрібно). Процес зашифрування надано у табл. 2.1.

Таблиця 2.1

Перетворення відкритого тексту в криптотекст
з врахуванням ключового слова

Вхідний текст *M*	*s*	*h*	*e*	*i*	*s*	*l*	*i*	*s*	*t*	*e*	*n*	*i*	*n*	*g*
Значення *M*	*18*	*07*	*04*	*08*	*18*	*11*	*08*	*18*	*19*	*04*	*13*	*08*	*13*	*06*
Потік ключів	*15*	*00*	*18*	*02*	*00*	*11*	*15*	*00*	*18*	*02*	*00*	*11*	*15*	*00*
Значення *C*	*07*	*07*	*22*	*10*	*18*	*22*	*23*	*18*	*11*	*6*	*13*	*19*	*02*	*06*
Зашифрований текст *C*	*H*	*H*	*W*	*K*	*S*	*W*	*X*	*S*	*L*	*G*	*N*	*T*	*C*	*G*

Отже, зашифрований текст буде мати такий вигляд:

“HHWKSWXSLGNTCG”.

Шифр Віженера може розглядатися як комбінації адитивних шифрів [5]. Рис. 2.6 показує, що вхідний текст попереднього прикладу можна розглядати як такий, що складається з декількох частин по шість елементів у кожному (хоча в одному не вистачило літер вхідного тексту), де кожний з елементів зашифрований окремо.

Рис. 2.6. Шифр Віженера як комбінація адитивних шифрів

Є *n* частин вхідного тексту, кожна з яких зашифрована різним ключем, щоб розділити зашифрований текст на *m* частин. *Адитивний шифр* — окремий випадок шифру Віженера, в якому *m = 1*.

Інший спосіб розгляду шифрів Віженера проводиться за допомогою ***таблиці Віженера*** [12].

Квадрат Віженера складається з *lA* стовпців і такої ж кількості рядків, де *lA* – довжина базового алфавіту. У якості базового алфавіту візьмемо

український алфавіт, який містить тільки прописні букви, тобто *A*={А,Б,В,Г,Ґ,Д,...,Ю,Я}. Це означає, що розмір квадрату буде 33*33.
Алгоритм генерації квадрату Віженера складається з наступних етапів:

1. У додатковий рядок і додатковий стовпець квадрату заноситься базовий алфавіт (перший совпець і перший рядок таблиці, див. Таблиця 2.2).

2. Задати ключ К1. Візьмемо, наприклад, ключ ***K1***=13572468.

3. До базового алфавіту застосовують початкову перестановку по ключу ***K1***, у результаті чого отримують перший рядок квадрату (див. Таблиця 2.2).

4. Другий рядок квадрату отримується шляхом циклічного зсуву на один символ уліво першого рядка. Витиснутий перший символ переноситься у кінець рядка.

5. Третій рядок формується із другого аналогічно: виконується циклічний зсув уліво другого рядка на один символ, а витиснутий перший символ переноситься у кінець рядка. Так продовжують, доки не буде остаточно заповнено усі рядки квадрату.

Слід відмітити, що завдяки перестановці по ключу ***K1*** на третьому етапі алгоритму, квадрат Віженера може містити ***lA*** різноманітних варіантів алфавітів $l_A * l_A$, тобто 33*33.

Таблиця Віженера, яку побудовано на базі алфавіту *A*={А,Б,В,Г,Ґ,Д,...,Ю,Я} з використанням ключа ***K1***=13572468 згідно викладеному алгоритму, наведено нижче (див. Таблиця 2.2).

Зашифрування – криптоперетворення ***S=f(*C,K1,K2*).***

Зашифрування виконується посимвольно. Кожному символу відкритого повідомлення ***C*** вибирається рядок, а по символу ключового слова ***K2*** стовпець квадрату Віженера, на перетинанні яких усередині квадрату визначається символ шифротексту.

Ключове слово ***K2*** може бути будь-якої довжини і складатися з будь-яких символів алфавіту. Якщо ключове слово коротше за повідомлення, його повторюють циклічно.

Розшифрування – криптоперетворення ***C=f(S,K1,K2)***.

За ключем ***K1*** формують таблицю Віженера. Потім по символу ключового слова ***K2*** вибирається рядок, в якому усередині квадрату знаходимо символ шифротексту. В стовпці, який відповідає символу шифротексту, визначаємо символ відкритого повідомлення.

Приклад 2.3: Зашифрувати повідомлення ***«КРИПТОСИСТЕМА»*** використовуючи ключ ***K1***=13572468 (див. Таблиця 2.2) та ключ ***K2****=«КЛЮЧ».*

Розв'язання. Відповідно до алгоритму зашифрування: на перетині рядка, в якому розміщена перша літера відкритого повідомлення «***К***» і стовпця, в якому міститься перший символ ключа ***К2*** (це символ «***К***») отримаємо символ «***Ш***», який є першим символ зашифрованого повідомлення. На перетині рядка, в якому міститься друга літера відкритого повідомлення «***Р***» і стовпця, в якому міститься другий символ ключа ***К2*** (це символ «***Л***») отримаємо символ «***Ґ***», який є другим символом зашифрованого повідомлення, і т. д. Отримаємо зашифроване повідомлення «***ШҐЖЙЙАПБҐББИК***».

Таблиця 2.2

Представлення таблиці Віженера з ключем K_1=13572468.

	А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
А	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
Б	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А
В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В
Г	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ
Ґ	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е
Д	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б
Е	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г
Є	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д
Ж	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є
З	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж
И	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З
І	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И
Ї	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І
Й	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї
К	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й
Л	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К
М	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л
Н	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М
О	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н
П	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О
Р	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П
С	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р
Т	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С
У	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ь	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ю	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь
Я	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю

2.4 Підготовка до завдання

Ознайомитися з теоретичними положеннями задач інформаційної безпеки стосовно захищеності даних, дій, що забезпечують рішення задач інформаційної безпеки.

Ознайомитися з базовими шифрами багатоалфавітної підстановки (заміни) - шифри Плейфера, Віженера, методами їх побудови та реалізації.

2.5 Практичне завдання

1. Записати своє прізвище, ім’я та по-батькові (ПІП) та представити у цифровій формі, згідно рис.1.10. Наприклад, ШЕВЧЕНКО -→ «28 06 02 27 06 17 14 18».
2. Для шифрування тексту використати багатоалфавітні шифри підстановки - шифр Плейфера та шифр Віженера із самостійно обраним ключем.
3. Провести зворотну процедуру по розшифруванню інформації. Порівняти отриманий результат з оригінальним текстом.
4. Виконати програмну реалізацію функцій шифрування і розшифрування по п.1-3. одного з шифрів (Плейфера або Віженера) за вибором.
5. Надати аналіз отриманим результатам.

2.6 Зміст протоколу роботи

Протокол до виконаного практикуму оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

2.7 Контрольні питання для самоперевірки

1. Навести загальну схему симетричного шифрування. Яке призначення закритого ключа в цій системі?
2. Дайте визначення шифрів підстановки (заміни). Сформулювати загальні принципи для методів шифрування підстановкою.
3. Пояснити сутність та розбіжності шифрів моноалфавітної та багатоалфавітної підстановок.
4. Пояснити сутність шифру підстановки Плейфера.
5. Пояснити сутність шифру підстановки Віженера.
6. Пояснити шифр Віженера як комбінацію адитивних шифрів.
7. Пояснити функціонали перетворення шифром Віженера при застосуванні функції криптоперетворення $S=f(C,K1,K2)$.
8. Скільки правил потрібно дотримуватися, щоб правильно зашифровувати та розшифровувати повідомлення, використовуючи шифр Плейфера?

2.8. Задачі до Практикуму №2

1. Використовуючи шифр Плейфера і кириличну матрицю (див. рис.2.1.), зашифрувати повідомлення українською мовою «УНІВЕРСИТЕТ» при використанні ключа «ПЛЕЙФЕР».
2. Використовуючи шифр Віженера та ключове слово «ЕКЗАМЕН», зашифрувати повідомлення «ВІДМІННО» українською мовою.
3. Використовуючи шифр Віженера та ключове слово «INK», розшифрувати повідомлення «KEIXGYTBQG» з використанням латинського алфавіту.
4. Розшифрувати повідомлення «GOGROSZY», яке було зашифроване шифром Плейфера при застосуванні латинського алфавіту (рис.2.1.) з ключем «HELLO».
5. Розшифрувати повідомлення «CUPI», яке було зашифроване шифром Плейфера при застосуванні латинського алфавіту (рис.2.1.) з ключем «BYTE».
6. Який ключ використано при застусуванні наступної матриці шифру Плейфера?

C	R	Y	P	T
O	A	B	D	E
F	G	H	I/J	K
L	M	N	Q	S
U	V	W	X	Z

7. Розшифрувати шифротекст «КЇЦБСУРД» при зашифруванні якого використовували автоключовий шифр (*k1 = 11*) та українську абетку (рис.1.10).

2.9 Завдання до самостійної роботи

1. Опишіть алгоритм шифру огорожі (*rail fence cipher*).
2. Пояснити сутність афінного шифру підстановки. Навести приклад шифрування та розшифрування.
3. Пояснити сутність автоключового шифру підстановки. Навести приклад шифрування та розшифрування.
4. Пояснити сутність шифру Вермана.

ПРАКТИКУМ № 3
Шифри одинарної перестановки

Мета роботи. Ознайомитися з теоретичними положеннями шифрів одинарної перестановки, на практичних прикладах продемонструвати створення ключа шифрів, провести зашифрування відкритого і розшифрування шифрованого повідомлення.

3.1 Основні поняття про перестановки

Перестановочний шифр — алгоритм шифрування, який полягає у перестановці знаків відкритого тексту згідно з певним правилом, яке є ключем [5, 22-24].

Шифр перестановки не замінює одним символом інший, замість цього він змінює місце розташування символів. Символ у першій позиції вхідного тексту може з'явитися в десятій позиції зашифрованого тексту. Символ, який знаходиться у восьмій позиції вхідного тексту, може з'явитися в першій позиції зашифрованого тексту і т.д. Іншими словами, шифр перестановки ставить (переміщає) в іншому порядку символи. Яскравим прикладом шифру перестановки є так звана Анаграма.

Анагра́ма (грец. ανα- — знову та γράμμα — літера) — переставлення літер у слові, завдяки чому утворюється нове значення, прочитуване у зворотному напрямку (*тік — кіт*), постають псевдоніми (*Симонов — Номис*) чи слова (*мука — кума, літо — тіло*). Винахідником анаграм вважають грецького граматика Лікофрона (III століття до н. е.). Приклади анаграм: *апельсин - спаниель, полковник - клоповник, горилка - рогалик, лепесток – телескоп* та ін.

Одним із прикладів шифру без використання ключа є ***шифр огорожі*** (*rail fence cipher*).

Приклад 3.1. Наприклад, текстове повідомлення «*Робіть те, у що вірите, і вірте в те, що робите*» (Нісаргадатта Махарадж) буде зашифроване як «*рбттуоіиевретщрбтоіьещвртііітвеооие*» в результаті зчитування (без знаків пунктуації), спочатку верхнього рядка, а потім нижнього (рис.3.1.)

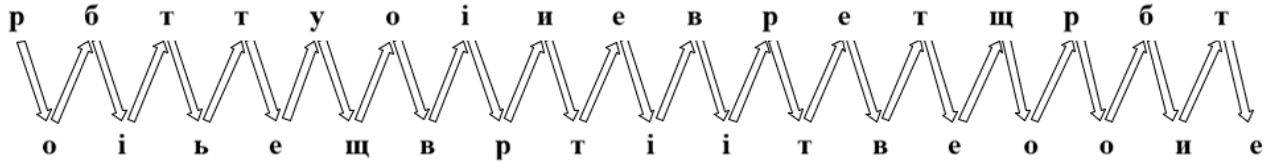


Рис.3.1. Демонстрація шифру огорожі

Одержувач отримує зашифрований текст і поділяє його навпіл. Перша половина форми — перший рядок, друга половина — другий.

Широке поширення отримали шифри перестановки, які використовують деяку геометричну фігуру (плоску або об'ємну). Перетворення полягають у тому, що в фігуру вихідний текст вписується по ходу одного маршруту, а

виписується по іншому. Одним із давній і історичних представників даного шифрування є **Скитала** (грец. Σκυτάλη «палиця») — пристрій шифрування, який використовувався у Стародавній Спарті (рис.3.2.).

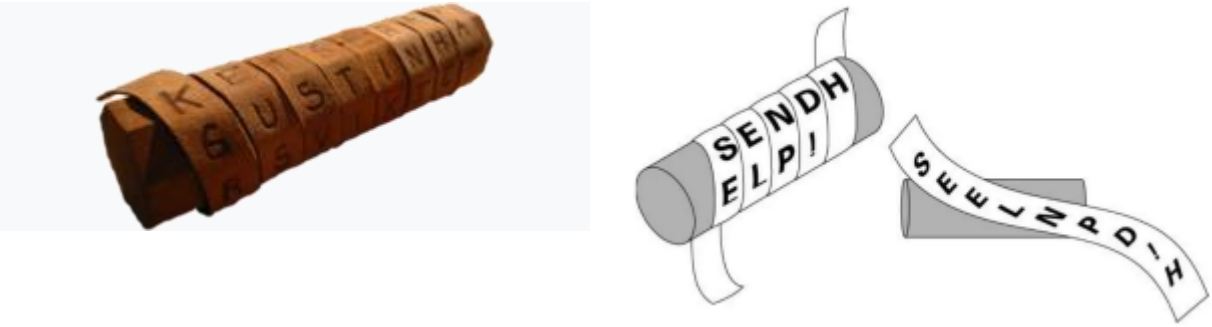


Рис.3.2. Скитала — стародавній пристрій шифрування (https://uk.wikipedia.org/wiki/Скитала)

Даний пристрій представляє собою дерев'яний циліндр, на який намотувалась шкіряна стрічка. Перпендикулярно стрічці писалось повідомлення, потім стрічка розмотувалась і передавалась одержувачу. Ключем є діаметр циліндра.

З широкими можливостями комп'ютерної техніки щодо обробки даних, шифр перестановки набув свого «нового дихання» і активно використовується в різноманітних сучасних шифрах.

3.2. Шифри перестановки з використанням ключа

Шифр перестановки не замінює одним символом інший, замість цього він змінює місце розташування символів. Символ у першій позиції вхідного тексту може з'явитися в десятій позиції зашифрованого тексту. Символ, який знаходиться у восьмій позиції вхідного тексту, може з'явитися в першій позиції зашифрованого тексту. Іншими словами, шифр перестановки ставить (переміщає) в іншому порядку символи.

Прості шифри перестановки, які застосовувалися в минулому, не використовували ключ. Є два методи перестановки символів. У першому методі текст записується в таблицю стовпець за стовпцем і потім передається рядок за рядком. У другому методі навпаки, текст записується в таблицю рядок за рядком і потім передається стовпець за стовпцем.

У загальному випадку для даного класу шифрів при шифруванняі і дешифрування використовується таблиця перестановок (рис.3.3):

1	2	3	…	n
I_1	I_2	I_3	…	I_n

Рис.3.3. Таблиця перестановок для визначення правила заміни

У першому рядку цієї таблиці вказується позиція символу в вхідному повідомленні, а в другій - його позиція в шифрограмі. Таким чином,

максимальна кількість ключів для шифрів перестановки n!, де n - довжина повідомлення.

Інший метод полягає в тому, щоб розділити вхідний текст на групи заздалегідь визначеного розміру, називані **блоками**, а потім використовувати ключ, щоб переставити символи в кожному блоці окремо.

Приклад 3.2. Відправник повинен передати одержувачу повідомлення «**Я ВИВЧАЮ КРИПТОЛОГІЮ**». Відправник і одержувач погодилися розділити текст на групи по п'ять символів і потім переставити символи в кожній групі.

Нижче показано угруповання після додавання фіктивного символу «**Ь**» в кінці, щоб зробити останню групу однаковою за розміром з іншими:

1	2	3	4
ЯВИВЧ	**АЮКРИ**	**ПТОЛО**	**ГІЮЬЬ**

Ключ, використовуваний для зашифрування й розшифрування, — ключ перестановки, який показує як переставляти символи. Для цього повідомлення приймемо, що відправник і одержувач використовували такий ключ (рис.3.4.):

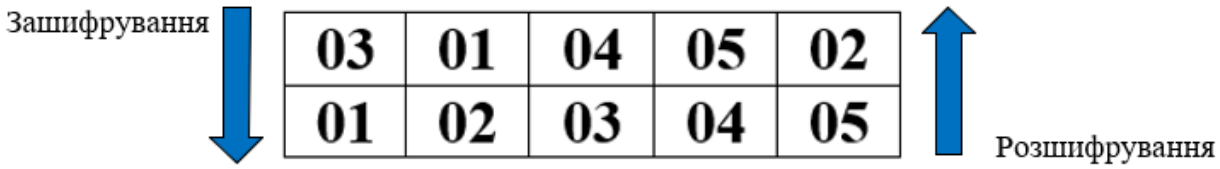


Рис.3.4. Ключ - таблиця перестановок

Третій символ у блоці вхідного тексту стає першим символом у зашифрованому тексті в блоці, перший символ у блоці вхідного тексту стає другим символом у блоці зашифрованого тексту і так далі. Результати перестановки такі:

«**ИЯВЧВ КАРИЮ ОПЛОТ ЮГЬЬІ**»

Відправник передає зашифрований текст одержувачу:

«ИЯВЧВ КАРИЮ ОПЛОТ ЮГЬЬІ»

Одержувач розділяє зашифрований текст на групи по п'ять символів і, використовуючи ключ в оберненому порядку, знаходить вхідний текст.

3.3. Шифри перестановки стовпців за ключем

Сучасні шифри перестановки обробляються таким чином, що вони можуть бути прийнятими тільки приймачем, який оснащений відповідним дешифратором. Зашифрування й розшифрування таких шифрів відбувається в три кроки.

Перший крок: текст пишеться в таблицю рядок за рядком.

Другий крок: робиться перестановка, змінюючи порядок проходження стовпців.

Третій крок: стовпець за стовпцем читається нова таблиця.

Перший і третій кроки забезпечують безключову глобальну зміну порядку проходження; другий крок забезпечує блокову ключову перестановку. Ці типи шифрів згадуються часто як ключові шифри перестановки стовпців.

Приклад 3.3. Припустимо, що відправник знову зашифровує повідомлення «**Я ВИВЧАЮ КРИПТОЛОГІЮ**», цього разу використовуючи об'єднаний підхід. Зашифрування й розшифрування показано на рис.3.5.

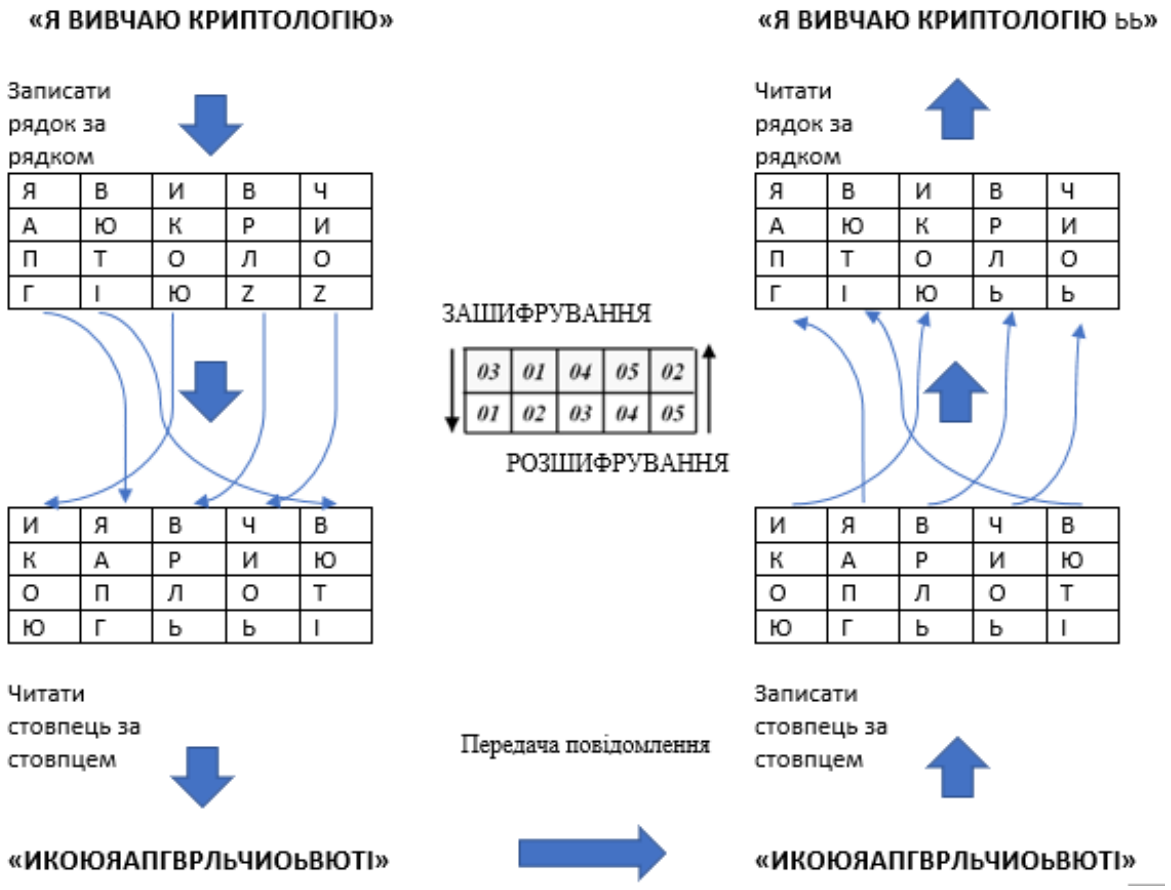


Рис.3.5. Приклад перестановки стовпців за ключем

Можна використовувати матриці, щоб показати процес зашифрування /розшифрування для шифру перестановки.

Оригінальний текст $M(l \times g)$ і зашифрований текст $C(l \times g)$ — матриці $l \times g$, що представляють числові значення символів; ключі $K(g \times g)$ — квадратні матриці розміру $g \times g$.

Тоді алгоритм зашифрування даних можна надати у вигляді

$$C(l \times g) = M(l \times g) \times K_{Ш}(g \times g)$$

а розшифрування

$$M(l \times g) = C(l \times g) \times K_{Р}(g \times g),$$

де $K_Ш(g \times g)$ і $K_P(g \times g)$ — ключові матриці (*матриці перестановки*), які використовуються для зашифрування й розшифрування відповідно. Ключові матриці є оберненими

$$K_P(g \times g) = K_Ш^{-1}(g \times g) \quad \text{і} \quad K_Ш(g \times g) = K_P^{-1}(g \times g)$$

Зашифрування виконується множенням матриці вхідного тексту на ключову матрицю, щоб отримати матрицю зашифрованого тексту; розшифрування виконується множенням зашифрованого тексту на обернену ключову матрицю, після чого отримуємо вхідний текст.

Приклад 3.4. Продемонструємо застосування матричного способу шифрування для прикладу, наведеного вище (рис.3.6):

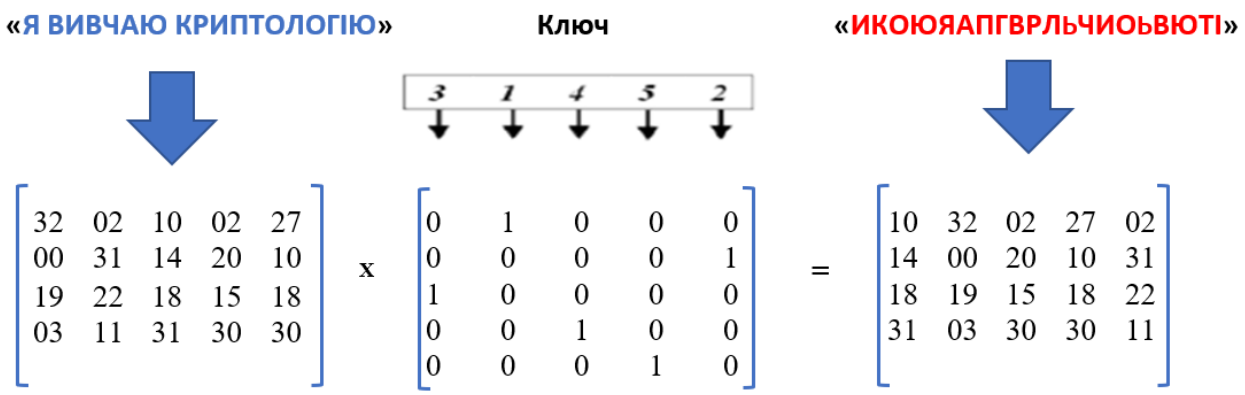


Рис.3.6. Приклад процесу зашифрування даних із використанням шифру перестановок

Повідомлення «**Я ВИВЧАЮ КРИПТОЛОГІЮ**» в цифровому еквіваленті, згідно відповідності літер цифрам для української мови (рис.1.10), має вигляд: «32, 02, 10, 02, 27, 00, 31, 14, 20, 10, 19, 22, 18, 15, 18, 03, 11, 31, 30, 30» із використанням перестановки стовпців матриці початкових даних 3, 1, 4, 5, 2.

Для довідки – множення матриць.

Множення матриць — це бінарна операція, яка використовуючи дві матриці, утворює нову матрицю, яка називається добу́тком матриць.

Нехай дано дві прямокутні матриці A та B розмірністю $m \times n$ $n \times q$ відповідно:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \quad B = \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1q} \\ b_{21} & b_{22} & \cdots & b_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nq} \end{bmatrix}.$$

Тоді матриця C розмірністю m×q називається їх добутком:

$$C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1q} \\ c_{21} & c_{22} & \cdots & c_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \cdots & c_{mq} \end{bmatrix},$$

де

$$c_{ij} = \sum_{r=1}^{n} a_{ir} b_{rj} \quad (i = 1, 2, \ldots m;\ j = 1, 2, \ldots q)$$

.

Приклад 3.5. Обчислити добуток двох матриць:

$$\begin{pmatrix} 3 & 5 \\ 6 & -1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ -3 & 2 \end{pmatrix}.$$

Розв'язок:

$$\begin{pmatrix} 3 & 5 \\ 6 & -1 \end{pmatrix} \begin{pmatrix} 2 & 1 \\ -3 & 2 \end{pmatrix} = \begin{pmatrix} 3 \cdot 2 + 5 \cdot (-3) & 3 \cdot 1 + 5 \cdot 2 \\ 6 \cdot 2 - 1 \cdot (-3) & 6 \cdot 1 - 1 \cdot 2 \end{pmatrix} = \begin{pmatrix} -9 & 13 \\ 15 & 4 \end{pmatrix}.$$

Множення матриці початкового тексту М(4 × 5) на ключову матрицю зашифрування $К_{Ш}$(5 × 5) дає матрицю зашифрованого тексту С(4×5).

Матрична маніпуляція вимагає зміни символів до їх числових значень (від 00 до 32 для української мови). Зауважимо, що матричне множення забезпечує тільки перестановку стовпців; читання й запис у матрицю повинні бути забезпечені іншою частиною алгоритму.

Отже, зашифроване повідомлення набуде вигляду 10, 32, 02, 27, 02, 14, 00, 20, 10, 31, 18, 19, 15, 18, 22, 31, 03, 30, 30, 11, що при зчитуванні стовпця за стовцем відповідає набору цифр «10, 14, 18, 31, 32, 00, ….», або зашифрованому повідомленню:

«ИКОЮЯАПГВРЛЬЧИОЬВЮТІ».

Приклад 3.6. На рис.3.7 показано процес розшифрування повідомлення із використанням перестановки стовпців матриці початкових даних 2, 5, 4, 1, 3, яка обернена матриці перестановки. Множення матриці зашифрованого тексту С(4 × 5) на ключову матрицю розшифрування (5 × 5), дає матрицю початкового тексту М(4 × 5).

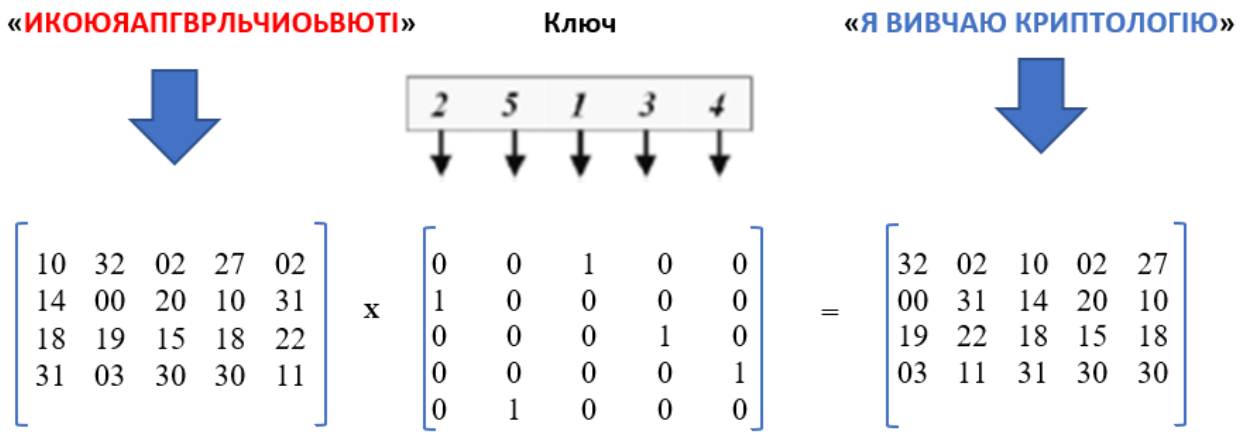


Рис.3.7. Приклад процесу розшифрування даних із використанням шифру перестановок

Отже, розшифрований текст в цифровому еквіваленті набуде вигляду «32, 02, 10, 02, 27, 00, 31, 14, 20, 10, 19, 22, 18, 15, 18, 03, 11, 31, 30, 30», що, згідно відповідності літер цифрам для української мови (рис.1.10), відповідає повідомленню:

«**Я ВИВЧАЮ КРИПТОЛОГІЮ**».

3.4. Підготовка до завдання

Ознайомитися з базовими шифрами одинарної перестановки, методами їх побудови та реалізації.

3.5 Практичне завдання

1. Записати своє прізвище, ім’я та по-батькові (ПІП) та представити у цифровій формі, згідно перетворенням, наведеним на рис.1.10. Наприклад, ШЕВЧЕНКО -→ «28 06 02 27 06 17 14 18».
2. Розбити даний текст на блоки (розмір обирається самостійно, 4-5 символів) з доповненням до блоку при необхідності додаткових фиктивних символів, наприклад знаку «Ь».
3. Для зашифрування тексту використати шифр перестановки із самостійно обраним ключем (одномірний масив), співставлений з розміром блока. Провести процедуру зашифрування тексту.
4. Провести зворотну процедуру по розшифруванню інформації. Провести порівняльний аналіз отриманого результату.
5. Виконати програмну реалізацію функцій зашифрування і розшифрування по п.1-4.
6. Надати аналіз отриманим результатам.

3.6 Зміст протоколу роботи

Протокол до виконаної практичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

3.7. Контрольні питання для самоперевірки

1. Дати визначення шифру перестановки.
2. Пояснити шифр простої одинарної перестановки.
3. Пояснити шифр блочної одинарної перестановки.
4. Дати визначення шифрів маршрутної табличної перестановки.
5. Пояснити сутність шифрів перестановки без використання ключа.
6. Пояснити сутність шифрів перестановки з використанням ключа.
7. У чому полягає відмінність процесів розшифрування та розкриття шифрів?
8. Сформулюйте алгоритм шифрування тексту подвійною перестановкою.

3.8. Задачі до практикуму №3

1. Виконати методом перестановки шифрування тексту «КРИПТОСТІЙКІСТЬ ШИФРУ ЗАЛЕЖИТЬ ВІД КЛЮЧА» із ключем 426135 (перестановка по стовпцях).

2. Виконати методом подвійної перестановки шифрування тексту «КРИПТОСТІЙКІСТЬ ШИФРУ ЗАЛЕЖИТЬ ВІД КЛЮЧА» із ключами 426135 (перестановка по стовпцях) і 254316 (перестановка по рядках).

3. Методом магічного квадрата, який має вид

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

,

виконати шифрування тексту «ВІРТУАЛЬНИЙ КАНАЛ».

4. Виконати шифрування тексту «СВІТ ЛОВИВ МЕНЕ, ТА НЕ СПІЙМАВ» одинарною перестановкою з ключем «ЗАХИСТ».

5. Ключ для шифра перестановки складаэться з 5 позицій. Чому дорівнює потужність ключового простору?

6. Для шифрування повідомлення «ОСВІТА» був використаний шифр підстановки з ключем у вигляді таблиці співвідношення літер:

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇	⬇
П	Д	Е	М	Л	О	Ґ	А	К	Щ	Ю	Я	С	І	Г	В	З	Р	Є	Б	Н	Й	Ч	Т	У	Ц	Ь	Ф	Х	Ж	Ш	Ї	И

При першому циклі шифрування повідомлення «ОСВІТА» заміниться на «ЄЙЕЯЧП». При виконанні ще раз цієї процедури шифрування повідомлення «ЄЙЕЯЧП» заміниться на «АЇИФБ», потім «АЇИФБ» заміниться на «ПЯЛЮУД» і т.д. Скільки потрібно виконати циклів шифрування, щоб знову побачити повідомлення «ОСВІТА»?

7. При використанні шифру Хілла та ключової матрицю *KEY*

$$KEY = \begin{pmatrix} 11 & 16 & 09 \\ 05 & 12 & 22 \\ 02 & 10 & 13 \end{pmatrix},$$

зашифрувати СВОЄ ПРІЗВИЩЕ та ІМЯ українською мовою при використанні подання літер, представлених на рис.1.10.

3.9 Завдання до самостійної роботи

1. Шифрування методом магічного квадрата. Яка кількість можливих різних комбінацій магічних квадратів розміром 3x3, 4x4, 5x5?
2. Шифрування методом шифру огорожі. Параметри шифру.
3. Опишіть алгоритм шифрування криптосистемою Хілла.
4. Що являє собою ключ в криптосистемі Хілла?
5. Поняття оберненої матриці. Алгоритм знаходження оберненої матриці.

ПРАКТИКУМ № 4
Гамування

Мета роботи. Ознайомитися з теоретичними основами шифру гамування, на практиці здійснити створення послідовності буквеної гами, провести зашифрування відкритого і розшифрування шифрованого повідомлення. Усвідомити сильні і слабкі сторони гамування.

4.1 Основні поняття про гамування

Суть методу гамування полягає в тому, що символи даних, які шифруються, послідовно складаються із символами деякої спеціальної послідовності, яка називається *гамою*.

Гамування - метод симетричного шифрування, що полягає в «накладенні» послідовності, що складається з випадкових чисел, на відкритий текст. Послідовність випадкових чисел називається гама-послідовністю і використовується для зашифровування і розшифрування даних. Додавання, зазвичай, виконується в будь-якому кінцевому полі, наприклад, в полі Галуа GF(2) додавання набуває вигляду операції «виключне АБО (XOR)».

Процедуру накладення *гами* на вхідні дані можна здійснити двома способами.

У першому способі символи вхідного тексту й гами замінюються цифровими еквівалентами, які потім складаються за модулем n, де n — кількість символів в алфавіті, тобто:

$$c_i = (m_i + g_i) mod\ n, \qquad (4.1)$$

де c_i, m_i, g_i — символи відповідно зашифрованого тексту, вхідного тексту й гами.

Процес шифрування (*гамування*) можна продемонструвати структурною схемою на рис.4.1, де в якості послідовності ключів виступає послідовність *гами* **К**, яка формується Генератором псевдовипадкової послідовності (ГПВП) при початковій ініціалізації на передавальній і приймальній стороні.

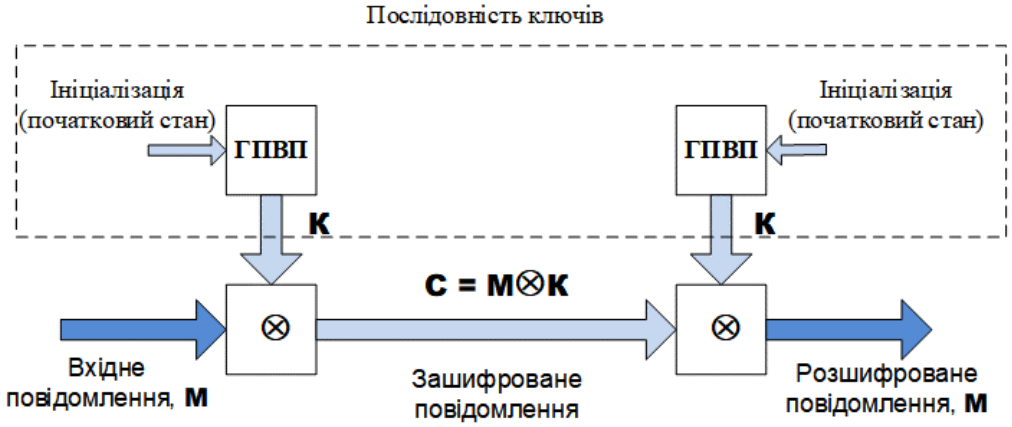


Рис.4.1. Процес шифрування повідомлення за допомогою гами

Приклад 4.1. Зашифруємо відкрите повідомлення «**УНІВЕРСИТЕТ**», цифровий еквівалент якого для для українського алфавіту n = 33 (рис.1.10), відповідає наступній послідовності чисел: «23 17 11 02 06 20 21 10 22 06 22» .

Оберемо в якості *гами* слово «**ЗАХИСТ**», цифрове представлення якої має вид «09 00 25 10 21 22». Гама буде повторюватися, якщо її розмір менше за повідомлення.

Розв'язання. Для зашифрування відкритого тексту скористаємося співвідношенням (4.1) і отримаємо:

C1 = (23+09)mod33 = 32 ; C7 = (21+09)mod33 = 30;

C2 = (17+00)mod33 = 17; C8 = (10+00)mod33 = 10;

C3 = (11+25)mod33 = 03; C9 = (22+25)mod33 = 14;

C4 = (02+10)mod33 = 12; C10 = (06+10)mod33 = 16;

C5 = (06+21)mod33 = 27; C11 = (22+21)mod33 = 10.

C6 = (20+22)mod33 = 09;

Отже, вийшов зашифроване повідомлення «32 17 03 12 27 09 30 10 14 16 10», яке з урахуванням переведення в еквівалент літер (рис. 1.10), представляється у вигляді:

«ЯНГЇЧЗЬИКМИ».

За *другого способу* символи вхідного тексту й гами представляються у вигляді двійкового коду, а потім відповідні розряди складаються за модулем *2*:

$$c_{ij} = m_{ij} \oplus g_{ij},\ i = 1, 2, 3, \ldots, n;\ j = 0, 1, \ldots, l, \qquad (4.2)$$

де $\oplus$ — операція підсумовування за модулем *2*; c_{ij}, m_{ij} і g_{ij} — двійкові символи зашифрованого тексту, вхідного тексту й гами відповідно; n – кількість символів даних, які шифруються (кількість символів зашифрованих даних; кількість символів гами); l — кількість двійкових бітів вхідних символів, гами й зашифрованих символів.

Розшифрування здійснюється шляхом застосування до символів шифротекста й гами оберненої операції:

$$c_i = (m_i - g_i) mod\ n.$$

Приклад 4.2. Розшифруємо шифротекст **«ЯНГЇЧЗЬИКМИ»** із попереднього прикладу, який представлений отриманою цифровою послідовністю «32 17 03 12 27 09 30 10 14 16 10».

Розв'язання.

Для розшифрування повідомлення виконаємо наступні операції:

m1 = (32 - 09)mod33 = 23 ; m7 = (30 - 09)mod33 = 21;

m2 = (17 - 00)mod33 = 17;

m3 = (03 - 25)mod33 = 11;

m4 = (12 - 10)mod33 = 02;

m5 = (27 - 21)mod33 = 06;

m6 = (09 - 22)mod33 = 20;

m8 = (10 - 00)mod33 = 10;

m9 = (14 - 25)mod33 = 22;

m10 = (16 - 10)mod33 = 06;

m11 = (10 - 21)mod33 = 22.

Таким чином, отримали цифрову послідовність «23 17 11 02 06 20 21 10 22 06 22», яка відповідає, згідно рис.1.10, оригінальному повідомленню:

«**УНІВЕРСИТЕТ**».

Стійкість шифрування методом гамування визначається, головним чином, властивостями гами — тривалістю періоду й рівномірністю статистичних характеристик. Остання властивість забезпечує відсутність закономірностей за появи різних символів у межах періоду.

Довжиною періоду гами називається мінімальна кількість символів, після якого послідовність цифр в гамі починає повторюватися. Випадковість розподілу символів по періоду означає відсутність закономірностей між появою різних символів в межах періоду.

Для наведеного вище прикладу, довжина періоду *гами* для слова «**ЗАХИСТ**» складає 6.

По довжині періоду розрізняються *гами* з *кінцевим і нескінченним періодом*. Якщо довжина періоду *гами* перевищує довжину шифротексту, гамма є істинно випадковою і не використовується для шифрування інших повідомлень. Таке перетворення є абсолютно стійким (***досконалий шифр***).

Як нескінченна *гама* може бути використана будь-яка послідовність випадкових символів, наприклад, послідовність цифр числа π або *e*.

Для отримання *гами* використовують різні види псевдовипадкових генераторів.

4.2. Генератор псевдовипадкових чисел на основі алгоритму *BBS*

Для формування *гами* потрібно використовувати випадкові послідовності. Для цих цілей часто використовують генератори псевдовипадкових чисел (ПВЧ).

Широке поширення набув алгоритм генерації ПВЧ, названий ***алгоритмом BBS*** (від прізвищ авторів — *L. Blum, M. Blum, M. Shub*) або *генератором із квадратичним залишком*. Для цілей криптографії цей метод запропонований в 1986 р.

Сутність цього алгоритму: спочатку вибираються два великих простих числа *p* та *q*, які потрібно порівняти з *3* за модулем *4*, тобто при діленні *p* і *q* на *4* повинен виходити однаковий залишок *3*.

Далі обчислюється число $M = p^* q$, так зване ***ціле число Блюма***. Потім вибирається інше випадкове ціле число x, взаємно просте (тобто не має загальних дільників, крім одиниці) з M і обчислюється

$$x_0 = x^2 \bmod M,$$

де x_0 – початкове число генератора.

На кожному n -му кроці роботи генератора обчислюється

$$x_{n+1} = x_n^2 \bmod M.$$

Результатом n -го кроку є один (зазвичай молодший) біт числа x_{n+1}. Іноді, як результат, приймають біт парності, тобто кількість одиниць у двійковому поданні елемента. Якщо кількість одиниць у записі числа парне, біт парності приймається таким, що дорівнює *0*, непарне — біт парності приймається таким, що дорівнює *1*.

Приклад 4.3. Нехай $p = 11$, $q = 19$
(переконуємося, що $11 \bmod 4 = 3$, $19 \bmod 4 = 3$). Тоді $M = p^* q = 11*19 = 209$.
Виберемо x, взаємно просте з M: нехай $x = 5$.
Визначимо початкове число генератора x_0:

$$x_0 = x^2 \bmod M = 5^2 \bmod 209 = 25$$

.

Визначимо перші 15 чисел x_i за алгоритмом ***BBS.***

Результати обчислень показують, що отримується послідовність чисел:

«207 4 16 47 119 158 93 80 130 180 5 25 207 4 16 …. » (рис.4.2.). Таким чином можна бачити, що період гами є обмеженим і дорівнює N=12.

Властивістю цього методу є те, що для отримання з послідовності n-го числа не потрібно обчислювати всі попередні n чисел x_i, x_n можна отримати за формулою:

$$x_n = x_0^{2^{n \bmod ((p-1)(q-1))}} \bmod M .$$

Приклад 4.4. Наприклад, обчислимо x_{15} одночасно з x_0:

$$x_{15} = x_0^{2^{15 \bmod ((11-1)(19-1))}} \bmod 209 =$$

$$= 25^{32768 \bmod 180} \bmod 209 = 16$$

.

$x_1 = 25^2 \ mod \ M = 207 \ ---\rightarrow$ "молодший біт = 1"

$x_2 = 207^2 \ mod \ M = 4 \ ---\rightarrow$ "молодший біт = 0"

$x_3 = 4^2 \ mod \ M = 16 \ ---\rightarrow$ "молодший біт = 0"

$x_4 = 16^2 \ mod \ M = 47 \ --\rightarrow$ "молодший біт = 1"

$x_5 = 47^2 \ mod \ M = 119 \ --\rightarrow$ "молодший біт = 1"

$x_6 = 119^2 \ mod \ M = 158 \ --\rightarrow$ "молодший біт = 0"

$x_7 = 158^2 \ mod \ M = 93 \ --\rightarrow$ "молодший біт = 1"

$x_8 = 93^2 \ mod \ M = 80 \ --\rightarrow$ "молодший біт = 0"

$x_9 = 80^2 \ mod \ M = 130 \ ---\rightarrow$ "молодший біт = 0"

$x_{10} = 130^2 \ mod \ M = 180 \ --\rightarrow$ "молодший біт = 0"

$x_{11} = 180^2 \ mod \ M = 5 \ --\rightarrow$ "молодший біт = 1"

$x_{12} = 5^2 \ mod \ M = 25 \ --\rightarrow$ "молодший біт = 1"

$x_{13} = 25^2 \ mod \ M = 207 \ --\rightarrow$ "молодший біт = 1"

$x_{14} = 207^2 \ mod \ M = 4 \ --\rightarrow$ "молодший біт = 0"

$x_{15} = 4^2 \ mod \ M = 16 \ --\rightarrow$ "молодший біт = 0"

Рис.4.2. Обчислення послідовності гами при застосуванні алгоритму ***BBS.***

4.3. Лінійний конгруентний генератор

Одним із найпростіших генераторів є так званий *лінійний конгруентний генератор*, який для обчислення чергового числа *ki* використовує формулу:

$$k_i = (a \cdot k_{i-1} + b) \ mod \ c,$$

де *a, b, c* — деякі константи, а *ki-1* — попереднє псевдовипадкове число.

Приклад 4.5. Нехай початкове значення генератора дорівнює k0=1, a=7, b=5, c=17. Визначити число на третьому кроці генератора.

Розв'язання. Запишемо вираз генерації псевдовипадкових чисел:

$$\boldsymbol{k_i = (7 * k_{i-1} + 5) mod 17.}$$

Тоді

k1 = (7*1 + 5)mod 17 = 12;
k2 = (7*12 + 5)mod 17 = 4;
k3 = (7*4 + 5)mod 17 = 16.

Якщо розглянути більшу послідовність генерації, то отримаємо наступний набір псевдовипадкових чисел6 «12 4 16 15 8 10 7 3 9 0 5 6 13 11 14 1 12 4 16 15 …..».

Легко бачити, що період гами буде складати N=16. Властивість циклічності характерна для всіх лінійних конгруентних генераторів. Змінюючи значення основних параметрів *a, b* і *c*, можна впливати на довжину періоду й на самі значення *ki*, які породжуються. Так, наприклад, збільшення числа *c* у загальному випадку приводить до збільшення періоду. Якщо

параметри *a, b* і *c* обрані правильно, то генератор буде породжувати випадкові числа з максимальним періодом, що дорівнює *c*. За програмної реалізації значення *c* зазвичай встановлюється таким, що дорівнює

$$2^b \text{ або } 2^{b-1},$$

де *b* — довжина слова *електронної обчислювальної машини* у бітах [5].

4.4. Метод Фібоначчі із запізненням

Метод Фібоначчі із запізненням (*Lagged Fibonacci Genera – tor*) — один із методів генерації ПВЧ, що дозволяє отримати більш високу "*якість*" ПВЧ. Відомі різні схеми використання методу Фібоначчі із запізненням. Один із широко поширених фібоначчієвих датчиків заснований на такій рекурентній формулі:

$$k_i = \begin{cases} k_{i-a} - k_{i-b}, & \text{якщо} \quad k_{i-a} \geq k_{i-b}; \\ k_{i-a} - k_{i-b} + 1, & \text{якщо} \quad k_{i\ a} < k_{i\ b}, \end{cases}$$

де *ki* — дійсні числа з діапазону *[0,1]*; *a, b* — цілі позитивні числа, параметри генератора.

Приклад 4.6. Яке число сформує на третьому кроці роботи (для значення $\boldsymbol{x_5}$) генератор Фібоначчі

$$\boldsymbol{x_i = (x_{i-3} + x_{i-1}) mod 19}$$

з початковими значеннями $\boldsymbol{x_0 = 3, x_1 = 1, x_2 = 1}$?

Розв'язання.

$$x_3 = (x_0 + x_2) mod 19 = (3 + 1) mod 19 = 4,$$

$$x_4 = (x_1 + x_3) mod 19 = (1 + 4) mod 19 = 5,$$

$$x_5 = (x_2 + x_4) mod 19 = (1 + 5) mod 19 = 6$$

Для генераторів, побудованих за методом Фібоначчі із запізненням, існують рекомендовані параметри *a* і *b*, для яких провели дослідження. Наприклад, пропонуються такі значення: (*a, b*) = (*55, 24*), (*17, 5*) або (*97, 33*). Якість одержуваних випадкових чисел залежить від значення константи *a*: чим воно більше, тим вище розмірність простору, в якому зберігається рівномірність випадкових векторів, утворених з отриманих випадкових чисел. У той самий час, із збільшенням значення константи *a*, збільшується обсяг використовуваної алгоритмом пам'яті.

Отже, значення (*a, b*) = (*17, 5*) рекомендуються для простих додатків. Значення (*a, b*) = (*55, 24*) дозволяють отримати числа, які задовольняють більшість криптографічних алгоритмів, вимогливих до якості випадкових чисел. Значення (*a, b*) = (*97, 33*) дозволяють отримати якісні випадкові числа й використовуються в алгоритмах, що працюють із випадковими векторами великої розмірності [5].

4.5. Генератори псевдовипадкових чисел на основі регістрів зсуву зі зворотним зв'язком (*LFSR*)

Регістри зсуву зі зворотним зв'язком (Linear Feedback Shift Registers - *LFSR*)) можуть застосовуватися для отримання потоку псевдовипадкових бітів і складаються з двох частин: в *n*-бітного регістра зсуву та пристрою зворотного зв'язку (рис. 4.3).

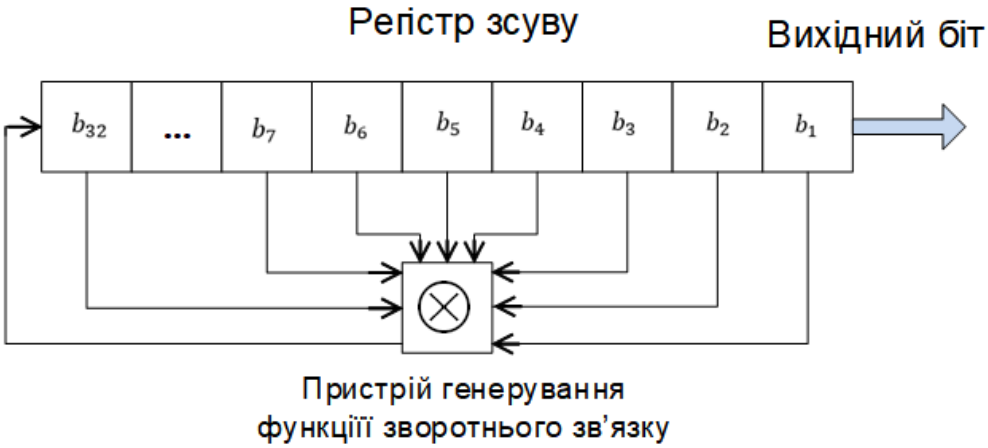


Рис. 4.3. Узагальнене представлення регістра зсуву із зворотним зв'язком

Отримати біти з регістра зсуву можна тільки по одному. Якщо необхідно отримати наступний біт, усі біти регістра зсуваються праворуч на один розряд. При цьому на вхід регістра ліворуч надходить новий біт, який формується пристроєм зворотного зв'язку (як правило застосовується функція XOR) й залежить від всіх інших бітів регістра зсуву. Завдяки цьому біти регістра змінюються за певним законом, який і визначає схему отримання ПВЧ. Очевидноо, що через деяку кількість тактів роботи регістра послідовність бітів почне повторюватися. Довжина одержуваної послідовності до початку її повторення називається *періодом регістра зсуву* максимальний період для *n* розрядного регістру дорівнює

$$2^n\text{-}1.$$

Приклад 4.7. Розглянемо регістр зсуву, представлений на рис.4.4.

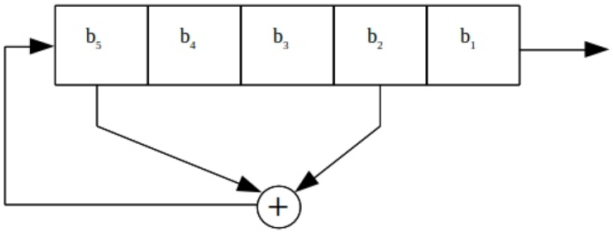


Рис. 4.4. Регістра зсуву із зворотним зв'язком

Даний регістр предтавляється наступним поліномом

$$x^5 + x^2 + 1,$$

з теоретично максимальним періодом

$$2^5 - 1 = 31.$$

Таким чином, при застосуванні подібних і інших генераторів можна формувати випадкові послідовності гами як для зашифрування, так і розшифрування даних.

4.6 Підготовка до завдання

Ознайомитися з шифрами гамування, методами їх побудови та реалізації. Розглянути різноманітні методи побудови генераторів псевдовипадкових послідовностей.

4.7 Практичне завдання

1. Записати своє прізвище, ім'я та по-батькові (ПІП) та представити у цифровій формі, згідно перетворенням, наведеним на рис.1.10. Наприклад, ШЕВЧЕНКО → «28 06 02 27 06 17 14 18».
2. Використати генератор ***BBS*** (або будь який інший) для формування гами. Задати параметри генератора.
3. Для відкритого повідомлення по п.1. зашифрувати дані із застосуванням ГПВЧ (п.2).
4. Провести зворотну процедуру по розшифруванню інформації. Порівняти отриманий результат з оригінальним повідомленням.
5. Надати аналіз отриманим результатам.

4.8. Зміст протоколу роботи

Протокол до виконаної лабораторної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки. Робота захищається індивідуально.

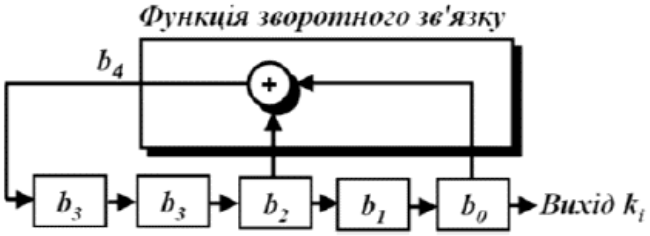
4.9 Контрольні питання для самоперевірки

1. У чому полягає суть шифрування гамуванням?
2. Наведіть класифікацію видів гамувальних послідовностей за різними ознаками. Які, у криптографічному розумінні, розбіжності між ними?
3. Які відмітні риси шифрування гамуванням? Порівняйте його із шифрами заміни, шифрами перестановки.
4. Поясніть, наскільки криптографічно стійким є шифрування гамуванням і від чого залежить його стійкість?
5. Які є способи формування послідовності гами, порівняйте їх за складністю і криптографічною стійкістю.
6. Дайте визначення потоковго шифру. Чим потоковий шифр відрізняється від блочного?
7. Яким чином організовується шифрування потоку даних змінної довжини?
8. Які числа називають псевдовипадковими?

9. Назвіть існуючі ГПВЧ. Які властивості повинен мати ГПВЧ для використання з криптографічною метою? Перерахуйте основні характеристики, переваги й недоліки кожного ГПВЧ.
10. Як для отримання псевдовипадкових чисел можуть використовуватися регістри зсуву зі зворотним зв'язком? Пояснити їх принцип роботи.
11. У чому різниця між генераторами випадкових і псевдовипадкових чисел?
12. Який закон розподілу характериний для послідовності істинно випадкових чисел?
13. Яку послідовність називать М-послідовністю?

4.10. Задачі до практикуму №4

1. Визначити послідовність із перших десяти чисел і період лінійного конгруентного ГПВЧ для різних параметрів *a*, *b* і *c* (k_0 прийняти таким, що дорівнює — *0*):
   а) *a = 3, b = 5, c = 11;* б) *a = 7, b = 13, c = 29.*
2. Для генератора ПВЧ на основі алгоритму BBS обчислити перші десять чисел, якщо *p = 23, q = 31, x = 3*. Визначити період генератора.
3. Яке число сформує на третьму кроці роботи (для значення $\boldsymbol{x_5}$) генератор Фібоначчі
   $$\boldsymbol{x_i = (x_{i-3} + x_{i-1}) mod 23}$$
   з початковими значеннями $\boldsymbol{x_0 = 5, x_1 = 3, x_2 = 7}$?
4. Визначити ключовий потік на *15* бітів, який буде згенерований представленим регістром зсуву зі зворотним зв'язком *LFSR*



5. Показати *LFSR* із характеристичним поліномом
   $$x^7 + x^5 + x^2 + 1$$
   .
Визначити максимальний період послідовності, що формується даними *LFSR*?
6. Для потокового алгоритму *RC4* показати перші *20* елементів ключового потоку, якщо ключ сеансу — *7* байтів зі значеннями *1*, *2*, *3*, *4*, *5*, *6* і *7*.

4.11 Завдання до самостійної роботи

1. Потокові шифри, їх призначення та різновиди.
2. Регістри зсуву зі зворотним зв'язком для побудови ГПВЧ.
3. Потоковий алгоритм *RC4*. Призначення, принцип формування всевдовипадкових послідовностей.
4. Поняття одноразового шифрувального блокноту.

5. Система шифрування Вермана. Принцип роботи, умова криптостійкості шифрування.
6. Самосинхронізовані потокові шифри. Приклади, особливості роботи.

ПРАКТИКУМ № 5
Складені шифри. Шифри ADFGX та ADFGVX

Мета роботи. Ознайомитися з теоретичними основами шифру ADFGX та ADFGVX, на практиці здійснити створення ключової послідовності, провести зашифрування відкритого і розшифрування шифрованого повідомлення. Усвідомити сильні і слабкі сторони шифрування.

5.1 Основні поняття про складені шифри

К.Шеннон увів поняття складених шифрів. Складений шифр — комплекс, який об'єднує підстановку, перестановку й інші компоненти. Складений шифр повинен дати можливість блоковим шифрам мати дві важливі властивості: *розсіювання* й *перемішування*.

Розсіювання повинне приховати відношення між зашифрованими й початковими даними, щоб не дати можливість зловмиснику використати статистику зашифрованих даних, щоб знайти початкові дані. Під розсіюванням розуміється, що кожний біт (символ) у зашифрованих даних залежить від одного або усіх бітів (символів) у початкових даних. Іншими словами, якщо один єдиний біт у початкових даних змінений, декілька або всі біти в зашифрованих даних будуть також змінені.

П*еремішування* — приховати залежність між зашифрованими даними та ключем, щоб не дати можливість зловмиснику використовувати зашифровані дані, щоб знайти ключ. Іншими словами, якщо один єдиний біт у ключі змінений, усі біти в зашифрованих даних будуть також змінені.

Розглянемо даний підхід на одному із сторично перших і відомих шифрів такого типу.

ADFGVX-шифр — один з найвідоміших шифрів, який використовувався військовими під час війни. Особливість шифру полягає в тому, що він побудований на поєднанні базових *операцій заміни та перестановки*. Частина шифру, що відповідає заміні, ґрунтується на квадрат Полібія.

До кінця Першої світової війни, в той час як більша частина країн світу використовувала або шифр заміни, або шифр перестановки, Німеччина почала використовувати нову систему шифрування ADFGX, яка об'єднала риси обох. Свою назву ця система отримала через те, що її шифрограми містили тільки літери «A», «D», «F», «G» і «X». Ці літери були обрані не випадковим чином. Якщо їх представити у вигляді крапок і тире коду Морзе, то вони будуть суттєво відрізнятися одна від одної. Таким чином, вибір цих літер мінімізує ризик появи помилок під час телеграфної передачі. Фактично це був квадрат Полібія, в який вписувався латинський алфавіт в певному порядку [25].

З метою ускладнення шифру, до шифру додали букву «V», тим самим збільшивши сітку шифрування до 36 символів. Це дозволило включити у відкритий текст цифри від 0 до 9 та літери I і J, які стали шифруватися порізному. Розширення шифру значно скоротило розмір повідомлень, що містять велику кількість цифр. Шифр став називатися ADFGVX.

5.2. Опис ADFGX шифру

Процес шифрування починається з малювання сітки розміру 5×5, кожна клітинка заповнюється 25 літерами латинського алфавіту (I та J шифруються однаково). Кожен рядок і стовпець сітки задається однією з 5 букв: «A», «D», «F», «G» і «X». Заповнення сітки здійснюється в ***довільному*** порядку (рис.5.1), тому одержувач повинен знати розташування кожного елемента, щоб зробити дешифрування.

Крок перший — заміна. Розглянемо процес шифрування на прикладі невеликого повідомлення: «**ATTACK AT DAWN**». На першому кроці кожен символ повідомлення замінюється на *пару літер*, що позначають рядок і стовпець відповідного символу в сітці. Наприклад, A буде замінено на FF, а B — на GA. Тоді зашифроване повідомлення буде мати вид, представлений на рис.5.2.

	A	D	F	G	X
A	F	N	H	E	Q
D	R	D	Z	O	C
F	I/J	S	A	G	U
G	B	V	K	P	W
X	X	M	Y	T	L

Рис.5.1. Приклад заповнення літерами квадрату ADFGX 5^x5

Повідомлення:	attack at dawn											
Відкритий текст:	a	t	t	a	c	k	a	t	d	a	w	n
Шифротекст на першому кроці:	FF	XG	XG	FF	DX	GF	FF	XG	DD	FF	GX	AD

Рис.5.2. Приклад шифрування повідомлення *«attack at dawn»*

Необхідно зауважити, що в цьому випадку застосування лише простої заміни було б не достатньо для побудови криптостійкого алгоритму. Застосування частотного аналізу було б достатньо, щоб розгадати повідомлення. Тому ускладнемо процедуру шифрування.

Крок другий — перестановка. На другому кроці застосовується перестановка, що значно ускладнює криптоаналіз. Перестановка здійснюється в залежності від ключового слова, яке повинно бути відомо одержувачу. Нехай, у нашому прикладі, таким словом буде «**BATTLE**». Процес перестановки полягає в наступному. Спочатку створюється нова сітка, у верхній частині якої записуються літери ключового слова. Потім, під цим

словом порядково записується, отриманий на першому кроці зашифрований текст (рис.5.3).

B	A	T	T	L	E
F	F	X	G	X	G
F	F	D	X	G	F
F	F	X	G	D	D
F	F	G	X	A	D

Рис.5.3. Приклад заповнення квадрату ключовим словом «BATTLE» та криптотекстом.

Далі, літери ключового слова переставляються в алфавітному порядку разом з відповідними стовпцями сітки (рис.5.4)

A	B	E	L	T	T
F	F	G	X	X	G
F	F	F	G	D	X
F	F	D	D	X	G
F	F	D	A	G	X

Рис.5.4. Приклад перестановки стовбців криптотексту згідно ключового слова «BATTLE».

Після чого літери кожного стовпця виписуються по черзі зверху вниз. Отримана послідовність літер утворює остаточний вигляд шифротексту.

Остаточний вигляд шифротекста:

«F F F F F F F F G F D D X G D A X D X G G X G X».

5.3. Опис ADFGVX шифру

Шифр ґрунтується на 6 літерах: «A», «D», «F», «G», «V» і «X». Аналогічно шифру ADFGX малюється таблиця розміру 6x6 і *випадковим чином* заповнюється 26 літерами і 10 цифрами. Розташування елементів в таблиці є частиною ключа (рис.5.5).

	A	D	F	G	V	X
A	1	J	R	4	H	D
D	E	2	A	V	9	M
F	8	P	I	N	K	Z
G	B	Y	U	F	6	T
V	5	G	X	S	3	O
X	W	L	Q	7	C	0

Рис.5.5. Приклад заповнення літерами квадрату ADFGVX 6x6.

Перший і другий крок дій аналогічний описаним вище. Створюється нова таблиця з ключовим словом у верхньому рядку. В якості ключа візьмемо слово «**SECRET**». Зазвичай використовуються більш довгі ключові слова або фрази (рис.5.6).

S	E	C	R	E	T
D	F	G	X	G	X
D	F	X	V	F	V
X	A	F	F	X	D
X	D	G	A	D	A
V	D	F	F	F	G
F	F	F	G	A	A
A	A	D	F	D	X

Рис.5.6. Приклад заповнення квадрату ключовим словом «SECRET» та криптотекстом.

За аналогією з шифром ADFGX, стовпці таблиці сортуються в алфавітному порядку (рис.5.7)

C	E	E	R	S	T
G	F	G	X	D	X
X	F	F	V	D	V
F	A	X	F	X	D
G	D	D	A	X	A
F	D	F	F	V	G
F	F	A	G	F	A
D	A	D	F	A	X

Рис.5.7. Приклад перестановки стовбців криптотексту згідно ключового слова «SECRET».

Після чого стовпці по черзі записуються в один рядок, утворюючи зашифрований текст.

Остаточний вигляд шифротексту:

«G X F G F F D F F A D D F A G F X D F A D X V F A F G F D D X X V F A X V D A G A X».

Для відновлення вихідного тексту необхідно виконати дії, зворотні шифруванню. Володіючи ключовим словом, послідовність стовпчиків можна привести до первісного порядку. Знаючи розташування символів у вихідній таблиці, можна розшифрувати текст.

5.4. Підготовка до завдання

Ознайомитися з шифрами ADFGX та ADFGVX, методами їх побудови та реалізації. Основні задачі розсіювання та перемішування при шифруванні.

5.5. Практичне завдання

1. Створити відкрите повідомлення у вигляді свого прізвища, ім’я та по-батькові (ПІП).
2. Для шифрування тексту використати шифр ADFGX або ADFGVX із самостійно обраним ключем (одномірний масив). Провести процедуру зашифрування тексту.
3. Провести зворотну процедуру по розшифруванню інформації. Порівняти отриманий результат з оригінальним повідомленням.
4. Написати програму, яка реалізує функцію зашифрування і розшифрування по п.1-3.
5. Надати аналіз отриманим результатам.

5.6. Зміст протоколу роботи

Протокол до виконаної лабораторної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

5.7. Контрольні питання для самоперевірки

1. У чому полягає суть шифрування ADFGX та ADFGVX? У чому їх схожість та відмінність?
2. Що розуміють під теоретико-інформаційною стійкістю шифра?
3. Що розуміють під гарантованою обчислювальною стійкістю шифра?
4. Який приховується зв’язок при перемішуванні інформації під час шифрування?
5. В чому полягає ефект розсіювання під час шифрування?
6. В чому полягає ефект перемішування під час шифрування?

5.8. Задачі до Практикуму №5

1. Таблиця для шифру ADFGVX з початковою ініціалізацією «125» представлена наступним чином:

\	A	D	F	G	V	X
A	1	2	5	A	B	C
D	D	E	F	G	H	I
F	J	K	L	M	N	O
G	P	Q	R	S	T	U
V	V	W	X	Y	Z	0
X	3	4	6	7	8	9

Зашифрувати повідомлення «Hello» з ключовим словом «ABCDEF». «DFVFDFDXFXFX»

2. Умова задачі шифрування аналогічна попередній. Зашифрувати повідомлення «Hello» з ключовим словом «BACDEF».

Порівняти та пояснити результати шифрування.

3. Таблиця шифрування шифру ADFGVX аналогічна першій задачі. Розшифрувати повідомлення «GDGAFVDDFDDFGAVFXG», якщо ключове слово «123456».

5.9 Завдання до самостійної роботи

1. Класифікація систем шифрування. Різновиди систем, їх призначення, та основні характеристики (швидкодія, криптостійкість т ін.).
2. Відмінності між сучасними і традиційним шифрами із симетричним ключем.
3. Поняття складених шифрів, їх класифікація та застосування.
4. Визначення та властивості шифрів Фейстеля. Принципи побудови та реалізації.
5. Визначення та властивості шифрів не Фейстеля. Принципи побудови та реалізації.

ПРАКТИКУМ № 6
Побудова сучасних блокових симетричних криптографічних систем

Мета роботи. Ознайомитися з теоретичними основами побудови сучасних симетричних криптографічних систем, провести зашифрування відкритого і розшифрування шифрованого повідомлення. Усвідомити сильні і слабкі сторони симетричного шифрування.

6.1. Основні поняття про симетричні криптографічні системи

Традиційні шифри із симетричним ключем орієнтуються на символи. Із появою і широким запровадженням комп'ютерних систем актуальними стали такі шифри, які орієнтовані на біти. Це пояснюється тим, що інформація, яку слід зашифрувати, не завжди складається тільки з тексту, але й містить числа, графіки, аудіо- та відеоданні тощо. Для зашифрування зручно перетворити ці типи даних у потік бітів і вже потім передавати зашифрований текст. Крім того, коли текст оброблено на розрядному рівні, кожний символ замінено на *8* (або *16*) бітів. Це означає, що кількість символів стає у *8* (або *16*) разів більше. Таким чином, обробка і змішування більшої кількості символів, які відповідають одному символу, збільшує безпеку. Одним із прийомів, який застосовують для обробки таких символів, це розміщення їх в блоки розміром - *n* (*n*-bit plaintext) з подальшим перетворенням.

Для реалізації сучасних блокових шифрів використовують симетричні системи, які передбачають застосування симетричних ключів. Симетричний ключ використовується як для зашифрування повідомлення (Encryption), так і для його розшифрування (Decryption). При використанні *n*-бітового блоку зашифрованих даних застосовують *k*-бітовий ключ (*k*-bit key).

Криптографічна система називається симетричною, оскільки той самий *k*-бітовий секретний ключ (key) має бути застосований як на стороні передачі повідомлення (*Plaintext*) при його шифруванні, так і на приймальній стороні при розшифруванні криптотекста (*Ciphertext*). Рис. 6.1 показує загальні процеси зашифрування й розшифрування сучасного блокового шифру.

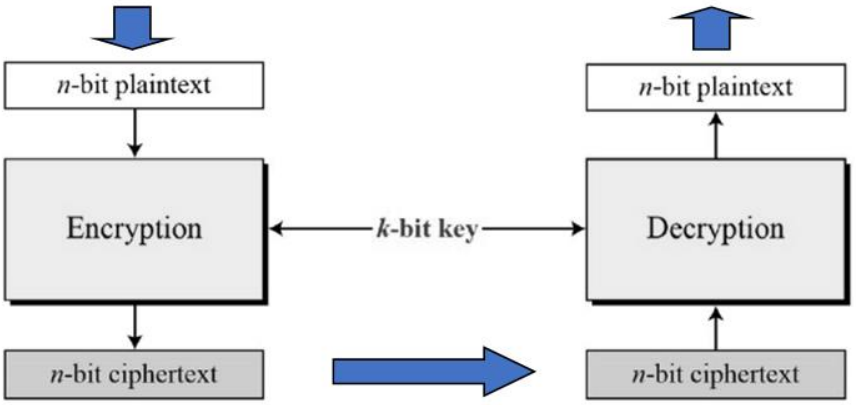


Рис.6.1. Структура сучасного блокового шифру

Якщо повідомлення має розмір менше ніж n бітів, слід додати заповнення, щоб створити цей *n*-розрядний блок; якщо повідомлення має більше ніж *n* бітів, його слід розділити на *n*-розрядні блоки, і у разі потреби додати до останнього блока відповідне заповнення. Загальні значення для *n* зазвичай 64, 128, 256 або 512 бітів.

Приклад 6.1. Визначити кількість додаткових бітів, які слід додати до повідомлення, що складається зі *50* символів, якщо для кодування використовується *ASCII* коди по *8* бітів і блоковий шифр приймає блоки довжиною *64* біти.

Розв'язання. Для кодування *50* символів, використовуючи *ASCII* коди по *8* бітів, довжина повідомлення буде містити 50*8 = 4*00* бітів. Умовою побудови заповнених блоків є ділення початкових даних без залишку на розмір блоку – *64* біти. Якщо |*Message*| і |*Padder*| — довжина повідомлення й довжина заповнення відповідно, то:

$$(|Message| + |Padder|) mod 64 = 0.$$

Тоді

$$|Padder| = -|Message| mod 64 = -400 mod 64 = 48.$$

Це означає, що до повідомлення слід додати 48 біт додатково заповнення (наприклад нулів за домовленістю). Тоді початкові дані будуть складатися з 448 бітів або сіми 64-розрядних блоків (7*64=448 біт). Зазначимо, що тільки останній блок містить заповнення. Шифратор використовує алгоритм зашифрування сім разів, щоб створити сім блоків зашифрованих даних.

Схема роботи блокового шифру описується функціями:

$$C = \mathbf{EnCrypt}(M, Key),$$
$$M = \mathbf{DeCrypt}(C, Key),$$

де ключ ***Key*** є параметром блокового криптоалгоритму, ***M*** – повідомлення, ***C*** – зашифрований криптотекст.

Криптоалгоритм називається ***ідеально стійким***, якщо прочитати зашифрований блок даних можна тільки перебравши всі можливі ключі, доти, поки повідомлення не виявиться осмисленим.

Функція стійкого блокового шифру

$$C = EnCrypt(M, Key)$$

повинна задовольняти таким умовам:

- функція **EnCrypt** повинна бути оборотною;
- не повинно існувати інших методів прочитання повідомлення ***M*** по відомому блоку ***C***, крім як повним **перебором ключів** Key (методом «грубої сили»);
- не повинно існувати інших методів визначення, яким ключем **Key** було зроблене перетворення відомого повідомлення ***M*** у повідомлення ***C***, окрім, як повним **перебором ключів**.

За дослідженнями К. Шеннона, **абсолютно стійкий криптоалгоритм** повинен відповідати наступним правилам:

1. довжина повідомлення повинна бути не більша вибраного випадковим чином ключа;
2. ключ при шифруванні повинен бути використаний лише один раз.

6.2. Шифри підстановки та перестановки

Як відомо, для шифрування активно використовують шифри підстановки (заміни) та перестановки (транспозиції). Зокрема, блоковий шифр може бути спроектований таким чином, щоб мати властивості шифру підстановки (заміни) або перестановки. Такі операції були вже розглянуті в попередніх практикумах, де особливістю є те, що замість операцій над символами оперують бітами.

При проектуванні шифру, як шифру підстановки, значення бітів «*1*» або «*0*» у вхідних даних можуть бути замінені на «*0*» або «*1*». З цього слідує, що вхідні дані та криптотекст можуть мати різну кількість одиниць та нулів.

Наприклад, блок вхідних даних обсягом *64* біти, який має *12* «0» та *52* «*1*», може бути представлений криптотекстом, наприклад, з *34* «0» і *30* «*1*».

При проектуванні шифру, як шифру перестановки (або транспозиції), біти змінюють тільки свій порядок (або переміщуються). При цьому зберігається та сама кількість символів у вхідних даних та криптотексті.

Оскільки один біт в блоці може приймати два можливих значення – «0» або «1», тоді кількість можливих *n*-бітових початкових або зашифрованих даних дорівнює 2^n.

Сучасні блокові шифри спроектовані як шифри підстановки, тому що властивості транспозиції (збереження кількості одиниць або нулів) роблять шифр уразливим до атак вичерпного пошуку [5].

6.3. Блокові ключові шифри

При формуванні блокових шифрів постає питання довжини ключа, який він повинен мати розмір. Як відомо, криптостійкість криптоалгоритму має залежати тільки від КЛЮЧА. Чим більший розмір ключа, тим більш криптостійким є криптоалгоритм. Але в цьому випадку він буде мати більший час для операцій шифрування/дешифрування. Тому потрібно мати баланс між швидкісью обробки та криптостійкістю.

Нехай ключ у блоковому шифрі є достатньо довгим, щоб створити відображення будь-яких можливих початкових даних у блоці у зашифровані дані. Такі блокові шифри будемо називати *повнорозмірними ключовими шифрами* [5]. На практиці реальні ключі є меншими. Великий розмір ключа можна використовують тільки для певних відображень вхідної інформації у вихідну.

Повнорозмірний ключовий блоковий шифр перестановки (транспозиції) переміщує біти між собою. В цьому випадку він може бути представлений, як шифр перестановки *n*-мірного об'єкта - $n!$ можливих варіантів перестановок. Ключ буде визначати, яка комбінація обміну відбулася і кількість таких

можливих ключів має бути так само *n!* Тоді ключ повинен мати довжину *[* $\log_2 n!$ *]* бітів.

Приклад 6.2. Побудуємо модель можливих комбінацій у шифрі перестановок блокового шифру, де розмір блока *n = 3* біти.

Розв'язання. Можлива кількість комбінацій перестановок визначена, як *n! = 3! = 6* – [(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)] (рис. 6.2).

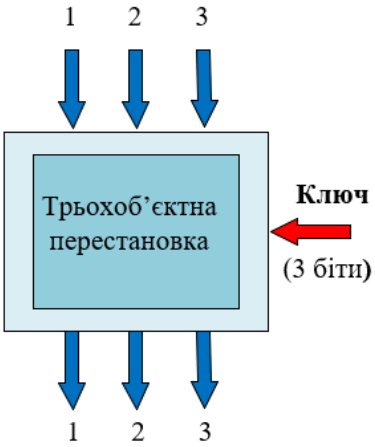


Рис. 6.2. Можливі варіанти перестановок трьохбітового блокового шифру при *n = 3.*

В цьому випадку ключ буде мати *[* $\log_2 n!$ *] = 3* біти. Тоді з можливих 8 різних комбінацій ($2^3 = 8$) нам знадобляться лише *6* із них.

Чи можливо шифр підстановки представити як шифр перестановки? Для цього можна застосувати добре відому модель представлення «one-hot encoding», де відбувається перетворення *n*-розрядного цілого числа в рядок із 2^n бітів, в якому присутня лише одна «1» а решта «0». Позиція такої одиниці буде вказувати на значення цілого числа у впорядкованій послідовності рядка. В цьому випадку шифр може бути змодельований, як перестановка $2^n!$ об'єктів.

Приклад 6.3. Побудувати модель шифру **перестановки** для блокового шифру **підстановки** для розміра блоку *n = 3* біти.

Розв'язання. Позначимо три вхідні блоки даних цілими числами від 0 до 7 та закодуємо кожен з них рядком на 8 бітів з однією «1». Наприклад, дані «*000*» можуть бути закодовані, як «*00000001*»; дані «*011*» можуть бути закодовані, як *00001000* і т.д. Тоді можлива кількість перестановок для 8 елементів буде 8! = 40320 і буде потребувати ключа розмірністю $[\log_2 8!]=16$ біт. Ключ розмірністю *16* біт може визначати *65536* різноманітних відображень (комбінацій), але ми використовуємо лише *40320*. Наприклад, ключ розмірністю в 15 біт дав би можливість робити тільки 32768 відображень (комбінацій), що було б недостатньо.

На рис. 6.3 представлена можлива комбінація перестановок.

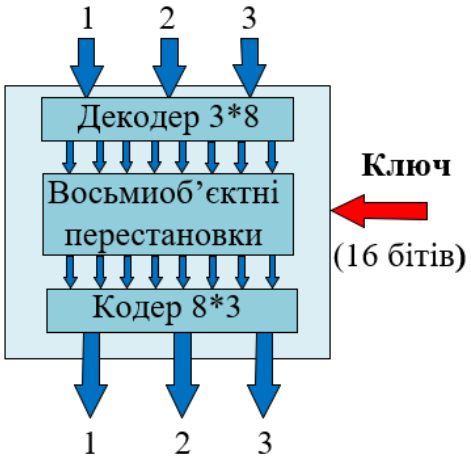


Рис. 6.3. Представлення трибітового блокового шифру підстановки у вигляді шифру перестановки

Повнорозмірна ключова транспозиція або шифр підстановки/перестановки показує, що якщо зашифрування (чи розшифрування) використовує більше ніж одну будь-яку комбінацію із цих шифрів, результат еквівалентний операції групової перестановки. Відомо, що дві або більше каскадні перестановки можуть завжди бути замінені однією перестановкою. Це означає, що не потрібно мати більше ніж один каскад повнорозмірних ключових шифрів тому, що ефект той самий, як і за наявності одного кроку [5].

На практиці шифри не можуть використовувати повнорозмірні ключі, тому що розмір ключа в цьому випадку стає занадто великим, особливо для блокового шифру підстановки. Наприклад, добре відомий блоковий шифр *DES* використовує блоки завдовжки в *64*-біта. При використанні повнорозмірнго ключа його розмір був би $log_2(2^{64}!)$ бітів. На практиці ключ для *DES* – тільки *56* бітів, що є надто маленьким фрагментом повнорозмірного ключа. Це означає, що *DES* використовує тільки *256* відображень із всіх можливих.

6.4. Безключові шифри (з фіксованим ключем)

Безключові шифри не представляють інтересу окрім випадків, коли вони є окремою частиною або компонентами ключових шифрів. Наприклад, безключові шифри транспозиції (чи із фіксованим ключем) можна розглядати як шифр транспозиції, реалізований в апаратних засобах. При єдиному правилі перестановки (фіксоване правило) ключ може бути представлений спеціальною таблицею при реалізації шифру. Далі будуть розглянуті елементи блокового шифрування, які отримали назву *P-блоки перестановки*, які використовуються як стандартні блоки сучасних блокових шифрів, *S*-блоки заміни.

Безключові шифри розуміють як заздалегідь визначене алгоритмом відображення вхідної інформації у вихідну. Таке відображення може бути представлене математичною функцією, підготовленою таблицею або в інший спосіб.

6.5. Компоненти сучасного блокового шифру

Як правило, блокові шифри є ключовими шифрами підстановки, де ключ реалізує тільки часткові відображення можливих входів інформації в можливі виходи. Щоб забезпечити необхідні властивості сучасного блокового шифру, такі як перемішування та розсіювання інформації, цей шифр формується як комбінація модулів транспозиції (***P*-блоків перестановки** - *P-blocks Permutation*), модулів підстановки (***S*-блоків заміни –** *Substitution Box*), ***циклічних зсувів*** *(Shift left/right)* та інших модулів.

***P*-блок перестановки**, подібний до традиційного шифру транспозиції символів, переміщає біти. Сучасні блокові шифри мають три типи *P*-блоків перестановки (рис.6.4.):

- *прямі P-блоки (Straight P-box – розмірність «*5 *x* 5*») - ;*
- *P-блоки стискання (Compression P-box – розмірність «*5 *x* 3*»);*
- *P-блоки розширення (Expansion - розмірність «*3 *x* 5*»).*

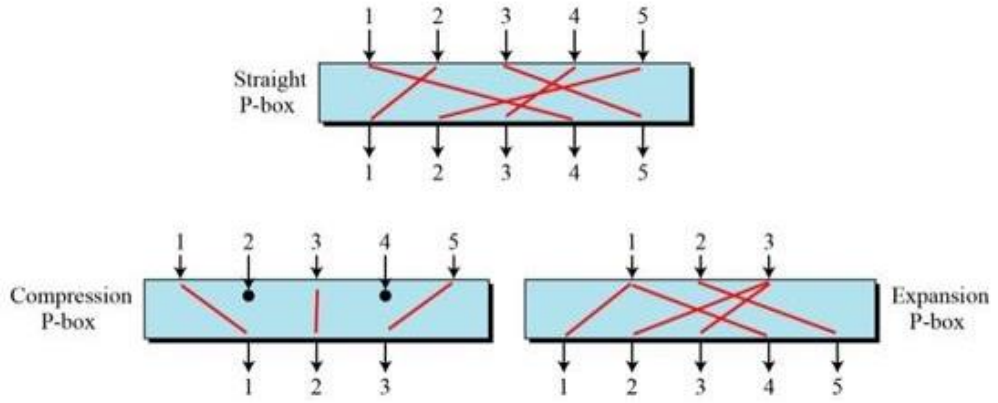


Рис.6.4. Представлення різних типів *P*-блоків перестановок

Прямий P-блок (straight) із *n* входами і *n* виходами — це перестановка з *n!* можливих відображень.

Приклад 6.4. Продемонструємо всі можливі набори перестановок для **прямого P-блока** розмірністю «3x3» (рис.6.5)

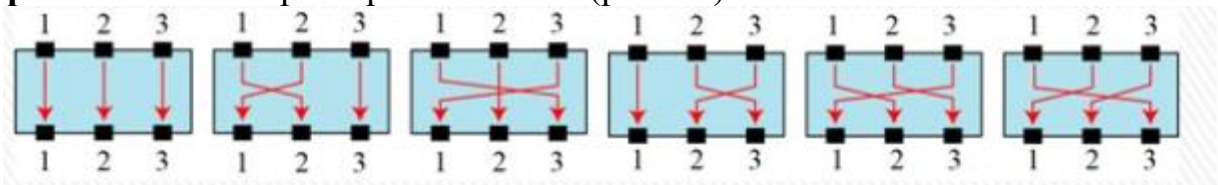


Рис. 6.5. Відображення прямого *P*-блока *3 x 3.*

Для визначення одного з *n!* можливих відображень в прямих *P*-блоках використовують ключ. Якщо ключ не застосовується, тоді відображення для *P*-блока задане заздалегідь. В цьому випадку таке відображення можна

реалізувати апаратно або програмно, використовуючи для цього таблиці перестановок. Табл. 6.1 наведений приклад таблиці перестановок для *n = 32*.

Таблиця 6.1

Таблиця перестановки для прямого *P*-блока

	Номери бітів															
Вхід	*26*	*18*	*10*	*02*	*28*	*20*	*12*	*04*	*30*	*22*	*14*	*06*	*32*	*24*	*16*	*08*
Вихід	*01*	*02*	*03*	*04*	*05*	*06*	*07*	*08*	*09*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
Вхід	*25*	*17*	*09*	*01*	*27*	*19*	*11*	*03*	*29*	*21*	*13*	*05*	*31*	*23*	*15*	*07*
Вихід	*17*	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*

В таблиці показана відповідність між *32* входами і виходами. Наприклад, вхід з номером «26» буде підключений до виходу з номером «01» і т.д. Іноді зручно зменшувати обсяги поданої інформації і не помічати виходи. Виходи будуть автоматично призначені як номери комірок (індекси), в яких стоять значення відповідних входів (табл. 6.2). Наприклад, вхід з номером «58» буде відповідати виходу під номером «1», вхід з номером «50» буде відповідати виходу під номером «2», і т.д.

Таблиця 6.2

Приклад таблиці перестановки для прямого *P*-блока при для *n = 64*

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

Зауважимо, що *прямі P-блоки є оберненими*. Таким чином, прямий *P*-бітовий блок можна використовувати як для зашифрування, так і для розшифрування. Таблиці перестановки також повинні бути оберненими у відношенні один до одного.

P-блок стискання (compression) — це *P*-блок із *n* входами і *m* виходами, де *m* < *n*. Деякі інформаційні входи блоковані й не пов'язані з виходом (див. рис. 6.4). Задавати такі *P*-блоки стискання також зручно у вигляді таблиць перестановок, які визначають правила перестановки бітів. Зауважимо, що не всі входи будуть відображені на виході, деякі з входів можуть бути відсутні (заблоковані). Табл. 6.3 показує приклад перестановки для *P*-блока стискання розмірністю «*32*x*24*», де такі входи, як *7*, *8*, *9*, *16*, *23*, *24* і *25* відсутні на виході (заблоковані).

Таблиця 6.3

Приклад таблиці перестановки для *P*-блока стискання «*32*x*24*»

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

P-блоки стискання використовують тоді, коли потрібно не тільки переставити біти, але й зменшити кількість бітів для наступного рівня обробки даних. Очевидно, що обернення таких *P*-блоків стискання є неможливим. Оскільки ключ для *P*-блоків відсутній, відновити відкинуті (заблоковані біти) буде неможливо.

P-блок розширення (Expansion) — *P*-блок із n входами і m виходами, де m > n. Деякі із входів пов'язані більше ніж з одним виходом (див. рис. 6.4). P-блоки розширення, які використовуються в сучасних блокових шифрах, зазвичай безключові і правила перестановки бітів вказуються в таблицях (табл.6.4). Оскільки виходів більше, ніж входів, деякі з входів зєднані з більш ніж одни виходом (входи 1, 3, 9, 12) зєднані з двома виходами.

Таблиця 6.4

Приклад таблиці перестановки для P-блока розширення «12x16»

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12

P-блоки розширення використовуються, коли потрібно переставити біти та водночас збільшити кількість бітів для наступного каскаду шифрування.

S-блоки підстановки. *S-блок підстановки* (заміни) реалізує локальний шифр підстановки над певною групою даних. *S-блок* може мати різну кількість *n* входів (***X***) і *m* виходів (***Y***). В загальному виді *S*-блоки підстановки можуть бути ключовим або безключовими. Сучасні шифри, як правило, використовують *S*-блоки підстановки як безключові схеми, де відповідне відображення (перетворення) вхідних даних у вихідні заздалегідь визначене. Наприклад, таке залежність вихідних даних від вхідних може бути представлене в аналітичному виді, як система функціональних перетворень:

$$y_1 = f_1(x_1, x_2, \ldots, x_n);$$
$$y_2 = f_2(x_1, x_2, \ldots, x_n);$$
$$\ldots$$
$$y_m = f_m(x_1, x_2, \ldots, x_n).$$

де $x_1, x_2, \ldots, x_n$ - вхідні дані, $y_1, y_2, \ldots, y_m$, - вихідні дані.

Функціональну залежність між ***Y*** та ***X*** можна також представляти у вигляді таблиці.

Приклад 6.5. Нехай аналітична залежність в *S*-блоці з трьома інформаційними входами ($n = 3$) та двома інформаційними виходами ($m = 2$) представлена у вигляді:

$$y_1 = x_1 \otimes x_2 \otimes x_3;\ y_2 = x_1.$$

Тоді таку залежність можна представити як:

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 111 \\ 100 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}.$$

Приклад 6.6. Нехай аналітична залежність в *S*-блоці з трьома інформаційними входами ($n = 3$) та двома інформаційними виходами ($m = 2$) представлена у вигляді

$$y_1 = x_2^2 + x_1,$$

$$y_2 = x_3^2 + x_1 x_2 + x_2^2,$$

де множення та складання проводиться в полі Галуа *GF(2)*. Зауважимо, що представлений *S*-блок є нелінійним, так як немає лінійних співвідношень між входами й виходами.

Циклічний зсув *(Shift left/right)*. Побітовий зсув може бути вліво або вправо. Кругова операція лівого зсуву зсуває кожний біт у *n*-бітовому слові на *k* позицій вліво; крайні ліві *k*-біти видаляються ліворуч і стають крайніми правими бітами. Кругова операція правого зсуву зсуває кожний біт у *n*-бітовому слові на *k* позицій вправоруч; крайні праві *k*-біти справа видаляються та стають крайніми лівими бітами (рис.6.6).

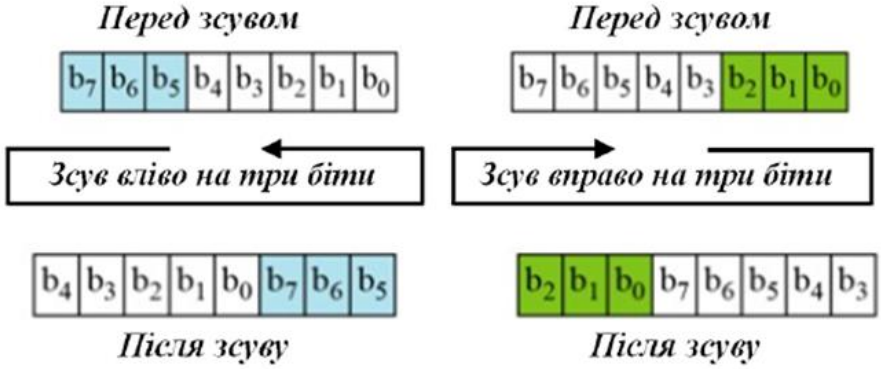


Рис. 6.6. Приклад циклічного зсув *8*-розрядного слова вліво та вправо

Циклічна *операція зсуву* зміщує біти в слові й допомагає змінити вхідне слово. Кількість позицій, на які можуть бути зсунуті біти, може бути визначена значенням ключ. Проте, як правило, циклічні операції зсуву є безключові і значення зсуву *k* встановлюється заздалегідь.

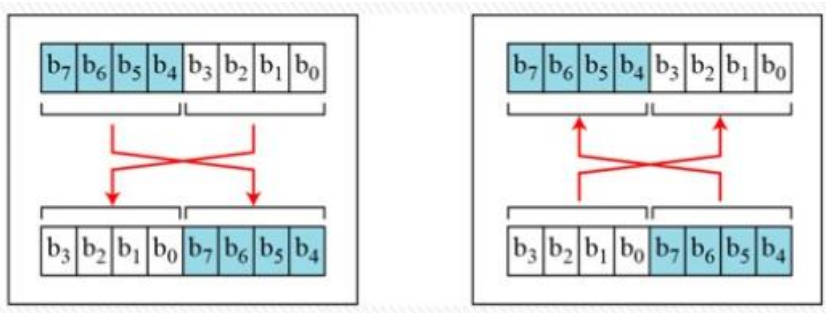
Циклічна операція лівого зсуву — інверсія операції правого зсуву, тобто є оберненою. Якщо одна з них використовується для зашифрування, то інша може застосовуватися для розшифрування.

Операція циклічного зсуву має дві властивості:

1) зміщення за модулем *n:* якщо $k = 0$ або $k = n$, тоді зсув не відбувається. Якщо k є більшим за n, тоді вхідна інформація в слові зсувається на $k \bmod n$ позицій;

2) операції циклічного зсуву над з'єднанням операцій є групою. Це означає, що якщо зміщення робиться неодноразово, то однакове значення слова може з'явитися кілька разів.

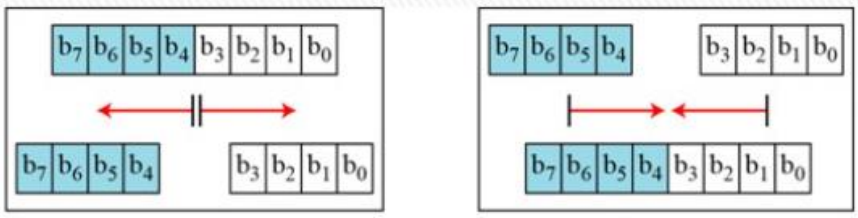
Заміна *(Swap Operation). Операція заміни* — спеціальний випадок операції циклічного зсуву, коли *k = n/2*. Це можливо реалізувати, коли *n* — парний номер. Оскільки зсув вліво *n/2* — те саме, що і зсув *n/2* вправо, то операція є оберненою. Операція заміни для зашифрування може бути повністю розкрита операцією заміни під час розшифрування. Рис. 6.7 ілюструє операцію заміни для слова з *n = 8* бітів.



а) б)

Рис. 6.7. Операція заміни у *8*-бітовому слові: а) зашифрування; б) розшифрування

Розбиття й об'єднання *(Split and Combine).* Розбиття зазвичай розділяє *n*-бітове слово по середині, створюючи два слова рівної довжини. Об'єднання зв'язує два слова рівної довжини, щоб створити *n*-бітове слово. Ці дві опе-рації інверсні одна одній і можуть використовуватися як пара, щоб урівноважити одна одну. Якщо одна використовується для зашиф-рування, то інша — для розшифрування. Рис. 6.8 показує ці операції для випадку *n = 8*.



а) б)

Рис.6.8. Операції розбиття (а) та об'єднання (б) для *8*-бітового слова

Клод Елвуд Шеннон (англ. Claude Elwood Shannon) вперше почав вивчати криптографію, використовуючи системний підхід. Він увів поняття складених шифрів - комплекс, який об'єднує підстановку, перестановку й інші компоненти, розглянуті вище.

Розсіювання й перемішування *(difusion and confusion).* Ідея К.Шеннона щодо складеного шифру повинна була дати можливість блоковим шифрам мати дві важливі властивості: *розсіювання* й *перемішування*.

Розсіювання повинне приховати відношення між зашифрованими й початковими даними. Це дезорієнтує зловмисника, який використовує статистику зашифрованих даних, щоб знайти початкові дані. Під розсіюванням розуміється, що кожний біт (символ) у зашифрованих даних залежить від одного або усіх бітів (символів) у початкових даних. Іншими словами, якщо один єдиний біт у початкових даних змінений, декілька або всі біти в зашифрованих даних будуть також змінені.

Ідея відносно *перемішування* — приховати залежність між зашифрованими даними та ключем. Це дезорієнтує зловмисника, який прагне використовувати зашифровані дані, щоб знайти ключ. Іншими словами, якщо один єдиний біт у ключі змінений, усі біти в зашифрованих даних будуть також змінені.

Раунди *(rounds).* Ефекти розсіювання та перемішування можна отримати при використанні складених шифрів, де кожна ітерація — комбінація *S-блоків, P-блоків* та інших компонентів. Кожна ітерація називається *раундом,* кількість яких задається криптоалгоритмом. Блоковий шифр використовує ключовий список, або генератор ключів (*round-key generator*), який створює різні ключі (K_1, K_2, …) для кожного раунду від основного ключа шифру (K). У N-раундовому шифрі, щоб створити зашифровані дані, початкові дані зашифровуються N разів. При цьому, для розшифрування так само використовують N ітерацій. Дані, створені на проміжних рівнях (між двома раундами), називаються *середніми даними (middle text).* Рис. 6.9. показує простий складений шифр із двома раундами. На практиці складені шифри мають більше ніж два раунди. На рис. 6.9 у кожному раунді проводяться три перетворення [7].

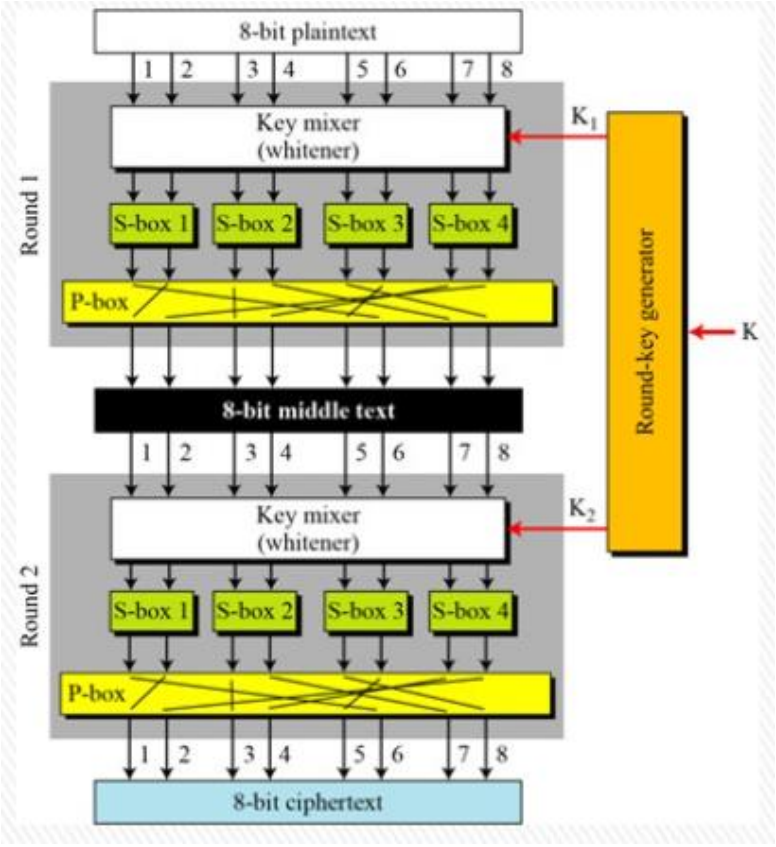


Рис.6.9. Складений шифр, що складається з двох раундів

Перше перетворення: *8*-бітові дані змішуються з ключем (*Key mixer*), щоб зробити біти даних рівноймовірними (приховати біти, використовуючи ключ) —"*вибілити*" дані (*whitener*). Це зазвичай робиться за допомогою операції *виключного або* (*xor*) слова на *8* бітів із ключем на *8* бітів.

Друге перетворення. Дані з виходів "*вибілювача*" (*whitener*) розбиваються на чотири групи по *2* біти і подаються в чотири *S*-блоки заміни. Значення бітів змінюються відповідно до функціонування *S*-блоків заміни у цьому перетворенні.

Третє перетворення. Із виходів *S*-блоків заміни дані надходять у *P*-блок перестановки, де біти переставляються в такий спосіб, щоб у наступному раунді результат із виходу кожного блока надійшов на різні входи.

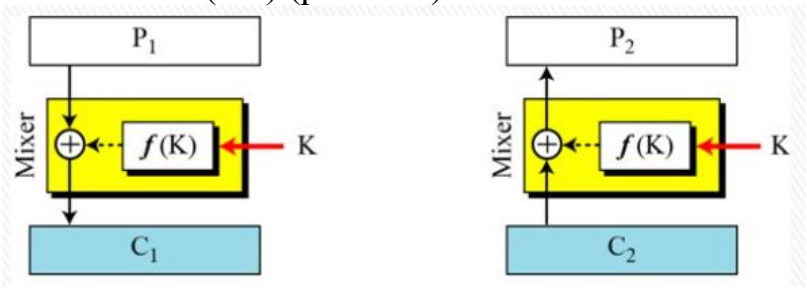
На рис.6.9. наведений приклад реалізації складеного шифру при застосування двох раундів, які містять комбінації *S-блоків заміни і P-блоків перестановки*, що може гарантувати розсіювання.

6.6. Класи складених шифрів

Сучасні блокові шифри є складеними, оскільки містять в собі різноманітні операції. Разом з тим, такі шифри розділені на два класи. Шифри першого класу використовують і обернені, і необернені компоненти. Ці шифри називають *шифрами Фейстеля*. Шифри другого класу застосовують тільки обернені компоненти і отримали назву - *шифри не-Фейстеля* (через відсутність іншої назви) [5].

Фейстель розробив інтелектуальний і цікавий шифр, який використовувався впродовж багатьох десятиліть. Шифр Фейстеля має три типи компонентів: *самообернений, обернений і необернений*.

Шифр Фейстеля містить у блоках усі необернені елементи та використовує один модуль в алгоритмах зашифрування й розшифрування. Питання в тому, як алгоритми зашифрування й розшифрування дозволяють інвертувати відкриті й закриті дані один до одного, якщо кожен містить необернений модуль. Фейстель показав, що вони можуть бути збалансовані. Показано, що ефектів необерненості можна позбутися, якщо використовувати операцію виключного або (xor) (рис.6.10).



а) б)

Рис.6.10. Ілюстрація застосування необерненого компонента для зашифрування (а) і розшифрування (б) в шифрі Фейстеля

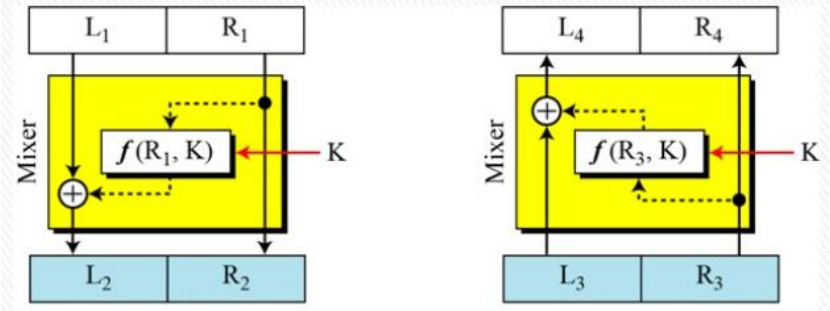
Приклад 6.7. Покажемо, що при використанні операції виключного або (xor) можна досягти ефекту оберненості на прикладі початкових даних M=0111, ключ K=101.

Зашифрування: C = M f(K) = 0111 1001 = 1110.

Розшифрування: M = C f(K) = 1110 1001 = 0111.

Продемонструємо, як можна вдосконалити даний шифр і отримати шифр Фейстеля.

Шифри Фейстеля. Візьмемо функцію f(K), щоб застосувати її для зашифрування частини початкових даних і розшифрування частини зашифрованих даних. Ключ може використовуватися як другий вхід до функції f(K). Завдяки цьому способу функція f(K) стає складним елементом із деякими неключовими й ключовими елементами. Щоб досягти мети, розділимо початкові й зашифровані дані на два блоки *рівної довжини* — ліві (L) і праві (R). Правий блок вводиться у функцію f(K) і набуває вигляду f(R1,K), а лівий блок складається за допомогою операції виключного або (xor) з виходом функції f(R1,K). Входи до функції f(R1,K) повинні точно співпадати як під час зашифрування, так і під час розшифрування (рис.6.11).



а) б)

Рис.6.11. Ілюстрація роботи схеми Фейстеля для режиму зашифрування (а) та розшифрування (б)

Схема шифру на рис.6.11 характеризується тим, що права половина початкових даних ніколи не змінюється. В цьому випадку цим може скористатися зловмисник і одразу знайти половину початкових даних. Таким чином, можна усунути даний недолік при реалізації додаткової заміни і збільшивши кількість раундів. На рис. 6.12 проілюстрований новий варіант шифру Фейстеля з двома раундами.

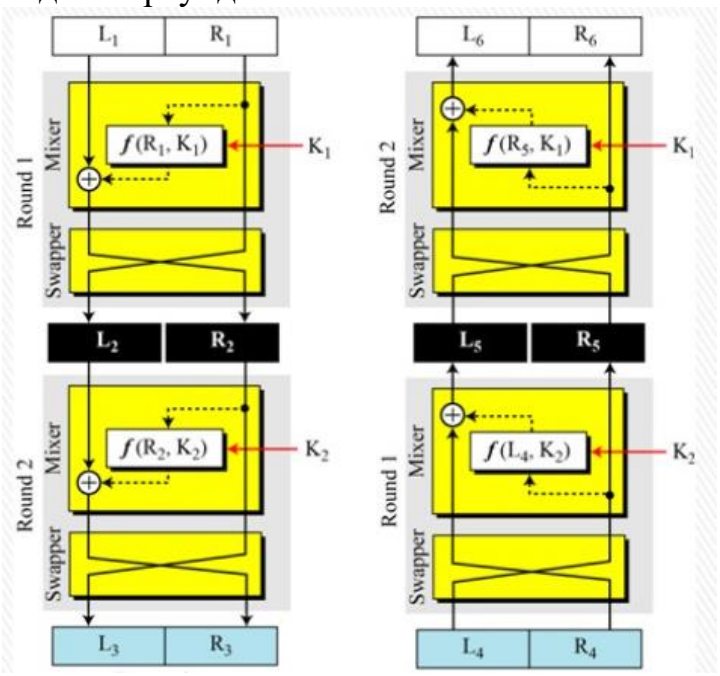


Рис.6.12. Ілюстрація роботи удосконаленого шифру Фейстеля з двома раундами

6.7. Підготовка до завдання

Ознайомитися з теоретичними положеннями побудови блокових складених шифрів, компонентами сучасних блокових шифрів, схемами реалізації шифрів Фейстеля.

6.8. Практичне завдання

1. Створити відкрите повідомлення (M) у вигляді свого прізвища.
2. Представити кожний символ повідомлення M (m_i) у вигляді ASCII (*American Standard Code for Information Interchange*) 8-бітного (бінарного) коду.
3. Створити блок даних на основі ASCII кодування. Наприклад, прізвище Шевченко буде мати наступний вид:

Ш	**е**	**в**	**ч**	**е**	**н**	**к**	**о**	**←-ПІП**
152	*165*	*162*	*231*	*165*	*173*	*170*	*174*	←Десяткове представлення
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	

0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	← Бінарне представлення
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
1	**0**	**0**	**0**	**0**	**1**	**1**	**1**	
0	**1**	**0**	**1**	**1**	**1**	**0**	**1**	
0	**0**	**1**	**1**	**0**	**0**	**1**	**1**	
0	**1**	**0**	**1**	**1**	**1**	**0**	**0**	

Нехай бінарне представлення кожної літери буде блоком розмірністю n=8 біт і буде підлягати обробці.

4. Кожна літера в даному випадку представляється як 1 байт інформації (8 біт - $b_7 \ldots b_0$). На основі такого представлення застосувати компоненти сучасного блокового шифру для кожного байту інформації, а саме:
   - реалізувати 1 раунд перетворень, представлених на рис. 6.9;
   - в якості ключа обрати 8-бітове слово (кожен студент обирає свій секретний ключ, а алгоритм обробки для всіх блоків залишається однаковий). Ключ взаємодіє з кожним вхідним байтом окремої літери (8-бітове представлення кожної літери) за правилом XOR;
   - чотири S-блоки мають функціонувати наступним чином

$S1\colon\ y_1 = x_1 + x_2 \quad y_2 = x_1 * x_2, \qquad S2\colon\ y_3 = x_4 \quad y_4 = x_3,$

$S3\colon\ y_5 = x_5 * x_6 \quad y_6 = x_5, \qquad S4\colon\ y_7 = x_7 + x_8 \quad y_8 = x_8.$

   - *прямий P-блок перестановки* (8*8) має реалізувати операцію зсуву вліво на 3 позиції.
5. Реалізувати даний алгоритм вручну. Результатом обробки має бути зашифроване 8-бітне представлення кожної окремої літери. Записати результат у двійковій і десятковій формах. Порівняти із вхідним початковим кодом.
6. Реалізувати зворотню процедуру по декодуванню. При коректному виконанні операцій має бути початкове вхідне повідомлення - Ваше прізвище.
7. Написати програму, яка реалізує функцію шифрування і розшифрування по п.1-4. Записати отриманий результат і перевірити з п.5 і п.6.
8. Змінити ключ і подивитися на результат шифрування.
9. Порівняти отримай результат шифрування і розшифрування з результом роботи програми Вашого колеги за умови вводу однакового прізвища і ключа (за умови різних ключів).
10. Надати аналіз отриманим результатам.

6.9 Зміст протоколу роботи

Протокол до виконаної практичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

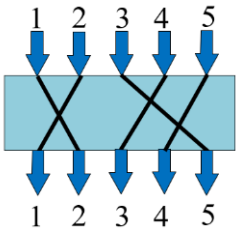
Робота захищається індивідуально.

6.10. Контрольні питання для самоперевірки

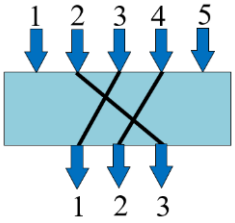
1. В чому різниця між потоковими і блоковими шифрами?
2. Чому сучасні блокові шифри спроектовані як шифри підстановки а не шифри транспозиції?
3. Чим визначається криптостійкість алгоритму? Чому?
4. Які необхідні реалізувати умови для побудови абсолютно криптостійкого алгоритму?
5. Перерахуйте компоненти сучасного блокового шифру.
6. Дайте визначення Р-блока, його різновиди і призначення. Які Р-блоки є оберненими?
7. Дайте визначення S-блока і його призначення. Які існують способи задання S-блоків і яка умова оберненості?
8. Дайте визначення складового шифру. Які існують два класи складових шифрів?
9. Дайте визначення розсіюванню і перемішуванню даних. Чим вони відрізнюються? Яким чином можна реалізувати ці функції?
10. Які шифри називають шифрами Фейстеля? Які властивості шифрів Фейстеля?

6.11. Задачі до практикуму №6

1. Повідомлення має 1000 символів, для подання яких використовуються ASCII коди. Це повідомлення буде зашифроване блоковим шифром розмірністю n=128 біт. Знайти розмір доповнення й кількість зашифрованих блоків.
2. Блок транспозиції має 5 входи та 5 виходи. Порахувати порядок групи перестановки та розмір ключової послідовності в бітах?
3. Блок підстановки має 5 входи та 5 виходи. Порахувати порядок групи перестановки та розмір ключової послідовності в бітах?
4. Для вхідних даних $(10011010)_2$ показати результат циклічного лівого зсуву на 3 біти.
5. Для вхідних даних $(11011101)_2$, показати результат циклічного правого зсуву на 2 біти.
6. Порахувати результат операцій:
   а) $(01001001)_2 \oplus (01101101)_2$; б) $(01011101)_2 \oplus (10100010)_2$;
7. Здійснити перестановку бітів для прямого Р-блока, якщо на його вході діє послідовність $(11010)_2$:



8. Здійснити перестановку бітів для Р-блока стискання, якщо на його вході діє послідовність $(10110)_2$:



9. Визначте, чи є *P*-блок з таблицею перестановки:

8	1	2	3	4	5	6	7

прямим P-блоком, P-блоком стискання або P-блоком розширення.

10. Визначити, чи є P-блок із таблицею перестановки

1	3	5	6	7

прямим P-блоком, P-блоком стискання або P-блоком розширення.

6.12 Завдання до самостійної роботи

1. Криптосистема DES. Принцип побудови та функціонування, основні параметри криптосистеми: довжина ключів, довжина блоку вхідного тексту, кількість раундів, крипостійкість, режими роботи.
2. Основні перетворення алгоритму DES. Призначення розширювальної перестановки в алгоритмі DES, яка подовжує 32-бітний правий півблок до 48 біт.
3. Режими роботи блокових шифрів: режим електронної кодової книги (Electronic Code Book — ECB); режим зчеплення блоків зашифрованих даних (Cipher Block Chaining — CBC); режим зворотного зв'язку за зашифрованими даними (Cipher Feedback — CFB); режим зворотного зв'язку за виходом (Output Feedback — OFB); режим лічильника (Counter Mode — CTR).

ПРАКТИКУМ № 7
Симетричне блокове шифрування даних. Алгоритм Advanced Encryption Standard - AES (Rijndael)

Мета роботи. Ознайомитися з теоретичними основами блокового шифрування даних, з основними перетвореннями інформації, що використовуються в блоковому шифрі *Advanced Encryption Standard (AES)*, а саме: перетворення повідомлення за допомогою циклових ключів; заміна байтів повідомлення; зсув у рядках.

7.1. Основні поняття про становлення стандарту

Потреба у новому стандарті шифрування постала у середині 1990-х років. Наявний тоді стандарт DES, довжиною ключа 56 біт, давав змогу застосувати метод грубої сили для дешифрування даних. Успішні злами даних відбулись уже наприкінці 1990-х. Крім того, архітектура DES орієнтувалась на апаратну реалізацію, а програмна реалізація на платформах з обмеженими ресурсами не давала необхідної швидкості застосування. Модифікація DES 3-DES мала достатню довжину ключа, але при цьому була ще повільнішою.

В жовтні 1997 р. NIST (Державний інститут стандартів і технологій, англ. National Institute of Standards and Technology) оголосив конкурс на обрання спадкоємця для DES, що був американським стандартом ще з 1977 року. Перед претендентами поставили такі основні вимоги:

- блочний шифр;
- довжина блоку 128 біт;
- ключі довжиною 128, 192 і 256 біт.

На відкритий конкурс було прийнято 15 алгоритмів, розроблених криптографами 12 країн — Австралії, Бельгії, Великобританії, Німеччини, Ізраїлю, Канади, Коста-Ріки, Норвегії, США, Франції, Південної Кореї та Японії.

Вибір алгоритму проходив у три етапи. В серпні 1998 року на 1-й конференції AES було оголошено список з 15 кандидатів. У фінал конкурсу вийшли такі алгоритми: *Mars*, *Twofish* і *RC6* (США), ***RIJNDAEL*** (Бельгія), *Serpent* (Великобританія, Ізраїль, Норвегія). За результатами доповідей 3-ї конференції, що проходила у Нью-Йорку у квітні 2000 року, 2 жовтня 2000 алгоритм, запропонований бельгійськими криптографами Д. Деймоном та В. Ріджменом, був оголошений переможцем конкурсу і почалась процедура стандартизації. В травні 2002 року AES був прийнятий як стандарт.

При відборі оцінювалися: реальна захищеність алгоритмів від криптоаналітичних атак; статистична безпека криптографічних алгоритмів; надійність математичної бази криптоалгоритмів; обчислювальна складність (швидкість) виконання шифрування і розшифрування; складність програмної, апаратної і апаратно-програмної реалізації; обчислювальна складність (швидкість) розгортання ключів; можливість роботи з різними довжинами інформаційних блоків і початкових ключів; можливість реалізації на існуючому спектрі програмних платформ і додатків; можливість вживання

алгоритму у всіх режимах роботи, що рекомендуються, – блокового шифрування, потокового шифрування, потокового шифрування із зворотним зв'язком, вироблення кодів автентифікації, хешування, а також генератора псевдовипадкових чисел; еквівалентність складності програмної, апаратної і апаратно-програмної реалізацій і ін.

7.2 Загальна характеристика криптоалгоритму AES

Алгоритм **RIJNDAEL (RD)** (або *Advanced Encryption Standard - AES)* є блоковим симетричним криптоалгоритмом. ***Довжина блоку*** l_n - 128 біт. Криптографічні перетворення в алгоритмі здійснюються за рахунок перетворення блоків інформації з використанням ключових даних. Ключ, що вводиться в засоби реалізації RIJNDAEL, називають початковим ключем K_u . ***Дозволеними довжинами початкового ключа*** є l_{k_u} =128, 192 і 256 біт.

Алгоритм побудований на **основі SP-мереж**. SP-мережа, або мережа замін-перестановок (*англ. substitution-permutation network*) — це ряд пов'язаних математичних операцій що використовуються в блочних шифрах, наприклад AES. Така мережа приймає блок відкритого тексту і ключ на вході, і застосовує декілька *«раундів» S-скринь і P-скринь*, які чергуються для отримання блоку шифротексту.

Ключі, що використовуються в циклах перетворення, формуються з початкового K_u . Для цього виконується процедура розгортання з початкового циклових ключів. Число розгорнених ключів N_p визначається, як $N_p = n_ц + 1$, а ***довжина*** l_p ***розгорненого ключа,*** як $l_p = (n_ц + 1)l_u$, де $n_ц$ – є число циклів перетворення (10, 12, 14), виконуваних в алгоритмі, а l_u – довжина інформаційного блоку (128 біт).

Алгоритм RD може застосовуватися в п'яти режимах:

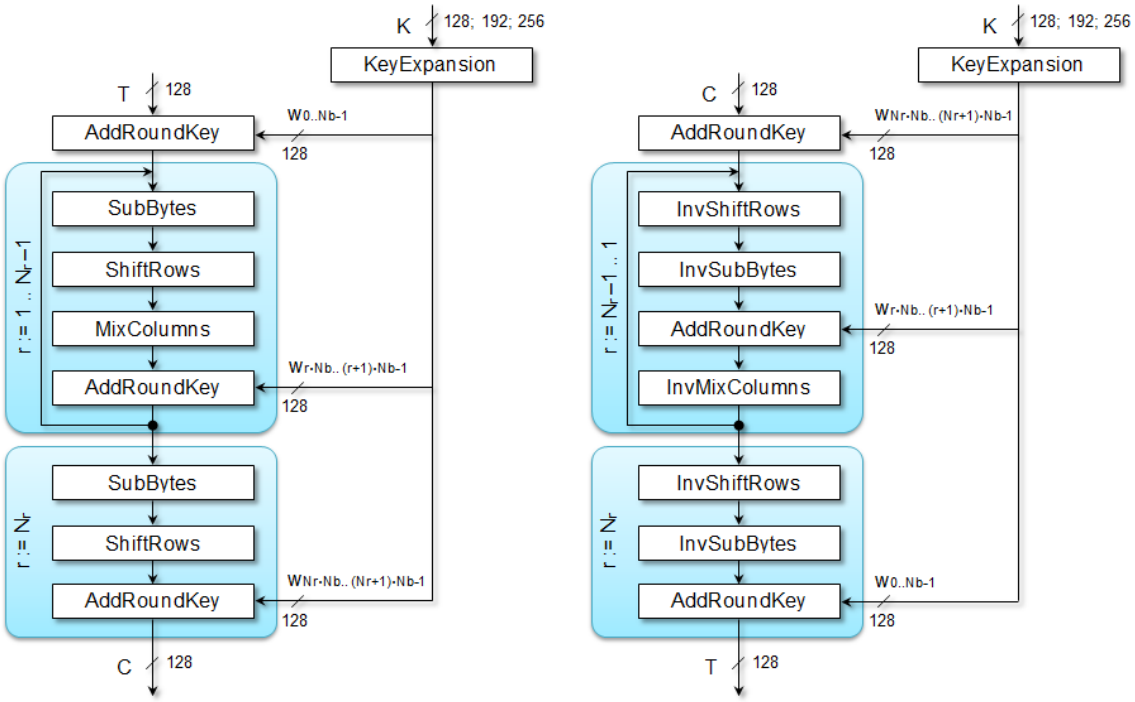
- *блокового шифрування;*
- *потокового шифрування;*
- *потокового шифрування із зворотним зв'язком*;
- *вироблення ключової хеш-функції (коду-автентифікації);*
- *генератора псевдовипадкових послідовностей.*

Алгоритм складається з **раундів** (10, 12 або 14 - це залежить від довжини ключа l_{k_u} =128, 192 і 256 біт), у яких послідовно виконуються такі операції:

- ***SubBytes*** - таблична підстановка 8x8 біт;
- ***ShiftRow*** - зсув рядків у двовимірному масиві;
- ***MixColumn*** - математичне перетворення, що перемішує дані в середині стовбця;
- ***AddRoundKey*** - додавання ключа операцією XOR.

В останньому раунді операція перемішування стовбців відсутня, що робить усю послідовність операцій симетричною.

Ілюстрація вказаних операцій даного алгоритму для зашифрування і розшифрування даних представлена на рис.7.1. [5, 7]



а) б)

Рис.7.1. Блок-схема шифрування (а) і розшифрування одного блоку (б) даних розміром 128 біт

7.3 Представлення перетворюваної інформації і ключів, цикл перетворення

У алгоритмі RD перетворення виконуються при сталій довжини блоків інформації l_u **=128 біт** і довжини початкових ключів l_{k_u} **=128, 192 і 256 біт**. Необхідна стійкість в алгоритмі забезпечується за рахунок багатоциклового перетворення, причому, в кожному з циклів застосовуються математичні і ключові перетворення, тобто перетворення по визначених формулах і розгорнених ключах. Нехай необхідно зашифрувати A_i блок інформації завдовжки **128 біт**. Назвемо значення $A_i = M_i$, де M_i – значення блоку, що має бути зашифрованим, початковим станом. Далі залежно від номера циклу j перетворення стан позначатимемо як A_{ij} , представляючи його у вигляді матриці байтів. Вхідні дані A_i і вихідні C_i розглядаються як одновимірні масиви з 8-бітових слів (байтів), пронумеровані по стовпцях від 0 до $4n_c - 1$, де n_c є кількість стовпців.

Для l_u =128 бітів (16 байтів) матриця стану має наступний вид:

$$A_{ij} = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{vmatrix} \quad (7.1)$$

Таким чином, стан A_{ij} описується матрицею байтів, в якій чотири рядки і чотири стовпці.

Приклад 7.1. Розглянемо, як можливо представити блок із *16* символами у вигляді матриці розміром *4x4.*

Нехай текстовий блок має наступне повідомлення «*AES uses a matrix*». Оскільки до 16 елементів блока не вистачає двух знаків, добавимо дві літери «Z»:

Текст	A	E	S	U	S	E	S	A	M	A	T	R	I	X	Z	Z
Десяткове значення	00	04	18	20	18	04	18	00	12	00	19	17	08	23	25	25
Шістнадцяткове значення	00	04	12	14	12	04	12	00	0c	00	13	11	08	17	19	19

Рис.7.2.Представлення відповідності текстового повідомлення його числовим еквівалентом

Тоді матриця стану набуде виду:

00	12	0c	08
04	04	00	23
12	12	13	19
14	00	11	19

Рис.7.3. Формування матриці стану

Аналогічно стану інформаційного блоку задається і початковий ключ k_u, наприклад, для довжини l_{k_i}=128 біт прийме наступний вид:

$$k_u = \begin{vmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{vmatrix}. \quad (7.2)$$

Для довжин ключів k_u, рівних 192 і 256 біт, початковий ключ задається аналогічно з більшим розміром матриці. У алгоритмі RD число циклів $n_ц$ перетворення залежить від довжини інформаційного блоку $l_м$ і довжини початкового ключа l_{ku}. У табл. 7.1 приведено значення $n_ц$ циклів перетворення як функції $l_м$ і l_{ku}.

Таблиця 7.1

Залежність кількості циклів перетворення від розміру ключа

$l_k \backslash l_м$	128	192	126
128	10	12	14
192	12	12	14
256	14	14	14

У кожному з циклів перетворення (за винятком останнього) виконується три табличні перетворення і одне криптографічне. В процесі перетворення байти прочитуються по стовпцях, тобто $a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, \ldots$ і т.д. На мові *псевдокоду* один цикл має вигляд:

Round(State); //цикли $\overline{1, n_ц - 1}$

{

SubBytes (State); //заміна байтів

ShiftRow(State); //зсув рядків

MixColumn(State); //перемішування в стовпцях

AddRoundKey(State, RoundKey); //додавання ключа

}

На останньому циклі перетворення MixColumn відсутнє, тобто перемішування в стовпцях не виконується. Алгоритм RD блокового шифрування на кількість циклів $n_ц$ представлений на рис. 7.4.

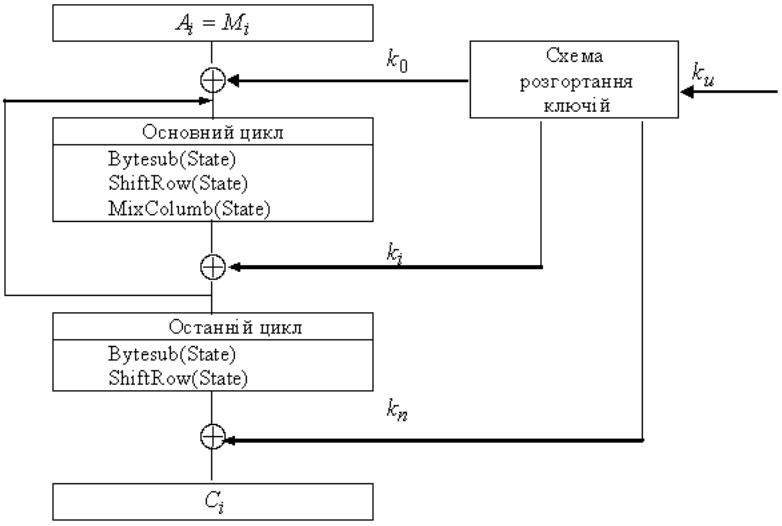


Рис.7.4. Алгоритм RD блочного шифрування.

7.4. Перетворення, що використовуються при зашифруванні

Згідно із засадничими принципами, сформульованими ще К.Шенноном, перетворення даних, використовуваних у шифрі, повинні надавати останньому дві основні властивості - ***розсіювання й перемішування***.

Розсіювання припускає поширення впливу кожного біта відкритого повідомлення, а також кожного біта ключа на значну кількість бітів зашифрованого повідомлення.

Перемішування призводить до втрати в процесі шифрування будь-яких залежностей між бітами відкритого повідомлення.

Саме ці дві властивості забезпечують захист від двох можливих загроз: підробки повідомлення та його розкриття.

За забезпечення цих двох властивостей відповідають функції ***MixColumns()*** **і** ***SubBytes().*** Розсіювання в шифрі *Rijndael*, забезпечується здебільше функцією *MixColumns()*, перемішування — здебільше функцією *SubBytes()*. Перетворення *SubBytes* (заміна байт) - це послідовна заміна байтів-елементів матриці повідомлення. Тобто послідовне виконання дій над кожним байтом повідомлення, представленого у вигляді матриці (рис. 7.5).

Рис. 7.5. Процедура заміни байтів в *SubBytes*

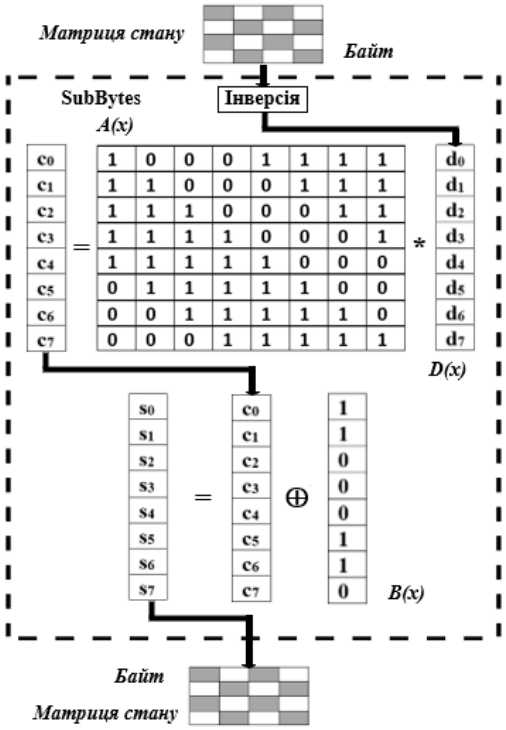
Дії, що виконуються над повідомленням, математично можна представити у вигляді :

$$Y = C \cdot x + C_1 \left(\mathrm{mod}\, x^8 + 1 \right). \tag{7.3}$$

У матричному представлені формула (7.3) має вигляд:

$$\begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{vmatrix} \oplus \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{vmatrix}. \tag{7.4}$$

Матриця 8x8 або подібні до неї, називається ***матрицею Тепліца*** (теплицева матриця) або діагонально-постійною матрицею. Візуально процедуру реалізації (7.3) та (7.4) можна представити на рис.7.6. [5].



$$S(x) = A(x) \cdot D(x) \oplus B(x),$$

де $S(x)$, $D(x)$ і $B(x)$ — вектор-стовпці розмірністю 8×1;

$A(x)$ — матриця розмірністю 8×8.

$$s(x) = \begin{vmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{vmatrix} \cdot \begin{vmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{vmatrix} \oplus \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{vmatrix}.$$

Рис.7.6. Представлення перетворення *SubBytes* матриці стану

Перетворення **ShiftRow** (зсув в рядках) забезпечує циклічний зсув в рядках матриці, якою представлено повідомлення. При цьому перший рядок циклічно не зсовується, а другий, третій і четвертий циклічно зсовуються на величину, вказану в табл. 7.2. Причому, величини зсуву C_1, C_2, C_3 залежать від числа стовпців n_c матриці повідомлення $A_i(4, 6 \text{ чи } 8)$.

Таблиця 7.2

Зсуву в рядках для матриць різних розмірів

n_c	C_1	C_2	C_3
4	1	2	3
6	1	2	3
8	1	3	4

На рис.7.7 приведений приклад перетворення «ShiftRow» для стану $A_i(4)$, довжина блоку 128 біт (16 байт).

a_{00}	a_{01}	a_{02}	a_{03}	Зсуву немає	a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}	Зсув на 1	a_{13}	a_{10}	a_{11}	a_{12}
a_{20}	a_{21}	a_{22}	a_{23}	Зсув на 2	a_{22}	a_{23}	a_{20}	a_{21}
a_{30}	a_{31}	a_{32}	a_{33}	зсув на 3	a_{31}	a_{32}	a_{33}	a_{30}

Рис. 7.7. Приклад перетворення «зсув рядків»

Перетворення типу **MixColumn** («перемішування в стовпцях») забезпечує послідовне табличне перетворення елементів-байтів стовпців. Стовпці, що завжди складаються з чотирьох байт, представляються поліномами третього ступеня:

$$a(x) = a_{3j}x^3 + a_{2j}x^2 + a_1 x + a_0 \left(\mathrm{mod}\, x^4 + 1\right), \quad j = \overline{0, n_c}\,. \tag{7.5}$$

Коефіцієнти $a(x)$ приймають в (7.5) значення в інтервалі від нуля (00000000) до 255 (11111111).

Перемішування проводиться множенням $a(x)$ на константу

$$C(x) = '3'x^3 + '1'x^2 + '1'x + '2',$$

тобто перетворений стовпець $a'(x)$ є

$$a'(x) = C(x) \cdot a(x) \left(\mathrm{mod}\, x^4 + 1\right) \tag{7.6}$$

У матричному вигляді перетворення (7.6) має вигляд:

$$\begin{vmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{vmatrix} = \begin{vmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{vmatrix}. \tag{7.7}$$

На рис. 7.8 показано перетворення типу «перемішування в стовпцях» для A_i вигляду (7.1).

a_{00}	a_{01}	a_{02}		a_{03}
a_{10}	a_{11}	a_{12}		a_{13}
a_{20}	a_{21}	a_{22}		a_{23}
a_{30}	a_{31}	a_{32}		a_{33}

$\otimes C(x)$

a'_{00}	a'_{01}	a'_{02}	a'_{03}
a'_{10}	a'_{11}	a'_{12}	a'_{13}
a'_{20}	a'_{21}	a'_{22}	a'_{23}
a'_{30}	a'_{31}	a'_{32}	a'_{33}

Рис.7.8 – Перетворення «перемішування в стовпцях»

Функцію розсіювання *MixColumns()* було вибрано з можливих лінійних перетворень *4 bytes → 4 bytes* (чотири байти стовпця масиву стану в чотири байти) виходячи з таких критеріїв вибору:

- оборотність;
- лінійність у полі $GF(2^4)$; ця властивість функції *MixColumns()* дозволяє застосовувати алгоритм прямого розшифрування;
- досить сильне розсіювання;
- швидкість виконання на *8*-розрядних процесорах;
- симетричність, тобто *MixColumns()* повинна працювати з усіма даними однаково;
- простота опису.

Приклад 7.2. Нехай на вхід функції *MixColumns()* алгоритму *AES* (*Nb = 4*) надходить масив стану, який дорівнює [5]:

$$State = «d4bf5d30e0b452aeb84111f11e2798e5»_{16}, \qquad (7.8)$$

при цьому

$$\begin{array}{llll} s_{0,0} = d4; & s_{0,1} = e0; & s_{0,2} = b8; & s_{0,3} = 1e; \\ s_{1,0} = bf; & s_{1,1} = b4; & s_{1,2} = 41; & s_{1,3} = 27; \\ s_{2,0} = 5d; & s_{2,1} = 52; & s_{2,2} = 11; & s_{2,3} = 98; \\ s_{3,0} = 30; & s_{3,1} = ae; & s_{3,2} = f1; & s_{3,3} = e5. \end{array}$$

Масив стану на виході буде визначатися виразом (7.7). Визначимо стовпець масиву стану на виході для випадку $c = 0$, використовуючи вираз (7.7). У даному випадку

$$s'_{0,0} = \{02\} \bullet s_{0,0} \oplus \{03\} \bullet s_{1,0} \oplus s_{2,0} \oplus s_{3,0} =$$
$$= \{02\} \bullet \{d4\} \oplus \{03\} \bullet \{bf\} \oplus \{5d\} \oplus \{30\} =$$
$$= \{02\} \bullet \{d4\} \oplus \{02\} \bullet \{bf\} \oplus \{bf\} \oplus \{5d\} \oplus \{30\}.$$

Представимо значення *{d4}*, *{bf}*, *{5d}* і *{30}* у вигляді поліномів:

$$\{d4\} \rightarrow x^7 + x^6 + x^4 + x^2,$$
$$\{bf\} \rightarrow x^7 + x^5 + x^4 + x^3 + x^2 + x + 1,$$
$$\{5d\} \rightarrow x^6 + x^4 + x^3 + x^2 + 1,$$
$$\{30\} \rightarrow x^5 + x^4.$$

Обчислимо *{02 }●{d4}*:

$$\{02\} \bullet \{d4\} \rightarrow x \cdot (x^7 + x^6 + x^4 + x^2) \bmod p(x) =$$
$$= (x^8 + x^7 + x^5 + x^3) \bmod (x^8 + x^4 + x^3 + x + 1) =$$
$$= x^7 + x^5 + x^4 + x + 1.$$

Обчислимо *{02}●{bf}*:

$$\{02\} \bullet \{bf\} \rightarrow x \cdot (x^7 + x^5 + x^4 + x^3 + x^2 + x + 1) \bmod p(x) =$$
$$= (x^8 + x^6 + x^5 + x^4 + x^3 + x^2 + x) \bmod (x^8 + x^4 + x^3 + x + 1) =$$
$$= x^6 + x^5 + x^2 + 1.$$

Обчислимо елемент $s'_{0,0}$:

$$s'_{0,0} = (x^7 + x^5 + x^4 + x + 1) \oplus (x^6 + x^5 + x^2 + 1) \oplus$$
$$\oplus (x^7 + x^5 + x^4 + x^3 + x^2 + x + 1) \oplus (x^6 + x^4 + x^3 + x^2 + 1) \oplus$$
$$\oplus (x^5 + x^4) = x^2 \rightarrow \{04\}.$$

При проведенні аналогічних обчислень отримаємо значення $s'_{1,0}$, $s'_{2,0}$ і $s'_{3,0}$:

$$s'_{1,0} = x^6 + x^5 + x^2 + x \rightarrow \{66\};$$
$$s'_{2,0} = x^7 + 1 \rightarrow \{81\};$$
$$s'_{3,0} = x^7 + x^6 + x^5 + x^2 + 1 \rightarrow \{e5\}.$$

При проведенні аналогічних перетворень для інших елементів, отримаємо масив стану на виході функції *MixColumns()* (рис. 7.9).

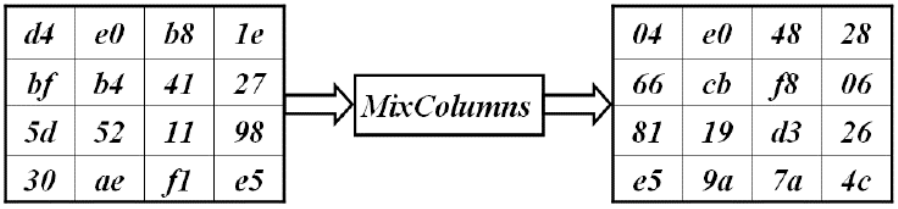


Рис.7.9. Перетворення масиву стану функцією *MixColumns()*

Таким чином, вхідне значення масиву стану (7.8.) перетвориться на вихідне:

$$State_{вих} = «\mathbf{046681e5e0cb199a48f8d37a2806264c}»_{16.}$$

Попередні перетворення є оберненими і відомими. Якщо не додавати ключовий шифр у кожному раунді, зловмисник може визначити вхідне повідомлення за відомим для нього шифрованими даними. У цьому випадку необхідно додавати ключ шифру для додаткових «таємних» перетворень.

Криптографічне перетворення **AddRoundKey** в алгоритмі RD здійснюється за допомогою додавання по модулю масиву стану A_i і k_ν ключа ν-го циклу перетворення. В загальному вигляді це перетворення можна представити як:

$$A'_{ij} = A_{ij} \oplus K_{\nu j} \ . \qquad (7.9)$$

У матричному вигляді для стану (7.9) його можна представити як:

$$\begin{vmatrix} a'_{00} & a'_{01} & a'_{02} & a'_{03} \\ a'_{10} & a'_{11} & a'_{12} & a'_{13} \\ a'_{20} & a'_{21} & a'_{22} & a'_{23} \\ a'_{30} & a'_{31} & a'_{32} & a'_{33} \end{vmatrix} = \begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{vmatrix} \oplus \begin{vmatrix} k_{00} & k_{01} & k_{02} & k_{03} \\ k_{10} & k_{11} & k_{12} & k_{13} \\ k_{20} & k_{21} & k_{22} & k_{23} \\ k_{30} & k_{31} & k_{32} & k_{33} \end{vmatrix} \qquad (7.10)$$

Таким чином, зашифрування проводиться за допомогою байтного складання байт A_i стану і байтів k_ν розгорнутого ключа.

Після закінчення процедури зашифрування блок-криптограма C_i представляє собою стан, одновимірний масив байти якого формуються прочитанням стовпців матриці

$$C_{0,0} = a'_{00}, C_{1,0} = a'_{10}, C_{3,0} = a'_{30}, C_{0,1} = a'_{01}, \ C_{0,2} = a'_{02}, \dots \ .$$

7.5. Зворотні перетворення (розшифрування)

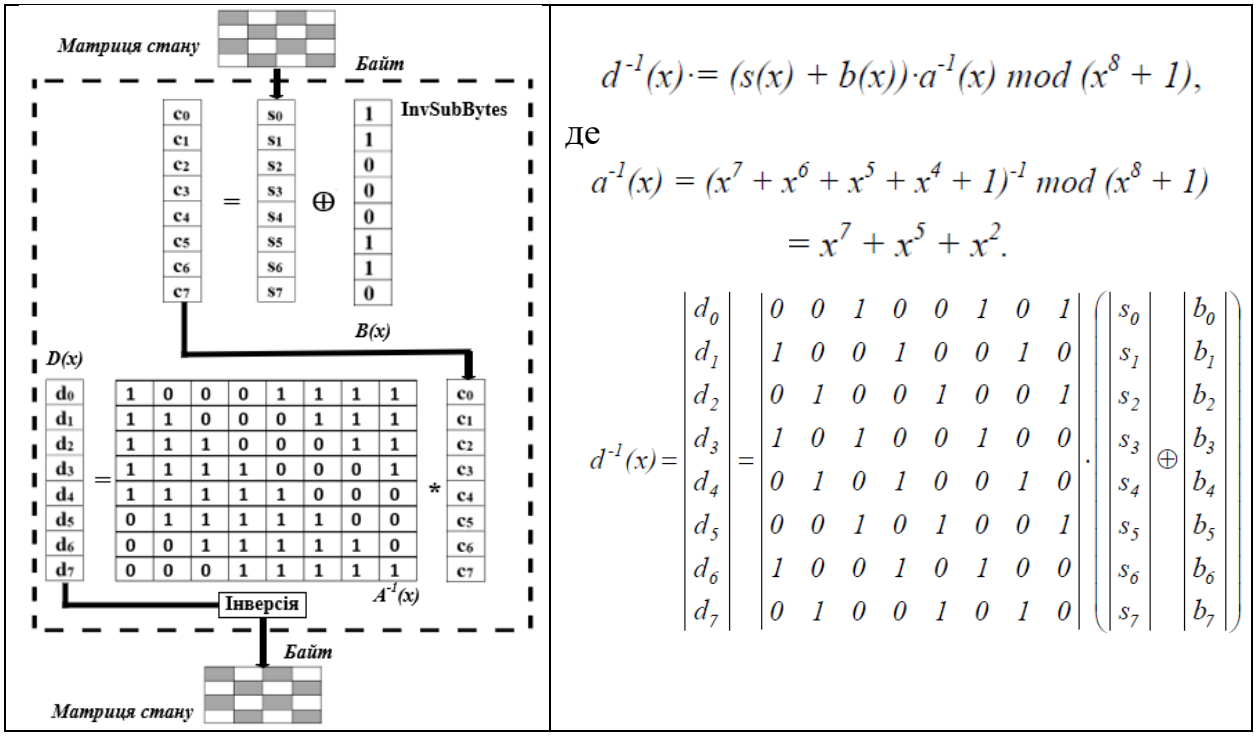
Алгоритм RD в явному вигляді не є симетричним. На рис. 7.1(б) приведена структурна схема розшифрування C_i блоку масиву стану.

У алгоритмі використовуються інверсні перетворення ***Inv***, особливістю яких є те, що при їх виконанні використовуються інші перетворення.

Перетворення ***InvSubBytes*** виконується аналогічно ***SubBytes,*** тільки замість формули (7.4) використовується наступний вираз:

$$\begin{vmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{vmatrix} \cdot \begin{vmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{vmatrix} \otimes \begin{vmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{vmatrix}.$$

Схема перетворення (рис.7.6.) зміниться та отримає наступний вид (рис.7.10) [5]:



$$d^{-1}(x) \cdot = (s(x) + b(x)) \cdot a^{-1}(x) \bmod (x^8 + 1),$$

де

$$a^{-1}(x) = (x^7 + x^6 + x^5 + x^4 + 1)^{-1} \bmod (x^8 + 1)$$
$$= x^7 + x^5 + x^2.$$

$$d^{-1}(x) = \begin{vmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{vmatrix} = \begin{vmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{vmatrix} \cdot \left(\begin{vmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{vmatrix} \oplus \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{vmatrix} \right)$$

Рис.7.10. Представлення перетворення *InvSubBytes* матриці стану

В перетворені ***InvShiftRow*** виконується також циклічний зсув, як і в *ShiftRow*, але напрям цього зсуву протилежний. В першому рядку циклічний зсув не проводиться, а в інших рядках проводиться відповідно до таблиці 7.2.

На рис.7.11 приведений приклад перетворення «зсув в рядках» для стану A_i (4), довжина блоку стану 128 біт (16 байт).

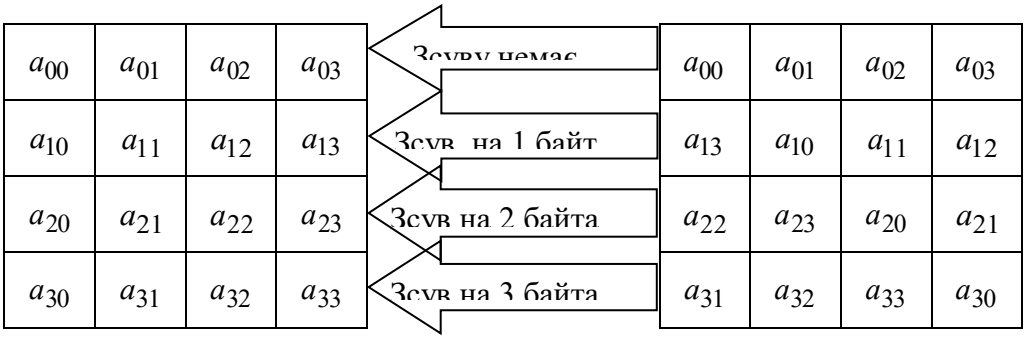


Рис.7.11 - Перетворення «зсув рядків»

InvMixColumn також проводиться аналогічно *MixColumn,* тільки для виконання дій замість формули (7.7) використовується наступне перетворення:

$$\begin{vmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{vmatrix} = \begin{vmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{vmatrix} \begin{vmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{vmatrix}.$$

Як показали дослідження, на вживані в RD початкові ключі не накладається ніяких обмежень. Тому можна припустити, що слабких ключів для цього алгоритму не існує.

7.6. Інструкція по користуванню програмою «Cript»

Для дослідження основних функціональних перетворень алгоритму скористаємося програмних додатком «**Cript**» (розроблено на кафедрі захисту інформації у Вінницькому національному технічному університеті). Додаток дозволяє досліджувати основні перетворення алгоритму для різних режимів, зокрема для навчання, контроля та тестування.

Після запуску програми з'являється головне меню, структура якого зображена на рис. 7.12.

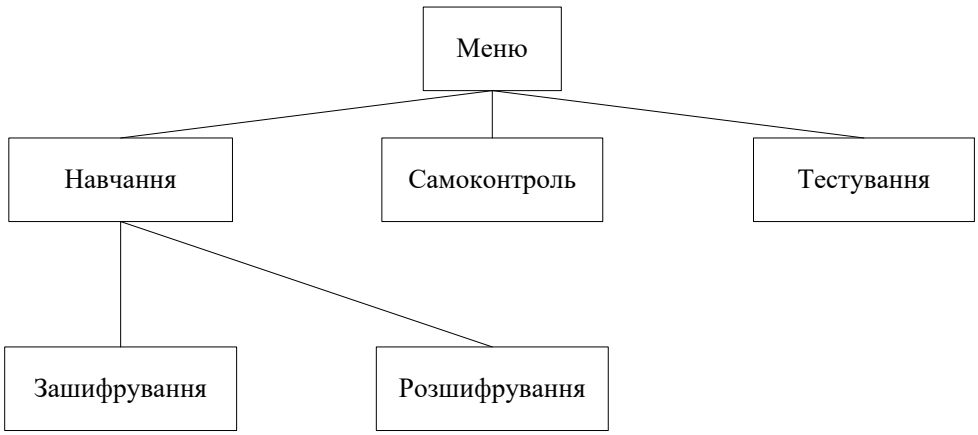


Рис.7.12 – Структура головного меню додатку «**Cript**»

Для того, що перейти до режиму «**НАВЧАННЯ**» необхідно зайти в пункт меню «Навчання». Можливі два типи навчання:

- **«зашифрування»** – зашифрування за допомогою блокового шифру AES;
- **«розшифрування»** – розшифрування за допомогою блокового шифру AES.

Для того, що вибрати навчання типу «з**ашифрування**», необхідно в головному меню вибрати пункт «Навчання», а потім «Зашифрування». На екрані з'явиться вікно для введення відкритого повідомлення і початкового ключа (рис.7.13). В поле «1» необхідно ввести відкрите повідомлення, а в поле «2» – початковий ключ (див. індивідуальні завдання в п.7.8).

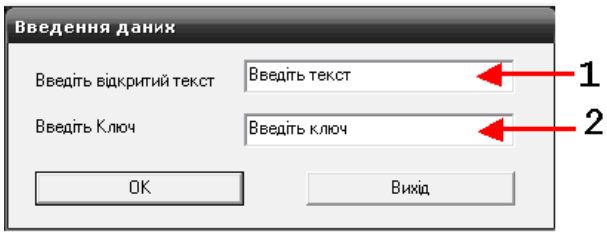


Рис.7.13 – Введення повідомлення і ключа

Наступним вікном, що з'явиться, буде вікно перетворення **«Додавання ключа»** (рис. 7.14). В нижній частині даного вікна знаходиться детальне пояснення дій на даному кроці. В комірці «1» даного вікна буде виведено перший байт матриці повідомлення, а комірці «2» – перший байт матриці ключа. Після натиснення кнопки «*Далі >>>*» з комірок «1» і «2» беруться значення і додаються за модулем 256 і виводяться в комірку «3». Після цього у відповідні наступні комірки виводяться наступні байти матриць повідомлення і ключа. Ці дії будуть виконуватись до тих пір, поки не будуть заповненні всі коміки. В даному вікні під «*індексом елементів*» розуміється значення індексів *i* і *j* елементів матриць повідомлення **A** і ключа **K**, над якими проводяться дії, а «*значення елементів*» - числові значення елементів матриць повідомлення **A** і ключа **K**, над якими проводяться дії.

В кінці виконання всіх дій стає активною кнопка «*Наступний крок>>>*», натиснувши яку дане вікно закривається і відкривається наступне, що реалізує перетворення **«Зміна байтів»** (рис.7.15).

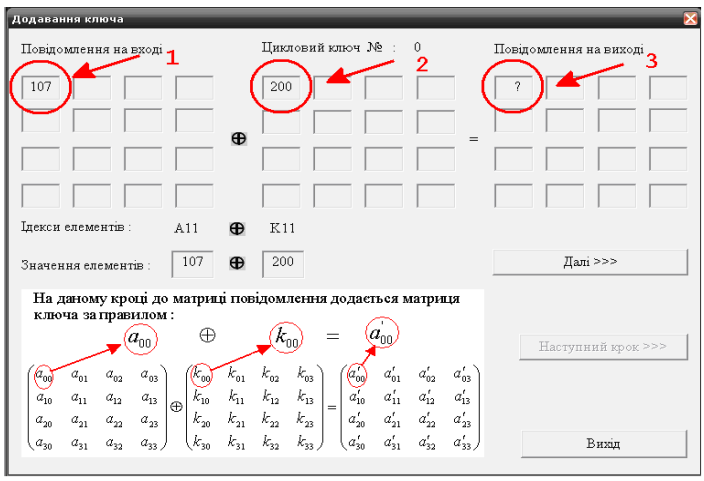


Рис.7.14. Вікно для додавання матриць повідомлення і ключа

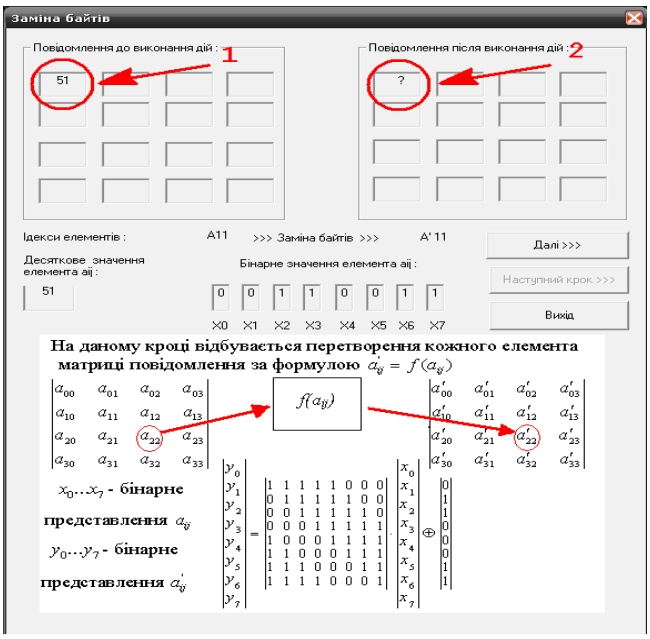


Рис.7.15 – Вікно для перетворення «Зміни байтів»

В нижній частині вікна для **«Зміни байтів»** знаходиться детальне пояснення дій на даному кроці. В комірці «1» даного вікна буде виведено перший байт матриці повідомлення до виконання дій. Після натиснення кнопки «*Далі >>>*» з комірки «1» береться значення елемента і проводиться дія «*Зміни байтів*» і її результат виводяться в комірку «2». Після цього у наступі комірки виводяться наступні байти матриць повідомлення. І так повторюється до заповнення всіх комірок. Стрічка «*Індекси елементів*» показує значення індексів ***i*** і ***j*** елемента вхідної матриці повідомлення ***i*** вихідної зашифрованої матриці повідомлення. «*Десяткове значення елемента* a_{ij}» - десяткове представлення елемента вхідної матриці повідомлення, а «*бінарне значення елемента*» - це представлення елемента вхідної матриці повідомлення в двійковій системі.

При заповненні всіх комірок відкривається кнопка «*Наступний крок>>>*», яка закриває поточне вікно і відкриває вікно перетворення **«Зсув у рядках»** (рис.7.16). В рядок «1» виводиться перший рядок матриці повідомлення. Далі при натисненні кнопки «*Далі >>>*» відбувається циклічний зсув рядка на величину, вказану в вікні. Далі аналогічно проводяться дії до заповнення всіх комірок.

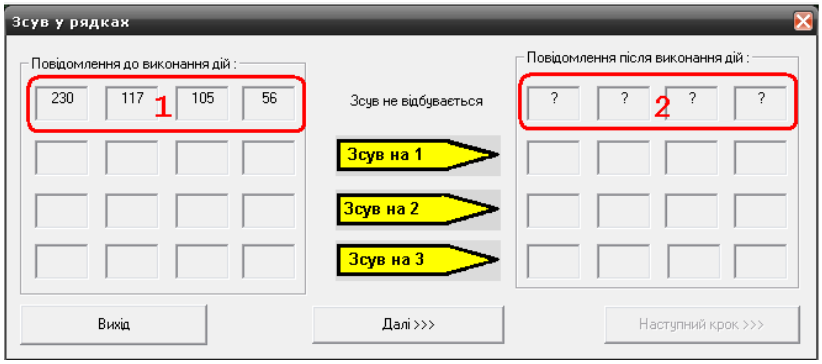


Рис.7.16 – Віно перетворення «Зсув у рядках»

Кнопка «*Наступний крок>>>*» закриває дане вікно і відкриває вікно перетворення **«Перемішування в колонках»** (рис. 7.17).

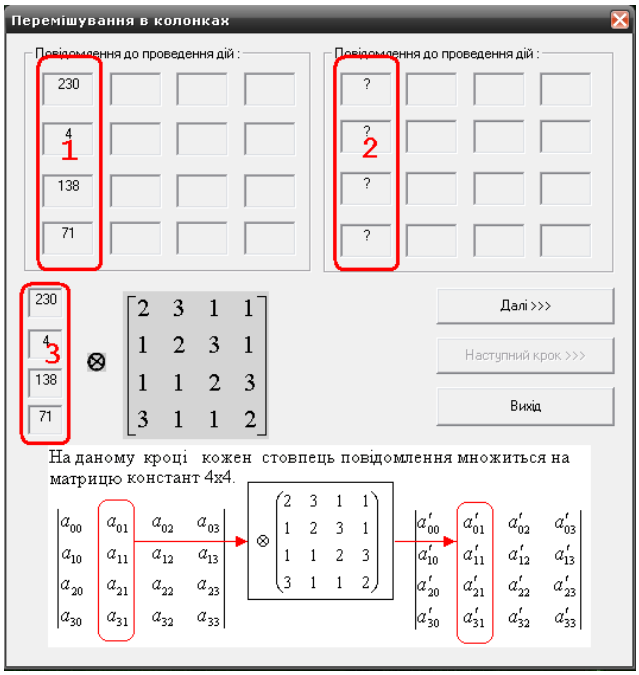


Рис.7.17. Вікно перетворення «Перемішування в колонках»

В нижній частині вікна для «Перемішування в колонках» знаходиться детальне пояснення дій на даному кроці. В стовпець «1» даного вікна буде виведено перший стовпець матриці повідомлення. Після натиснення кнопки «*Далі>>>*» береться стовпець «1» і проводиться дія над ним і результат записується в стовпець «2». І так для кожного наступного стовпця, аж поки не буде заповнено всі комірки.

Кнопка «*Наступний крок>>>*» закриває дане вікно і відкриває вікно – **«Виведення зашифрованого повідомлення»** (рис. 7.18), яке є завершальним для зашифрування. В полі «1» виводиться зашифрований текст, який можна зберегти в файл при натисканні кнопки «Зберегти в файл». А для того, щоб завершити зашифрування, потрібно натиснути кнопку «**ОК**».

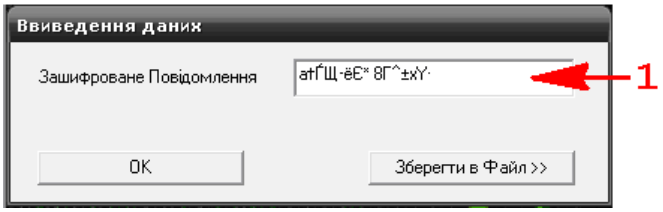


Рис.7.18 – Вікно виведення зашифрованого повідомлення

Тип режиму **«Навчання» - «Розшифрування»** Проводиться аналогічно до **«Зашифрування»,** тільки вікна виводяться в зворотному порядку і

формули, за якими проводяться обчислення, також інші (див. теоретичні відомості).

Режим **«САМОКОНТРОЛЬ»** викликається при виборі пункту меню «Самоконтроль» (див. рис. 7.12). На екрані з'явиться вікно для введення відкритого повідомлення і початкового ключа (рис. 7.19). В поле «1» необхідно ввести відкрите повідомлення, а в поле «2» – початковий ключ.

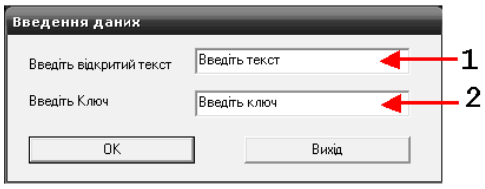


Рис.7.19 – Введення повідомлення і ключа

Наступним вікном, що з'явиться, буде вікно перетворення **«Додавання ключа»** (рис. 7.20).

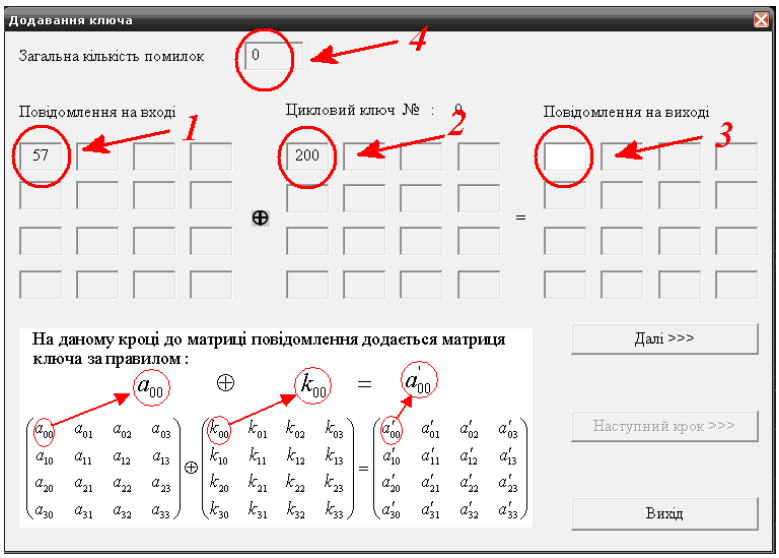


Рис.7.20 – Вікно перетворення «Додавання ключа»

В комірках «1» і «2» буде виведено ASCII коди перших символів матриць повідомлення і ключа відповідно. Комірка «3» буде порожньою. В неї вводиться результат даної дії, який студент обраховує сам. Після натиснення кнопки «*Далі* >>>» програма проводить перевірку введеного результату. Якщо результат вірний, то виводяться наступні значення матриць повідомлення і ключа і відкривається наступна порожня комірка. Якщо результат не вірний, то на екран виводиться повідомлення про помилку і лічильник помилок «4» збільшується. Дії проводяться до тих пір, поки не буде заповнено всі комірки. По завершенні заповнення відкривається кнопка «*Наступний крок* >>>», натиснувши яку дане вікно закриється, а вікно перетворення **«Зміна байтів»** (рис. 7.21) з'явиться на екрані.

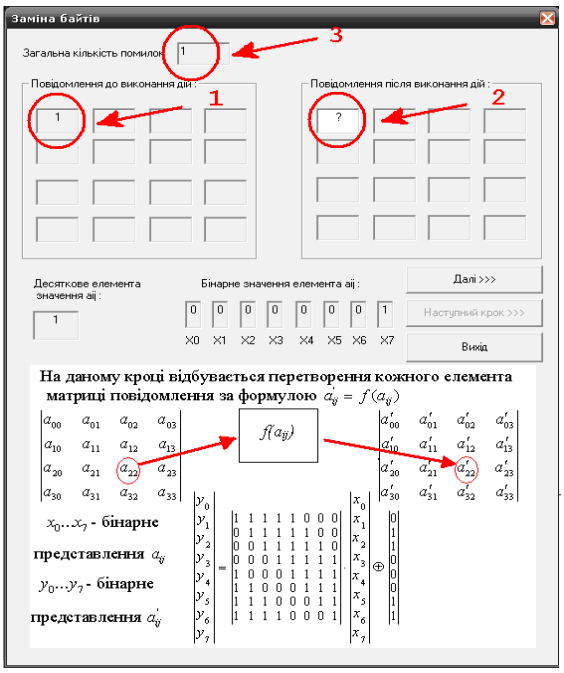


Рис. 7.21 – Вікно перетворення «Зміна байтів»

В комірці «1» буде виведено ASCII коди першого символу матриці повідомлення. Комірка «2» буде порожньою. В неї вводиться результат даної дії, який студент обраховує сам. Після натиснення кнопки «*Далі* >>>» програма проводить перевірку введеного результату. Якщо результат вірний, то виводиться наступний елемент матриці повідомлення і відкривається наступна порожня комірка. Якщо результат не вірний, то на екран виводиться повідомлення про помилку і лічильник помилок «3» збільшується. Дії проводяться до тих пір, поки не буде заповнено всі комірки.

По заповненню всіх комірок відкривається кнопка «*Наступний крок*>>>». При її натисканні вікно закриється, а вікно перетворення **«Зсув у рядках»** (рис. 7.22) з'явиться на екрані.

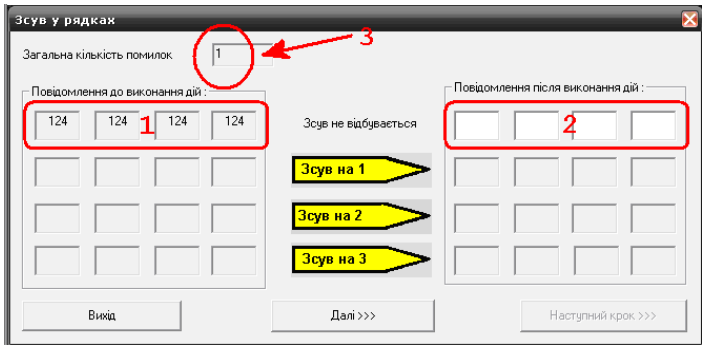


Рис. 7.22 – Вікно перетворення «Зсув у рядках»

В полі «1» буде виведено ASCII коди першого рядка матриці повідомлення. Значення в поле «2» будуть порожні. В них вводиться результат даної дії, який студент обраховує сам. Після натиснення кнопки «*Далі* >>>» програма проводить перевірку введеного результату. Якщо

результат вірний, то виводиться наступний рядок матриці повідомлення і відкривається наступний порожній рядок. Якщо результат не вірний, або нічого не було введено, то на екран виводиться повідомлення про помилки і лічильник помилок «З» збільшується. Дії проводяться до тих пір поки не буде заповнено всі комірки.

По заповненню всіх комірок відкривається кнопка «Наступний крок >>>», натиснувши яку дане вікно закриється, а вікно перетворення **«Перемішування у колонках»** (рис. 7.23) з'явиться на екрані.

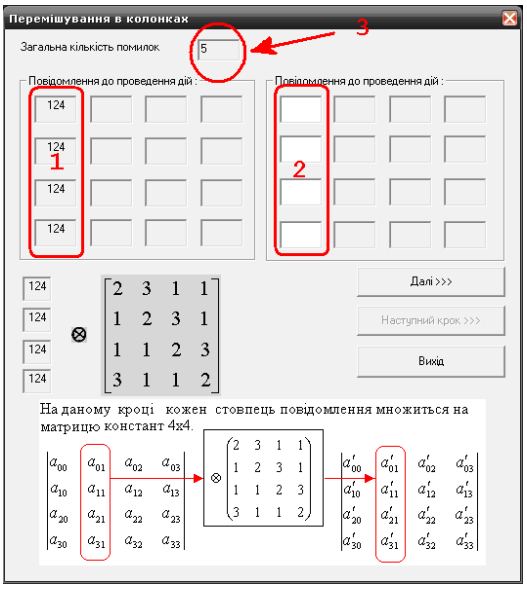


Рис.7.23 – Вікно перетворення «Перемішування в колонках»

В полі «1» буде виведено ASCII коди першого стовпця матриці повідомлення. Поле «2» – порожній стовпець. В них вводиться результат даної дії, який студент обраховує сам. Після натиснення кнопки «*Далі* >>>» програма проводить перевірку введеного результату. Якщо результат вірний, то в виводиться наступний стовпець матриці повідомлення і відкривається наступний порожній стовпець. Якщо результат не вірний, або зовсім не було нічого введено, на екран виводиться повідомлення про помилки і лічильник помилок «З» збільшується. Дії проводяться до тих пір поки не буде заповнено всі комірки.

По завершенню цих дій стає доступною кнопка «*Наступний крок* >>>», при натсканні якої вікно закриється, а вікно з результатами (рис.7.24) з'явиться на екрані. В вікні результатів в полі «1» виводиться зашифроване повідомлення. Поле «2» при самоконтролі не використовується. А в комірках «З» і «4» виводяться загальна кількість помилок, допущених на всіх кроках і відповідна оцінка. Для завершення режиму «**Самоконтроль**» потрібно натиснути кнопку

«!!! Завершення тестування !!!».

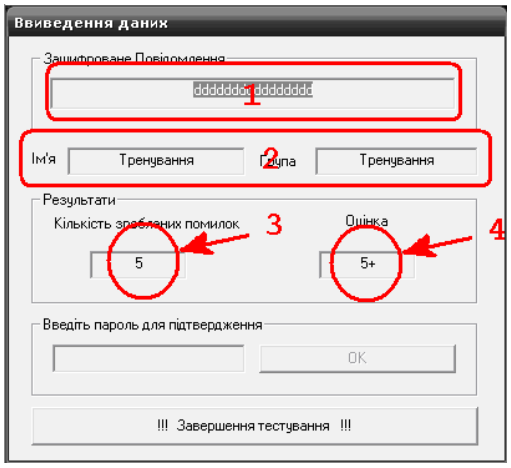


Рис.7.24 – Вікно з результатами

Для запуску режиму **«ТЕСТУВАННЯ»** потрібно вибрати пункт головного меню **«Тестування»**. Режим **«Тестування»** починається з вікна, яке заповнюється викладачем. Потім слідує вікно з інформацією про студента (рис. 7.25), в поля кого студент вводить своє прізвище, ім'я, по батькові і групу.

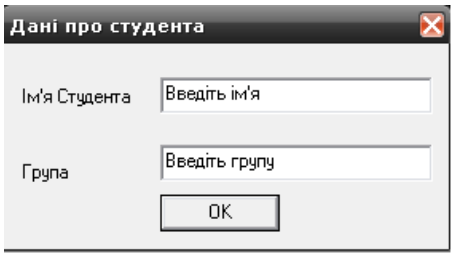


Рис.7.25 – Інформація про студента

Далі відбувається перевірка знань студентів так як у режимі **«Самоконтроль».** Тільки в режимі навчання можна вийти лише тоді, коли пройдені всі кроки.

При проходженні всіх кроків на екрані з'явиться вікно з результатами (рис.7.26). В полі «1» даного вікна виводиться зашифрований текст, в полі «2» група і ім'я того хто тестується. В полях «3» і «4» виводиться кількість помилок і оцінка, яку студент заробив.

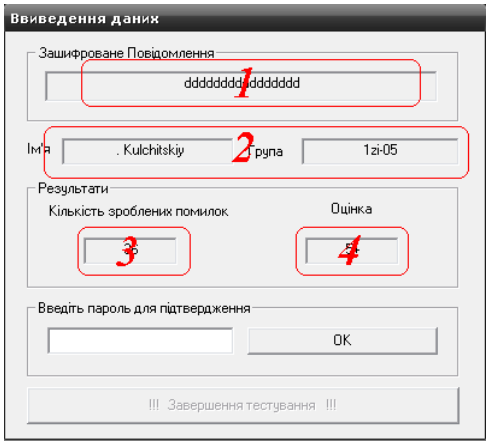


Рис.7.26 – Вікно з результатами

7.7. Підготовка до завдання

Ознайомитися з теоретичними положеннями побудови блокових шифрів симетричного шифрування, принципами функціонування та основними перетвореннями алгоритму *Advanced Encryption Standard (AES)*.

7.8. Практичне завдання

1.Ознайомитись з теоретичними відомостями функціонування алгоритму ***AES*** та математичним апаратом його реалізації для шифрування даних.
2.Запустити програмний модуль **Cript** та ознайомитися з інтерфейсом, пунктами меню та функціональними можливостями програми.
3.Набути практичних навичок математичних перетворень алгоритму ***AES*** при використанні програми **Cript** в режимі **«Навчання»**.
4.Перевірити засвоєння матеріалу при використанні програми **Cript** в режимі **«Самоконтроль»**.
5.Згідно варіанту (номер за списком в журналі групи) оберіть завдання з пункту **«Індивідуальні завдання»** і перевірте свій рівень підготовки за допомогою режиму програми **«Тестування»**.
6.На етапі **«Самоконтроль»** та **«Тестування»** та для зручності можна скористатися допоміжним програмним кодом, написаним самостійно, щоб полегшити обтяжливі процедури перемноження матриць та двійкових векторів.

Варіанти індивідуальних завдань

№ варіанту	*Відкрите повідомлення*	*Початковий ключ*
1	В последнее врем	Дуже важлива зна
2	Computers are us	заметно возрос и
3	звернення прогр	are almost every
4	с нетрадиционной	ми до файлів оск
5	The banking indu	методам лечения,
6	це дає майже нео	uses computers t
7	зарубежной медиц	можливості для к
8	enormous numbers	Немедикаментозны
9	програмою, що ла	documents such a
10	чаще всего связы	Наприклад навіть
11	and deposit slip	медициной в сфер
12	програма виконує	applications inc
13	Авторы не отрица	проводить зчитув
14	funds transfer w	спорности ряда п
15	команд з власног	in some areas, e
16	теории Бэйтса. Н	файлу до операти
17	allows banking t	совершенно неопр
18	пам’яті вже не к	In manufacturing
19	отрицать этот ме	про інші прихова
20	aided design and	Когда стороннико

7.9. Зміст протоколу роботи

Протокол до виконаної практичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки. Робота захищається індивідуально.

7.10 Контрольні питання для самоперевірки

1 Перерахувати параметри (розмір блока, розмір ключа та кількість раундів) для трьох версій AES.
2 Які функціональні перетворення відбуваються в одному раундовому перетворенні?
3 Яке призначення раудового ключа при виконанні перетворення?
4 Що розуміють під розширенням ключа і яке призначення цієї операції?
5 Що розуміють під кількістю розгорнутих ключів?
6 Що уявляє собою довжина рогорненого ключа?
7 Які режими роботи підтримує алгоритм RD?
8 Яке функіональне перетворення реалізує функція *SubBytes?*
9 Яке функіональне перетворення реалізує функція *ShiftRow?*
10 Яке функіональне перетворення реалізує функція *MixColumn?*
11 Що таке матриця стану і який вона має розмір?
12 Яке свівідношення між «бітом», «байтом», «словом», «блоком»?
13 Які раундові операції в алгоритмі RD призводять до нелінійних перетворень?
14 Скільки перетворень є в кожній версії AES? Який розмір ключів необхідно для кожної версії?
15 Які із чотирьох перетворень, визначених для AES, змінюють зміст байтів, а які не змінюють?
16 Яка мережа використовується для побудови криптосистеми AES? Що лежить в основі фунціонування цієї мережі?
17 Які функції виконують «S» і «P» блоки в мережі криптосистеми AES?
18 Що розуміють під термінами «розсіювання» та «перемішування»? Які блоки реалізують функції «розсіювання» та «перемішування» в алгоритмі AES?
19 Що у арифметиці полів Галуа називають непривідний поліном? Які його властивості? Навести приклад незвідних поліномів.
20 Які матриці називають матрицями Тепліца? Які їх властивості?

7.11. Задачі до практикуму №7

1. Для зашифрування взято повідомлення «***Відомо, що вчитися ніколи не пізно***». Представити повідомлення у вигляді матриці станів у числовому еквіваенті в шеснацятирічній системі з врахуванням ASCII кодування. При необхідності, заповнити пусті клітинки матриці довільними символами.

2. При шифруванні було використано стандартний блок даних і 10 ітерацій. Чому дорівнює загальна кількість біт ключів ітерації?

3. Представити число «*cb*» у шістнадцятірічній формі через бінарне і поліноміальне представлення.

4. Виконати операцію множення двох чисел «*57*» та «*83*» у шістнадцятірічній формі представлення у кінцевому полі $GF(2^8)$.

5. Зробити складання двох елементів кінцевого поля GF(2)

x^7+x^3+x+1 *і* $x^6+x^3+x^2+1$.

6. Визначити частку й залишок від ділення елемента

$x^{13}+x11+x^9+x^7+x^5+x+1$ на елемент $x^6+x^3+x^2+1$ в кінцевому полі GF(2^8).

7.Визначити мультиплікативно-обернений елемент до «*2d*» у шістнадцятірічній формі представлення у кінцевому полі $GF(2^8)$ із незвідним багаточленом $p(x) = x^8 + x^4 + x^3 + x + 1$.

8. Зробити складання двох багаточленів із коефіцієнтами кінцевого поля GF(2^8) $\{f5\}\cdot x^7+\{03\}\cdot x^3+\{09\}\cdot x+\{02\}$ і $\{07\}\cdot x^7+\{1b\}\cdot x^5+\{1e\}\cdot x+\{1f\}$. Незвідний багаточлен $p(x) = x^8+x^4+x^3+x+1$.

9. Зробити множення двох багаточленів із коефіцієнтами з кінцевого поля GF(2^8) $\{02\}\cdot x^3+\{05\}\cdot x^2+\{03\}\cdot x+\{04\}$ і $\{03\}\cdot x^3+\{05\}\cdot x^2+\{04\}\cdot x+\{01\}$. Незвідний багаточлен $p(x) = x^8+x^4+x^3+x+1$.

7.12 Завдання до самостійної роботи

1. Криптосистема ДСТУ 7624:2014 "Калина". Принцип побудови та функціонування, основні параметри криптосистеми: довжина ключів, довжина блоку вхідного тексту, кількість раундів, крипостійкість, режими роботи.

2. Провести порівняння перестановок в криптоалгоритмах DES і AES. Пояснити потребу в розширенні та стисненні перестановок в DES та їх відсутність в AES?

3. Провести порівняння раундових ключів в криптоалгоритмах DES і AES. В якому криптоалгоритмі розмір ключа раунду дорівнює розміру блока?

ПРАКТИКУМ № 8
Побудова криптографічної системи з відкритим ключем. Система Діффі–Хеллмана

Мета роботи. Ознайомитися з принципами побудови асиметричних криптосистем на прикладі реалізації системи Діффі–Хеллмана, провести зашифрування відкритого і розшифрування шифрованого повідомлення.

8.1. Основні поняття про асиметричні криптографічні системи

Асиметричні криптосистеми, також відомі як криптосистеми з відкритим ключем, є ефективними засобами криптографічного захисту даних. У таких системах для зашифрування даних використовують один ключ, який є відкритим і доступним для використання всіма користувачами системи, що шифрують дані. Розшифрування даних за допомогою відкритого ключа неможливе, оскільки для цього потрібен інший, секретний ключ. Зрозуміло, що ключ розшифрування не може бути визначений з ключа шифрування, тому що це два різні ключі [5, 7, 26]. Математичні перетворення, які лежать в основі асиметричних криптографічних систем викладені в чисельних спеціалізованих виданнях, зокрема [5, 27-29] та ін. Різноманітні задачі на цю та іншу тематику криптографічних перетворень представлені в [30].

Основним досягненням асиметричного шифрування є можливість обміну секретними повідомленнями між людьми, які не мають наперед встановленої угоди щодо безпеки. Більше не потрібно, щоб відправник та одержувач повідомлення погоджували таємний ключ через безпечний канал звязку. Криптосистеми з відкритим ключем, такі як схема Ель-Гамаля, RSA, Діффі-Геллмана і DSA (Digital Signature Algorithm, розроблений Девідом Кравіцом), є прикладами такого шифрування.

У 1975 році Уітфілд Діффі та Мартін Геллман запропонували концепцію криптографії з відкритим ключем, яка успішно вирішила проблему керування ключами. Це асиметрична схема шифрування, в якій використовуються пари ключів: відкритий ключ (***public key***), який використовується для шифрування даних, і відповідний йому закритий ключ (***private key***), який використовується для їх розшифрування. Ви можете поширювати свій відкритий ключ у всьому світі, тоді як закритий ключ залишається конфіденційним. Будь-хто, хто має копію вашого відкритого ключа, може зашифрувати повідомлення, яке може бути розшифроване тільки вами, навіть якщо ви ніколи не зустрічалися з цією людиною.

Хоча відкритий та закритий ключі математично пов'язані, на практиці обчислення закритого ключа з відкритого є неможливим. Будь-хто може зашифрувати дані, використовуючи відкритий ключ, але їх розшифрування можливе тільки за допомогою відповідного закритого ключа. Поява шифрування з відкритим ключем вважається технологічною революцією, яка забезпечила доступність стійкої криптографії для широкого загалу.

Основна ідея криптографії з відкритим ключем полягає в використанні односторонніх функцій, що представляють собою функції ***f(x)***, які просто

вираховують значення ***f(x)*** за відомим значенням x, але обернене визначення x за значенням ***f(x)*** є складним з точки зору теорії. Однак, сама по собі одностороння функція має обмежені застосування, оскільки її неможливо використовувати для розшифрування повідомлення. Тому в криптографії з відкритим ключем використовують односторонні функції з лазівкою. Лазівка - це секретний ключ, який допомагає відновити вихідне повідомлення з зашифрованого. Іншими словами, для даної функції ***f(x)***, існує таке y, що дозволяє використовувати функцію для розшифрування повідомлення, знаючи лише ***f(x)***. Це подібно до того, як можна розібрати якись електронний прилад на деталі, але потрібно мати інструкцію (схему) по складанню (лазівку), щоб зібрати його знову.

8.2. Модель криптосистеми з відкритим ключем

Асиметрична криптографія базується на використанні двох ключів - публічного для шифрування повідомлення та приватного для розшифрування його отримувачем. Публічний ключ може бути переданий через відкритий канал і його знання не дає можливості зловмисникам отримати доступ до змісту повідомлення. На рис.8.1 зображено структурну схему криптосистеми з використанням відкритого ключа [26].

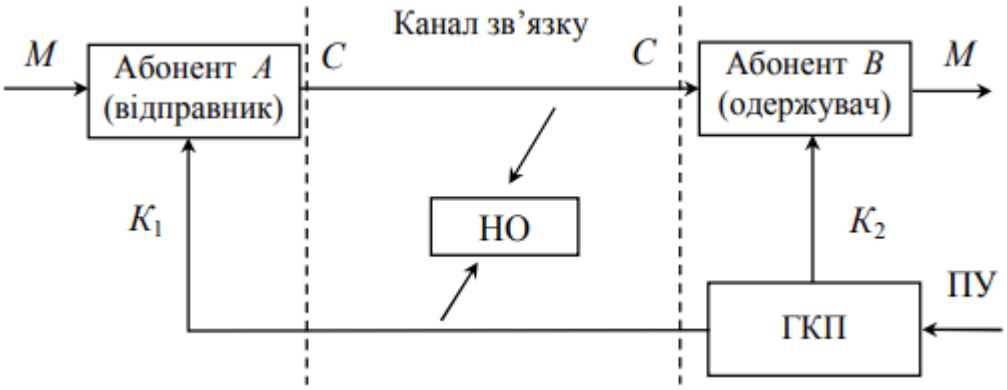


Рис.8.1 – Структурна схема криптосистеми з відкритим ключем

Генератор ключів (ГКП) створює пару ключів (К1, К2) на основі початкових умов, які відомі тільки одержувачеві повідомлення. Відкритий ключ К1 передається відправникові через незахищений канал зв'язку. Відправник використовує К1 для зашифрування повідомлення М, після чого шифротекст С передається одержувачеві через незахищений канал зв'язку. Одержувач використовує секретний ключ К2 для розшифрування криптограми і відновлення початкового повідомлення. Зловмисник (несанкціонована особа - НО) може перехопити криптограму С і відкритий ключ К1 через доступ до незахищених каналів зв'язку, але вона не може отримати ключ К2.

Найбільш відомі і поширені системи з відкритим ключем: – криптосистема ***RSA***; – ***криптосистема Ель–Гамаля*** *(El Gamal Cryptosystem)*; – криптосистема ***Діффі–Хеллмана*** *(Diffie–Hellman)*; – криптосистема,

базована на властивостях ***еліптичних кривих*** *(Elliptic Curve Cryptosystem)*; – ранцева криптосистема ***Меркле–Хеллмана*** *(Knapsack Cryptosystem)* та ін.

8.3. Криптографічна система Діффі–Хеллмана

У середині 1970-х років американські вчені Діффі та Хеллман розробили криптосистему, яка значно змінила криптографію та її практичне застосування. Ця система була першою, що дозволяла захищати інформацію без використання секретних ключів, які передавалися по захищених каналах. Щоб продемонструвати схему застосування такої системи, можна розглянути мережу зв'язку з N користувачами, де N є великим числом. Якщо використовувати звичайну систему розподілу секретних ключів, то для кожної пари абонентів потрібен свій секретний ключ, що займе значну кількість ресурсів, зокрема

$$\tilde{N}_N^2 = \frac{N(N-1)}{2} \approx \frac{N^2}{2}$$

ключів.

Наприклад, якщо ми маємо 100 абонентів, то нам потрібно згенерувати 5000 секретних ключів. А при 10000 абонентах - вже $5*10^7$ ключів. Якщо кількість абонентів зростає, то система постачання секретних ключів стає надзвичайно складною та дорогою. Діффі та Хеллман розробили систему, яка використовує відкрите поширення та обчислення ключів, щоб вирішити цю проблему. Для цього вони запропонували використовувати велике просте число p та число g ($1<g<p-1$), яке може бути використане для представлення всіх чисел з множини $\{1,2,\ldots, p-1\}$ як різні степені $g \bmod p$ (існують різні методи для знаходження такого числа g, один з яких буде описаний далі). Кожен абонент має свій секретний та відкритий ключ, що використовується для побудови системи зв'язку між абонентами A, B, C і т.д.

Нехай усі абоненти мережі знають числа p і g. Абоненти вибирають великі числа x_A, x_B, x_C, які є секретними (зазвичай, рекомендують випадковий вибір за допомогою генераторів випадкових чисел). Кожен абонент обчислює відповідне число y (y_A, y_B, y_C) і передає його іншим абонентам (відкриті числа):

$$\begin{cases} y_A = g^{x_A} \bmod p; \\ y_B = g^{x_B} \bmod p; \\ y_C = g^{x_C} \bmod p. \end{cases} \qquad (8.1)$$

Результат обчислень можна представити в таблиці 8.1:

Таблиця 8.1

Розподіл ключів серед користувачів

Абонент	Секретний ключ	Відкритий ключ
A	x_A	y_A
B	x_B	y_B
C	x_C	y_C

Припустімо, що абонент **A** бажає здійснити зв'язок з абонентом **B**, обмінюючись інформацією з таблиці 8.1. Обом абонентам доступна відкрита інформація (відкритий ключ). Абонент **A** інформує **B** по відкритому каналу про свій намір передати повідомлення. Потім абонент **A** розраховує значення:

$$Z_{AB} = (y_B)^{x_A} \bmod p .$$

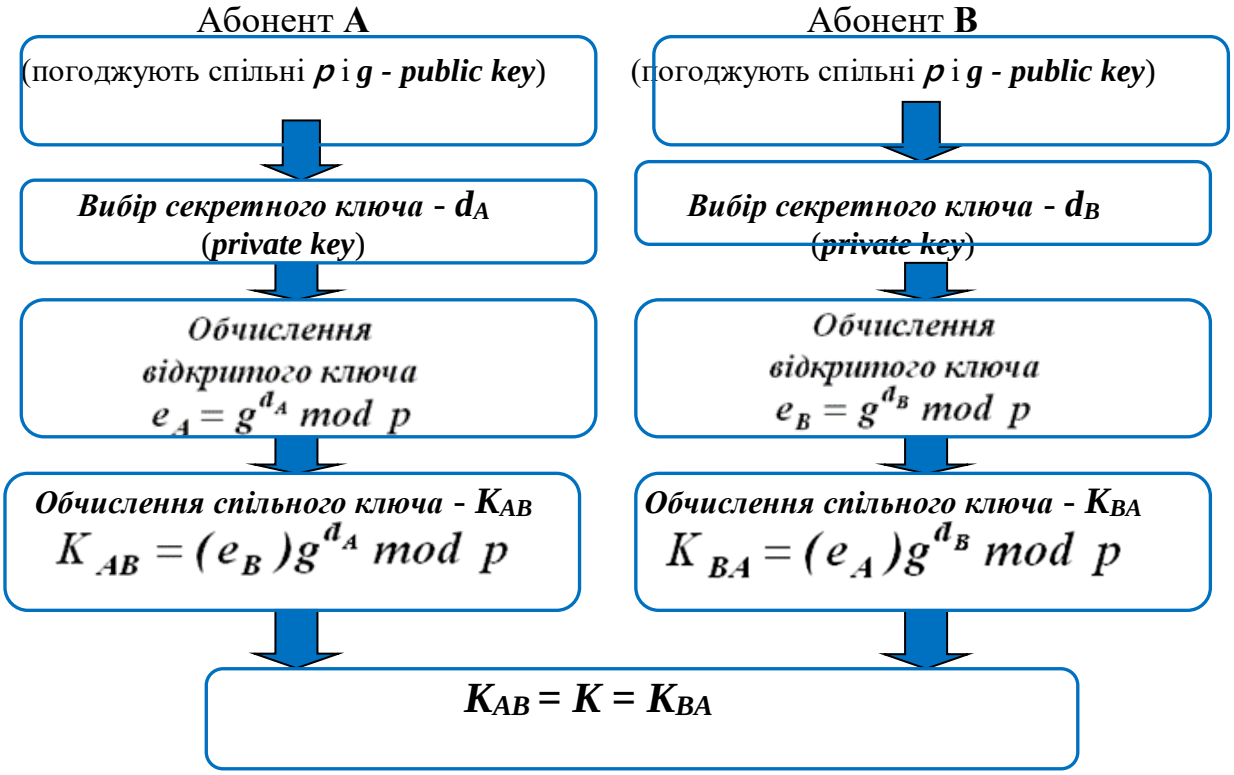
Окрім абонента **A** цього зробити не може, тому що число x_A є секретним і нікому не відомо. В своє чергу, абонент **B** обчислює свій аналогічний вираз:

$$Z_{BA} = (y_A)^{x_B} \bmod p .$$

Тоді можна стверджувати, що два вирази Z_{AB} і Z_{BA} будуть однакові, тобто відбувся обмін секретним ключами без їх безпосередньої передачі по каналам звязку:

$$Z_{AB} = (y_B)^{x_A} \bmod p = (g^{x_B})^{x_A} \bmod p =$$
$$= g^{x_A x_B} \bmod p = (y_A)^{x_B} = Z_{BA}.$$

Таким чином, криптографічну систему Діффі–Хеллмана можна представити у вигляді схеми, яка зображена на рис. 8.2.



*(обмін спільним ключем **K**)*

Рис. 8.2. Реалізація криптографічної системи Діффі–Хеллмана

Зазначимо головні властивості системи Діффі–Хеллмана:

1) абоненти **A** і **B** отримали те саме число $K_{AB} = K = K_{BA}$, яке не передавалося по відкритому каналу зв'язку;

2) зловмисник (третя особа) не знає секретних чисел d_A і d_B, і тому не може обчислити числа K_{AB} та K_{BA}.

3) Абоненти **A** і **B** можуть використати $K_{AB} = K = K_{BA}$, як **секретний ключ** для зашифрування та розшифрування даних, наприклад в симетричних криптографічних системах.

Необхідно зазначити правила вибору числа g, яке використовується в алгоритмі. При довільно заданому значенні p, це може стати складною задачею, пов'язаною з розкладанням числа p-1 на прості множники. Для забезпечення високої стійкості системи число p-1 повинно містити великий простий множник для високої криптостійкості алгоритму. Тому рекомендують вибирати просте число p так, щоб задовольнялася наступна рівність:

$$p = 2 \cdot q + 1,$$

де q — також просте число. Число g можна взяти будь-яким, для якого справедливі нерівності:

$$1 < g < p - 1 \quad \text{і} \quad g^q \bmod p \neq 1.$$

Приклад 8.1. Розглянемо приклад підбору параметрів g і p. Нехай

$$p = 39 = 2*19 + 1 \ (q=19).$$

Виберемо число g. Нехай $g = 3$ і перевіримо умову:

$$g^q \bmod p = 3^{11} \bmod 39 \neq 1,$$

а отже, g підходить.

Таким чином, можна вибрати параметри $p = 39$, $g = 3$.

Приклад 8.2. Для обміну спільним ключем кожний абонент обирає своє **секретне число** та обчислює відповідне йому відкрите число. Нехай абоненти **A** і **B** обрали секретні числа: $x_A = 7$, $x_B = 17$. Тоді, згідно виразу (8.1), отримаємо наступні **відкриті числа**:

$$y_A = 3^7 \bmod 39 = 3, \qquad y_B = 3^{17} \bmod 39 = 9.$$

Тоді спільні ключі будуь обчислюватися, як:

$$K_{AB} = (y_B)^{x_A} \bmod p = 9^7 \bmod 39 = 9,$$

$$K_{BA} = (y_A)^{x_B} \bmod p = 3^{17} \bmod 39 = 9,$$

Таким чином обидва абоненти отримали однаковий спільний ключ, який дорівнює $K = K_{AB} = K_{BA} = 9$ і ніколи не передавався по каналу зв'язку. При отриманні такого спільного секретного ключа можна шифрувати і розшифровувати повідомлення, в тому числі використовувати його, наприклад, як секретний ключ для симетричних криптоалгоритмів.

8.4. Підготовка до завдання

Ознайомитися з теоретичними положеннями задач побудови та реалізації асиметричних криптосистем на прикладі системи Діффі–Хеллмана. Пригадати елементи теорії чисел (поняття простих чисел, модульних перетворень, подільності, теорема Ферма, теорема Ейлера тощо).

8.5. Практичне завдання

1. Створити повідомлення (M) у вигляді свого прізвища.
2. Представити кожний символ повідомлення M (m_i) у вигляді ASCII (*American Standard Code for Information Interchange*) 8-бітного (бінарного) коду. Наприклад, прізвище Шевченко буде мати наступний вид:

Ш	**е**	**в**	**ч**	**е**	**н**	**к**	**о**	**←-ПІП**
152	*165*	*162*	*231*	*165*	*173*	*170*	*174*	←Десяткове представлення
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**	
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	
1	**0**	**0**	**0**	**0**	**1**	**1**	**1**	← Бінарне представлення
0	**1**	**0**	**1**	**1**	**1**	**0**	**1**	
0	**0**	**1**	**1**	**0**	**0**	**1**	**1**	
0	**1**	**0**	**1**	**1**	**1**	**0**	**0**	

   Кожна літера в даному випадку представляється як 1 байт інформації (8 біт - $b_7 \ldots b_0$).
3. Використовуючи алгоритм **Діффі–Хеллмана** створити спільний секретний ключ для шифрування Вашого прізвища та передачі інформації по каналу звязку з наступним розшифруванням. Для цього потрібно обрати прості числа ***p*** і ***g*** таким чином, щоб вони задовольняли наведеним умовам:

$$p = 2 \cdot q + 1,$$

$$1 < g < p - 1 \quad \text{і} \quad g^q \bmod p \neq 1.$$

4. Обрати два секретних числа x_A, x_B (імітація вибору двома сторонами **A** і **B** відповідно). Обрахувати за (1.1) відкриті ключі y_A, y_B.
5. Обрахувати секретні ключ Z_{AB} і Z_{BA}. Перевірити, чи співпадають вони.

6. Маючи такий спільний ключ реалізувати алгоритм шифрування і розшифрування Вашого прізвища, яке представлено в цифровій формі, наприклад, використовуючи гамування (дивись попередній практикум). Отриманий секретний ключ використати для початкової ініціалізації генератора гами.
7. Написати програми, які реалізують функцію шифрування і розшифрування за п.1-6.
8. Надати аналіз отриманим результатам.

8.6. Зміст протоколу роботи

Протокол до практикуму оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

8.7. Контрольні питання для самоперевірки

1. Які числа називають простими і складеними? Навести приклади простих і складених чисел.
2. Які числа називають взаємнопростими та не взаємнопростими? Навести приклади таких чисел.
3. Що таке мультиплікативно зворотнє числа $\boldsymbol{x}$ за модулем $\boldsymbol{n}$? Яка умова існування мультиплікативно зворотнього числа? Навести приклади існування і знаходження таких чисел.
4. Яка повинна виконуватися умова, якщо числа $\boldsymbol{a}$ і $\boldsymbol{b}$ можна порівняти за модулем $\boldsymbol{n}$?
5. Дати визначення функції Ейлера. Основні методи знаходження функції Ейлера.
6. На чому базується криптостійкість більшості асиметричних криптоалгоритмів?
7. Що таке односпрямовані функції (one-way function)? Область застосування.
8 Що таке односпрямовані функції з секретом? Які основні властивості застосування односпрямованих функцій в криптографії?
9. Що таке факторизація чисел? Область застосування факторизації чисел.
10. Наведіть і поясніть схему криптосистеми з відкритим ключем, принцип роботи.
11. Яке основне призначення асиметричних криптографічних алгоритмів? Область застосування алгоритму Діффі–Хеллмана.
12. Чому алгоритм Діффі–Хеллмана називають асиметричним криптографічним алгоритмом?
13. Навести послідовність дій, які виконуються при застосуванні алгоритму Діффі–Хеллмана.
14. Які характерні вразливості для алгоритму Діффі–Хеллмана з точки зору атак?

8.8. Задачі до практикуму №8

1. Обчислити вирази за заданим модулем: 7; 11; 25; -4; -17; 7 + 8; 5 – 9:
   а) за модулем 9; б) за модулем 15.
2. Розкласти числа на прості множники: 66; 77; 100; 137.
3. Визначити, які з пар чисел: (5, 7); (5, 10); (24, 32); (21, 31) *є* взаємно простими?
4. Яке число є результатом обчислення виразу: $\left(\frac{13}{11}\right) mod 27$?
5. Яке число є результатом обчислення виразу: 7^{-2} mod 13?
6. Обчислити, яке число є мультиплікативно оберненим числа 11 за модулем 13?
7. При використанні основних теорем теорії чисел визначити, які числа є результатом обчислення виразів:
   а) 3^{14} mod 21; б) 7^{10} mod 13; в) 2^{7} mod 10*;* г*)* 7^{19} mod 100?
8. Знайти значення функції Ейлера *φ(10), φ(15), φ(29).*
9. При застосуванні теореми Ферма, обчислити наступні вирази: 3^{13} *mod 13,* 5^{22} *mod 11.*
10. При застосуванні теореми Ейлера, обчислити наступні вирази: 2^{14} *mod 21;* 2^{107} *mod 159.*
11. Згенерувати спільний ключ K при застосуванні алгоритму Діффі–Хеллмана, якщо задані наступні відкриті параметри ***p*** та ***g***, а також секретні ключі x_A та x_B:
а) p=19, g = 2, x_A = 7, x_B = 10;
б) p = 23, g = 7, x_A = 11, x_B =9.

8.9. Завдання до самостійної роботи

1. Ранцева криптосистема Меркле–Хеллмана.
2. Алгоритм Рабіна та його застосування.
3. Алгоритм Шаміра та його застосування.
4. Алгоритм Вільямса.

ПРАКТИКУМ № 9
Побудова та дослідження асиметричної криптографічної системи RSA та Ель Гамаля (ЕГ)

Мета роботи. Ознайомитися з принципами побудови асиметричних криптосистем на прикладі реалізації *RSA* та *ЕЛЬ ГАМАЛЬ*, провести зашифрування відкритого і розшифрування шифрованого повідомлення.

9.1. Криптографічна система RSA

Одним з найбільш широко використовуваних криптографічних систем є RSA, яка отримала свою назву на честь її розробників: Рональда Рівеста (Ron **R**ivest), Аді Шаміра (Adi **S**hamir) та Леонарда Аделмана (Leonard **A**ddleman) [5, 27].

RSA - це криптографічний алгоритм, який використовує відкритий ключ і ґрунтується на складності обчислень, пов'язаних з факторизацією великих цілих чисел. Факторизація цілого числа - це процес розкладання заданого числа на прості множники, подібно до розкладу оператора на добуток кількох операторів нижчого рівня. наприклад: 15=5*3.

Криптосистема RSA базується на використанні односторонньої функції, яка формує добуток двох великих простих цілих чисел. Розкладання такого великого числа на прості множники є значно складнішим процесом. Хоча це можливо зробити, проте витрати ресурсів, таких як час або обчислювальна потужність комп'ютерів, значно перевищують простий перебір всіх можливих комбінацій ключів. Однак наявність гарантованої криптостійкості алгоритму забезпечує сприятливі умови для поширення криптосистеми RSA порівняно з іншими алгоритмами. Це одна з причин, чому RSA стала популярною у багатьох системах для роботи з віддаленими клієнтами (див. рис. 9.1) [31].

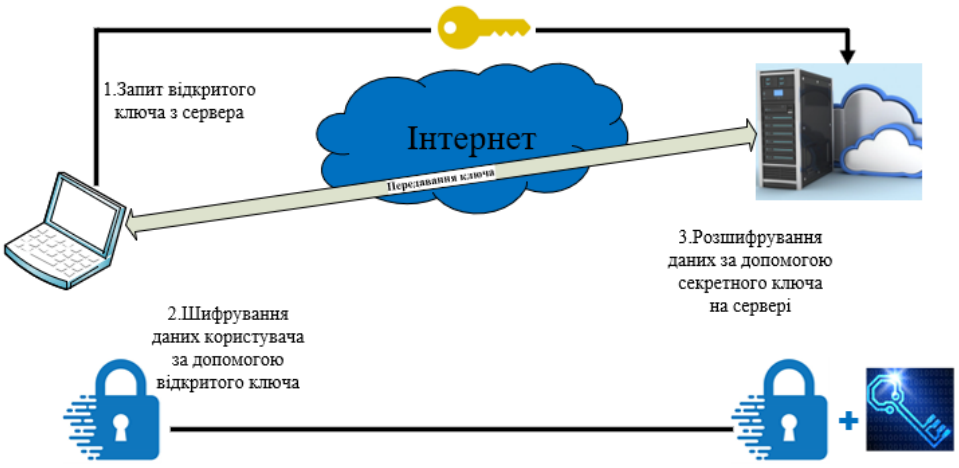


Рис.9.1. Принципи функціонування системи RSA.

Концепція криптографії з відкритим ключем тісно пов'язана з ідеєю односторонніх функцій, які мають таку властивість, що для даного значення аргумента «x» легко обчислити значення функціїї «f(x)», але неможливо знаходити зворотне значення «x» за розумний проміжок часу, виходячи зі

значення функціїї «f(x)». Однак сама одностороння функція має обмежене застосування, оскільки з нею можна зашифрувати повідомлення, але розшифрування стає складним процесом або взагалі неможливим. Тому криптографічні системи з відкритим ключем використовують односторонні функції з так званою "лазівкою" (trapdoor function). "Лазівка" представляє собою секрет, який допомагає розшифрувати повідомлення. Іншими словами, існує спеціальне значення «y», при знанні якого та функції «f(x)» можна відновити значення «x». Наприклад, якщо розібрати якись пристрій на окремі вузли та компоненти, то потім буде достатньо проблематично зібрати цей самий пристрій знову. Проте, якщо пронумерувати всі вузли та компоненти перед розбиранням та записати послідовність розбирання відповідно до нумерації, можна буде легко відновити цей самий пристрій, використовуючи записану послідовність та зворотну нумерацію.

Проілюструємо принцип дії такого підходу на прикладі збереження паролів у комп'ютерній системі. Кожен абонент в мережі має власний пароль, який він використовує при вході та вказує своє ім'я та вводить цей таємний пароль. Однак, якщо пароль зберігати в памяті комп'ютерної системи, то існує ризик, що хтось його зможе прочитати і отримати доступ до конфіденційної інформації. Для вирішення цієї проблеми використовується одностороння функція. Під час створення таємного пароля комп'ютер зберігає не сам пароль, а результат обчислення функції від цього пароля та імені користувача.

Одностороння функцією з "*лазівкою*" базується на наступному:

- перевірка числа на простоту - це відносно просте завдання;
- розкладання числа ***n*** на множники у вигляді добутку ***p*** і ***q*** (де ***p*** і ***q*** - прості числа) є надзвичайно складною задачею, особливо якщо ми знаємо лише ***n***, а ***p*** і ***q*** - великі числа. Такий розклад відомий, як задача *факторизації*.

Розглянемо алгоритм реалізації даної криптосистеми.

1. Нехай в криптографічній системі присутні абоненти ***A***, ***B***, ***C*** і так далі. Кожен з них випадковим чином вибирає два великі прості числа ***p*** і ***q*** (не менше 128 біт). Після цього абонент обчислює число, яке ***відкрите*** для інших

$$n = p \cdot q.$$

Знайдене число ***n*** *буде* відкритою інформацією, яка доступна для всіх інших абонентів. Далі абоненти *A, B, C* обчислюють *функцію Ейлера* від ***n:***

$$\varphi(n) = (p-1)\cdot(q-1)$$

і вибирають такі числа ***e***, які задовольняють умовам:

$$e < \varphi(n),\ НСД(e, \varphi(n)) = 1 \qquad (9.1)$$

і за узагальненим алгоритмом Евкліда знаходять мультиплікативно зворотні числа ***d:***

$$(d \times e) \bmod \varphi(n) = 1. \qquad (9.2)$$

Приклад 9.1. Візьмемо $p = 7$, $q = 11$. Знайдемо ***модуль*** криптосистеми

$$n = p \times q = 77.$$

Обчислимо функцію Ейлера для цього модуля з врахуванням, що p і q – прості числа:

$$\varphi(n) = (p - 1)(q - 1).$$

Таким чином, *функція Ейлера* дорівнює: $\varphi(77) = 6 \times 10 = 60$.

2. Згенеруємо ціле число, яке задовольняє умові (9.1), наприклад $e = 13$. Тоді, пара чисел **(*e, n*)** буде ***відкритий ключем*** криптосистеми. Для нашого випадку це будуть числа е пара **(13, 77).**

3. Обчислимо ***секретний ключ d***. Для цього потрібно розв'язати рівняння:

$$(d \times e) \bmod \varphi(n) = 1.$$

Відповідно до обчислень мультиплікативно зворотнього числа отримаємо:

$$d = (1 + k \times \varphi(n))/e,$$

де k – ціле число. Для даного прикладу простим перебором отримуємо, що дана рівність буде задовольнятися для $k = 8$, де $d = 37$.

Отже, ***секретним ключем***, парним до відкритого ключа **(13, 77)** буде пара чисел

$$(d, n) \rightarrow (37, 77).$$

Отриману інформацію, яка пов'язана з абонентами, їх ключами, надано в табл.9.1

Таблиця 9.1

Інформація про ключі абонентів при реалізації системи RSA

Абонент	***Відкритий ключ***	***Секретний ключ***
A	(e_A, n_A)	d_A
B	(e_B, n_B)	d_B
C	(e_C, n_C)	d_C

Примітка –нижній індекс при ключі вказує на його приналежність до відповідного абонента.

Опис протоколу RSA.

Припустимо, що абонент ***A*** бажає надіслати повідомлення M абоненту ***B***, де M вважається числом, яке задовольняє нерівності: $M < n_B$. Запишемо послідовність дій у вигляді таких кроків.

1. Абонент ***A*** шифрує повідомлення за наступним виразом

$$C = (M^{e_B}) mod\ n_B \qquad (9.3)$$

і пересилає дане зашифроване повідомлення (*криптотекст*) по відкритій лінії абоненту ***B***.

2. Абонент ***B,*** який отримав повідомлення ***C*** (*криптотекст*), обчислює вираз:

$$M^* = (C^{d_B}) mod\ n_B \qquad (9.4)$$

Запишемо деякі властивості протоколу RSA:

1) для описаного протоколу передачі даних $M^* = M$, тобто абонент ***B*** отримує від абонента ***A*** теж саме повідомлення;

2) протокол забезпечує зашифрування й розшифрування повідомлень M коректно;

3) зловмисник, який перехоплює зашифровані повідомлення (криптотекст ***C***) і знає відкриту інформацію **(*e, n*)**, не зможе знайти повідомлення M за великих значень параметрів ***p*** і ***q***.

Для доведення першої властивості скоритаємося тим, що вираз (9.2) є еквівалентним для деякого цілого числа k наступному рівнянню :

$$e_B\, d_B = k\, \varphi_B + 1.$$

Тоді має місце наступна рівність:

$$M^* = (M^{e_B d_B}) mod\ n_B = \left(M^{k\varphi(n_B)+1}\right) mod\ n_B = M$$

Для доведення другої властивості відмітимо, що зловмиснику відомі тільки відкриті параметри ***n*** і ***e***. Для знаходження секретного числа ***d*** він повинен знати значення функціїї Ейлера ***φ(n) = (p–1)(q–1)***, що в свою чергу потребує знання простих чисел ***p*** і ***q***. Але отримати (розкласти) з числа ***n*** множники ***p*** і ***q*** є складним завданням для великих значень за прийнятний час.

Показано [5], що одностороння функція $y = x^d\ mod\ n$, що застосовується в системі *RSA*, має так звану «*лазівку*», яка дозволяє досить легко обчислити обернену функцію

$$x = \left(\sqrt[d]{y}\right)\ mod\ n.$$

при умові відомого розкладання ***n*** на прості множники. Дійсно, легко обчислити значення функції Ейлера ***φ(n) = (p–1)(q–1)***, а потім знайти

$$d = e^{-1} mod\ \varphi(n).$$

Якщо ***p*** і ***q*** невідомі, то обчислення оберненої функції є досить важкою задачею за прийнятний час, а знайти ***p*** і ***q*** за відомим ***n*** досить складно. Таким чином, знання параметрів ***p*** і ***q*** є так званою «лазівкою».

Необхідно відмітити, що за даною схемою шифрування важливо, щоб кожний абонент вибирав свою власну пару простих чисел ***p*** і ***q***, що призведе до різних модулів системи $\boldsymbol{n_A, n_B, n_C}$ (щоб інший абонент не зміг прочитати повідомлення, які для нього не призначені). При цьому, другий відкритий параметра ***e*** може бути однаковим для всіх абонентів. Рекомендується вибирати ***e***=*3* (при відповідному виборі параметрів ***p*** і ***q***). В цьому випадку зашифрування буде виконуватися досить швидко, лише за дві операції множення.

Блок схема криптографічної системи *RSA* передачі повідомлень від абонента ***A*** до абонента ***B*** показана на рис. 9.2. В цьому випадку оригінальне повідомлення M, яке шифрується і передається, збігається з результатом розшифрування M^*.

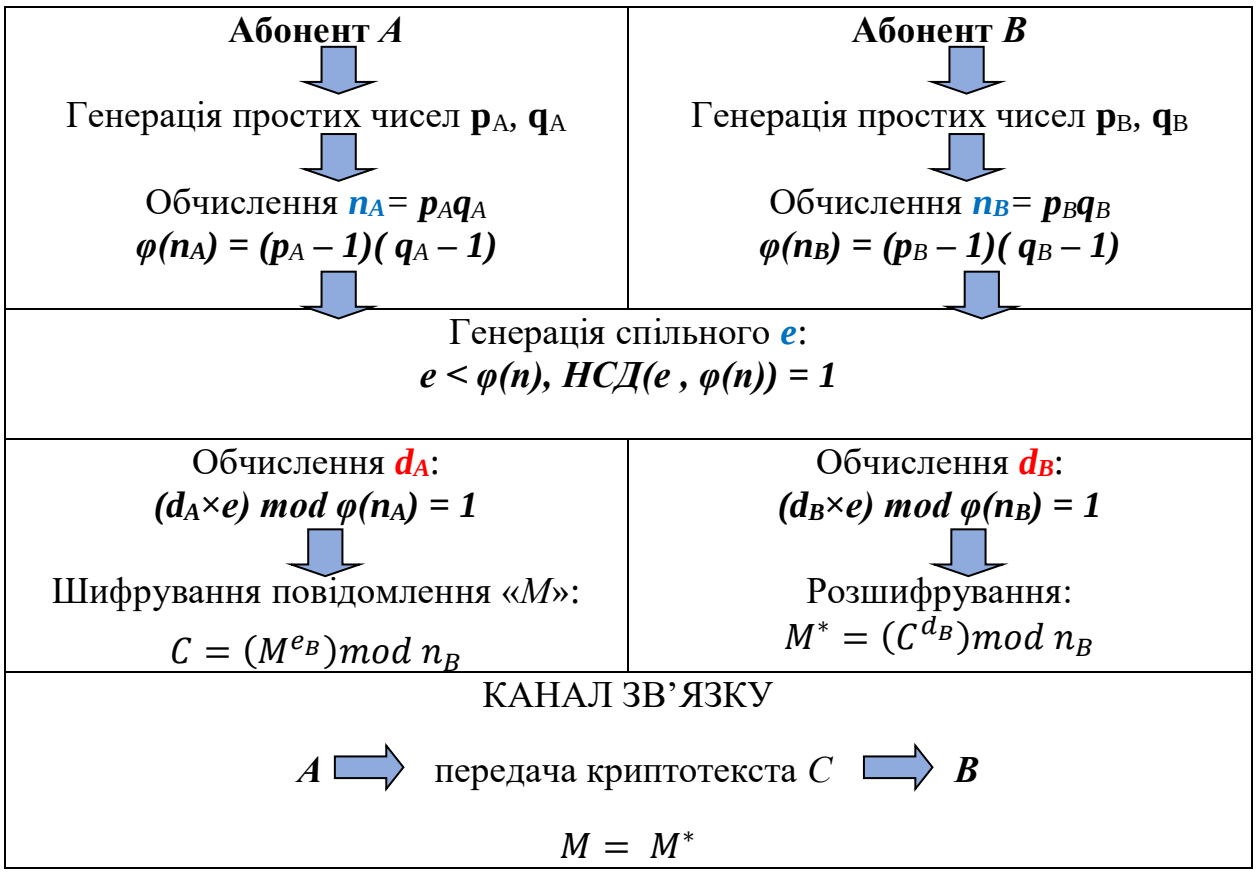


Рис. 9.2. Схема криптографічної системи RSA передачі повідомлення від абонента **A** до абонента **B.**

Приклад 9.2. Нехай абонент ***A*** хоче передати абоненту ***B*** повідомлення $M = 6$. Згідно алгоритму, абонент ***B*** обирає параметри, нехай: $\mathbf{p}_B=7$, $\mathbf{q}_B=17$. Визначимо модуль криптосистеми:

$$n_B = p_B q_B = 7*17 = 119.$$

Визначимо функцію Ейлера:

$$\varphi(n_B) = (p_B - 1)(q_B - 1) = (17-1)*(7-1) = 96.$$

Оберемо відкритий параметр ***e***, наприкла $e = 5$.

Визначимо ***секретний ключ*** d_B. Відповідно до обчислень мультиплікативно зворотнього числа отримаємо:

$$d = (1 + k \times \varphi(n))/e:$$

$k = 1$: $d_B = (1 + 1 \times 96))/5 = 19.4;$
$k = 2$: $d_B = (1 + 2 \times 96))/5 = 38.6;$
$k = 3$: $d_B = (1 + 3 \times 96))/5 = 57.8;$
$k = 4$: $d_B = (1 + 4 \times 96))/5 = 77.$

Визначено значення секретного ключа $d_B = 77$, тому що результатом обчислення є ціле число.

Таким чином маємо, що відкритим ключем є пара чисел:

$$(e_B, n_B) = (5, 119),$$

а секретним ключем:

$$(d_B, n_B) = (77, 119).$$

Зашифруємо повідомленн $M = 6$. Для цього скористаємося виразом (9.3) та отримаємо значення криптотекста, яке шифрується та передається від абонента ***A*** до абонента ***B***:

$$C = 6^5 \ mod\ 119 = 41.$$

Абонент ***B***, який прийняв криптотекст C = 41, буде розшифровувати його при застосуванні виразу (9.4):

$$M^* = 41^{77} mod\ 119 = 6$$

Таким чином, абонент ***B*** отримав таке саме значення, яке передавав абонент ***A***. В результаті відбувся обмін секретною інформацією (ключем) без беспосередньої передачі цієї інформації по каналу зв'язку і абонент ***B*** розшифрував повідомлення абонента ***A***.

Зазначимо, що криптосистема RSA симетрична відносно застосування парних ключів, а саме:

- можна зашифровувати інформацію на відкритому ключі ***(e, n)*** та розшифровувати на секретному ***(d, n)*** – так як наведено в прикладі 9.2;
- зашифровувати інформацію на секретному ключі ***(d, n)***, а розшифровувати на парному до нього відкритому ключі ***(e, n)***.

Система RSAпризначена, в тому числі, для шифрування блочних повідомлень.

Приклад 9.3. Зашифруємо абревіатуру **UKR** через блокове представлення алгоритму RSA. Для цього літери **U**, **K** та **R** представимо у вигляді пятирозрядних двійкових векторів, скориставшись двійковим записом їх порядкових номерів у англійській абетці (див. рис.1.19):

$$\mathbf{U} = 20_{10} = 10100_2;\ \mathbf{K} = 10_{10} = 01010_2;\ \mathbf{R} = 17_{10} = 10001_2.$$

Нехай візьмемо числа, як в наведеному вище прикладі 9.2: ***p***=7, ***q***=17. Тоді модуль криптосистеми ***n***= ***pq*** = *7*17 = 119*, а функція Ейлера ***φ(n)*** = *96*.

Можна обєднати всі двійкові послідовності в одне число, яке і буде використовуватися для шифрування та передачі. В цьому випадку отримаємо

$$101000101010001_2 = 20817_{10}.$$

Але такий підхід буде не коректним і не робочим, оскільки має виконуватися вимога:

$$M < n.$$

Така вимога не виконується, оскільки M = *20817*, а ***n***=*119*. В цьому випадку потрібно або збільшувати ***n***, або ділити M на блоки. Наприклад, візьмемо в якості блокового представлення пять розрядів для кожного блока, так як і представлені літери:

$$\mathbf{UKR} = (10100), (01010), (10001) = (M_1 = 20, M_2 = 10, M_3 = 17).$$

Умова $M < \boldsymbol{n}$ виконується, отже можемо застосувати даний алгоритм. Розмір блоків можна змінювати, наприклад збільшувати (може знаходитися в інтервалі $\overline{0, \boldsymbol{n}}$). В цьому випадку кількість переданих цифр може бути меншою від кількості оригінального повідомлення. Наприклад, можна було б створити блоки розміром по 8 розрядів. Тоді дане повідомлення буде перетворене на наступне:

UKR = {10100), (01010), (10001)} = {$(10100010)_2 = 162_{10}$, $(10100010)_2 = 162_{10}$},

де останній біт є додатковим бітом для формування повного восьмирозрядного блоку.

Тоді замість трьох цифр ($M_1 = 20$, $M_2 = 10$, $M_3 = 17$) потрібно було б передати тільки дві ($M_1 = 162$, $M_2 = 162$). Причому, дві однакові цифри при передачі не означають дублювання змісту повідомлення. Але в даному випадку умова $M<\boldsymbol{n}$ не виконується, отже ділення на такі блоки не може бути реалізовано при заданому $\boldsymbol{n}$. При збільшення $\boldsymbol{n}$ повідомлення може бути зашифроване і передано в такий спосіб. Для модельного представлення, залишимо пятирозрядні блоки для передачі загального повідомлення.

Оберемо відкритий параметр $\boldsymbol{e}$**,** наприкла $\boldsymbol{e} = 5$ та визначимо секретний ключ $\boldsymbol{d} = 77$. Зашифруємо повідомлення M_1, M_2 та M_3:

$$C_1 = (M_1^e) mod\ n = \ 20^5 mod\ 119 = 90,$$

$$C_2 = (M_2^e) mod\ n = \ 10^5 mod\ 119 = 40,$$

$$C_3 = (M_2^e) mod\ n = \ 17^5 mod\ 119 = 68.$$

В результаті шифрування отримали шифротекст, який передається від абонента ***A*** до абонента ***B***:

$$\mathrm{C} = (90, 40, 68):\ (\mathrm{C}_1 = 90, \mathrm{C}_2 = 40, \mathrm{C}_3 = 68).$$

Розшифруємо дане повідомлення на сторні абонента ***B:***

$$M_1^* = \left(C_1^d\right) mod\ n = \ (90^{77}) mod\ 119 = 20\ ,$$

$$M_2^* = \left(C_2^d\right) mod\ n = \ (40^{77}) mod\ 119 = 10\ ,$$

$$M_3^* = \left(C_3^d\right) mod\ n = \ (68^{77}) mod\ 119 = 17\ ,$$

Після представлення в двійковому коді набору чисел:

$$(M_1^* = 20, M_2^* = 10, M_3^* = 17) = (10100), (01010), (10001)$$

отримаємо повідомлення, яке відповідає оригінальному:

U = 10100; **K** = 01010; **R** =10001.

Наведений *приклад 9.2.* для зручності подання представимо у вигляді таблиці 9.2., де показані етапи алгоритму (генерація ключів, шифрування та розштфрування, аналіз оригінального і розшифрованого повідомлення), короткий опис основних операцій алгоритму, результати обчислення.

Таблиця 9.2.

Алгоритм шифрування та розшифрування даних в системі RSA

Етап алгоритму	Опис основних операцій	Результат обчислення
Генерація ключів	Обрати два простих числа	$p_B=7$, $q_B=17$
	Обчислити добуток (модуль криптосистеми)	$n_B = p_B q_B = 7*17 = 119$
	Обчислити функцію Ейлера	$\varphi(n_B) = (p_B - 1)(q_B - 1) = (17-1)*(7-1) = 96$
	Обрати відкриту експоненту	$e = 5$
	Обчислити секретну експоненту	$d_B = 77$
	Опублікувати відкритий ключ	$(e_B, n_B) = (5, 119)$
	Записати секретний ключ	$(d_B, n_B) = (77, 119)$
Шифрування	Обрати оригінальний текст для шифрування	$M = 6$
	Обчислити криптотекст	$C = (M^{e_B}) mod\ n_B$ $C = (6^5) mod\ 119 = 41$
Розшифрування	Розшифрувати вхідне повідомлення	$M^* = (C^{d_B}) mod\ n_B$ $M^* = 41^{77} mod\ 119 = 6$
Аналіз повідомлень	Порівняння M і M^*	$M = M^* = 6$

Для зручності роботи з асиметричними алгоритмами та виконанням математичних перетворень наведена коротка інформація про основні співвідношення і теореми, які допомагають спростити обчислення (табл.9.3.).

Таблиця 9.3.

Основні співвідношення і теореми

Назва функції (теореми)	Співвідношення
Значення функції Ейлера для простого числа n	$\varphi(n) = n - 1$
Значення функції Ейлера від аргумента n ($n=pq$, де p, q - прості числа)	$\varphi(n) = (p - 1)(q - 1)$
Значення функції Ейлера від аргумента p в степені k(де p - просте число)	$\varphi(p^k) = p^k - p^{k-1}$
Значення функції Ейлера від аргумента p – ціле додатнє число з розкладанням на прості множники, n – всі прості числа, які є дільниками числа p.	$p = n^{\alpha_1} n^{\alpha_2} n^{\alpha_3} \dots n^{\alpha_n}$ $\varphi(p) = \prod_{i=1}^{t} \left[n_i^{\alpha_{i-1}} (n_i - 1) \right] = p \prod_{i=1}^{t} \left(1 - \frac{1}{n_i} \right)$
Теорема Ейлера	$a^{\varphi(b)} mod\ b = 1$, де a і b взаємно прості числа
Теорема Ферма	$a^{p-1}\ mod\ p = 1$, де p – просте число
Деякі співвідношеня при виконанні модулярних обчислень	$(a + b) mod\ n = \big((a\ mod\ n) + (b\ mod\ n)\big) mod\ n$ $(a * b) mod\ n = \big((a\ mod\ n) * (b\ mod\ n)\big) mod\ n$ $(a\ mod\ n) k\ mod\ n = (a * k)\ mod\ n$
Знаходження мультипликативно зворотнього числа	$(A * A^{-1}) mod\ n = 1$ $A^{-1} = (1 + i * n)/A$

9.2. Криптографічна система Ель Гамаля (EG)

Схема Ель-Гамаля (*Elgamal*) - це криптосистема з відкритим ключем, яка базується на складності обчислення дискретних логарифмів у кінцевому полі. Вона складається з алгоритму шифрування і алгоритму цифрового підпису.

Ця криптосистема була створена *Тахером Ель-Гамалем* у 1984 році. Ель-Гамаль розробив одну з варіацій алгоритму Діффі-Хеллмана, вдосконаливши його і створивши два алгоритми, які використовуються для шифрування та автентифікації. На відміну від RSA, алгоритм Ель-Гамаля не був запатентований, тому став доступною та більш економічною альтернативою, оскільки не потребував плати за ліцензію. Вважається, що цей алгоритм підпадає під дію патенту Діффі-Хеллмана.

Роботу алгоритму можна представити у вигляді наступних етапів.

Генерація ключів:

1. Генерується просте випадкове число p;
2. Вибирається число g, для якого виконуються умови:

   $1<g<p-1$ та $g^{p-1}\ mod\ p = 1$;
3. Вибирається число x, яке: $1<x<p-1$;
4. Обчислюється значення $y = g^x\ mod\ p$.
5. *Відкритими* даними є p, g, y.
6. *Закритим* ключем є x.

Шифрування повідомлення M:

1. Вибирається сесійний ключ – випадкове число k, яке $1<k<p-1$;
2. Обчислюються значення $a = g^k \bmod p$ та $b = y^k M \bmod p$.
3. Отримана пара чисел (a, b) є криптотекстом: $C = (a, b)$.

Розшифрування криптотекста C:

1. $M = b(a^x)^{-1}\ mod\ p$; або
2. $M = ba^{(p-1-x)}\ mod\ p$.

Приклад 9.4.

1. Нехай для передачі повідомлення абонентом ***A*** для абонента ***B*** потрібно зашифрувати $M = 10$.
2. Згенеруємо ключі шифрування. Нехай p=11, g=5.
3. Оберемо закритий ключ x=7 – випадкове ціле число x, для якого виконується умова $1<x<p-1$.
4. Обчислимо $y = g^x\ mod\ p = 5^7\ mod\ 11 = 3$.
5. Таким чином, відкритими даними є числа: $(p, g, y) = (11, 5, 3)$, закритим ключем є число x=7.
6. Для шифрування вибираємо випадковий сесійний ключ: k=9 таке, що $1<k<p-1$.

7. Обчислюємо $a = g^k \bmod p = 5^9 \bmod 11 = 9$.
8. Обчислюємо $b = y^k M \bmod p = 3^9 * 10 \bmod 11 = 7$.
9. Таким чином, утворена пара чисел C = (**9, 7**) є шифротекстом при секретному ключі ***x*=7**.
10. Для дешифрування повідомлення абонент обчислює M за формулою:

$$M = ba^{(p-1-x)} \; mod \; p = 7 * 9^{(11-1-7)} \; mod \; 11 = 10.$$

Таким чином, отримали початкове повідомлення M=10.

9.3. Підготовка до завдання

Ознайомитися з теоретичними положеннями побудови та реалізації асиметричних криптосистем на прикладі систем RSA та Ель Гамаля. Пригадати елементи теорії чисел (поняття простих чисел, модульних перетворень, подільності, теорема Ферма, теорема Ейлера тощо).

9.4. Практичне завдання

1. Створити повідомлення (M) у вигляді свого прізвища.
2. Представити кожний символ повідомлення M (m_i) у вигляді ASCII (*American Standard Code for Information Interchange*) 8-бітного (бінарного) коду. Наприклад, прізвище Шевченко буде мати наступний вид:

Ш	**е**	**в**	**ч**	**е**	**н**	**к**	**о**	**←-ППП**
152	*165*	*162*	*231*	*165*	*173*	*170*	*174*	←Десяткове представлення

Кожна літера в даному випадку представляється в десяткові формі.
3. Виконання лабораторної роботи передбачає роботу в команді по 2 особи (або імітація роботи двох абонентів). Використовуючи алгоритм **RSA** кожна команда задається (обирає) простими числами ***p*** *і* ***q*** .
4. Обраховується модуль криптосистеми ***n*= *pq***.
5. Обчислюється функція Ейлера для цього модуля: ***φ(n)*** і генерується **спільний ключ e,** який є взаємно простим як з **n**, так і з **φ(n)**. Таким чином формується ***відкритий ключ*** криптосистеми у вигляді пари чисел **(*e, n*).**
6. Студенти в команді обмінюються відкритими ключами, тобто парами чисел **(*e, n*).**
6. Кожен в команді обчислює ***секретний ключ d*** (їх буде 2: d_A і d_B, секретні ключі НЕ РОЗГОЛОШУЮТЬСЯ НІКОМУ)**,** парний до обраного відкритого, де формується відповідна пара чисел **(d_A, *n*)** і **(d_B, *n*).**
7. Провести шифрування повідомлення за виразом (9.3) (шифрується окремо кожна літера у форматі ASCII, представлена десятковим числом) і передати один одному в команді для розшифрування.
8. Провести розшифрування повідомлення за виразом (9.4) і порівняти з оригінальним повідомленням свого колеги.
9. Написати програму, яка реалізує функцію шифрування і дешифрування по п.1-8.

10. Провести шифрування та розшифрування свого прізвища за алгоритмом Ель-Гамаля.
11. Надати аналіз отриманим результатам.

9.5. Зміст протоколу роботи

Протокол до виконаної практичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

9.6. Контрольні питання для самоперевірки

1. Яке прикладне призначення алгоритмів RSA та Ель Гамаля?
2. Описати послідовність дій для шифрування та розшифрування повідомлень при застосуванні алгоритму RSA.
3. Описати блоковий принцип передачі повідомлень при застосуванні алгоритму RSA. З чим пов'язаний цей принцип?
4. Що потрібно зробити, аби зменшити кількість зашифрованих цифр при передачі повідомлень. Відповідь обгрунтуйте на вище розглянутому прикладі при шифруванні вашого прізвища алгоритмом RSA.
5. Описати послідовність дій для шифрування та розшифрування повідомлень при застосуванні алгоритму Ель Гамаля.
6. На чому базується криптостійкість алгоритмів RSA та Ель Гамаля? Відповідь обгрунтуйте.
7. Що таке односторонні функції з "лазівкою"? Опишіть практичне застосування таких функцій.
8. Що таке мультиплікативно обернене число за заданим модулем? Яка умова існування мультиплікативно оберненого числа? Навести приклади існування і знаходження таких чисел.
9. Дати визначення функції Ейлера, навести приклади.
10. Основні методи знаходження функції Ейлера.

9.7. Задачі до практикуму №9

1. Знайти значення функції Ейлера $\varphi(5)$, $\varphi(10)$. $\varphi(15)$ і $\varphi(30)$ різними способами, описаними в таблиці 9.3, та порівняти результати.
2. Обчислити вирази $3^{11} mod\ 11$, $7^{33} mod\ 11$ і $3^{19} mod\ 5$ при застосуванні теореми Ферма.
3. За допомогою алгоритму Евкліда знайти *НСД(27, 12), НСД (20, 12), НСД (26, 50)* і *НСД (37,14)*.
4. Для криптосистеми Ель-Гамаля із заданими параметрами p, g і k задатися відсутніми параметрами і описати процес передачі повідомлення M абоненту $\boldsymbol{B}$ від абонента $\boldsymbol{A}$ у вигляді криптотекста з подальшим розшифруванням:

а) $p = 23,\ g = 5,\ k = 11,\ M = 15$;

б) $p = 19,\ g = 2,\ k = 5,\ M = 7$;

в) $p = 11$, $g = 2$, $k = 5$, $M = 10$

5. Для алгоритму *RSA* з заданими параметрами абонента ***A*** (p_A, q_A і d_A) задатися відсутніми параметрами абонента ***B*** та описати процес отримання спільного ключа:

а) $p_A = 5$, $q_A = 11$, $d_A = 3$, $M = 12$;

б) $p_A = 5$, $q_A = 13$, $d_A = 5$, $M = 20$.

6. Розшифруйте повідомлення 1552 - 352, отримане при шифруванні за допомогою криптосистеми RSA, модуль якої n=1739, а закрита експонента d=5 (абетка українська).

7. Використовуючи метод *"крок немовляти, крок велетня"*, вирішити такі рівняння:

а) $2^x \bmod 37 = 12$;

б) $6^x \bmod 41 = 21$;

9.8. Завдання до самостійної роботи

1. Криптостійкість алгоритму RSA.
2. Порівняльний аналіз асиметричних криптографічних систем.
3. Принципи розповсюдження ключів.
4. Методи зламу криптографічних систем. Метод «Крок немовляти, крок велетня».

ПРАКТИКУМ № 10
Дослідження властивостей асиметричних криптоперетворень в групі точок еліптичних кривих

Мета роботи. Ознайомитися з принципами побудови асиметричних криптосистем на прикладі криптоперетворень в групі точок еліптичних кривих та реалізувати процедури обміну ключової інформації між двома абонентами.

10.1. Вступ до криптографії на основі еліптичних кривих

Більшість продуктів і стандартів, у яких для шифрування і цифрових підписів застосовується метод криптографії з відкритим ключем, базується на застосуванні криптосистм RSA, Ель-Гамаля та ін. Наприклад, кількість бітів ключа, яка необхідна для надійної захищеності криптосистеми RSA, за останні роки різко зросла. Це обумовило відповідне зростання завантаження систем у додатках, які використовують RSA. Це породило множину проблем, особливо для вузлів, що спеціалізуються на електронній комерції, де потрібен захист великої кількості транзакцій.

Алгоритм RSA десятиліттями добре служив у світі безпеки. Проте математика та обчислювальна потужність, необхідні для розкладання великих чисел і «розриву» ключів RSA, стають все більш доступними та практичними для використання зловмисниками. Щоб протистояти загрозам, створеним передовими обчисленнями та математикою, розміри ключів RSA повинні збільшуватися, що стає нежиттєздатним у довгостроковій перспективі. Останніми роками кілька дослідників знайшли вразливі місця у відкритих ключах RSA. Наприклад, у 2020 році дослідники Keyfactor проаналізували понад 75 мільйонів активних ключів RSA в Інтернеті, виявивши, що кожен із 172 сертифікатів, які використовують ключі RSA, вразливий до практичної атаки, відомої як «факторинг».

Але є підхід, який може конкурувати з RSA – це криптографія на основі еліптичних кривих (**ECC – Elliptic curve cryptography**). Привабливість підходу на основі еліптичних кривих у порівняні з RSA полягає в тому, що з в цьому випадку забезпечується еквівалентний криптографічних захист при значно меншій кількості розрядів ключа (табл.10.1) [32].

Таблиця 10.1

Порівняльна характеристика еквівалентної довжини ключів

Еквівалентна довжина ключа (біти)			
ECC	**RSA**	**Відншення довжини ключів ECC/RSA**	**Довжина ключа AES**
163	1024	1:6	
256	3072	1:12	128
384	7680	1:20	192
512	15360	1:30	256

Такий результат суттєво зменшує завантаження процесору і прискорює процес обробки даних.

Використання еліптичних кривих для створення криптосистеми було незалежно запропоновано **Нілом Кобліцем** (англ. Koblitz Neal) та **Віктором Міллером** в 1985 р. З 1998 року використання еліптичних кривих для вирішення криптографічних завдань було закріплено в стандартах США ANSI X9.62 (Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA), 1999) і FIPS 186-2 (FIPS 186-3 з 2009 року).

В Україні на рівні національного стандарту (ДСТУ 4145-2002 - Державний стандарт України. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка) прийнято алгоритм цифрового підпису, що ґрунтується на еліптичних кривих. 2020 року запроваджено новий Національний стандарт ДСТУ 9041:2020 - Інформаційні технології. Криптографічний захист інформації. Алгоритм шифрування коротких повідомлень, що ґрунтується на скручених еліптичних кривих Едвардса.

ECC — це криптографічний алгоритм із відкритим ключем, який використовується для виконання критичних функцій безпеки, зокрема шифрування, автентифікації та цифрових підписів. ECC базується на теорії еліптичних кривих, яка генерує ключі за допомогою властивостей рівняння еліптичної кривої на відміну від традиційних методів розкладання на множники дуже великих простих чисел.

ECC вважається більш безпечним, ніж RSA, оскільки RSA базується на розкладанні великих чисел на множники — проблему, яку комп'ютери вирішили. Навпаки, криптографія еліптичної кривої базується на проблемі дискретного логарифмування, яку набагато важче вирішити.Було доведено, що навіть із сучасними технологіями для зворотного проектування ключа, згенерованого за допомогою ECC, знадобиться більше часу, ніж вік Всесвіту.

Ключі **ECC** також набагато коротші за ключі RSA — найпоширеніший тип ключа, який використовується в криптографії з відкритим ключем. Коротші ключі також означають, що для шифрування та дешифрування даних потрібна менша потужність обробки, що робить ECC більш ефективним, ніж інші алгоритми.

Оскільки **ECC** пропонує еквівалентну безпеку з меншою обчислювальною потужністю та використанням ресурсів батареї різноманітних мобільних пристроїв, він стає все більш широко використовуваним у криптовалютних платформах, включаючи Bitcoin та Ethereum, мобільних додатках і пристроях з низьким енергоспоживанням, які мають обмежену обчислювальну потужність. Нарешті, ECC можна використовувати для цифрових підписів, обміну ключами та інших цілей, що робить його універсальним інструментом для багатьох різних застосувань.

10.2. Основи побудови еліптичних кривих та їх властивості

Еліптичні криві (ECC – Elliptic curve cryptography) називаються так тому, що описуються кубічними рівняннями, подібними до тих, які використовуються для обчислювання кривої еліпса.

Еліптична крива (ЕК) над полем ***P*** — це множина точок проективної площини над ***P***, що задовольняють рівнянню

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

разом з точкою на нескінченності (рис.10.1.), де a_i – дійсні числа, які дадовольняють певним умовам.

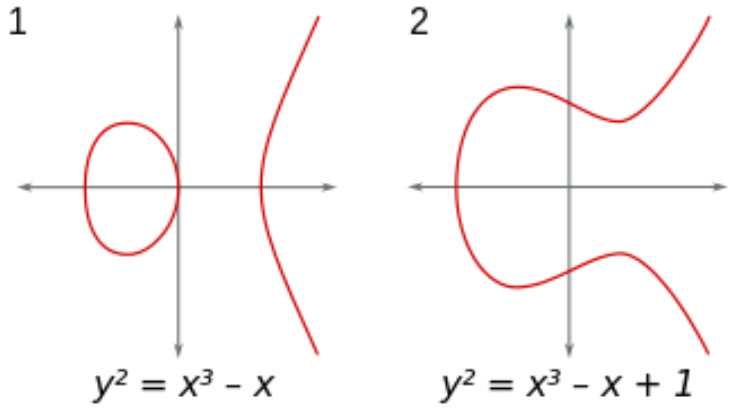


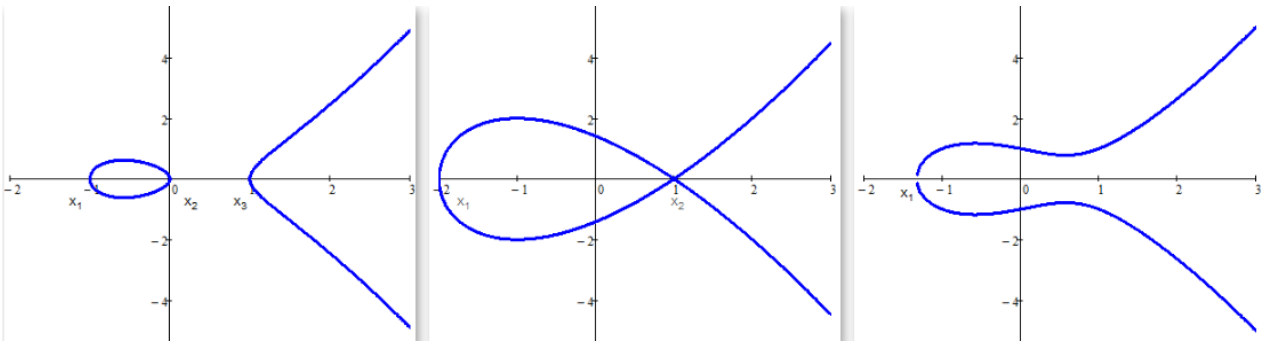
Рис.10.1. Загальний вид різних еліптичних кривих (https://uk.wikipedia.org/wiki/Еліптична_крива)

Еліптична крива може описуватися і більш простим рівнянням, у якого деякі коефіцієнти дорівнюють нулю. Наприклад,

$$y^2 = x^3 + ax + b$$

,

де $4a^3 + 27b^2 \neq 0$ - це множина точок, що задовольняє цьому рівнянню (рис.10.2.)



а) б) в)

Рис.10.2. Деякі приклади ЕК для трьох випадків:

а) $y^2 = x^3 - x$, б) $y^2 = x^3 - 3x + 2$, в) $y^2 = x^3 - x + 1$.

Еліптичні криві безпосередньо пов'язані з терміном **поля Галуа** та операціями над ними, а відповідно, і над **групами**.

В математиці **група** - це **множина**, для якого визначили двійкову операцію, звану «складанням» і позначається символом +.

Щоб **множина G** була **групою**, складання потрібно визначити таким чином, щоб воно відповідало чотирьом наступним властивостям:

1. **замикання**: якщо a і b входять в **G**, то a+b входить в **G**;
2. **асоціативність:**

$$(a + b) + c = a + (b + c);$$

3. існування **одиничного елемент 0:**

$$a + 0 = 0 + a = a;$$

4. у кожного елемента є **зворотна величина**, тобто: для кожного **a** існує таке **b**, що

$$a + b = 0.$$

На 10.3. наведені множини чисел, але не всі вони можуть бути групою згіно наданого визначення. Наприклад, множина **Z** – є групою, а множина **N** – ні, оскільки для неї не виконується четверта умова, тобто не існує зворотнього елемента.

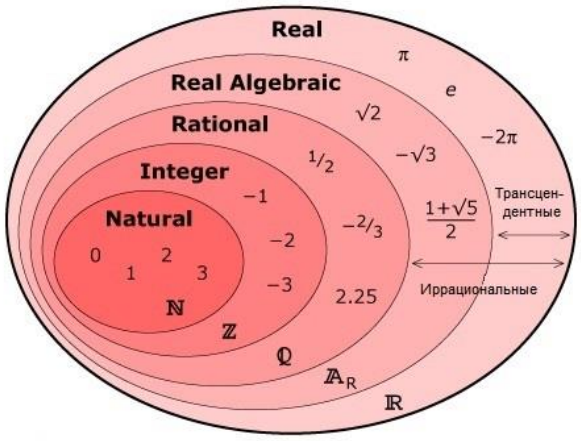


Рис.10.3. Представлення різних типів чисел: **N** – натуральні числа; **Z** – цілі числа; **Q** – раціональні числа; $\mathbf{A_R}$ – ірраціональні числа; **R** – дійсні числа (https://in.pinterest.com/pin/820218150896444303/)

Можемо визначити групу для еліптичних кривих, а саме:

- **елементи групи** є точками еліптичної кривої**;**
- **одиничний елемент -** це нескінченно віддалена **точка 0,** в якій збігаються всі вертикальні прямі;
- **зворотна величина точки P -** це точка, **симетрична** щодо вісі **OX**;

- складання задається наступним правилом**: сума трьох ненульових точок P, Q та R,** що лежать на одній прямій, буде дорівнювати **0**:

$$P + Q + R = 0.$$

Потрібно врахувати, що в останньому правилі нам потрібні тільки три точки на одній прямій і порядок розташування цих трьох точок не важливий:

$$P + (Q + R) = Q + (P + R) = R + (P + Q) = \cdots = 0.$$

З геометричного складання точок еліптичної кривої слідує наступна рівність:

$$P + Q = -R.$$

і геометрична інтерпритація представлена на рис.10.4. Іншими словами, сума двох точок *P* та *Q* утворює симетричну точку (-*R*) для точки *R* відносно вісі OX.

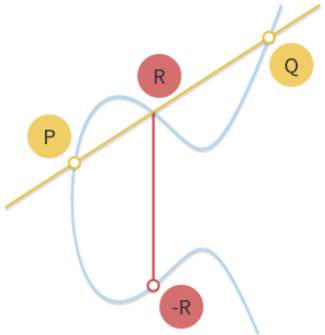


Рис.10.4. Геометрична інтерпритація взаємного розташування точок на еліптичній кривій (https://andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/)

Загальне представлення точок на еліптичній кривій зображено на рис.10.5.

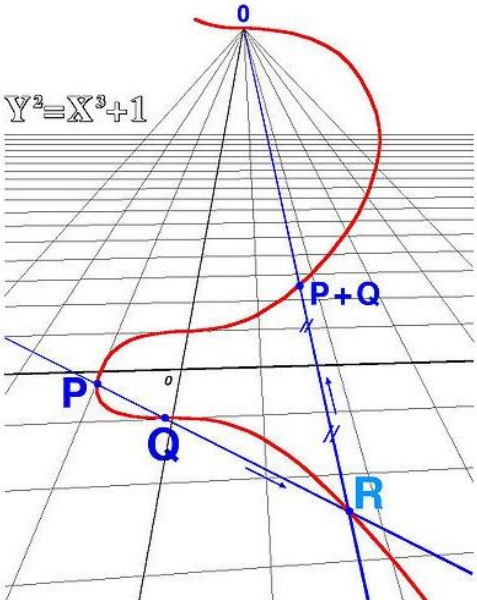


Рис.10.5. Взаємне розташування точок на еліптичній кривій (https://programmingpraxis.com/2009/07/28/elliptic-curves/)

Геометричний спосіб дозволяє досить зручно інтерпретувати результат додавання точок, але потребує вдосконалення. Зокрема, потрібно відповісти на кілька запитань.

1) Що, якщо **P**=0 або **Q**=0? Зрозуміло, ми не зможемо провести пряму (**0** не перебуває на площині XY). Але оскільки ми визначили **0,** як одиничний елемент, то

$$P + 0 = P \qquad 0 + Q = Q$$

2) Що, якщо

$$P = -Q.$$

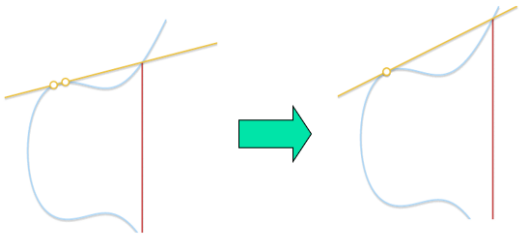
В цьому випадку пряма, що проходить через дві точки, **вертикальна**, і не перетинає третю точку:

$$P + Q = P + (-P) = 0.$$

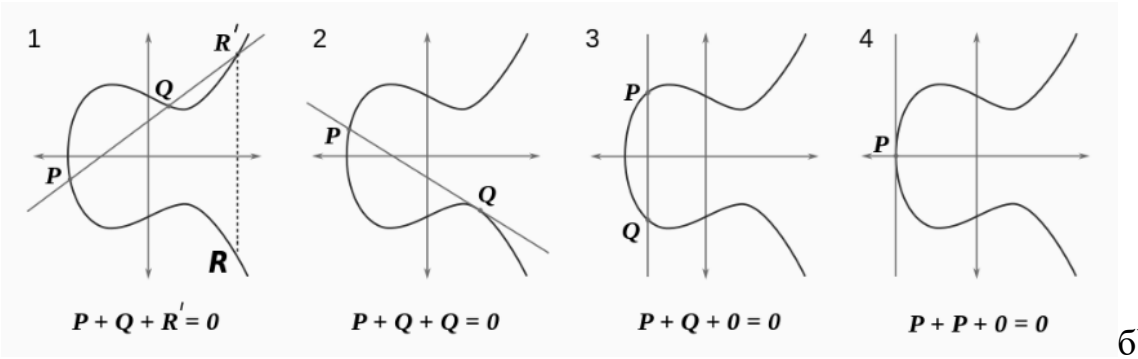
3) Що, якщо

$$P = Q$$

У цьому випадку через точку проходить нескінченна кількість прямих. При зближенні двох точок пряма стає дотичною до кривої (рис.10.6).



а)



б)

Рис.10.6. Формування дотичної для еліптичної кривої при зближенні двох точок, коли P=Q (а), візуалізація варіантів формування прямої на еліптичній кривій (б) (https://uk.wikipedia.org/wiki/Еліптична_крива)

На практиці зручніше працювати в скінченній множині точок. У криптографії застосовуються еліптичні криві, визначені над скінченнимим **полями Галуа *GF* (*q*).** У цьому випадку вони являють собою множину точок $E(GF(q))$, координати яких задовольняють рівнянню кривої. В цьому випадку неперервний графік еліптичної кривої перетвориться на множину точок на площині, розмірність якої буде дорівнювати основі модуля поля. На рис.10.7. представлена ілюстрація еліптичних кривих в полі 19 та 97.

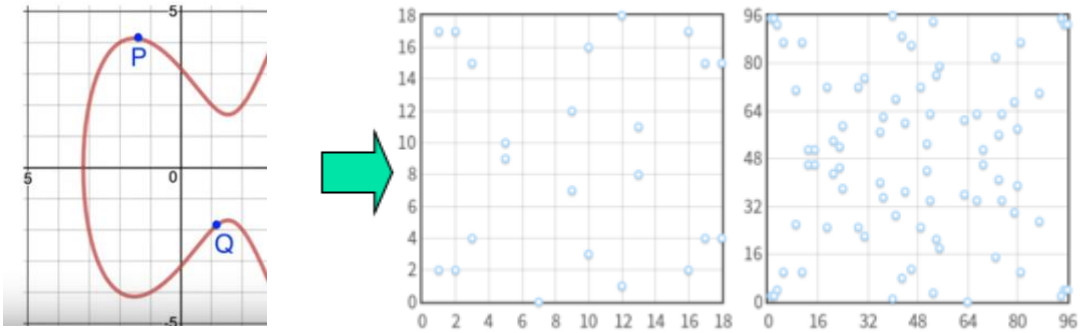


Рис.10.7. Перехід від неперервного представлення еліптичної кривої до дискретного для різних скінченних полів

Такий перехід в скінченні поля буде еквівалентний наступним виразам:

$$y^2 = x^3 + ax + b \qquad \Rightarrow \qquad y^2 = x^3 + ax + b \bmod p$$

$$4a^3 + 27b^2 \neq 0 \qquad\qquad 4a^3 + 27b^2 \neq 0 \bmod p$$

У реальних криптосистемах використовується рівняння виду:

$$y^2 \equiv (x^3 + ax + b) \bmod p, \quad \text{де} \quad a, b \in GF(p);$$

$$(4a^3 + 27b^2) \bmod p \neq 0; \quad p > 3$$

,

де p – просте.

Група $E(GF(p))$ складається з усіх точок

$$(x, y), \quad x \geq 0, \quad p > y,$$

які задовольняють рівнянню

$$y^2 \equiv (x^3 + ax + b) \bmod p,$$

і нескінченно віддаленої точки O.

Множина $Ep(a, b)$ має свою кількість точок і ця величина відіграє важливе значення для криптостійкості системи.

Приклад 10.1. Дано еліптичну криву $\mathbf{y^2 = x^3 + x + 1 (mod\ 5)}$ над полем $GF(5)$. Цьому рівнянню задовольняють 8 скінченних точок із координатами в полі $GF(5)$. На рис.10.3 подано приклад цієї кривої з координатами точок:

(0, 1), (0, 4), (2, 1), (2, 4), (3, 1), (3, 4), (4, 2), (4, 3).

Для знаходження координат цих точок перебираються всі можливі комбінації для x (в даному випадку від 0 до 4), обраховується права частина рівняння за заданим модулем (p=5), обраховується ліва частина рівняння (y^2) за цим же модулем при різних комбінаціях y (так само від 0 до 4). Значення пар для x та y, для яких ліва і права частини рівняння будуть однаковими, і будуть координатами точок для заданої еліптичної кривої в цьому полі.

Зазначимо, що розташування точок завжди буде симетричним відносно вісі OX.

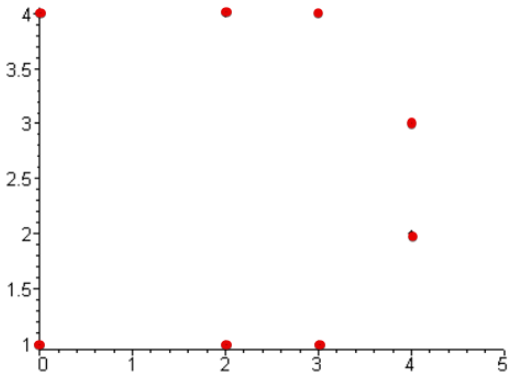


Рисунок 10.8 – Точки, координати яких належать еліптичній кривій $y^2 = x^3 + x + 1 \pmod 5$ над полем $GF(5)$.

Розглянемо еліптичні криві над простим полем $GF(p)$:

$$y^2 \equiv (x^3 + ax + b) \bmod p, \quad \text{де} \quad a, b \in GF(p), \tag{10.1}$$

де p - просте число, модуль перетворень; параметри кривої a, b належать $GF(p)$ - невідємні і задовольняють умові гладкості кривої

$$(4a^3 + 27b^2) \bmod p \neq 0.$$

Змінні x, y належать $GF(p)$.

Окрім точок (x, y), що задовольняють даному рівнянню, у множину точок еліптичної кривої також включається нескінченно віддалена точка **O** .

Приклад 10.2. При $p = 2$ точками еліптичної кривої

$$y^2 = x^3 + x + 1 \pmod 2$$

є три точки - (0,1), (1,1) і **O,** координати яких задовольняють даному рівнянню в заданому полі.

Приклад 10.3. Знайдемо точки еліптичної кривої $\boldsymbol{E_7(2, 6)}$:

$$\boldsymbol{y^2 = x^3 + 2x + 6 \pmod 7}.$$

Перевіримо умову: $(4*2^3 + 27*6^2) \bmod 7 = 3$ (не дорівнює 0 – умова виконується).

Визначемо можливі значення змінної $x = \{0, 1, 2, 3, 4, 5, 6\}$. При визначених значеннях x знайдемо значення правої частини рівняння y^2 (табл.10.2):

Таблиця 10.2.

Значення рівняння еліптичної кривої

x	0	1	2	3	4	5	6
$x^3 + 2x + 6 \pmod 7$	6	2	4	4	1	1	3
y	0	1	2	3	4	5	6
$y^2 \bmod 7$	0	1	4	2	2	4	1

Група $E_7(2, 6)$ складається з точок (x, y), для яких виконується рівність:

$$y^2 (\mathrm{mod}\, 7) = x^3 + 2x + 6 (\mathrm{mod}\, 7).$$

Такими точками для даної еліптичної кривої в заданому полі будуть:
(1, 3), (1, 4), (2, 2), (2, 5), (3, 2), (3, 5), (4, 1), (4, 6), (5, 1), (5, 6) і **O.**

Визначення. Порядком групи точок еліптичної кривої ***E***p(a,b) називається число елементів цієї групи.

10.3. Операції над точками еліптичної кривої

Позначимо через $Ep(a,b)$ множину точок еліптичної кривої виду (10.1). На множині $Ep(a,b)$ введемо операцію додавання «+» за правилом:
якщо $P = (x1, y1)$ і $Q = (x2, y2)$, то $P + Q = (x3, y3)$ визначаються за формулами:

$$\begin{aligned} x_3 &\equiv (\lambda^2 - x_1 - x_2) \bmod p; \\ y_3 &\equiv (\lambda\,(x_1 - x_3) - y_1) \bmod p, \end{aligned} \qquad (10.2)$$

де

$$\lambda \equiv \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} (\mathrm{mod}\, p), & \text{за } P \neq Q; \\ \dfrac{3x_1^2 + a}{2y_1} (\mathrm{mod}\, p), & \text{за } P = Q. \end{cases}$$

Приклад 10.4. Розглянемо приклад додавання точок еліптичної кривої над заданим полем $E_5(1, 1)$:

$$y^2 = x^3 + x + 1 (\mathrm{mod}\ 5) \qquad (10.3).$$

Визначимо можливі точки та їх координати для заданих параметрів. Для цього складемо таблицю можливих значень для лівої і правої частини рівняння (табл.10.3)

Таблиця 10.3.

Значення рівняння еліптичної кривої

x	0	1	2	3	4
$x^3 + x + 1 (\mathrm{mod}\ 5)$	1	3	1	1	4
y	0	1	2	3	4
y^2 mod(5)	0	1	4	4	1

Таким чином точки, координати яких будуть задовольняти рівнянню (10.3), мають наступні координати:

(0, 1), (0, 4), (2, 1), (2, 4), (3, 1), (3, 4), (4, 2), (4, 3).

Продемонструємо операції додавання точок, наприклад точки P(0, 1) з точкою з такими самими координатами, тобто P(0, 1) + P(0, 1) = 2P(0, 1).
Для обрахунку скористаємося виразом (10.2):

$x_1 = x_2 = 0$;
$y_1 = y_2 = 1$;
$\lambda = [(3*0 + 1)/(2*1)] \bmod 5 = (1/2) \bmod 5 = 3$
$x_3 = (3^2 – 0 – 0) \bmod 5 = 4$
$y_3 = (3(0 – 4) – 1) \bmod 5 = -13 \bmod 5 = 2$

Таким чином, отримаємо координати подвоєної точки 2P(4, 2).
При проведенні подібних розрахунків отримаємо:
P = (0, 1); 2P = (4, 2); 3P = (2, 1); 4P = (3, 4); 5P = (3, 1); 6P = (2, 4); 7P = (4, 3); 8P = (0, 4); 9P = **O**, при цьому 10P = P. При подальшому додаванні будемо отримувати циклічний результат, наведений вище.

Точка P = (0, 1) породжує циклічну підгрупу порядку 9:

(0, 1); (4, 2); (2, 1); (3, 4); (3, 1); (2, 4); (4, 3); (0, 4); **O.**

Таким чином ми отримали адитивну групу $E(Z_5)$ для заданої еліптичної кривої (рис. 10.9).

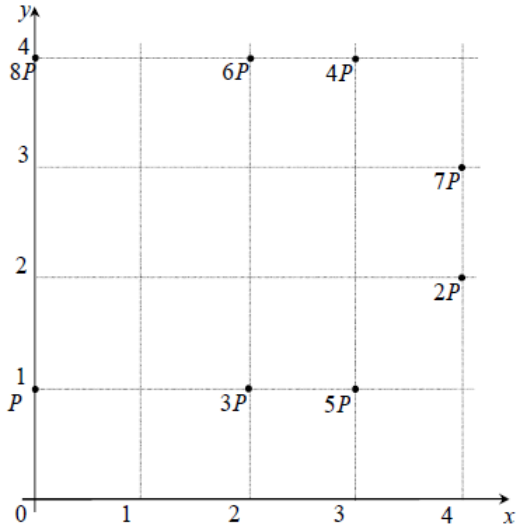


Рисунок 10.9 – Точки, координати яких належать еліптичній кривій $y^2 = x^3 + x + 1$ (mod5) над полем $GF(5)$.

Визначення. Точка $P \in Ep(a,b)$ називається базовою точкою підгрупи точок еліптичної кривої $Ep(a,b)$, якщо будь-яка точка Q цієї підгрупи може бути подана у вигляді $Q = k*P$, де k=1,2,…,n, де ***n*** - порядок підгрупи.

Для базової точки P має місце рівність $n*P$ = **O** .

Приклад 10.5. Точка P = (0,1) є базовою для групи точок еліптичної кривої $y^2 = x^3 + x + 1$ (mod 5) .

Твердження (теорема Хасе). Значення порядку групи точок еліптичної кривої над полем Галуа GF(q) має верхню і нижню границі:

$$q+1-2\sqrt{q} \le \#E_q(a,b) \le q+1+2\sqrt{q}$$

Приклад 10.6. Нехай Q = (2,4) - точка еліптичної кривої

$$y^2 = x^3 + x + 1 \ (mod\ 5).$$

Тоді $2Q$ = (2,1), $3Q$ = **O**.
Точка Q = (2,4) породжує циклічну підгрупу порядку n = 3:

(2,4), (2,1), **O**.

Таким чином можна бачити, що від вибору базової точки залежить, скільки різних варіантів інших точок можна отримати. Від вибору базової точки залежить порядок циклічної підгрупи обраної еліптичної кривої, що безпосередньо впливає на криптостійкість системи.

10.4. Алгоритм обчислення порядку точки еліптичної кривої

Нехай задана еліптична крива $E_p(a, b)$: $y^2 = x^3 + ax + b$ (mod p) над полем GF(p), і точка $P(x, y)$ належить заданій кривій. Знайдемо порядок точки $P(x , y)$.

Для цього необхідно вирішити рівняння і знайти невідоме значення n:

$$n*P = O,$$

яке є адитивним аналогом задачі дискретного логарифмування.

Застосуємо алгоритм великих-малих кроків.

Нехай n - максимальна оцінка порядку групи точок $P(x, y)$ еліптичної кривої, отримана по теоремі Хасе.
Обчислимо m, як $m = \sqrt{n}$ з округленням залишку.

4. Будуємо таблицю пар (j, jP) для j = 1, 2, …, α.
5. Обчислимо $\alpha = -mP$.
6. $\gamma = O$.
7. Виконаємо циклічні дії по i від 0 до $m - 1$:
   4.1. перевіряємо, чи є γ другою компонентою в таблиці п.1;
   4.2. якщо $\gamma = jP$, то вважаємо, що $n = m*i + j$;
   4.3. $\gamma = \gamma + \alpha$.

Приклад 10.7. Візьмемо еліптичну криву виду $E_5(1, 1)$:

$$y^2 = x^3 + x + 1 \ (mod\ 5)$$

над простим полем $GF(p)$ = 5. По теоремі Хасе максимальна оцінка порядку групи точок кривої дорівнює:

$$n = p + 1 + 2\sqrt{p} = 5 + 1 + 2\sqrt{5} \approx 10.$$

Звідси знайдемо, що m = 4.

За допомогою алгоритму великих-малих кроків знайдемо порядок точки $P(0, 1)$. Для цього побудуємо таблицю 10.4, скориставшись отриманими результатами в прикладі 10.4:

Таблиця 10.4.

j	1	2	3	4
jP	(0, 1)	(4, 2)	(2, 1)	(3, 4)

Знайдемо $\alpha = -mP = -4(0, 1) = -(3, 4) = (3, -4) = (3, 1)$.
$\gamma = \boldsymbol{O}$.
$i = 0$ $\gamma = \gamma + \alpha = \boldsymbol{O} + (3, 1) = (3, 1)$.
$i = 1$ $\gamma = \gamma + \alpha = (3, 1) + (3, 1) = 2(3, 1) = (0, 1)$ (значення належить таблиці для $j = 1$).
$i = 2$ $j = 1 \Rightarrow n = m*i + j = 4*2 + 1 = 9$.

Таким чином, порядок точки P(0, 1): $n = 9$. Отримай результат співпадає з результатом, отриманим методом перебору в прикладі 10.4.

10.5. Побудова криптографічної системи на основі еліптичних кривих (ECDH)

ECDH (англ. Elliptic curve Diffie–Hellman) - це різновид алгоритму Діффі-Хеллмана для еліптичних кривих. Насправді, це скоріше протокол узгодження ключів, а не алгоритм шифрування. По суті, це означає, що ECDH задає (певною мірою) **порядок генерування ключів** та обміну ними. **Спосіб шифрування** даних за допомогою таких ключів ми можемо вибирати самі.

Алгоритми еліптичних кривих працюватимуть в циклічній підгрупі еліптичної кривої над кінцевим полем. Алгоритмам будуть потрібні наступні параметри:

- просте ***p***, що задає розмір кінцевого поля;
- коефіцієнти ***a*** і ***b*** рівняння еліптичної кривої;
- базова точка ***G***, яка генерує підгрупу;
- порядок ***n*** підгрупи;
- кофактор ***h*** підгрупи.

Параметрами області визначення для алгоритмів є:

$$(p, a, b, G, n, h).$$

Закритий ключ - це випадкове ціле ***d***, вибране з

$$\{1, \ldots, n-1\},$$

де ***n*** - порядок підгрупи.

Відкритий ключ - це точка

$$H = dG,$$

де **G**- базова точка підгрупи.

Якщо ми знаємо ***d*** і **G** (разом з іншими параметрами області визначення), то знайти ***H*** **«просто»**!

Але якщо ми знаємо ***H*** і ***G***, то пошук закритого ключа ***d*** є **«складним»** завданням, тому що вимагає рішення задачі дискретного логарифмування. Фактично, це задача з застосуванням односпрямовуваних функцій на основі еліптичних кривих.

Криптографічний протокол за схемою Діффі-Хелмана на основі еліптичних кривих (ECDH)

1) Для встановлення секретного зв'язку два користувача **A (Alice)** і **B (Bob)** вибирають еліптичну криву з параметрами $E_P(a, b)$ і точку $P(x, y)$ на ній.

2) Потім **A** і **B** генерують незалежно один від одного по секретному числу **α** і **β**.

3) Користувач **A** обчислює добуток **αP** і пересилає його **B**, а користувач **B** обчислює **βP** і пересилає його **A**. При цьому параметри кривої, координати точки на ній, значення добутків є відкритими і можуть передаватися по незахищених каналах зв'язку.

4) Далі користувач **A** примножує прислане значення **βP** на α, отримавши після цього **αβP**, користувач **B** перемножує отримане значення **αP** на **β,** отримуючи такий же результат - **αβP**.

5) Таким чином, обидва користувачі отримали загальне секретне значення (координати точки **αβP** на ЕК), яке вони можуть використати як секретний ключ шифрування.

Приклад 10.8. Шифрування даних на ECDH

Нехай абоненти **A (Alice)** і **B (Bob)** вирішили провести передачу повідомлень при використанні протоколу **ECDH.** Нехай для цього вони обрали еліптичну криву виду $E_{743}(157, 223)$:

$$E: y^2 = x^3 + 157x + 223 \bmod(743)$$

і точку P (117,692) на ній.

Потім **A** генерує секретний число **α = 735**. Користувач **B** генерує секретний число **β = 529**. Користувач **A** обчислює добуток:

α*P = 735*(117, 692) = **(517,488)**

і пересилає його **B.** Користувач **B** обчислює добуток:

β*P = 529*(117, 692) = (472,687)

і пересилає його **A**.

Далі користувач **A** перемножує прислане значення **β*P** на **α:**

β*P*α = 735*(472,687) = (332,590).

Користувач **B** перемножує прислане значення **α*P** на **β,** отримуючи такий же результат:

α*P*β = 529 * (517,488) = (332,590).

Таким чином, обидва користувачі отримали **загальне секретне значення** (координати точки **αβP** на еліптичній кривій **(332,590)**, яке вони будуть використовувати в якості **загального секретного ключа**.

Зверніть увагу на те, що спільний секретний ключ є парою чисел. Якщо цей ключ передбачається використовувати як сеансовий ключ для традиційного шифрування, то з цієї пари чисел треба генерувати одне відповідне значення. Можна, наприклад, використовувати просто координату x чи певну просту функцію від x, або заздалегідь визначену комбінацію над парою (x, y). Окрім цього, в літературі можна знайти і інші підходи щодо використання координат загального секретного ключа для зашифрування/розшифрування даних.

Криптографічний протокол за схемою Ель–Гамаля на основі еліптичних кривих

Для користувачів обираються загальна еліптична крива $Ep(a, b)$ й точка G на ній такі, що G, $2G$, $3G$, ..., nG – різні точки і $nG = O$ для певного простого числа n [27].

Кожен користувач мережі обирає число k, $0 < k < n$, яке зберігає як власний секретний ключ, і обчислює точку на кривій $Y = kG$, яка слугуватиме за його відкритий ключ. Параметри кривої та перелік відкритих ключів передаються всім користувачам мережі.

Припустімо, що користувач A хоче передати повідомлення користувачеві B. Вважатимемо, що повідомлення подано у формі числа $M < p$. Користувач A виконує такі дії:

1) обирає випадкове число r, $0 < r < n$;
2) обчислює $R = rG$, $P = rY_b = (x, y)$;
3) зашифровує $C = (Mx) \bmod p$;
4) надсилає користувачеві B шифртекст (R, C).

Користувач B після отримання шифртексту (R, C) виконує такі дії:

1) обчислює $Q = k_bR = (x, y)$;
2) розшифровує $M = (Cx^{-1}) \bmod p$.

Обґрунтування протоколу полягає в наступному:

$$k_bR = k_b(rG) = r(k_bG) = rY_b,$$

тобто $Q = P$.

Координата x точки Q залишається секретною для криптоаналітика, оскільки йому невідоме число r. Криптоаналітик може спробувати обчислити r з точки P, але для цього йому потрібно розв’язати проблему дискретного логарифмування на кривій, що вважається за неможливе. Найбільш імовірним варіантом використання розглянутого алгоритму буде передавання в якості числа M секретного ключа для блокового чи потокового шифру. В цьому разі доречно обирати параметри кривої у такий спосіб, щоб $\log n$ приблизно удвічі перевищував довжину ключа шифру.

Приклад 10.9. Шифрування даних за схемою Ель–Гамаля на основі ЕС

Нехай $p = 211$, $G = (2, 2)$, $Ep(0, -4)$, що відповідає кривій $y^2 = x^3 - 4$. Можна підрахувати, що $241G = O$, тоді обираємо $r = 43$. Власним ключем користувача B є $k_b = 91$, тому відкритим ключем B буде

$$Y_b = 91(2, 2) = (206, 121).$$

Користувач A хоче передати повідомлення $M = 25$ користувачеві B. Користувач A обчислює:

1) $R = 43(2, 2) = (124, 119)$;
2) $P = 43(206, 121) = (142, 15)$;
3) $C = (25 * 142) \bmod 211 = 174$;
4) надсилає користувачеві B шифртекст $\{(124, 119), 174)\}$.

Користувач B обчислює:

1) $Q = 91(124, 119) = (142, 15)$;
2) $M = (174 \cdot 142^{-1}) \bmod 211 = (174 * 159) \bmod 211 = 25$.

Таким чином, відбувся обмін ключами (інформацією) на основі схеми Ель–Гамаля при застосуванні еліптичних кривих.

10.6. Підготовка до завдання

Ознайомитися з теоретичними положеннями задач побудови та реалізації асиметричних криптосистем на прикладі системи **ECC**. Пригадати елементи теорії чисел (поняття простих чисел, модульних перетворень, арифметичних операцій в полях Галуа).

10.7. Практичне завдання

1. Дано еліптичну криву $\boldsymbol{E_{13}(a, b)}$:

$$y^2 = x^3 + ax + b \,(\mathrm{mod}\, 13).$$

Параметри a, b відповідають номеру варіанта (табл. 10.5). Необхідно:

- знайти всі точки, що належать заданій кривій і порядок кривої.
- знайти базову точку, яка б відповідала найбільшому значенню порядку;
- представити кожну точку ЕС, як кратну базовій (знайти показник кратності).

Таблиця10.5.

Номер варіанта

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

2. Робота передбачає роботу в команді по 2 особи. Команда задається параметрами ЕС:

$$y^2 = x^3 + ax + b \,(\mathrm{mod} 13)\,.$$

(можна взяти один із варіантів учасників команди згідно табл.10.3, наприклад варіант №4, a=1, b=5).
3. Задається точка P (дивись теоретичний матеріал).
4. Кожен із учасників команди задається секретним простим числом **α** і **β** відповідно (дивись приклад вище).
5. Кожен із учасників команди обчислює **αP** і **βP** та обмінюються між собою.
6. Кожен із учасників команди обчислює **αβP.**
7. Порівняти результати. Отримане значення використовується як **ключ**.

По суті, це означає, що ECDH задає (певною мірою) **порядок генерування ключів** та обміну ними. **Спосіб шифрування** даних за допомогою таких ключів можна вибирати самостійно.
8. Написати програму, яка реалізує операції по п.1-6.
9. Надати аналіз отриманим результатам.

10.8. Зміст протоколуроботи

Протокол до пактичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

10.9. Контрольні питання для самоперевірки

1. Дайте визначення еліптичної кривої.
2. Що називається групою, перерахуйте властивості групи.
3. Чому множина натуральних чисел не є групою?
4. Що є одиничним елементом для еліптичної кривої?
5. Опишіть груповий закон для еліптичних кривих.
6. Яке існує правило для трьох точок, які належать еліптичній кривій?
7. Яка основна причина застосування скінченних полів для еліптичних кривих? Пояснити поняття еліптичніих кривих над кінцевими полями.
8. На чому грунтується криптографічна стійкість застосування перетворень на основі еліптичних кривих?
9. Що таке порядок групи ECC? Алгоритм його знаходження.
10. Описати послідовність дій при використанні алгоритму обміну даними при застосуванні ECC.
11. Описати алгоримт додавання точок в полі FG(n).
12. Описати алгоримт множення в полі FG(n).
13. Теорема Хассе та її практичне застосування. Навести приклад.
14. Надати алгоритм пошуку базової точки в ECC.
15. Що таке ефімерне ECDH, надати коротку характеристику та призначення.

10.10. Задачі до практикуму №10

1. Визначитипорядок групи для еліптичної кривої $\boldsymbol{E_{11}(-7, 10)}$.
2. Для еліптичної кривої $\boldsymbol{E_7(2, 6)}$ обчислити значення точок:

$$2(2, 2);\ 2(4, 6);\ (1, 3) + (1, 4);\ (3, 5) + (5, 1).$$

3. Провести зашифрування повідомлення M та його розшифрування з використанням алгоритму Ель–Гамаля на основі заданої еліптичної кривої: $E_{211}(0, -4)$, $G = (2, 2)$, $r = 2$, $k_b = 2$, $M = 2$.

10.11. Завдання до самостійної роботи

1. Різновиди та властивості еліптичних кривих.
2. Різновиди канонічних рівняннь еліптичних кривих і арифметичні операції з точками.
3. Алгоритм Шуфа для підрахунку числа точок на еліптичній кривій.
4. Застосування алгоритму великих-малих кроків для знаходження порядку точки для заданої еліптичної кривої.
5. Протокол Мессі–Омури на основі застосування еліптичних кривих.

ПРАКТИКУМ № 11
Методи та засоби побудови хеш-функцій

Мета роботи. Ознайомитися з призначенням, властивостями та принципами побудови геш-функцій. Провести процедуру гешування над заданими даними.

11.1. Призначення хеш-функцій та їх властивості

Забезпечення цілісності інформаційних ресурсів у сучасних автоматизованих та інформаційних системах є одним з важливих етапів проектування та розробки підсистем захисту інформації. Це пов'язано з постійним зростанням послуг, які надаються різними мережевим службами. Більшість послуг надаються при відсутності фіксованих мережевих адрес клієнтів, що в цілому підвищує ризик порушення цілісності та автентичності інформації. Для захисту від таких загроз безпеки інформації, як правило, використовують механізми **хешування даних** – ключові та безключові геш-функції [33-35].

Хеш-функції (або **геш-функції**) також можуть використовуватись у складі електронного цифрового підпису (**ЕЦП**), який є потужним механізмом забезпечення автентифікації в сучасних автоматизованих системах, перевірки цілісності даних, створення послідовностей псевдовипадкових чисел та генерування сеансових ключів.

Хешування (англ. hashing)— перетворення вхідного масиву даних *довільної довжини* у вихідний бітовий рядок *фіксованої довжини*. Такі перетворення також називаються **хеш-функціями**, або **функціями згортання**. Результат перетворення називають **хеш-сумою**, **хешем, хеш-кодом, дайджестом повідомлення** (англ. message digest) (рис.11.1).

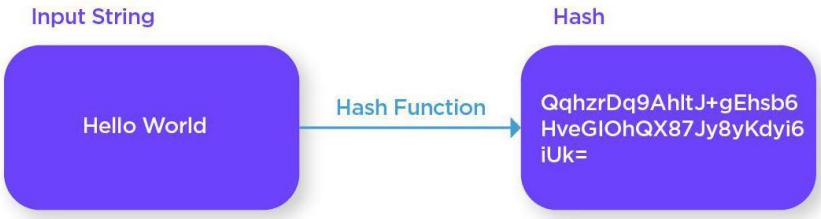


Рис.11.1. Приклад утворення хеш-функції (https://www.radixdlt.com/blog/primer-on-hashes-and-hash-functions)

Хешування застосовується для побудови асоціативних масивів, пошуку дублікатів в серіях наборів даних, побудови унікальних ідентифікаторів для наборів даних, контрольного підсумовування з метою виявлення випадкових або навмисних помилок при зберіганні або передачі, для зберігання паролів в системах захисту, при виробленні електронного цифрового підпису (на

практиці часто підписується не саме повідомлення, яке може бути достньо великим, а підписується його хеш-образ).

Розроблено багато алгоритмів хешування з різними властивостями (розрядність хеш-функцій, її обчислювальна складність та криптостійкість тощо). Вибір тієї чи іншої хеш-функції визначається специфікою розв'язуваної задачі. Найпростішими прикладами хеш-функцій можуть служити контрольна сума або CRC - циклічний надлишковий код (англ. *Cyclic redundancy check*), який реалізує обчислення контрольної суми, призначеної для перевірки цілісності даних.

Для забезпечення автентичності даних та надання інформації на електронних носіях юридичної значимості використовуються **ЕЦП**, елементом яких є хеш-значення даних, що підписуються. Основою електронного цифрового підпису є криптографічні перетворення з відкритим ключем. Особливістю таких перетворень є тривалий час зашифрування/розшифрування даних, оскільки вони базується на специфічних операціях, які є складними і неприродними для традиційних мікропроцесорів.

Для підписування великих файлів використовують їх **хеш-значення**, що суттєво скорочує процес обробки.

Цікавим моментом такого підписування документів на основі застосування хеш-функції є те, що це дозволяє підписувати секретні дані без їх розголошення (ознайомлення з ними). Це пояснюється тим, що сторона, яка підписує, має змогу підписати документ без ознайомлення з ним по суті. Тобто особа, що підписує, оперує тільки закодованим хеш-значенням документу, без можливості безпосереднього ознайомлення з ним.

Часто виникають ситуації, коли одержувач повинен вміти довести достовірність повідомлення зовнішній особі. Щоб мати таку можливість, до передавальних повідомлень мають бути приписані так звані **цифрові сигнатури.**

Цифрова сигнатура – це рядок символів, який залежить як від індентифікатора відправника, так і від змісту повідомлення. Використання цифрової сигнатури передбачає застосування деяких функцій шифрування:

$$S=H(k, T),$$

де S – сигнатура, k – ключ, T – вихідний текст, функція $H(k,T)$ – хеш – функція.

У загальному випадку однозначної відповідності між вихідними даними і хеш-кодом немає в силу того, що кількість різних значень геш-функцій менша, ніж число варіантів значень вхідного масиву даних. Може трапитися ситуація, коли масиви даних з різним вмістом дають однакові хеш-коди. В цьому випадку прийнято говорити про навність **колізій**. Імовірність виникнення колізій відіграє важливу роль в оцінці якості хеш-функцій.

На рис.11.2. наведений приклад виникнення колізії. Припустимо, що хеш-функція ставить у відповідність іменам ціле число від 0 до 15. З наведеної ілюстрації видно, що є суперечність (колізія) між іменами «John Smith» та

«Sandra Dee», яким відповідає однакове результуюче числове значення «02», чого не повинно бути.

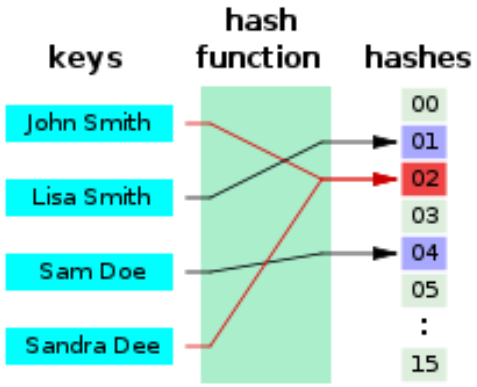


Рис.11.2. Приклад виникнення колізії при аналізі хеш-функцій (https://uk.wikipedia.org/wiki/Хеш-функція).

Колізією для функції **h** називається пара значень **x, y, x ≠ y,** для яких значення хеш-функцій будуть однаковими, тобто **h(x) = h(y).**

Таким чином, хеш-функція повинна мати наступні властивості:

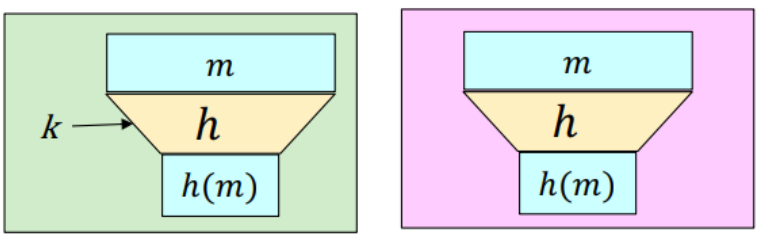
- для даного значення **h(x)** неможливо знайти значення аргументу **x.** Такі хеш-функції називають стійкими в сенсі звернення або стійкими в сильному сенсі;
- для даного аргументу **x** неможливо знайти інший аргумент **y** такий, що **h(x) = h(y).** Такі хеш-функції називають стійкими в сенсі обчислення колізій або стійкими в слабкому сенсі.

При практичному використанні хеш-функцій повинні виконуватися такі вимоги:

- алгоритм повинен володіти високою швидкістю обробки інформації (це особливо актуально для банківських операцій, де необхідна особлива оперативність обробки інформації);
- хеш-функція повинна бути стійкою до атаки методом «грубої сили»;
- програмна реалізація хеш-функції повинна бути оптимізована під використання на сучасній апаратно-програмній базі.

Ці вимоги повинен задовольняти як сам алгоритм створення хеш-функції, так і хеш-функція.

Хеш-функції поділяють на **безключові** і **ключові** (рис.11.3).



а) б)

Рис.11.3. Представлення результату обробки даних *m* в їх хеш-образ *h(m)* з ключем *h* (а) та без ключа (б).

До **безключових** хеш-функцій відносяться коди виявлення змін повідомлення (**MDC-код**, *modification detection code*), також відомі як коди виявлення маніпуляцій над повідомленнями, або коди цілісності повідомлень (рис.11.4). За даною технологією з повідомлення M формується хеш-функція Hash, з якої створюється дайджест MDC і який передається по захищеному (безпечному) каналу звязку. На приймальній стороні створюється інший дайджест хеш-функціїї і порівнюється з отриманим дайджестом. На основі їх порівняння робиться висновок про внесення змін в повідомлення.

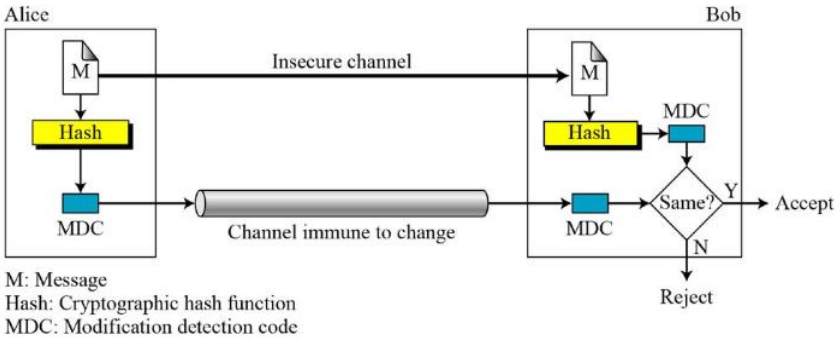


Рис.11.4. Процедура виявлення змін в повідомленні на основі MDC-коду.

Недоліком безключових хеш-функцій є те, що вони незахищені від можливості по підбору такого ж самого повідомлення з однаковим значенням геша. Наприклад, якщо повідомлення та MDC передаються через ненадійний канал, *Єва* (трєтя сторона) може перехопити повідомлення, змінити його, створити новий MDC із повідомлення та передати їх *Bob*. *Bob* ніколи не дізнається, що повідомлення надійшло від *Єви*. Разом з тим, MDC-коди забезпечують, спільно з іншими механізмами, цілісність даних.

До **ключових** хеш-функцій відносяться **MAC** коди - (англ. *message authentication code* — код аутентифікації повідомлення). Найбільш загальним використання MAC кодів у блоковому шифрі є використання шифру в режимі блокового з'єднання шифру - CBC (англ. *cipher-block chaining*). Ключ MAC використовується як код (Key) на кожному кроці ітерації, і блок повідомлення, що обробляється на поточній ітерації, виступає в якості перетворення відкритого тексту в шифр після побітового додавання з вихідним значенням шифротекста з попереднього кроку. На рис.11.5 наведений приклад реалізації ключової геш-функцій на основі MAC кодів з порівнянням отриманого результату на приймальній стороні (рис.11.5).

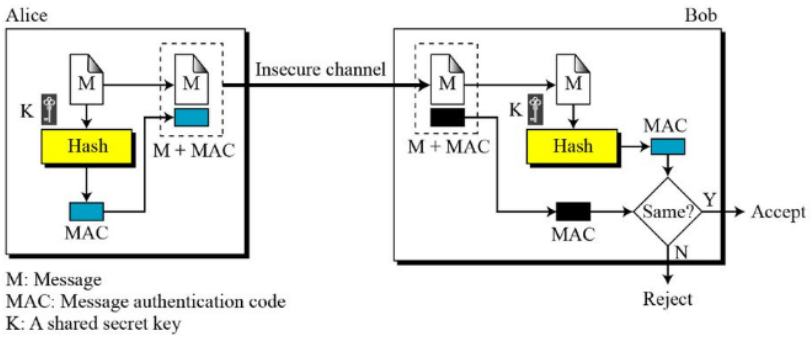


Рис.11.5. Режим реалізації ключової хеш-функцій на основі MAC кодів.

У разі, коли значення хеш-функції залежить не тільки від прообразу, а й закритого ключа, то це значення називають:

- кодом перевірки автентичності повідомлень (Message Authentication Code, MAC);
- кодом перевірки достовірності даних (Data Authentication Code, DAC);
- кодом імітовставки.

Таким чином, найпоширене **застосування хеш-функції** наступне:
- для прискорення пошуку даних в базах даних;
- для перевірки цілісності та автентичності повідомлень;
- для створення стисненого образу, що застосовується в процедурах ЕЦП;
- для захисту пароля в процедурах аутентифікації.

Основні **властивості хеш-функції** наступні:

1. *Детермінованість* – хеш-функція має повертати однакові хеш-значення $h(M)$ для однакових повідомлень M;

2. *Відсутність колізій* – ймовірність співпадіння двох хеш-значень $h_1(M_1)$ та $h_2(M_2)$ для двох різних повідомлень M_1 та M_2 повинна бути дуже низькою.

3. *Односторонність* – за аналізом значенням $h(M)$ неможливо відтворити повідомлення M.

4. *Висока швидкість роботи* – висока швидкість отримання $h(M)$ з повідомлення M.

Основні методи, які використовуються до побудови геш-функцій:

- на основі якої-небудь складнообчислювальної математичної задачі;
- на основі алгоритмів блокового шифрування;
- розроблені з нуля.

11.2. Загальні принципи побудови хеш-функцій

Загальні принципи побудови функцій хешування характеризуються наступними властивостями:

1. Дані, що вводяться (повідомлення, файл і т.д.), розглядаються як послідовність $\boldsymbol{n}$ – бітових блоків;

2. Значення функції хешування одержується шляхом послідовної обробки блок за блоком.
3. Усі проміжні та остаточні значення функції хешування мають *n*-бітове представлення, що в кінцевому результаті призводить до трансформації вхідних даних довільного обсягу (обмеження залежать від виду хеш-функції) до значень хеш-коду фіксованої розмірності (рис.11.6).

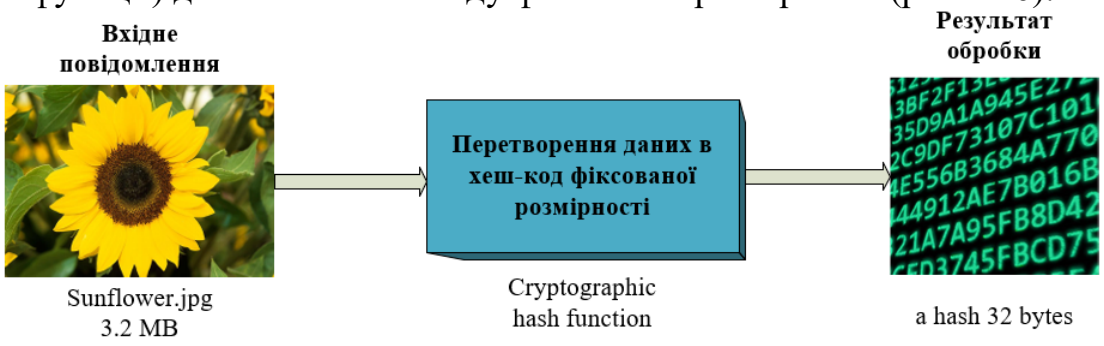


Рис.11.6. Приклад перетворення файла з зображенням в хеш-код фіксованої розмірності.

Прості функції хешування.

Однією з найпростіших реалізацій функцій хешування є зв'язування всіх блоків даних операцією порозрядного виключного «АБО» (XOR):

$$C_i = b_{i1} \oplus b_{i2} \oplus \ldots \oplus b_{im},$$

де C_i - i-й біт хеш-коду, m - число n-бітових блоків вихідних даних, b_{ij} - i-й біт у j-му блоці, $\oplus$ - операція XOR.

Приклад 11.1. Обчислити значення хеш-функції для повідомлення

1100.0010.1010.1100.

Результат обчислення хеш-функції наведений на рис.11.7. у вигляді послідовності C_1-C_4.

	b_1	b_2	b_3	b_4
Блок1	1	1	0	0
⊕	⊕	⊕	⊕	⊕
Блок2	0	0	1	0
⊕	⊕	⊕	⊕	⊕
Блок3	1	0	1	0
⊕	⊕	⊕	⊕	⊕
Блок4	1	1	0	0
	1	0	0	0
	C_1	C_2	C_3	C_4

Рис.11.7. Приклад обчислення простої хеш-функції.

Представимо один із способів реалізації **удосконаленої хеш-функції.** Така реалізація складається з наступних кроків.

1. Початкова ініціалізація *n*-бітового значення функції хешування нульовим значенням.
2. Послідовна обробка *n*-бітових блоків даних за такими правилами:
   - виконання циклічного зсуву поточного значення функції хешування ліворуч на один біт;
   - додавання поточного блоку до значення функції хешування за допомогою операції XOR.

Приклад 11.2. Обчислити значення хеш-функції для повідомлення 1100.0010.1010.1100 за удосконаленою процедурою.

Результат обчислення хеш-функції наведений на рис.11.8.

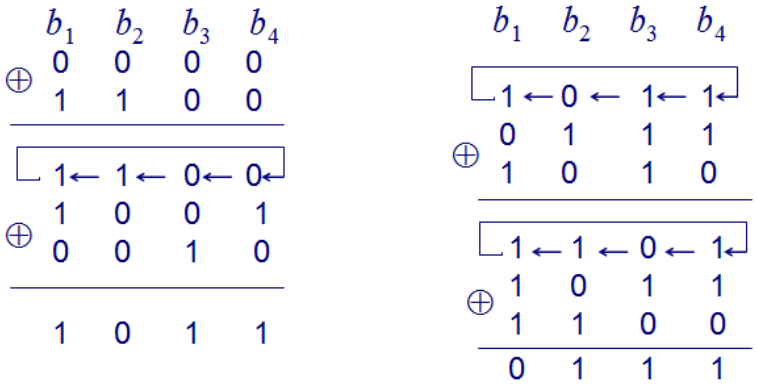


Рис.11.8. Приклад обчислення удосконаленої хеш-функції.

Існують інші, більш складніші і ефективніші подходи до побудови хеш-функцій.

Ефективність хеш-функції – це імовірність (***P***) того, що з появою помилки в даних значення функції хешування залишиться колишнім.

Якщо значення функції хешування *n*-розрядне, то така ймовірність запишеться у вигляді

$$P = 2^{-n}$$

для довільного виду даних і у вигляді

$$P < 2^{-n}$$

для форматованих даних.

Випадкова модель «Oracle».

Випадкова модель «Oracle» була запропонована в 1993 р. Белларом (Bellare) та Роджеєм (Rogaway). Це ідеальна математична модель для хеш-функції. Функція, яка заснована на цій моделі, має такі властивості.

Коли надходить нове повідомлення будь-якої довжини, «Oracle» породжує та виробляє на виході дайджест-повідомлення фіксованої довжини, які складаються із випадкових рядків нулів та одиниць. Це oracle-запис повідомлення та дайджест-повідомлення.

Коли надсилається повідомлення, для якого існує дайджест, «Oracle» просто вставляє дайджест у запис. Дайджест для нового повідомлення має бути вибраний незалежно від попередніх дайджестів. Це означає, що модель

«Oracle» не може використовувати формулу або алгоритм для обчислення дайджесту.

Проблеми «Дня народження».

У теорії ймовірностей парадокс днів народження оцінює ймовірність того, що у випадково вибраній групі людей збігатимуться дні народження в якоїсь пари. В групах кількістю не менших 23 випадково вибраних людей, ймовірність збігу днів народження в якоїсь пари становить більше 50%.

З точки зору вибору хеш-функції висновки з рішення даної проблеми будуть наступними.

Для хеш-коду довжиною *m* біт достатньо обрати

$$2^{m/2}$$

повідомлень, щоб ймовірність збігу хеш-кодів була більша за 0.5.

Приклад 11.3. Нехай оригінальна хеш-функція має довжинк 64 біт і алгоритм піддається атаці на виявлення колізії. В цьому випадку знадобиться провести $2^{64/2} = 2^{32}$ спроб, щоб стверджувати, що ймовірність колізій буде більша за 0.5. Якщо припустити, що за одну секунду зловмисник робить 2^{20} операцій (один мільйон), то таку атаку він здійснить за $2^{32}/ 2^{20}= 2^{12}$ **с –** майже за 1 год. Якщо значення довжини хеш-функціїї збільшиться вдвічі (128), і нехай швидкість перебору буде становити 2^{30} операцій в 1 сек. (більше ніж 1 міліард), то за таких самих умов на це піде $2^{64}/ 2^{30}= 2^{34}$ **с** - більше ніж 500 років!

11.3. Стандартні алгоритми хешування

На сьогодні існує досить великий вибір різноманітних функцій хешування, які мають свої характеристики і криптографічну стійкість. Серед цієї множини найбільш відомі такі алгоритми для побудови хеш-образів повідомлень, як SHA, MD5, RIPEMD, HAVAL, національний стандарт ДСТУ 7564:2014 - «Купина» та ін. Надамо коротку характеристику декількох алгоритмів хешування.

SHA-1 (Secure Hash Algorithm 1) — алгоритм криптографічного хешування для вхідного повідомлення довільної довжини (максимум $2^{64} – 1$ біт, що приблизно дорівнює 2 ексабайти — $2*10^{18}$ байт). Довжина блоку вхідних даних для обробки – 512 біт. Алгоритм генерує 160-бітове хеш-значення, відоме також дайджестом повідомлення. Вважається, що SHA-1 не гарантує достатнього захисту проти атак. Вже в 2005 дослідниками були відкриті методи атаки на SHA-1, які поставили під сумнів тривалість використання цього алгоритму.

SHA-2 – альтернативний варіант алгоритму SHA-1 і є наступною версією, яка дає можливість отримати односторонні хеш-функції SHA-224, SHA-256, SHA-384 і SHA-512. Цифри вказують на довжину хеш-значення в бітах і такі алгоритми є значно криптостійкими у порівнянні з першою версією алгоритму.

MD5 – представник сімейства алгоритмів обчислення геш-функцій MD (Message Digest Algorithm), запропонованого Р. Рівестом і розроблених в 1991 р. Прийшов на зміну MD4, що був недосконалим. Довжина блоку вхідних даних для обробки – 512 біт. Даний алгоритм перетворює інформаційну послідовність довільної довжини на хеш-образ розрядністю 128 біт. Хеш містить 128 біт (16 байт) і зазвичай представляється як послідовність з 32 шістнадцяткових цифр. Існує декілька надбудов до MD5.

MD5 (*HMAC*) — *Keyed-Hashing for Message Authentication* (хешування з ключем для аутентифікації повідомлення) - алгоритм дозволяє хешувати вхідне повідомлення L з деяким ключем K, таке змішування дозволяє аутентифікувати підпис.

MD5 (*Base64*) —отриманий MD5-геш кодується алгоритмом Base64

MD5 (*Unix*) — алгоритм викликає тисячу разів стандартний MD5 для ускладнення процесу. Також відомий як MD5crypt.

«**Купина**» (англ. *Kupyna*) — у грудні 2014 року прийнято національний стандарт ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція хешування». Стандарт набрав чинності з 1 квітня 2015р. Хеш-функція «Купина» використовується зокрема й для створення та перевірки електронного цифрового підпису, що визначений у ДСТУ 4145. Довжина вхідних даних: до $2^{96} - 1$ біт, довжина хешу: від 8 до 512 біт. Варіант, який повертає n біт, позначається як «Купина-n». Кількість ітерацій: 10 для $8 \leq n \leq 256$, 14 для $256 < n \leq 512$. Автори алгоритму запевняють, що диференціальні атаки і rebound-атаки неефективні вже після 4 ітерацій функцій перестановок. Як вказується в літературі, у результаті незалежного криптоаналізу вдалося провести атаку тільки на перші 5 раундів. Складність знаходження колізії для скороченої до 5 раундів функції Купина-256 складає 2^{120}. Даний алгоритм є у відритому доступі, розміщений в бібліотеці cppcrypto — Бібліотека з відкритим вихідним кодом на C++ для використання в програмних додатках.

Стандарт ДСТУ 7564:2014 розроблено задля поступової заміни міждержавного стандарту ГОСТ 34.311-95. Згідно чинного наказу Мінцифри є обов'язковим для застосування після 1 січня 2022р. замість функції хешування за ГОСТ 34.311-95.

Різноманіття побудови хеш-функцій представлено на рис.11.9., де можна побачити сучасні алгоритми отримання хеш-образів.

Окрім того, існує багато онлайнових версій для отримання хеш-значень над данними, зокрема http://www.sha1-online.com. Користувач може швидко отримати хеш-образ за довільно обраною функцією (рис.11.10).

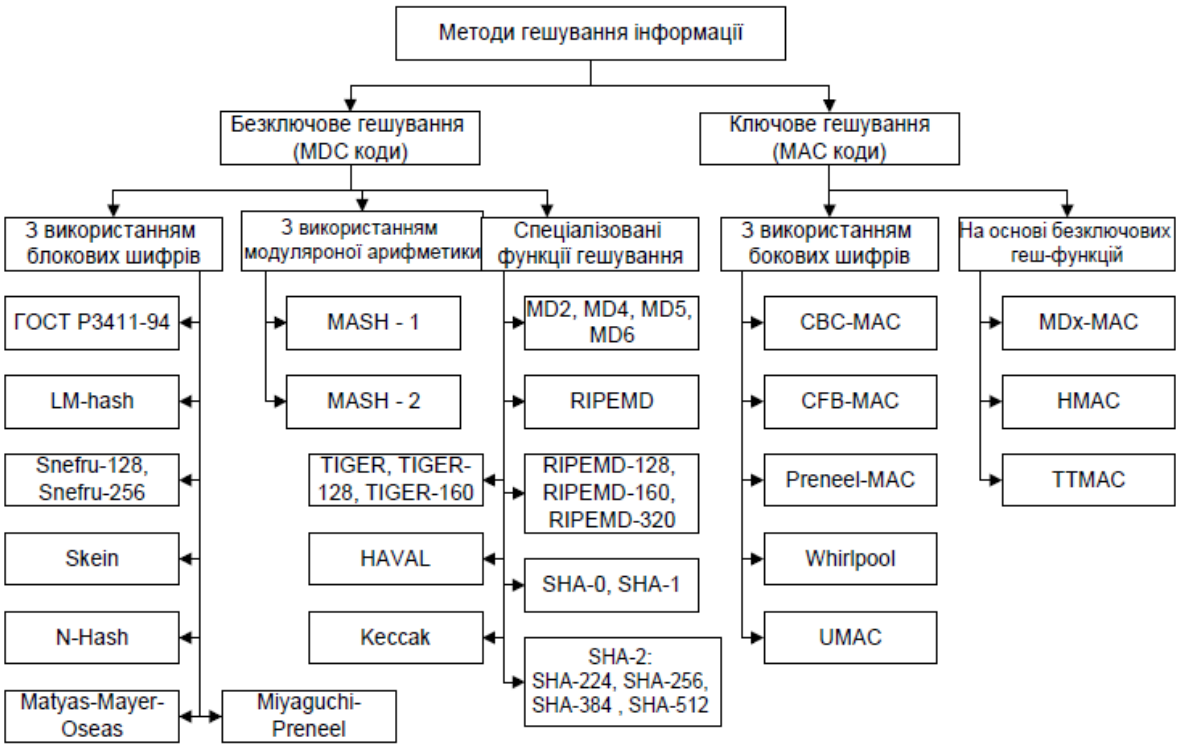


Рис.11.9. Класифікація хеш-функцій

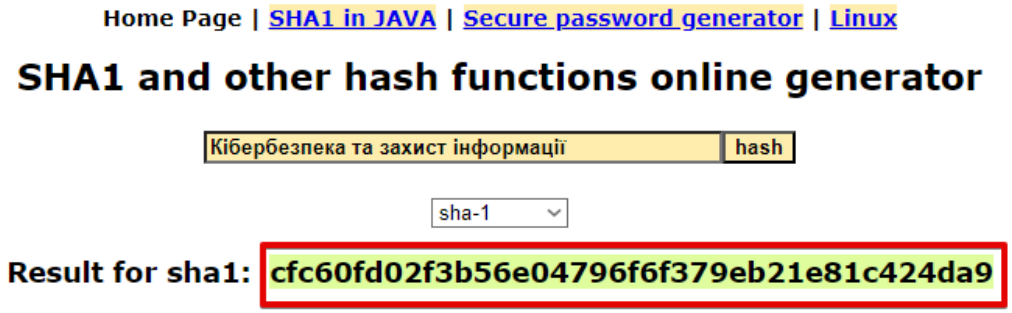


Рис.11.10. Приклад отримання хеш-значення словосволучення «Кібербезпека та захист інформації» при застосуванні стандартної хеш-функції SHA-1 в режимі онлайн (доступ на ресурсі http://www.sha1-online.com)

11.4 Підготовка завдання

Ознайомитися з призначенням, властивостями і вимогами до отримання хеш-функцій, теоретичними положеннями щодо забезпечення цілісності і автентичності даних на прикладі функцій хешування.

11.5 Практичне завдання

В якості практичного завдання стоїть задача побудувати власний алгоритм для отримання хеш-функції для текстового повідомлення, яке

складається з Вашого Прізвища. Для цього необхідно виконати наступні кроки:

1. Написати повністю власне Прізвище студента.
2. Представити своє Прізвище у вигляді ASCII (*American Standard Code for Information Interchange*) 8-бітного (бінарного) коду.
Результат записати у вигляді двійкової цифрової послідовності.

Приклад. Для прізвища «Шевченко» такий код буде мати вид: 10011000.10100101.10100010.11100111. і т.д., де 10011000 представляє літеру «**Ш**», 10100101 – «**е**» і т.д. (табл.11.1)

Таблиця 11.1

Приклад представлення літер у вигляді ASCII коду

Ш	**е**	**в**	**ч**	**е**	**н**	**к**	**о**
152	*165*	*162*	*231*	*165*	*173*	*170*	*174*
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**1**	**1**	**1**	**1**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**	**1**	**1**	**1**
0	**1**	**0**	**1**	**1**	**1**	**0**	**1**
0	**0**	**1**	**1**	**0**	**0**	**1**	**1**
0	**1**	**0**	**1**	**1**	**1**	**0**	**0**

3. Отриману двійкову цифрову послідовність Прізвища розбити на блоки, в кожному з яких знаходиться по 8 біт. Для наведеного прикладу першим блоком T_1 буде «10011000», другим блоком T_2 буде «10100101» і т.д. Всього буде n блоків, кількість яких відповідає кількості літер.
4. Виконати процедуру зчеплення блоків T_1, T_2, …., T_n, як зображено на рис.11.11., де початковий блок C_0 нульовий, C_0 = «00000000». Операції додавання виконуються за модулем 2.

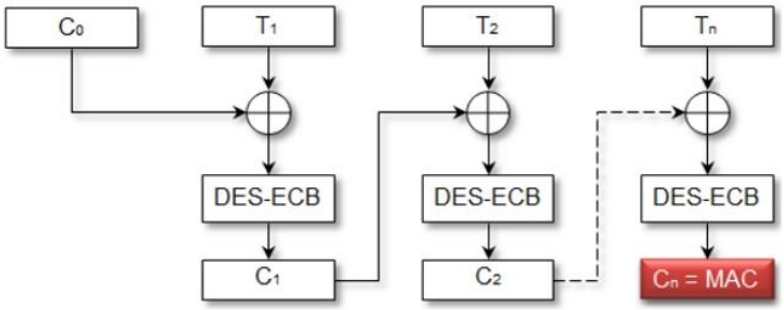


Рис.11.11. Приклад зчеплення блоків

4. Після виконання зчеплення всіх блоків на виході отримується восьми розрядний код (MAC), який і буде результатом хеш-функції над Прізвищем студента.
5. Виконати попередній пункт 4 з реалізацією **ключової хеш-функцій** на основі MAC кодів (див.рис.11.5). Значення ключа обрати самостійно.
6. Написати програму, яка реалізує операції по п.2-5.
7. Внести заміну літери у власному прізвищі, або додати літеру.
8. Повторити виконання дій по п.2-5 і порівняти кінцевий результат хеш образа з попереднім результатом.
9. Надати аналіз отриманим результатам.
10. Ознайомитися з онлайновими ресурсами отримання хеш-коду (наприклад з ресурсу http://www.sha1-online.com або ін.), з програмою обчисленя хеш-функцій HashTab http://implbits.com/products/hashtab/. Набути досвіду отримання значень хеш-функції довільного файлу, порівняння значеннь хеш-функції при внесенні змін до файла при використанні різних хеш-функцій: CRC32, MD5, SHA-1 та ін.
11. Записати значення хеш-функцій (CRC32, MD5, SHA-1) файла до внесених змін і після. Порівняти результати досліджень і занести їх у звіт до лабораторної роботи.

11.6. Зміст протоколу роботи

Протокол до виконаної практичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

11.7. Контрольні питання для самоперевірки

1. Що таке хеш-функція, яке її призначення?
2. Якими основними властивостями має бути наділена хеш-функція?
3. Дати визначення цифрової сигнатури.
4. Що таке колізії при використанні хеш-функції? Від чого залежить виникнення колізій?
5. Які вимоги висуваються до хеш-функції?
6. Чим відрізняються ключові і безключові хеш-функції?
7. Опишіть процедуру виявлення змін в повідомленні на основі MDC-коду.
8. Опишіть режим реалізації ключової хеш-функцій на основі MAC кодів.
9. В чому різниця між MAC і MDC кодами?
10. Опишіть загальні принципи побудови простої і удосконаленої функцій хешування.
11. В чому полягає принцип зчеплення блоків? Опишіть режим зчеплення блоків CBC (*cipher-block chaining).*

12. Що таке ефективність хеш-функції? Дайте визначення стійкості геш-функції.

11.8. Задачі до практикума №11

1. Чи можемо ми використовувати звичайний метод стиснення без втрат, такий, наприклад, як zip, для реалізації криптографічної хеш-функції? Надайте обгрунтування відповіді.
2. Чи можемо ми використовувати функцію контрольної суми як криптографічну хеш-функцію? Надайте обгрунтування відповіді.
3. Нехай використовується алгоритм SHA-512 для отримання хеш-образу повідомлення. Скільки різних документів необхідно проаналізувати, щоб зявилася колізія з ймовірністю більша наж 0.5?
4. Нехай використовується алгоритм SHA-1 для отримання хеш-образу повідомлення. Скільки часу потрібно зловмиснику для атаки на даний алгоритм для знаходження колізії з ймовірністю більшу за 0.5, якщо він проводить 2^{30} тестів за одну секунду методом повного перебору? Надайте обгрунтування відповіді.

11.9. Завдання до самостійної роботи

1. Найпоширеніші стандартні функції хешування: SHA, MD5, RIPEMD, HAVAL. Надайте їх коротку харатеристику, основні параметри та оцінку криптосійкості.
2. Охарактеризуйте стандарт хешування ДСТУ 7564:2014 - «Купина». Основні параметри та перетворення. Надайте оцінку криптостійкості та галузі застосування.
3. Парадокс «Дня народження» та його інтерпритація для аналізу стійкості хеш-функцій.

ПРАКТИКУМ № 12
Розробка та дослідження цифрових підписів на базі алгоритмів RSA та ЕЛЬ-ГАМАЛЯ. Електронний цифровий підпис DSA

Мета роботи. Ознайомитися з принципами побудови електронних цифрових підписів на базі алгоритмів RSA, Ель-Гамаля, DSA.

12.1. Визначення електронного цифрового підпису (ЕЦП), застосування та властивості

Необхідність забезпечення надійного функціонування комп'ютеризованих систем оброблення інформації ставить високі вимоги щодо цілісності та автентичності даних, які надходять, зберігаються та обробляються в цих системах. Автентифікація є процедура, яка встановлює достовірність твердження, що об'єкт (чи суб'єкт) має очікувані властивості. Зокрема, автентифікація повідомлення - перевірка того, що повідомлення було передано без порушення цілісності з очікуваного джерела. Автентифікація здійснюється, виходячи з аналізу структури відповідних даних, за узгодженими алгоритмами.

Одним з найефективніших та найнадійніших підходів, які застосовуються для розв'язку задач, пов'язаних з автентифікацією даних та джерел повідомлень, є процедури **цифрового підписування**, побудовані на основі асиметричних криптографічних алгоритмів [35].

Цифровий підпис повідомлення – це блок даних невеликого розміру, одержаний в результаті криптографічного перетворення повідомлення довільної довжини з використанням особистого (таємного) ключа відправника. Процедура обчислення цифрового підпису побудована таким чином, що кожний цифровий підпис має унікальну структуру, пов'язану з *повідомленням* та *ідентифікаційними даними власника* особистого ключа. Перевірка цифрового підпису полягає в установленні істинності деяких алгебричних співвідношень між цифровим підписом та величинами, обчисленими за повідомленням, виходячи із зв'язку між відкритим та особистим ключами. Цей зв'язок не дає змоги відновити особистий ключ з відкритого. Таким чином, відкритий ключ є унікальний параметр, що дає змогу здійснити перевірку цифрового підпису конкретної особи. Унікальність цифрового підпису і відкритого ключа означає, що обчислювально неможливо визначити особистий ключ цифрового підпису за доступними даними й обчислювально неможливо знайти два повідомлення з однаковим цифровим підписом.

Цифровий підпис забезпечує автентичність повідомлення та неспростовність застосування особистого ключа (автентифікація власника цифрового підпису). Розглядають різні стандарти реалізації ЕЦП, зокрема державний стандарт України «ДСТУ 4145-2002 Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих» - https://dbn.co.ua/load/normativy/dstu/4145/5-1-0-1798).

Електро́нний цифрови́й пі́дпис (англ. *digital signature*)— вид електронного підпису, отриманого за результатом криптографічного

перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача. Електронний цифровий підпис накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа (https://uk.wikipedia.org/wiki/Електронний_цифровий_підпис).

Надійний засіб електронного цифрового підпису — засіб електронного цифрового підпису, що має сертифікат відповідності або позитивний експертний висновок за результатами державної експертизи у сфері криптографічного захисту інформації.

Одним із елементів обов'язкового реквізиту є електронний підпис, який використовується для аутентифікації автора та/або підписувача електронного документу іншими суб'єктами електронного документообігу.

Оригіналом електронного документа вважається електронний примірник з електронним цифровим підписом автора.

Електронний цифровий підпис є складовою частиною інфраструктури відкритих ключів.

ЕЦП, як і звичайний підпис, має:

• гарантувати **істинність документа** шляхом звірення підпису зі зразком;

• гарантувати **авторства документа** (з юридичної точки зору).

До властивостей підписів можна віднести:

• **підпис автентичний**, тобто з його допомогою одержувачу документа можна довести, що він належить власнику (на практиці це визначається графологічною експертизою);

• **підпис не підроблюваний**, тобто служить доказом, що тільки та людина, чий автограф стоїть на документі, міг підписати даний документ, і ніхто інший не зміг би цього зробити;

• **підпис такий, що не переноситься**, тобто є частиною документа і тому перенести його на інший документ неможливо;

• **документ з підписом є незмінним**, тобто після підписування його неможливо змінити, залишивши даний факт непоміченим;

• **підпис незаперечний**, тобто людина, яка підписала документ, в разі визнання експертизою, що саме вона засвідчила даний документ, не може заперечити факт підписання;

• будь-яка особа, що має зразок підпису, може, упевнитися в тому, що даний **документ підписаний власником підпису**.

З переходом до безпаперових способів передачі і зберігання даних, а також з розвитком систем електронного переказу грошових коштів, в основі яких – електронний аналог паперового платіжного доручення, проблема **віртуального підтвердження автентичності** документа набула особливої гостроти.

Для аутентифікації електронних документів, що передаються телекомунікаційними каналами зв'язку, використовується спеціально створюваний **електронний цифровий підпис (ЕЦП)**

Електронний цифровий підпис - реквізит електронного документа, призначений для захисту даного документа від підробки, отриманий в результаті криптографічного перетворення інформації з використанням закритого ключа **ЕЦП,** що дозволяє ідентифікувати власника сертифіката ключа підпису, а також встановити відсутність спотворення інформації в електронному документі.

Основними стандартами ЕЦП є:

1. Міжнародний стандарт **ISO/IEC 9796**, який визначає ЕЦП з відновленням повідомлення (digital signature with message recovery);
2. Міжнародний стандарт **ISO/ IEC 14888**, який визначає ЕЦП з додаванням (digital signature with appendix);
3. Американський національний стандарт цифрового підпису (**FIPS 186**);
4. Американський фінансовий стандарт цифрового підпису з додаванням еліптичною кривої (**ANSI X9.62**);
5. Стандарт на ЕЦП **PKCS #1**, який визначає ЕЦП на основі алгоритму RSA;
6. Стандарт цифрового підпису з додаванням і відновленням повідомлення **IEEE 1363**;
7. Стандарт цифрового підпису з додаванням еліптичної кривої **IEEE P1363**;
8. Міжнародний стандарт **ISO/IEC CD 15946-2** стандартизується ЕЦП еліптичною кривою з додаванням;
9. Державний стандарт України **ДСТУ-4145 – 2002 -** «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка»
   та інші.

За способом побудови схеми ЕЦП діляться на два класи:

1. Схема ЕЦП із відновленням повідомлення;
2. Схема ЕЦП із додаванням повідомлення.

За кількістю учасників ЕЦП підрозділяється на:

1. Одиночну схему ЕЦП;
2. Групову схему ЕЦП.

За способом перевірки ЕЦП поділяються на два класи:

1. Інтерактивні схеми ЕЦП, що вимагають протокольної взаємодії;
2. Не інтерактивні схеми ЕЦП, які не потребують протокольної взаємодії.

Існуючі алгоритми ЕЦП можна розділити також за типами використовуваних односпрямованих функцій із секретом:

1. Схеми ЕЦП, засновані на стійкості факторизації великого числа;
2. Схеми ЕЦП, засновані на стійкості дискретного логарифма;
3. Схеми ЕЦП, засновані на стійкості дискретного логарифма в групі точок ЕК.

Узагальнена класифікція ЕЦП приведена на рис.12.1.

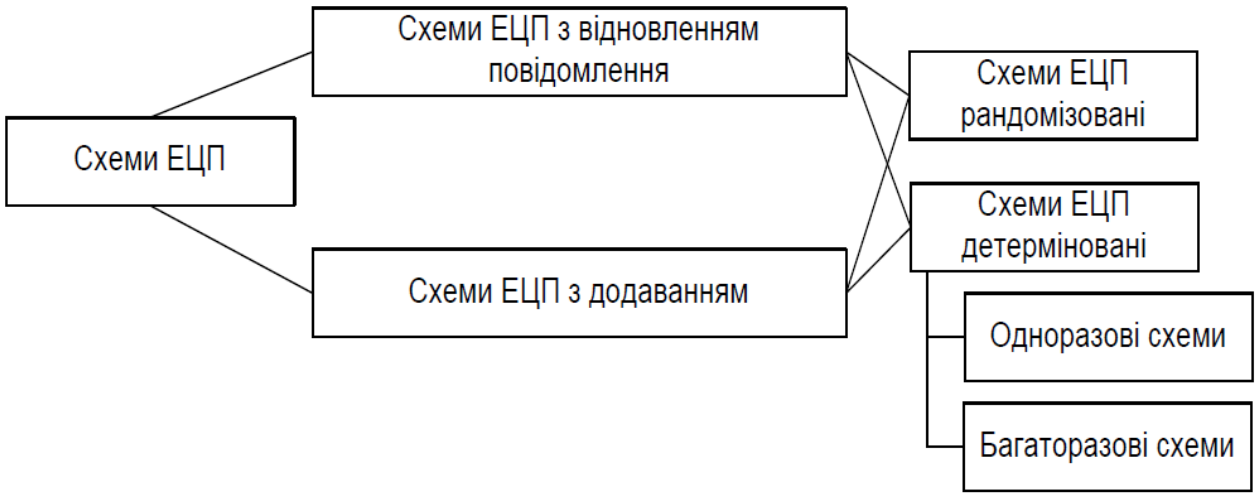


Рис.12.1. Узагальнена класифікація ЕЦП

У схемах ЕЦП із відновленням повідомлення всі або частина підписаного повідомлення може бути відновлена безпосередньо з цифрового підпису. Таким чином, на вхід алгоритму верифікації надходить лише цифровий підпис ***s*.**

У схемах ЕЦП із додаванням цифровий підпис приєднується до повідомлення й у такому вигляді відправляється адресату. Для верифікації такого ЕЦП необхідно мати і підпис ***s,*** і відповідне повідомлення ***m***.

Для опису процесів обробки інформації з використанням механізмів ЕЦП скористаємося такою термінологією:

1. ***Алгоритм генерації ЕЦП*** – це метод формування ЕЦП.

2. ***Алгоритм перевірки (верифікації) ЕЦП*** – метод перевірки того, що підпис є автентичним, тобто дійсно створений конкретним об'єктом і не модифікований при передачі.

3. ***Схема ЕЦП* (або *механізм ЕЦП*) –** сукупність взаємозалежних алгоритмів генерації і верифікації цифрового підпису.

4. ***Процес* (*процедура*) *накладання ЕЦП*** – це сукупність математичного алгоритму генерації ЕЦП і методів представлення (форматування) даних, що підписуються.

5. ***Процес* (*процедура*) *зняття ЕЦП*** – сукупність алгоритму верифікації ЕЦП і методів відновлення даних.

Для побудови схеми ЕЦП необхідно визначити два алгоритми: *алгоритм генерації ЕЦП і алгоритм верифікації ЕЦП.*

1) **Алгоритм верифікації** доступний для всіх потенційних одержувачів підписаних повідомлень;

2) **Алгоритм генерації ЕЦП** відомий тільки особі, яка підписує, що для деякого повідомлення **m** ∈ **M** визначає відповідний підпис **s** ∈ **S**. Верифікатор, одержавши пари **(m, s)** і деяку відкриту інформацію про особу, що підписує, застосовує відповідний алгоритм верифікації ЕЦП. Цей алгоритм видає двійковий результат: "**так**", якщо підпис правильний (автентична) і "**ні**" – у протилежному випадку.

Існує кілька методів побудови схем ЕЦП, а саме:

1. шифрування електронного документа (ЕД) на основі симетричних алгоритмів. Дана схема передбачає наявність у системі **третьої особи (арбітра),** що користується довірою учасників обміну підписаними подібним чином електронними документами;
2. використання **асиметричних алгоритмів шифрування**. Фактом підписання документа в даній схемі є зашифрування документа секретним ключем його відправника. Ця схема теж використовується досить рідко внаслідок того, що довжина ЕД може виявитися критичною;
3. розвитком попередньої ідеї (2) стала найбільш поширена схема ЕЦП, а саме зашифрування остаточного результату обробки ЕД **хеш-функцією** за допомогою **асиметричного алгоритму**.

Розглянемо деякі алгоритми реалізації ЕЦП.

12.2. Реаліація ЕЦП на основі алгоритму RSA

Наведемо приклад генерації ЕЦП, спрощена схема якого представлена на рис.12.2. За даною схемою виконуються наступні дії.

1) Учасник **A** (**Alice**) обчислює геш-код від ЕД. Отриманий геш-код проходить процедуру перетворення з використанням свого секретного ключа. Отримане перетворення (яке і є ЕЦП) разом з ЕД відправляється учаснику **B** (**Bob**).

2) Учасник **B** отримує ЕД з ЕЦП та сертифікований відкритий ключ від учасника **A**. Учасник **B** проводить розшифрування ЕЦП за допомогою цього ключа і отримує значення геш-коду, яке передавалося. Окрім того, ЕД піддається операції гешування, після чого результати порівнюються. Якщо два геш-коди співпадають, то ЕЦП визнається істинним, в іншому випадку помилковим.

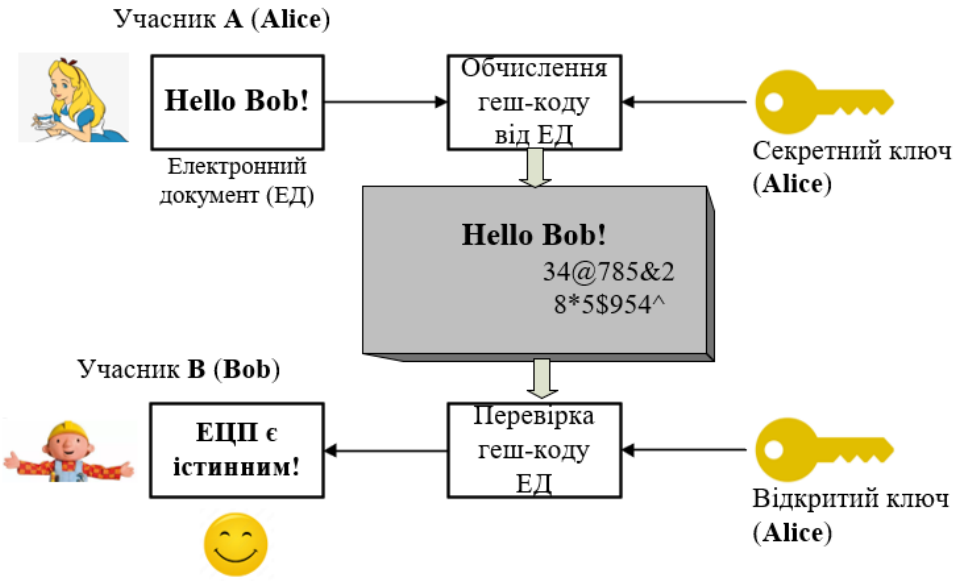


Рис.12.2. Структура зашифрування ЕД геш-функцією

Розширена структура формування ЕЦП представлена на рис.12.3.

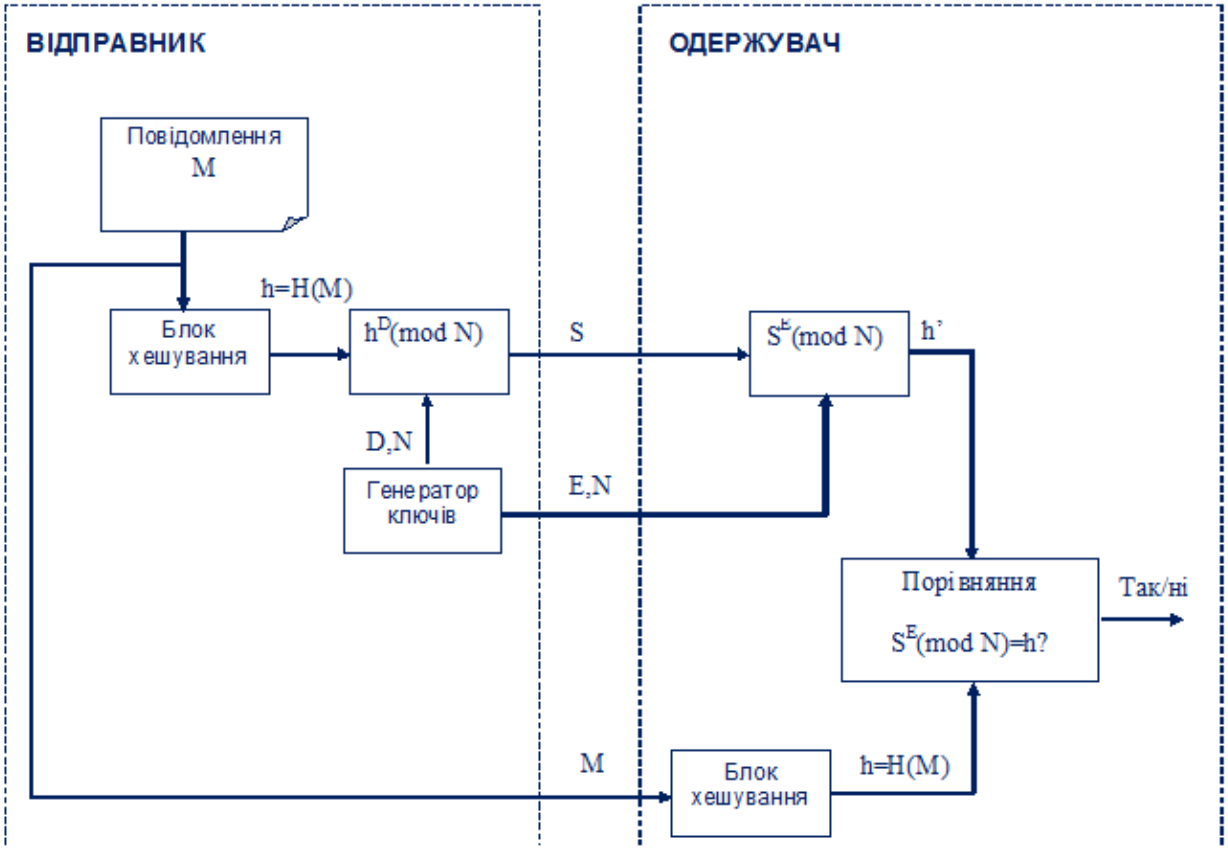


Рис.12.3. Структурна схема формування ЕЦП

Із заданого повідомлення **M** формується геш-функція **h=H(M)**, яка перетворюється за допомогою секретного ключа **D** і відкритого ключа **N** в ЕЦП виду **S**. Одержувач **B** на приймальній стороні отримує ЕЦП виду **S** і при застосуванні пари відкритих ключів **(E,N)** отримує геш-функцію **h*.** Значення

цієї геш-функції порівнюється з геш-функцією **h**, яка отримана з оригінального повідомлення М. При свівпаданні **h*** і **h** робиться висновок про ідентичність документів, відправлених відправником одержувачу.

Алгоритм вибору ключів для ЕЦП:

1. Вибираються два великих простих числа ***p*** і ***q***.
2. Обчислюються ***N=p*q*** **і** ***φ*** **(*N*) = (*p*-1)(*q* -1).**
3. Вибирається число ***E***, що задовольняє умовам

$$E < \varphi(N),\ \text{НСД}(E, \varphi(N)) = 1,$$

де *φ (N) – функція Ейлера, або кількість взаємопростих чисел в діапазоні від 1 до N-1*.

Обчислюється ***D***, що задовольняє умові $E*D \equiv 1 (\text{mod} \varphi(N))$.

Пара чисел **(*D, N*)** зберігається відправником, як **секретний ключ** для підписування.

Пара чисел **(*E, N*)** є **відкритим ключем**, яка відправником передається одержувачам для перевірки його цифрових підписів.

ЕЦП повідомлення $h(M)$ на основі алгоритму RSA буде мати вигляд:

$$S = h(M)^d \bmod N.$$

Приклад 12.1. На основі застосування алгоритму RSA підписати та перевірити підпис для повідомлення *M,* хеш-функція якого $h(M)=88$.

Нехай $p=17$ і $q=11$, тоді $N=p \cdot q=17 \cdot 11=187$.

Обчислимо функцію Ейлера - $\varphi(187)=16 \cdot 10=160$.

Виберемо відкритий ключ $e=7$ та перевіримо умови:

$$1<7<160,\ \text{НСД}(7, 160) =1.$$

Знайдемо закритий ключ $D=23$ за розширеним алгоритмом Евкліда з рівняння $7*D \equiv 1 (mod\ 160)$.

Обчислимо ЕЦП за допомогою закритого ключа підписувача:

$$S = h(M)^d \bmod N = 88^{23} mod\ 187=11.$$

Для перевірки ЕЦП повідомлення M обчислимо його хеш-значення $h(M)=88$ та порівняємо із значенням, отриманим за допомогою відкритого ключа:

$$S^e\ mod\ N=11^7\ mod\ 187=88.$$

Таким чином можна стверджувати, що ЕЦП справжній.

12.3. Реаліація ЕЦП на основі алгоритму Ель-Гамаля (El Gamal Signature Algorithm)

Наведемо приклад формування ЕЦП на базі алгоритму **Ель-Гамаля (**El Gamal Signature Algorithm).

ВІДПРАВНИК

1) Відправник, який має підписати свій документ, вибирає деяке велике **просте ціле число** ***P***. Це число є **відкритим** і передається усім отримувачам документів відправника. Реальні значення близькі до $2^{1024}(\approx 10^{308})$
Наприклад, відправник вибирає число ***P = 11.***

2) Відправник вибирає також велике ціле число ***G***, яке має задовольняти умовам $1 < G < P$. Це число також є **відкритим** і передається усім отримувачам документів відправника. Реальні значення близькі до $2^{512}(\approx 10^{154})$.
Наприклад, відправник вибирає число ***G = 2.***

3) Відправник вибирає ціле число ***X***, яке має задовольняти умовам $1<X<P$. Це число є **секретним ключем** відправника для підписування документів.
Наприклад, відправник вибирає секретний ключ **X= *8***.

4) Відправник обчислює число $Y=G^X \bmod P$.
Це число є **відкритим ключем** відправника, яке використовується отримувачами для перевірки його підпису. Це число також передається усім отримувачам документів відправника.
Наприклад, відправник обчислює число $Y = G^X \bmod P = 2^8 \bmod 11 = 3$.

5) Відправник обчислює хеш-значення ***H*** свого документу **M**. Кількість біт хеш-значення має бути на **одиницю меншою** кількості біт значення ***P***.

У загальному випадку геш-значення ***H*** документу **M** відправника має задовольняти умовам $1 < H < (P - 1)$.

Наприклад, відправник обчислює геш-значення ***H*** свого документу ***M*** і отримує ***H = 5.*** Згадані умови стосовно кількості біт виконуються. Дійсно, значення ***H=5*** має **3 біта (101)**, а значення $P - 1 = 10$ має **4 біта (1010)**.

6) Відправник вибирає випадкове ціле число ***K***, яке має задовольняти умовам $1 < K < (P - 1)$. Крім того, числа ***K*** і ***P-1*** мають бути **взаємно простими**, тобто їх найбільший спільний дільник $НСД(K, P-1) = 1$.
Це число також є **секретним числом відправника** для підписування його документу **M**.

Наприклад, відправник вибирає випадкове ціле число $K = 9$.

7) Відправник обчислює ціле число $a = G^K \bmod P$. Це число є першою складовою електронного цифрового підпису документу.

Наприклад, відправник обчислює число $a = G^K \bmod P = 2^9 \bmod 11 = 6$.

8) Відправник знаходить ціле число ***b*** з наступного рівняння

$$H=(X\cdot a + K\cdot b) \bmod (P-1).$$

Це число є другою складовою електронного цифрового підпису документу.

Наприклад, відправник знаходить ціле число ***b*** із рівняння:

$$H=(X\cdot a + K\cdot b) \bmod (P - 1) \quad \text{або} \quad 5 = (8\cdot 6 + 9\cdot b) \bmod 10.$$

Це число можна знайти шляхом послідовного перебору:

$b = 1;\ (8\cdot 6 + 9\cdot 1) \bmod 10 = 7;$

$b = 2;\ (8\cdot 6 + 9\cdot 2) \bmod 10 = 6;$

$\mathbf{b = 3;\ (8\cdot 6 + 9\cdot 3) \bmod 10 = 5.}$

Таким чином отримуємо, що $\mathbf{b = 3}$.

9). Відправник передає отримувачу документ **M**, а також його електронний цифровий підпис у вигляді пари чисел $\mathbf{S = (a, b).}$

Наприклад, відправник передає отримувачу документ **M,** а також його **електронний цифровий підпис** у вигляді пари чисел $\mathbf{S = (6, 3).}$

Таким чином підписане повідомлення буде мати вид: **(M, a, b)**.

ОТРИМУВАЧ

Отримавши документ M, а також його електронний цифровий підпис

$$\mathbf{S = (a, b),}$$

отримувач повинен перевірити, чи відповідає цей підпис документу.

1) Отримувач обчислює хеш-значення **H** отриманого документу **M**.

Наприклад, отримувач обчислює геш-значення $\boldsymbol{H}$ документу **M** і отримує $\mathbf{H = 5}$.

2) Отримувач обчислює ціле число $\mathbf{A_1 = (Y^{a} * a^{b}) \bmod P.}$

Наприклад, отримувач обчислює ціле число

$$\mathbf{A_1 = (Y^a * a^b) \bmod P = (3^6 * 6^3) \bmod 11 = 10.}$$

3) Отримувач обчислює ціле число $\mathbf{A_2 = G^{H} \bmod P.}$

Наприклад, отримувач обчислює ціле число

$$\mathbf{A_2 = G^{*H} \bmod P = 2^{*5} \bmod 11 = 10.}$$

4) Отримувач порівнює знайдені числа $\boldsymbol{A_1}$ **і** $\boldsymbol{A_2}$. Ці числа будуть рівні тоді і тільки тоді, коли електронний цифровий підпис відповідає документу, тобто відправником документу M є дійсно власник секретного ключа X, і що відправник підписав саме цей документ M.

ЕЦП $\mathbf{S = (6, 3)}$ відповідає отриманому документу $\boldsymbol{M}$. Використання алгоритму електронного цифрового підпису Ель-Гамаля вимагає вибору щоразу іншого випадкового цілого числа **K**. Інакше, якщо зловмисник розкриє повторно використовуване відправником число **K**, то він зможе розкрити і його секретний ключ **X**.

Отже, алгоритм Ель-Гамаля являє собою протокол з обчисленнями і цифровим підписом вважається пара чисел. Обчислення що проводяться в протилежну сторону базуються на практично нерозв’язній задачі дискретного логарифмування.

Приклад 12.2. Нехай обрано наступні параметри для формування ЕЦП на основі алгоритму Ель-Гамаля:

$$P = 19;\ G = 10;\ K = 5;\ H(M) = 14;\ X = 16,$$

де $H(M)$ - хеш-значення повідомлення M.

Відправник обчислює відкритий ключ:

$$Y = G^X \bmod P = 10^{16} \pmod{19} = 4.$$

Обчислює параметр a, який і буде одним з елементів ЕЦП:

$$a = G^K \bmod P = 10^5 \pmod{19} = 3.$$

Другий параметр ЕЦП b знаходимо з рівняння:

$$H = (X*a + K*b) \bmod (P - 1) \quad \text{або} \quad 14 = (16*3 + 5*b) \bmod 18:$$

$$b = 1;\ (16*3 + 5*1) \bmod 18 = 17\ ;$$
$$b = 2;\ (16*3 + 5*2) \bmod 18 = 4\ ;$$
$$b = 3;\ (16*3 + 5*3) \bmod 18 = 9\ ;$$
$$b = 4;\ (16*3 + 5*4) \bmod 18 = 14.$$

Таким чином, знайдений параметр $b = 14$.
Отже, за цифровий підпис приймається повідомдення **(M, 3, 4).**
Перевіримо, чи ЕЦП дійсний.

$$A_1 = (Y^a * a^b) \bmod P = (4^3 * 3^4) \bmod 19 = 16$$

$$A_2 = G^{*H} \bmod P = 10*14 \bmod 19 = 16$$

Таким чином можна зробити висновок, що підпис дійсний.

12.4. Реаліація ЕЦП на основі алгоритму DSA (Digital Signature Algorithm)

Національний інститут стандартів і технології США (NIST) затвердив федеральний стандарт цифрового підпису DSS (Digital Signature Standard). Для створення цифрового підпису використовується алгоритм DSA (Digital Signature Algorithm). В якості хеш-коду стандарт передбачає використання алгоритму SHA-1 (Secure Hash Algorithm). В стандарті DSA підпис становить собою два великих цілих числа, які отримані у відповідності до процедур, прописаних в DSS. Криптографічна стійкість системи в цілому ґрунтується на складності знаходження дискретних логарифмів у скінченних полях.

У стандарті DSS використовується перетворення, які призначені забезпечити лише функцію ЕЦП. На відміну від RSA, DSA не може бути використано для шифрування чи обміну ключами.

Алгоритм генерації ключів.

1. Генерується просте число p, таке що $2^{L-1} < p < 2^L$, $512 \le L \le 1024$ і L кратне 64.
2. Обирається q – простий дільник $p-1$, таке $2^{159} < q < 2^{160}$.

3. Обчислюється $g=h^{(p-1)/q} \, mod \, p$, де h будь-яке ціле число таке, що $0 \leq h \leq p-1$ та $h^{(p-1)/q} \, mod \, p > 1$.
4. Вибирається x – випадкове ціле число, таке що $0<x<q$.
5. Обчислюється $y=g^x \bmod p$.
6. x і y є закритим і відкритим ключами, відповідно.

Вибирається випадкове ціле число k – разовий секретний ключ, де $0<k<q$. Підпис повідомлення M із використанням закритого ключа підписувача виглядає наступним чином:

$$r = (g^k mod \, p) \, mod \, q \quad \text{та} \quad s = k^{-1}(h(M) + x * r) \, mod \, q,$$

де $h(M)$ – значення хеш-функції SHA-1 від повідомлення M.
Підписом для повідомлення $\boldsymbol{M}$ є **пара $(\boldsymbol{r}, \boldsymbol{s})$**.

Перевірка підпису із використанням відкритого ключа підписувача виглядає наступним чином. Обчислюються наступні значення:

$$\boldsymbol{w = s^{-1} mod \, q, \; u_1 = (h(M) * w) \, mod \, q,}$$

$$\boldsymbol{u_2 = (r * w) \, mod \, q, \; v = (g^{u_1} y^{u_2}) \, mod \, q.}$$

Підпис дійсний, якщо $v=r$.

Приклад 12.3. Нехай обрано наступні параметри для формування ЕЦП:

p = 211; q = 7; g = 144; k = 2; x = 3;

значення хеш-функції від повідомленн - $h(M)$ = 15.

Розрахуємо відкритий ключ користувача:

$$y = 144^2 \text{ (mod 211)} = 58.$$

Проведемо створення ЕЦП:

$$r = [144^3 \text{ (mod 211)}] \text{ mod 7} = 123 \text{ (mod 7)} = 4;$$

$$s = [3^{-1}(15 + 2*4)] \text{ mod 7} = (3^{-1}*23) \text{ mod 7} = 3.$$

Таким чином, отриманий ЕЦП має значення **(4, 3)**.
Перевірка підпису:

$$w = (3^{-1}) \text{ mod 7} = 5; \; u_1 = (15*5) \text{ mod 7} = 5; \; u_2 = (4*5) \text{ mod 7} = 6;$$

$$v = [(144^5 * 58^6) \text{ mod 211}] \text{ mod 7} = 123 \text{ (mod 7)} = 4.$$

Таким чином, $\boldsymbol{v = r = 4}$, отже підпис є справжній.

12.5. Підготовка до завдання

Ознайомитися з теоретичними положеннями механізмів забезпечення цілісності і автентичності на прикладі застосування електронних цифрових підписів.

12.6. Практичне завдання

В якості лабораторного завдання стоїть задача утворити ЕЦП з текстового повідомлення, яке складається з Вашого Прізвища. Для цього необхідно виконати наступні кроки

Для **ВІДПРАВНИКА:**

1. Написати повністю власне Прізвище студента.
2. Представити Прізвище у вигляді ASCII (*American Standard Code for Information Interchange*) 8-бітного (бінарного) коду. (див. п.6.8. до л.р. №6).
3. Скористатися результатом попередньої роботи №11 по утворенню геш-функції з власного прізвища, записавши його як ***H***.
4. Обрати просте число ***P***, ціле число ***G***, які є відкритими. У загальному випадку геш-значення ***H*** документу ***M*** відправника має задовольняти умовам ***1 <H <(P - 1)*** (дивись приклад вище).
5. Задатися секретним числом **X** так, щоб ***1<X<P.***
6. Очислити відкритий ключ відправника $Y=G^X \bmod P$.
7. Обрати випадкове **секретне** ціле число ***K***, яке має задовольняти умовам ***1<K<(P-1).*** Крім того, числа ***K*** і ***P-1*** мають бути **взаємно простими**, тобто їх найбільший спільний дільник ***НСД(K, P-1) = 1***.
8. Обчислити ціле число $a = G^K \bmod P$. Це число (***a***) є <u>першою складовою</u> електронного цифрового підпису документу.
9. Знайти ціле число ***b***, використовуючи рівняння

$$H=(X\cdot a + K\cdot b) \bmod (P-1).$$

Це число (***b***) є <u>другою складовою</u> електронного цифрового підпису документу.

10. Відправник передає отримувачу документ **M**, а також його електронний цифровий підпис у вигляді пари чисел ***S = (a, b).***

ДЛЯ **ОТРИМУВАЧА:**

11. Отримавши документ **M**, а також його електронний цифровий підпис

$$S = (a, b),$$

отримувач (це так само ВИ) повинен перевірити, чи відповідає цей підпис документу.

Отримувач обчислює геш-значення ***H*** отриманого документу **M**. У Вашому випадку цей пункт вже виконано (див.п.3).

12. Отримувач обчислює ціле число $A_1 = (Y^a * a^b) \bmod P.$
13. Отримувач обчислює ціле число $A_2 = G^H \bmod P.$
14. Отримувач порівнює знайдені числа A_1 **і** A_2. Ці числа будуть рівні тоді і тільки тоді, коли електронний цифровий підпис відповідає документу, тобто відправником документу **M** є дійсно власник секретного ключа **X**, і що відправник підписав саме цей документ **M**.
15. Написати програму, яка реалізує операції по п.2-14.
16. Надати аналіз отриманим результатам.

12.7. Зміст протоколу роботи

Протокол до виконаної лабораторної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

12.8. Контрольні питання для самоперевірки

1. Що таке ЕЦП та його застосування?
2. Надайте класифікацію ЕЦП та основні властивості.
3. Визначіть поширені стандарти ЕЦП.
4. Визначіть основні відмінності ЕЦП із відновленням та додаванням повідомлення, одиночна та групова ЕЦП.
5. Пояснити терміни «Алгоритм генерації ЕЦП» та «Алгоритм верифікації ЕЦП».
6. Надайте класифікацію ЕЦП за типом односпрямованих функцій.
7. Опишіть реалізацію ЕЦП на основі RSA.
8. Опишіть реалізацію ЕЦП на основі алгоритму Ель-Гамаля.
9. Опишіть реалізацію ЕЦП на основі DSA.
10. На чому ґрунтується криптостійкість алгоритмів шифрування даних: Ель-Гамаля, RSA, DSA, ECDSS?

12.9. Задачі до практикуму №12

1. Сформуйте ЕЦП для хеш-значення повідомлення $h(M)$=8 за алгоритмом RSA при p=13, q=17 та самостійно визначеному відкритому ключі e. Провести обчислення закритого ключа d.

2. Сформуйте ЕЦП для хеш-значення повідомлення $h(M)$=10 за алгоритмом Ель-Гамаля при P=17 та G=7. Самостійно оберіть закритий ключ X, сесійний ключ K та обчисліть відкритий ключ Y.

3. Виконати перевірку ЕЦП для алгоритму Ель-Гамаля при значенні хеш-функції повідомлення $H(M)$=8, (a=15, b=22), якщо P=29 та G=8, відкритий ключ Y=13, сесійний ключ K=9.

4. В корпоративній мережі використовується ЕЦП стандарту DSA із загальними параметрами p = 251; q = 25; $h(M)$ = 17. Для заданих секретних параметрів абонентів знайти відкритий ключ Y і побудувати підпис для повідомлення M та виконати перевірку підпису:

   a) k = 11, x = 2, g = 64;
   b) k = 15, x = 6, g = 219.

5. В корпоративній мережі використовується ЕЦП стандарту DSA із загальним параметром p = 251. Знайти відкритий ключ Y і побудувати ЕЦП для повідомлення m = h(M) та виконати перевірку підпису:

   a) k = 11, x = 3, g = 20; q = 25; h(M) = 59.
   b) k = 15, x = 6, g = 21; q = 125; h(M) = 93.

12.10. Завдання до самостійної роботи

1. Державний стандарт України ДСТУ-4145 – 2002 - «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка». Криптостійкість, основні принципи побудови.
2. Стандарт цифрового підпису ECDSS (Elliptic Curve Digest Signature Algorithm). Криптостійкість, основні принципи побудови.
3. Поняття атаки на ЕЦП та їх різновиди.
4. Поняття «сліпого підпису», групового підпису, розподіленого підпису, конфіденційного підпису, беззаперечного підпису.

ПРАКТИКУМ № 13
Методи побудови криптографічних протоколів автентифікації

Мета роботи. Ознайомитися з призначенням, класифікацією та основними принципами побудови протоколів автентифікації. Побудувати протоколи автентифікації з нульовою передачею знань.

13.1. Поняття ідентифікації, автентифікації та авторизації об’єкта

Ідентифікація, аутентифікація та авторизація об’єкта.

Кожен об'єкт, що бере участь у функціонуванні комп'ютерної системи (КС), пов'язується з певними наборами даними, які ідентифікують цей об'єкт. Ці ідентифікаційні дані можуть бути числами, рядками тексту або чимось іншим. Якщо ідентифікаційні дані зареєстровані в КС, то відповідний об'єкт вважається законним. З іншого боку, якщо ідентифікаційні дані не зареєстровані в КС, то об'єкт вважається нелегальним.

Коли об'єкт намагається приєднатися до КС для участі у її роботі, система захисту КС виконує три захисні процедури:

1) ідентифікація об’єкта;
2) автентифікація об’єкта;
3) авторизація об’єкта;
4) адміністрування об’єкта;

Послідовність дій об'єктів (суб'єктів) для досягнення певної мети називається **протоколом** [7, 36].

Протоколи, у яких використовуються криптографічні перетворення інформації, називаються **криптографічними протоколами**.

Однією з найважливіших завдань забезпечення безпеки є розробка методів та засобів, які дозволяють перевіряючій стороні переконатися в ідентичності сторони, яка намагається довести свою правдивість.

Основним міжнародним стандартом з криптографічних протоколів автентифікації є стандарт Міжнародної організації зі стандартизації та Міжнародної електротехнічної комісії ISO/ IEC9798 – Information technology – Security techniques – Entity authentication, що складається з наступних складових [37]:

– ISO/IEC9798–1:2010 Part 1: General;
– ISO/IEC9798–2:2019 Part 2: Mechanisms using authenticated encryption;
– ISO/IEC9798–3:2019 Part 3: Mechanisms using digital signature techniques;
– ISO/IEC9798–4:1999 Part 4: Mechanisms using a cryptographic check function;
– ISO/IEC9798–5:2009 Part 5: Mechanisms using zeroknowledge techniques;
– ISO/IEC9798–6:2010 Part 6: Mechanisms using manual data transfer.

Розглянемо основні терміни та методи реалізації протоколів автентифікації.

Ідентифікація об’єкта (*Identification*) - є першою лінією захисту в протоколі безпеки комп’ютерної системи. Перед отриманням доступу до системи об’єкт повинен ідентифікувати себе шляхом надання КС свого імені та унікального ідентифікаційного номера (ідентифікатора). Ім'я об'єкта використовується для зв'язку з системою, а ідентифікатор служить окремою міткою для об'єкта. Комп’ютерна система звіряє ідентифікатор з реєстром зареєстрованих об’єктів. Якщо ідентифікатор дійсний, до об’єкта надається доступ і всі його ідентифікаційні дані записуються. Потім система переходить до виконання подальших операцій безпеки, пов’язаних з об’єктом. Однак, якщо об’єкт буде визнано незаконним, залежно від політики безпеки система може заборонити доступ або вимагати від об’єкта пройти процедури реєстрації.

Автентифікація об’єкта (*Authentication*) в системі захисту КС полягає у перевірці, чи є цей об'єкт тим, за кого він себе видає. Перевірка може проводитися шляхом запиту пароля, однак цей метод не є безпечним, оскільки може стати доступним зловмисникам. У разі впевненості в правах КС, об'єкт повідомляє свій пароль, і після перевірки КС підтверджує його істинність. У разі невпевненості в правах КС, використовуються інші ідентифікаційні дані, такі як елементи апаратного забезпечення об'єкту або характерні риси особистості.

Ідентифікаційні дані можуть включати в себе ключі, магнітні картки, інші елементи апаратного забезпечення, які можуть бути унікальними для кожного об'єкту. Також можуть використовуватися характеристики особистості, такі як відбитки пальців, тембр голосу, особливості поведінки та стиль роботи, освітній рівень, вихованість, звички тощо.

Оскільки ці дані є унікальними для кожного об’єкта, вони можуть бути використані для підтвердження ідентичності об’єкта без необхідності використання пароля. Використання цих та інших ідентифікаційних даних може бути більш безпечним, оскільки вони не можуть бути отримані зловмисниками шляхом перехоплення.

У загальному контексті, аутентифікація об'єкта є важливою складовою захисту КС та забезпечення безпеки інформації, що зберігається в системі. КС повинна використовувати найбільш ефективні методи аутентифікації об'єктів, щоб унеможливити несанкціонований доступ до конфіденційної інформації та запобігти можливим кібератакам.

Протоколи автентифікації — це методи або процедури, які використовуються для перевірки ідентичності користувача, пристрою або системи. Ці протоколи призначені для забезпечення того, щоб лише авторизовані користувачі або пристрої мали доступ до захищених ресурсів, а також для запобігання несанкціонованому доступу або втручанню.

Виділяють декілька видів автентифікації.

Слабка автентифікація. Автентифікація за допомогою пароля, яку називають однофакторною або слабкою, є традиційною, але не є надійною, оскільки при наявності достатніх ресурсів зловмисник може перехопити або підібрати пароль. Людський фактор також грає важливу роль - пароль, який складно зламати, часто складно запам'ятати, що збільшує ймовірність його записування, підвищуючи ризик його перехоплення або викрадення. З іншого боку, паролі, які легко запам'ятовувати (наприклад, часто вживані слова або фрази, дати народження, імена близьких, назви моніторів чи найближчого обладнання), не є надійними в плані стійкості до злому. Щоб вирішити цю проблему, використовують одноразові паролі, але і вони можуть бути перехоплені зловмисником.

Багатофакторна автентифікація - автентифікація, що здійснюється з використанням двох чи більше факторів. Наприклад, згідно з вимогами Європейської директиви PSD2 (Payment Service Directive 2), в платіжних системах використовується посилена автентифікація, яка передбачає використання принаймні двох різних типів факторів для автентифікації. Ці фактори можуть бути класифіковані як:

- щось, чим володіє суб'єкт, наприклад, фізичний об'єкт або характеристика;
- знання - інформація, якою володіє суб'єкт;
- володіння - річ, якою володіє суб'єкт.

Двофакторна автентифікація – це тип автентифікації, який вимагає від користувача надати дві форми ідентифікації, наприклад пароль і маркер безпеки, щоб увійти в систему або отримати доступ до захищеного ресурсу. Двофакторна автентифікація може забезпечити додатковий рівень безпеки, але може бути *незручною для користувачів* і може потребувати додаткової інфраструктури для підтримки.

Біометрична автентифікація – це тип автентифікації, який використовує фізичні або поведінкові характеристики, як-от відбиток пальця або розпізнавання обличчя, для перевірки особи користувача. Біометрична автентифікація може бути дуже безпечною, але може бути дорогою для реалізації та може не працювати належним чином для всіх користувачів (наприклад, через відмінності у фізичних характеристиках).

Сувора автентифікація. Автентифікація, під час якої використовується інформація без розкриття цієї інформації. Як правило, реалізується за допомогою асиметричних криптографічних алгоритмів.

Приклад 13.1. Автентифікація при використанні карток.

Автентифікація користувача по паролю будується лише по одному фактору: людина знає певний набір символів. Альтернативою цього методу є двофакторна автентифікація. У цьому випадку у користувача є персональний ідентифікатор (перший фактор), і він має певну спеціальну інформацію (PIN-

коди, другий фактор). Як ідентифікатор зазвичай виступають смарт-карти, токени, таблетки iButton і т.д. Картки поділяють на два типи:

- *пасивні* (картки з пам'яттю);
- *активні* (інтелектуальні картки).

Найпоширенішими є пасивні картки з магнітною смугою, які зчитуються спеціальним пристроєм, що процесор. У разі використання вказаної картки користувач вводить свій ідентифікаційний номер. У разі збігу з електронним варіантом, закодованим у картці, користувач отримує доступ до системи. Це дозволяє достовірно встановити особу, яка отримала доступ до системи та виключити несанкціоноване використання картки зловмисником (наприклад, при її втраті). Такий спосіб часто називають двокомпонентною автентифікацією. Іноді (зазвичай при фізичному контролі доступу) картки застосовують без запиту особистого ідентифікаційного номера.

До переваги використання карток відносять те, що обробка автентифікаційної інформації виконується пристроєм зчитування без передачі в пам'ять комп'ютера. Це виключає можливість електронного перехоплення каналами зв'язку. До недоліку пасивних карток відносять: вони істотно дорожчі у порівнянні простого введення пароля, вимагають спеціальних пристроїв читання, їх використання передбачає спеціальні процедури безпечного обліку та розподілу. Їх також потрібно оберігати від зловмисників, і, звичайно, не залишати в пристроях читання. Відомі випадки підробки пасивних карток.

Інтелектуальні картки, окрім власної пам'яті, мають свій мікропроцесор. Це дозволяє реалізувати різні варіанти парольних методів захисту: багаторазові паролі, одноразові паролі, звичайні запит-відповідні методи. Усі картки забезпечують двокомпонентну автентифікацію. До зазначених переваг інтелектуальних карток слід додати їх багатофункціональність. Їх можна застосовувати не тільки з метою безпеки, а й, наприклад, для фінансових операцій. Супутнім недоліком карток є їхня висока вартість.

Авторизація об’єкта (*Authorization*). Авторизація об'єкта, яку виконує система захисту КС, має на меті встановлення допустимих дій об'єкту в КС та призначення доступних йому ресурсів КС. Ця операція також відома як надання повноважень об'єкту. Разом з іншими операціями, такими як ідентифікація та автентифікація, вона складає процедуру ініціалізації, яка є односторонньою і стосується конкретного об'єкта КС. Однак, у деяких випадках, для забезпечення взаємодії між об'єктами, які зв'язані каналом передачі даних, необхідне двостороннє підтвердження їх ідентичності. Загальна структура доступу до інформаційних ресурсів зображена на рис. 13.1.

Адміністрування (*Accounting*) – це реєстрування абсолютно всіх дій, дозволених користувачу в системі, зокрема, доступ до ресурсів.

Адміністрування забезпечує виявлення та аналіз різних інцидентів безпеки в системі.

Рис.13.1. Алгоритм надання доступу до інформаційних ресурсів

Взаємна перевірка істинності сторін для інформаційного обміну

Зазвичай на початку зв'язку проводиться взаємна перевірка істинності сторін інформаційного обміну (автентифікація). Мета цієї процедури полягає в тому, щоб забезпечити достатній рівень впевненості обох абонентів щодо чотирьох складових:

- отримувач повинен бути впевнений в тому, що відправник даних є достовірним;
- отримувач повинен бути впевнений в істинності переданих йому даних;
- відправник повинен бути впевнений в тому, що дані були успішно доставлені отримувачеві;
- відправник повинен бути впевнений в істинності доставлених даних.

Найбільш надійним способом взаємної перевірки істинності є процедура «***рукостискання***». Згідно з цією процедурою, сторони, які мають доступ до однакового секретного ключа, перевіряють його правильність. Іншими словами, кожна сторона доводить правильність свого ключа, після чого сторони визнають одна одну дійсними. Варто зазначити, що звичайну передачу секретного ключа неможливо перевірити на істинність.

Розглянемо процедуру рукостискання між двома абонентами **Alice (A)** і **Bob (B)**, які використовують спільний секретний ключ **Key(AB)** у симетричній криптосистемі. Будь-який з абонентів може ініціювати процедуру, і якщо це робить **A**, він надсилає свій ідентифікатор $\mathbf{ІД_A}$ незахищеним каналом зв'язку до **B**. В свою чергу **B**, отримавши $\mathbf{ІД_A}$, знаходить секретний ключ **Key(AB)** у своїй базі даних. Обидва абоненти налаштовують свої криптосистеми на ключ **Key(AB)**, після чого вони готові виконати процедуру «*рукостискання*».

Обидва абоненти в цій процедурі мають використовувати однакову відкриту односторонню функцію **h**, що дозволяє отримати перетворення **h(x)** аргумента x, але не дозволяє відновити значення аргумента:

1. Починаючи першим процедуру «*рукостискання*», абонент **A** генерує випадкову послідовність **S**, шифрує її та відправляє криптограму **E(S)** абоненту **B**.

2. Абонент **B** дешифрує криптограму **E(S)** та отримує послідовність **S**, використовуючи ту ж саму односторонню функцію **h**. Потім обидва абоненти отримують перетворення $\mathbf{h_{A(S)}}$ та $\mathbf{h_{B(S)}}$.
3. Абонент **B** шифрує перетворення $\mathbf{h_{B(S)}}$ та відправляє його абоненту **A**.
4. Абонент **A** дешифрує криптограму та порівнює отримане перетворення $\mathbf{h_{B(S)}}$ зі своїм перетворенням $\mathbf{h_{A(S)}}$. Якщо перетворення співпадають, то абонент **A** переконується в тому, що абонент **B** володіє спільним секретним ключем **Key(AB)** і можна його вважати істинним.
5. Якщо абонент **A** хоче перевірити істинність абонента **B**, то він повинен повторити процедуру.

Процедура рукостискання має *перевагу* в тому, що для підтвердження істинності не потрібно передавати секретні дані, але має *недолік* у тому, що абоненти можуть не мати спільного секретного ключа.

13.2. Найпоширеніші протоколи автентифікації

Наведемо найпоширеніші протоколи автентифікації, які використовуються на практиці:

- Kerberos;
- LDAP;
- OAuth2;
- SAML;
- РАДІУС;
- DIAMETER та ін.

Kerberos — це протокол автентифікації, який використовується для безпечної ідентифікації користувачів і пристроїв у мережі. Він призначений для запобігання таким атакам, як підслуховування та повторне відтворення, а також для того, щоб дозволити користувачам безпечно отримувати доступ до мережевих ресурсів без передачі своїх паролів через мережу.

Протокол Kerberos працює за допомогою довіреної третьої сторони, відомої як сервер автентифікації Kerberos, для перевірки ідентичності користувачів і пристроїв. Коли користувач або пристрій хоче отримати доступ до мережевого ресурсу, вони запитують доступ у сервера автентифікації Kerberos. Сервер автентифікації перевіряє особу користувача та видає користувачеві квиток (TGT), який можна використовувати для запиту доступу до певних ресурсів у мережі.

Потім користувач або пристрій може використовувати TGT для запиту доступу до певного мережевого ресурсу від сервера автентифікації. Сервер автентифікації перевіряє TGT і видає сервісний квиток (ST) користувачеві або пристрою, який можна використовувати для доступу до запитуваного ресурсу. Користувач або пристрій представляє ST серверу ресурсів, який надає доступ, якщо ST дійсний.

LDAP (Lightweight Directory Access Protocol) — це мережевий протокол, який використовується для доступу та керування службами каталогів, такими як служби Active Directory або OpenLDAP. LDAP розроблений як простий, швидкий і безпечний протокол для доступу до служб каталогів через мережу.

Служби каталогів LDAP використовуються для зберігання та керування інформацією про користувачів, пристрої та інші об'єкти в організації. Ця інформація організована в ієрархічній структурі, де кожен об'єкт представлено записом у каталозі. LDAP дозволяє користувачам і програмам отримувати доступ і маніпулювати цією інформацією через мережу за допомогою стандартних команд і протоколів.

LDAP зазвичай використовується для автентифікації користувачів і пристроїв, для пошуку інформації про користувачів і пристрої та для керування доступом до мережевих ресурсів. Він часто використовується в поєднанні з іншими протоколами, такими як Kerberos, щоб забезпечити повне рішення для автентифікації та контролю доступу.

OAuth2 (Open Authorization 2.0) — це відкритий стандарт авторизації, який дозволяє користувачам надавати стороннім програмам доступ до своїх ресурсів (таких як дані чи служби), не повідомляючи свої паролі. OAuth2 використовується для безпечної авторизації з веб-додатків, мобільних і настільних програм.

SAML (Security Assertion Markup Language) — це стандартний протокол, який використовується для безпечного обміну даними автентифікації та авторизації між організаціями. Він зазвичай використовується для ввімкнення системи єдиного входу (SSO) і забезпечення безпечного доступу до веб-ресурсів.

Протокол SAML працює, дозволяючи користувачеві автентифікуватися за допомогою постачальника ідентифікаційної інформації SAML (IdP), який є системою, яка перевіряє особу користувача та видає твердження про особу користувача. Потім твердження надається постачальнику послуг SAML (SP), який є системою, яка надає доступ до веб-ресурсу. SP використовує твердження, щоб надати користувачеві доступ до ресурсу, не вимагаючи від користувача повторної автентифікації.

RADIUS (Remote Authentication Dial-In User Service) — це мережевий протокол, який використовується для керування та автентифікації користувачів, які підключаються до мережі. Він зазвичай використовується для автентифікації користувачів, які підключаються до мережі за допомогою комутованого з'єднання, але його також можна використовувати для автентифікації користувачів, які підключаються до мережі за допомогою інших технологій, таких як бездротова мережа або VPN.

Протокол RADIUS працює, дозволяючи користувачеві автентифікуватися за допомогою сервера RADIUS, який є системою та перевіряє особу користувача та авторизує його доступ до мережі. Коли користувач намагається підключитися до мережі, RADIUS-сервер отримує запит на доступ і автентифікує користувача за допомогою його облікових

даних (таких як ім'я користувача та пароль). Якщо користувач автентифікований, RADIUS-сервер надає доступ до мережі та призначає користувачеві набір мережевих параметрів (таких як IP-адреса та маска підмережі).

Challenge-Handshake Authentication Protocol (CHAP) — ця система періодично повторно автентифікує користувачів, навіть під час одного сеансу. Кожне завдання відрізняється від попередньої версії.

DIAMETER — цей протокол забезпечує основу для автентифікації та обліку повідомлень. Він походить від RADIUS і вважається вдосконаленням цього протоколу.

13.3. Кількісна оцінка стійкості пароля

Нехай A – потужність алфавіту паролів (число символів, які можна використовувати для складання пароля), L - довжина пароля, $S = A^L$ - кількість всіх можливих паролів довжини L. Нехай швидкість підбору пароля зловмисником буде складати V, а максимальний термін актуальності пароля позначимо через T. Тоді ймовірність підбору пароля за час T дорівнює

$$P = V*T/S = (V*T) / A^L.$$

Визначимо нижню межу кількості всіх паролів, як $S^* = (V*T) / P$.

Якщо $S^* =< S < A^L$, то можливість підбору буде менше або дорівнює заданій ймовірності P.

Приклад 13.2. Визначити мінімальну потужність A алфавіту і довжину паролів L, що забезпечують ймовірність підбору $P=10^{-6}$ пароля зловмисником при максимальному терміні дії пароля T = 2 дня і швидкості підбору пароля V =100 (паролей/хв).

Рішення: Визначимо кількість паролів, які можуть згенеровані жддля підбору за термін сталого його існування в системі:

V = 100*60*24*2 = 288000 паролів/2дня.

Тоді нижня межа кількості всіх паролів запишеться як:

$$S^* = (V*T) / P = (288000*1) / 10^{-6} = 288*10^8.$$

$$288*10^8 =< S < A^L.$$

Якщо припустити, що використовується латинський алфавіт (26 літер) разом з цифрами (10), то A = 36. Тоді для виконання данної умови значення довжини пароля має бути не менше 7 літер, тобто $L>=7$ ($36^7 = 783,6*10^8$).

13.4. Атаки на протоколи автентифікації

Захист даних може супроводжуватися діями, які повязані з намаганнями порушити їх цілісність, видати себе за іншу особу та інше, що повязано з терміном «атака». Наведемо найбільш поширені типи атак на протоколи автентифікації.

САМОЗВАНСТВО (impersonation). Полягає в тому, що один користувач намагається видати себе за іншого;

ПОВТОРНЕ ПЕРЕДАВАННЯ (replay attack). Полягає в повторному передаванні автентифікаційних даних яким-небудь користувачем;

ПІДМІНА СТОРОНИ АВТЕНТИФІКАЦІЙНОГО ОБМІНУ (interleaving attack). Зловмисник у ході даної атаки бере участь у процесі автентифікаційного обміну між двома сторонами і має можливість модифікації трафіка, що проходить через нього.

ВІДОБРАЖЕННЯ ПЕРЕДАВАННЯ (reflection attack). Атака, у ході якої зловмисник у рамках даної сесії протоколу пересилає зворотно перехоплену інформацію;

ЗМУШЕНА ЗАТРИМКА (forced delay). Зловмисник перехоплює деяку інформацію і передає її через деякий час;

АТАКА З ВИБІРКОЮ ТЕКСТУ (chosen-text attack). Зловмисник перехоплює автентифікаційний трафікі намагається одержати інформацію про довгострокові ключі.

Деякі види атак і шляхи протидії представлені в табл.13.1

13.5. Протоколи автентифікації з нульовим розголошенням (zero knowledge proof)

Сучасні системи автентифікації використовують криптографію та концепцію секретних ключів, дозволяючи учасникам спілкуватися, лише якщо вони мають належну ідентифікацію: власні відкритий і закритий ключі. Ця система покладається на те, що обидві сторони попередньо домовилися про систему секретних ключів, і вимагає від кожного користувача підтримувати безпечне володіння своїми власними ключами. Хоча цей метод може здатися криптографічно безпечним, існує безліч вразливостей. Зловмисники могли отримати закритий ключ користувача або підслухати розмову та отримати передані дані. Протокол з нульовим розголошенням є надійною альтернативою сучасним системам автентифікації, оскільки він дозволяє автентифікувати користувача без передачі життєво важливих даних. Оскільки нинішній рівень конфіденційності в Інтернеті є тривожно низьким, зараз як ніколи існує потреба в криптографічно надійних протоколах, які дозволяють здійснювати безпечні та надійні транзакції.

Табл.13.1.

Різновиди атак і методи протидії

Різновиди атаки	Методи протидії
Підміна сторони	Зв'язування всіх повідомлень у рамках даної сесії протоколу (наприклад, використовуючи одноразовий й унікальний ідентифікатор у всіх повідомленнях)
Вибірка тексту	Використання протоколів, що володіють властивістю доказу з нульовим знанням; включення в кожне повідомлення випадкових чисел
Відображення	Використання ідентифікатора сторін у переданих повідомленнях; побудова протоколу таким чином, щоб кожне повідомлення мало відмінну від інших форм
Повторне передавання	Використання протоколів типу "запит-відповідь"; використання міток часу чи випадкових чисел; вставка у відповіді інформації, що ідентифікує
Затримка передавання	Комбінування використання міток часу разом з випадковими числами
Після проходження автентифікації між двома користувачами і встановлення з'єднання порушник підмінює якого-небудь користувача із з'єднання і продовжує роботу від замість нього	періодичне виконання процедур автентифікації в рамках вже встановленого сеансу зв'язку; прив'язка результату автентифікації до наступних дій користувачів у рамках системи. Прикладом подібного підходу може служити автентифікаційний обмін секретними сеансовими ключами, з використанням яких здійснюється подальша взаємодія користувачів.

У симетричних і асиметричних схемах автентифікації є багато вразливостей, що викликає потребу в унікальній системі, яка не піддається звичайним атакам. Докази з нульовим розголошенням забезпечують метод автентифікації за допомогою безпечної форми зв’язку. Основна концепція доведення з нульовим розголошенням полягає в тому, щоб підтвердити знання про деякі секретні дані без прямого розкриття будь-якої інформації про них. Протоколи доведення з нульовим розголошенням дозволяють користувачеві довести іншій стороні, що він знає «секрет», однак сам секрет ніколи не передається між сторонами. Такі протоколи можуть також використовуватись для доведення *без розголошення*, коли один абонент (абонент **А**) доводить іншому (абоненту **В**), що він знає певні секретні дані, *не розголошуючи* їх. Головна ідея цих протоколів полягає в тому, що абонент **В** може перевірити наявність секретних даних у абонента **А** без отримання самого секрету або будь-якої частини його. Зазвичай у таких протоколах абонент **В** задає ряд запитань абоненту **А**, відповіді на які повинні бути у формі «так/ні» відповідей. Зі збільшенням кількості правильних відповідей (**t**) у абонента **В**, його впевненість у достовірності абонента **А** зростає за виразом: $\mathbf{p=1-0.5^t}$.

Розглянемо поняття квадратичного залишку за модулем і його властивості. Нехай є деяке натуральне $n>2$, тоді деяке натуральне число a, для якого $a<n$, називається ***квадратичним лишком за модулем n***, якщо воно є взаємно простим за модулем n, тобто $НСД(a, n)=1$, та окрім цього виконується умова $x^2=a \pmod n$, де число x є квадратичним коренем за модулем n.

Протокол **Фейга-Фіата-Шаміра** (*Feige-Fiat-Shamir*) - це один із протоколів автентифікації з нульовим розголошенням. Ґрунтується на складності обчислення квадратного кореня числа за модулем великого числа з невідомим розкладанням на прості множники. Був запропонований 1986 р. У.Фейге, А.Фіатом та А.Шаміром. У протоколі передбачається, що обом сторонам наперед відомо деяке число $n = pq$. При цьому розкладання числа n на прості множники вважається невідомим всім учасників протоколу.

Опишемо даний протокол.

Нехай одна із сторін **A** (Аліса) вибирає секретне число S (що стає її **секретним ключем**), яке є взаємно простим з n: $1<S<n$ і $НСД(S, n)=1$. Далі обчислюється значення $V \equiv S^2 \bmod n$ і публікується значення, яке оголошується його **відкритим ключем**. Для обчислення секретного ключа S за відкритим ключем V потрібно вирішити квадратичне порівняння за складовим модулем, що обчислювально еквівалентно розв'язанню задачі факторизації. Оскільки завдання факторизації вважається складним завданням, для вирішення якої поки що не існує поліноміальних алгоритмів, то і відновити секретний ключ по відкритому без знання множників p і q за реальний час є неможливим.

Приклад 13.3. Нехай вибрано модуль **n=77**, який є добутком двох простих чисел **7** і **11.** Для прикладу обрано два невеликих числа, на практиці це мають бути великі числа: **n** > 512 біт (для спрощення наступних обчислень рекомендують користуватися наступним співвідношенням: **n**=(4**p**+3)*(4**q**+3). Нехай обрано число **S=31,** яке буде секретним ключем. Тоді в результаті обчислень

$$V=S^2 \bmod n=31^2 \bmod 77=37$$

отримано відкритий ключ - **V=37**.

Отже **n** та **V** – відкриті параметри, а **S** – секретний параметр.

Далі слідує виконання протоколу автентифікації. Наведемо даний протокол для задачі, коли одна сторона, наприклад **A** (це може бути, наприклад, деяка *платіжна картка*), доводить свою істиність іншій стороні **B** (наприклад, *банку*), яка приймає рішення про обслуговування сторони **A**.

Приклад протоколу автентифікації.

1. Сторона **A** вибирає деяке випадкове число r, $1 < r < n - 1$.
2. Сторона **A** обчислює число $x \equiv r^2 \bmod n$ і посилає його стороні, що перевіряє - **B**. Наприклад, сторона **A** обрала випадкове число $r = 11$,

$x \equiv r^2 \bmod n \equiv 11^2 \bmod 77 \equiv 44$ і відправляє число $x=44$ стороні **B.**

3. Сторона **B** обирає випадковий біт ***b*** («0» або «1») і пересилає його стороні **A**.
4. Сторона **A** обчислює число ***y***: якщо $b = 0$, то $y = r$, інакше $y \equiv r\ S \bmod n$, і пересилає число ***y*** стороні **B**.

Приклад 13.4.
Якщо $b = 0$, то $y = 11$, якщо $b = 1$, то $y = 11*31 \bmod 77 = 33$.
5. Сторона **B** перевіряє рівняння: $y^2 \equiv xV^b \bmod n$.
Якщо $b = 0$, то

$$y^2 \equiv x \bmod n,$$

і перевірка означає, що вираз

$$x \equiv r^2 \bmod n,\ y = r$$

вірний.

Приклад 13.5. $44=11^2 \bmod 77 = 44$, що є вірно.
Аналогічно, якщо $b = 1$, то сторона **B** перевіряє співвідношення:

$$y \equiv r\ S \bmod n,\ y^2 \equiv r^2\ S^2 \bmod n,\ xV^b = xV \equiv x\ S^2 \bmod n.$$

Перевірка також означає, що вираз $r^2\ S^2 \equiv x\ S^2 \bmod n$ вірний.
Тоді сторона **B** перевіряє співвідношення $xV=y^2 \pmod n$, що означає перевірку знання секретного ключа **S**.

Приклад 13.6. Для обраних чисел маємо: $11^2\ 31^2 = 44\ 31^2 \bmod 77$, що є вірно. Або $44*37 = 33^2 \bmod 77$, що є вірно.

Ці кроки утворюють один цикл протоколу, який називають акредитацією. Сторони **A** і **B** повторюють цей цикл **t** разів при різних випадкових значеннях ***r*** і ***b*** доти, доки сторони **B** не переконається, що сторона **A** знає значення ***S***.

Для забезпечення достатньої надійності протоколу вважається, що його необхідно повторити $t=[\log_2 n]$ разів. Якщо перевірка у всіх циклах була успішною, то сторона **A** може переконати сторону **B** у своєму знанні секретного ключа з ймовірністю **1**. Якщо сторона **A** не знає секретного ключа, то сторона **B** може викрити її неістинність з ймовірністю $p=0.5$ в кожному з циклів і з ймовірністю $p=1-0.5^t$ в останньому з **t** циклів, незалежно від того, які числа **x** та **y** сторона **A** відправлятиме.

Для успішної роботи протоколу необхідно уникати повторного використання значення випадкового числа ***r*** істинною стороною **A**, оскільки сторона **B** може легко визначити значення секретного ключа **S**. Структурна схема реалізації протоколу Фейга-Фіата-Шаміра представлена на рис.13.2.

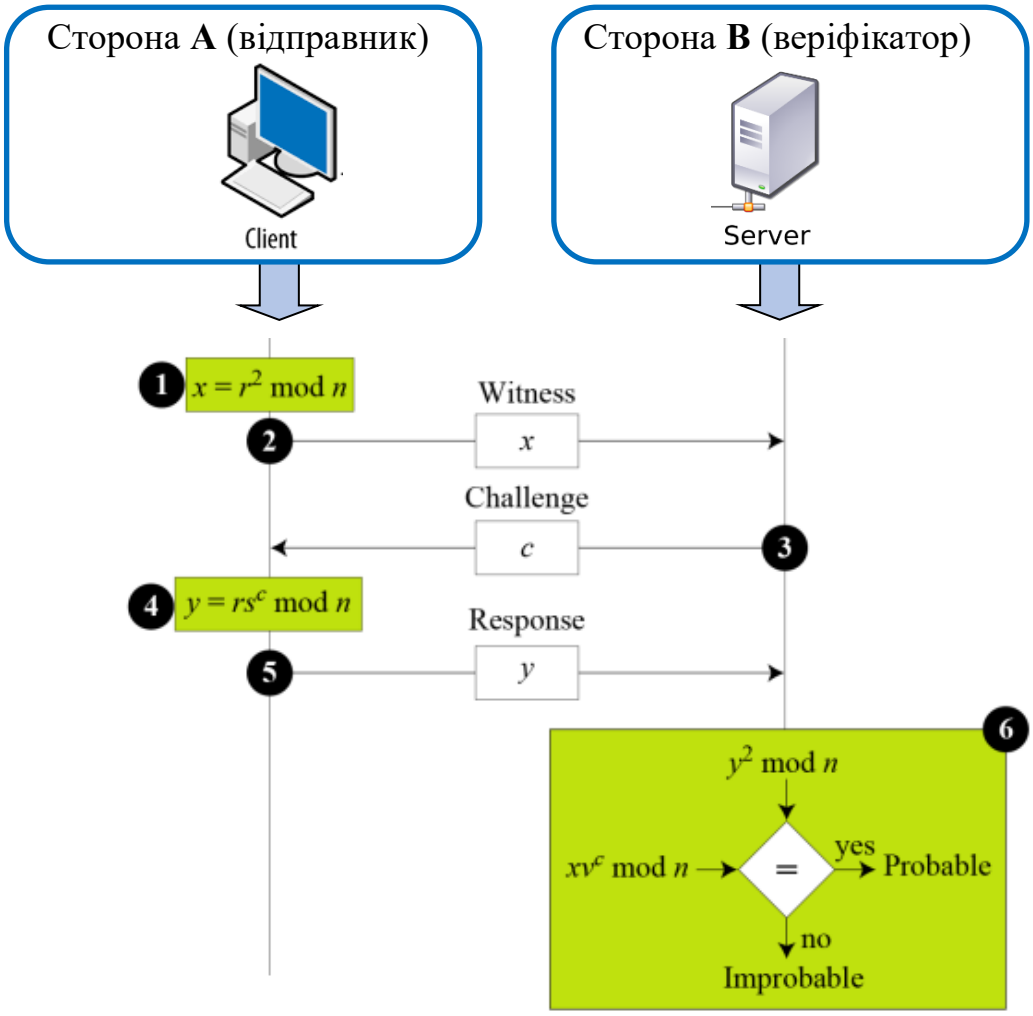


Рис.13.2. Графічне представлення протоколу Фейга-Фіата-Шаміра:
s – секретний параметр сторони **A,** ***v*** – відкритий параметр сторони **A,**
r **–** випадкове число, ***n*** - відкритий параметр, ***c*** = «0» або «1».

13.4. Підготовка до завдання

Ознайомитися з теоретичними положеннями та призначенням ідентифікації, аутентифікації та авторизації об'єкта, побудови протоколу автентифікації з нульовою передачею знань.

13.5. Практичне завдання

В якості практичного завдання стоїть задача автентифікації користувачів. Для цього потрібно зробити імітацію передачі і перевірки даних за протоколом *Фейга-Фіата-Шаміра*, який наведено вище, а саме:

1. Обрати значення числа **n**, яке являє собою добуток двох секретних великих простих чисел. Сам модуль **n** є **відкритим;**
2. Обрати значення секретного ключа ***S***;
3. Обчислити *відкритий ключ* $V=S^2 \bmod n$;

4. Здійснити процедури, наведені в п.1-4 за протоколом *Фейга-Фіата-Шаміра* (див. вище) з врахуванням заданої ймовірністю довіри до сторони обміну.
5. Написати програму, яка реалізує операції по п.1-4;
6. Надати аналіз отриманим результатам.

13.6. Зміст протоколу роботи

Протокол до виконаної практичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

13.7. Контрольні питання для самоперевірки

1. Загальні вимоги до реалізації криптографічних алгоритмів.
2. Переваги та недоліки апаратних засобів криптографічного захисту.
3. Переваги та недоліки програмних засобів криптографічного захисту.
4. Програмно-апаратна реалізація криптографічного захисту.
5. Що таке криптографічний протокол?
6. Роль криптографічних протоколів у загальній задачі забезпечення інформаційної безпеки.
7. Поняття ідентифікації, автентифікації, авторизації та адміністрування об’єкта.
8. Класифікація протоколів автентифікації та їх характристика (слабка, багатофакторна, біометрична, сувора).
9. Найпоширеніші протоколи автентифікації та їх коротка характеристика (Kerberos, LDAP, OAuth2, SAML, РАДІУС та ін.).
10. Атаки на протоколи автентифікації та шляхи протидії.
11. Властивості протоколів автентифікації.
12. Протоколи автентифікації з нульовою передачею знань.
13. Вимоги, що висуваються до протоколів автентифікації.
14. Основні задачі забезпечення інформаційної безпеки, що розв‘язуються за допомогою криптографічних протоколів.
15. Взаємна перевірка істинності сторін інформаційного обміну на прикладі процедури «рукостискання».
16. Протокол Фейга-Фіата-Шаміра та алгоритм його реалізації.

13.8. Задачі до практикуму №13

1. При використанні латинського алфавіту разом з цифрами визначити мінімальну довжину паролів $\boldsymbol{L}$, що забезпечують ймовірність підбору $\boldsymbol{P}=10^{-8}$ пароля зловмисником при максимальному терміні дії пароля $\boldsymbol{T}$ = 1 день і швидкості підбору пароля $\boldsymbol{V}=10^{6}$ (паролей/хв).

2. Яка з наведених категорій не використовується для опису методів автентифікації? Відповідь обгрунтуйте.

а) Щось, що вам подобається?
б) Щось, що ти знаєш?
в) Щось, чим ти володієш?
г) Щось, ким ти є?

3. Як розглядається пароль у схемі багатофакторної автентифікації? Відповідь обгрунтуйте.

а) Щось, що вам подобається?
б) Щось, що ти знаєш?
в) Щось, чим ти володієш?
г) Щось, ким ти є?

4. Що видає сервер автентифікації Kerberos клієнту, який успішно пройшов автентифікацію?

13.9. Завдання до самостійної роботи

1. Міжнародні стандарти з криптографічних протоколів автентифікації ISO/ IEC9798 – Information technology – Security techniques – Entity authentication (ISO/IEC9798–1:2010 Part 1, ISO/IEC9798–1:2010 Part 2, ISO/IEC9798–1:2010 Part 3, ISO/IEC9798–4:1999 Part 4, ISO/IEC9798–5:2009 Part 5, ISO/IEC9798–6:2010 Part 6).
2. Біометричні системи автентифікації: статичні, динамічні. Переваги, недоліки, особливості реалізації.
3. Системи автентифікації на основі графічниї паролів.
4. Основні атаки на протоколи автентифікації та їх характеристики.
5. Взаємна автентифікація на основі ЕЦП.
6. Протоколи «запит – відповідь» з використанням симетричних криптосистем.
7. Сучасні протоколи автентифікації: Kerberos, LDAP, OAuth2, SAML, RADIUS, DIAMETER. Призначення, область застосування.

ПРАКТИКУМ №14
Реалізація та дослідження методів стенографії в системах захисту інформації

Мета роботи. Ознайомитися з основами стеганографії та її практичним застосуванням.

14.1. Поняття стеганографії та її класифікація

Переваги представлення та передачі даних у цифровій формі (легкість відновлення, висока потенційна завадостійкість, можливість використання універсальних апаратних і програмних рішень) можуть бути легко перекреслені сучасними методами аналізу даних, їх викрадення та заміни. Тому в усьому світі визріває проблема розробки методів (засобів) захисту інформації організаційного, системного і технічного характеру, в тому числі криптографічних і стеганографічних методів.

Криптографічний захист інформації – це система, яка модифікує дані з метою зробити вміст повідомлень незрозумілим через шифрування. Цей захист не вирішує повністю проблему захисту інформації. Наявність зашифрованого повідомлення привертає увагу, і зловмисник, який отримує криптографічно захищений файл (дані), швидко розуміє, що в ньому міститься секретна інформація і виділяє певні ресурси на розшифрування. Приховування самого факту існування секретної інформації під час передачі, зберігання робить їх більш захищеними. Такий підхід реалізується стеганографією - наукою, що вивчає моделі та методи приховування конфіденційної інформації.

Стеганогра́фія — (перекладається з грецької як пишу приховано: Στεγανός γράφω) — тайнопис, який дозволяє кодувати повідомлення таким чином, що воно не виглядає приховани повідомленням на відміну від криптографії. При кодуванні в такий спосіб непосвячена особа не зможе розшифрувати повідомлення по тій причині, що не знає навіть про факт його існування.

Якщо криптографія приховує зміст повідомлення, то *стеганографія приховує сам факт існування повідомлення*.

Поняття стеганографії вперше було введено в 1499 році, але сам метод існує вже давно. Легенда свідчить, що цей метод використовувався в Римській імперії. Для передачі послання вибирали раба, голову якого голили, а потім за допомогою татуювання наносили текст. Після того, як волосся відростало, рабів відправляли в пункт призначення. Одержувач послання знову голив рабу голову і зчитував повідомлення. Сучасним прикладом є випадок роздрукування на принтері контрактів з малопомітними викривленнями обрисів окремих символів тексту — так вносилась шифрована інформація про умови складання контракту

На сьогодні стеганографія є наукою, яка швидко розвивається, використовує сучасні методи й досягнення криптографії, цифрової обробки сигналів, теорії зв’язку та інформації [38-43]. З 1996 р. постійно проводяться

різноманітні міжнародні заходи із проблем приховання даних (Information Workshop on Information Hiding). Перша конференція, присвячена стеганографії, відбулася в липні 2002 р.

Стеганографічні методи використовуються в різноманітних сферах діяльності людини, що ілюструється на рис.14.1. [38]

Рис.14.1. Галузі застосування стеганографії

Наприкінці 90-х років виділилося кілька напрямків стеганографії, які отриали свій розвиток і динамічно розвиваються:

- *класична стеганографія;*
- *лінгвістична стеганографія*
- *комп'ютерна стеганографія;*
- *цифрова стеганографія;*
- *квантова стеганографія та ін.*

Класична стеганографія

Класична стеганографія – спосіб приховування даних, що здійснюється за допомогою технічних засобів захисту інформації. Сучасна класична стеганографія складається з різноманітних хімічних та фізичних методів (рис.14.2).

Рис.14.2. Класифікація класичної стеганографії

Одним із найпоширеніших методів класичної стеганографії є використання невидимих чорнил. Зазвичай процес запису відбувається такий спосіб, коли спочатку наноситься секретний запис невидимими чорнилами, а потім формується запис видимими чорнилами. Зазвичай другий запис не має ніякого змістовного навантаження і нічого не значить. Перший шар невидимих чорнил проявляється тільки при певних умовах, наприклад, нагріванні, підсвічуванні, впливі хімічного проявника та ін. (рис.14.3). Речовинами для невидимих (симпатични) чорнил можуть бути молоко, яблучний сік, лимонна кислота та ін. Прояв таких речовин може зявлятися при підвищенні температури, підсвічування ультрафіолетом, нанесенні інших речовин та ін.

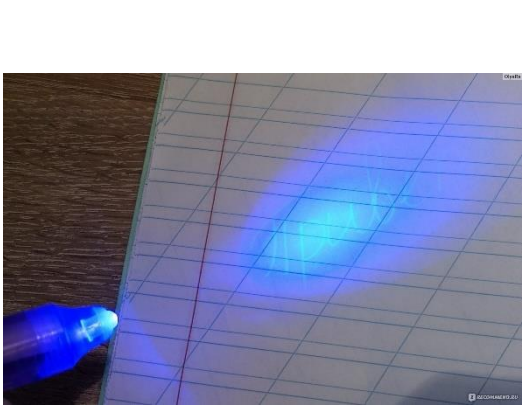


Рис.14.3. Застосування невидимих (симпатичних) чорнил для приховування інформації

У 2001 р. в Австралії було розроблено технологію нанесення мікроточок, що містять Персональний Ідентифікаційний Номер (ПІН), на найважливіші деталі виробу (зазвичай - автомобіля). Такі прозорі мікроточки, що виготовлені за допомогою лазера, наклеюються в непримітних місцях безпосередньо на складальному конвеєрі. Побачити їх можна лише при освітленні ультрафіолетовим світлом. Цей процес, дешевий та ефективний, ускладнює викрадачам автомобілів легальний продаж вкраденої та розібраної машини у вигляді "запчастин".

Також відомі технології для виконання мікронадпісів (*microdots*) спеціальними механічними пристроями, які використовувалися ще в XVIII столітті в Англії і Франції. Однин з найдосконаліших таких пристроїв (Peter's Machine for Microscopic Writing - 1862 г.), зберігається в музеї Оксфордського університету. Такий пристрій дозволяв виконувати надписи з висотою символів всього в 2,5 мікрона.

Іншим прикладом класичної стеганографії є додавання виробником кольорових принтерів так званих «жовтих крапок» в спеціальні місця при друку документів, які є малопомітними на білому фоні. При збільшенні масштабу можна побачити «жовті крапки» на рис.14.4.

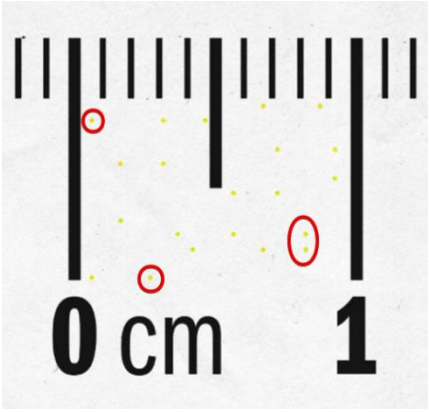


Рис.14.4. Додавання «жовтих крапок» до тексту повідомлення (https://uk.wikipedia.org/wiki/Жовті_крапки)

Ці крапки, що ледь видно неозброєним оком (для прикладу, три з них обведені червоними колами), друкувалися на кожній сторінці і містили в собі інформацію про серійний номер принтера, а також дату та час друку. Підтверджено використання цього методу у принтерах, що випускаються під торговими марками Brother, Canon, Dell, Epson, Hewlett-Packard, IBM, Konica, Kyocera, Lanier, Lexmark, NRG, Panasonic, Ricoh, Savin, Toshiba, Xerox. Введення цього заходу, згідно з коментарями виробників, було частиною співпраці з урядом та консорціумом банків, спрямованої на боротьбу з фальшивомонетниками.

У 2011 р Мануель Паласіос (Manuel Palacios) з університету Тафтса і Джордж Уайтсайдс (George Whitesides) з Гарварда спробували заховати повідомлення в масиві, що складається з семи штаммів бактерій Escherichia coli (E. coli). Техніку жартома назвали SPAM (Steganography by Printed Arrays of Microbes), що можна перевести як стеганографія за допомогою друкованих масивів мікробів (рис.14.5).

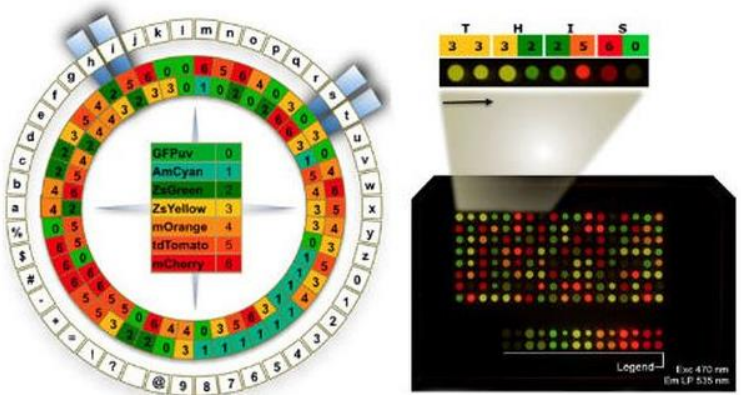


Рис.14.5. Техніка приховування повідомлень при використанні штаммів бактерій (https://chemlaba.wordpress.com/2011/09/30/цветовой-шифр-для-записи-информации-в/)

Зручність в тому, що невидиме оку повідомлення можна нанести на звичайну листівку і надіслати її поштою. Одержувачу необхідно мати під рукою невеликий набір для дешифрування і, звичайно ж, знати сам шифр.

Вчені створили сім штамів бактерій, кожен з яких виробляє свій білок, що світиться при певному світлі. Колонії бактерій наносяться на основу, як набір точок. Кожна пара точок (кольорів) є кодом для літер, цифр або символів. Сім кольорів на кожні дві таких точки дають 49 комбінацій. Автори роботи використовували їх для кодування 26 літер та 23 буквенно-цифрових символів (таких як «**@**» або «***$***»). Наприклад, дві жовті точки позначають букву «***t***», а комбінація помаранчевої та зеленої — «***d***» і т.д. Отримувач, знаючи коди дешифровки, легко прочитає надіслане повідомлення.

Екзотичним способом зберігання та передачі інформації є використання для цих цілей ДНК-молекул. У клітинах тварин та рослин ДНК знаходиться в ядрі клітини у складі хромосом. ДНК-молекули є компактним та надійним носієм інформації. Група вчених Гарварда підрахувала, що пам'ять із структур ДНК вагою лише 4 грами теоретично може зберігати всю інформацію, яку виробляє все людство сучасності за один рік. На користь надійності говорить той факт, що інформація з ДНК може бути зчитана через сотні тисяч і навіть мільйони років. ДНК можна зберігати в далеко не ідеальних умовах - наприклад, у мертвих тварин. При цьому вона збережеться, і через 400 тис. років ми все ще здатні її прочитати. Запис у ДНК буде зберігатися набагато довше, ніж на диску Blu-Ray.

Лінгвістична стеганографія – напрям, який вивчає методи приховування конфіденційної інформації в непримітний текст, застосовуючи мовні властивості та лінгвістичні ресурси. Лінгвістичні методи стеганографії поділяються на дві основні категорії: умовне письмо і семаграми (рис.14.6)

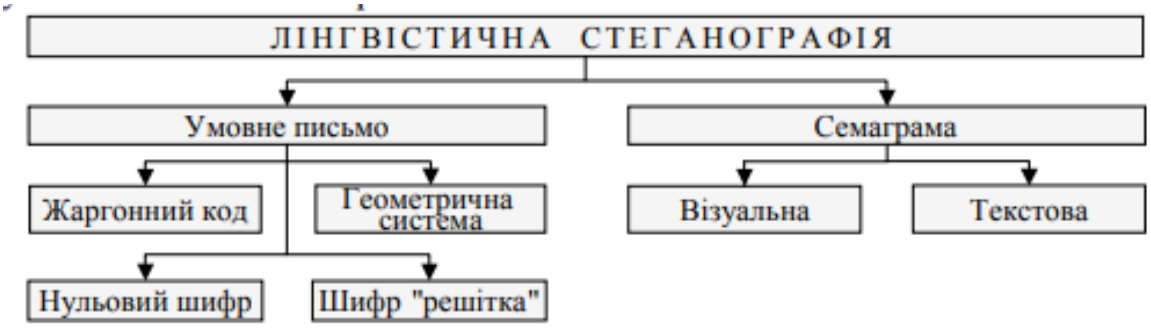


Рис.14.6. Класифікація лінгвістичної стеганографії

Добре відомі різноманітні **літературні прийоми**, призначені для приховування таємної інформації у зовні невинних посланнях, наприклад – «пустий шифр» та ін. Одним з простих прикладів таких літературних прийомів є написання **акровіршів** - таємних посланнь, які складається з перших літер рядків віршів.

«Пустий шифр». У разі використання цього шифру (точніше методу) слова або літери секретного повідомлення записуються у певних позиціях. Наприклад, читаються кожне п'яте слово або перша літера кожного слова, у той час як решта літер або слів служать як «пустишка» для приховування значущого тексту (рис.14.7).

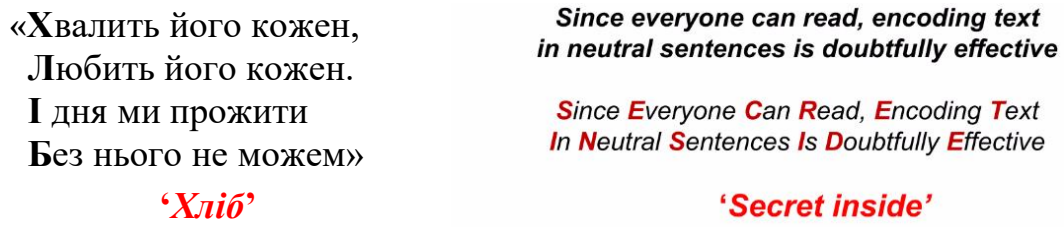


Рис.14.7. Приклад реалізації літературних прийомів для передачі повідомлень

Одним із різновидів стеганографії є **Соціальна стеганографія**. У спільнотах із соціальними чи державними табу чи цензурою люди використовують культурну стеганографію, приховуючи повідомлення в ідіомах, згадки про поп-культуру та інші повідомлення, які вони публічно поширюють і вважають, що вони відстежуються. Це залежить від соціального контексту, щоб зробити базові повідомлення видимими лише для певних читачів. Наприклад, написання з помилками імен або слів, популярних у засобах масової інформації протягом певного тижня, щоб запропонувати альтернативне значення. Приховування повідомлення в заголовку та контексті спільного відео чи зображення та ін.

Без сумніву, на сьогодні найпоширеним видом приховування інформації є комп'ютерна та цифрова стеганографія, що пояснюється активним використанням комп'ютерних засобів та цифрової форми представлення інформації.

Приклад 14.1. Одним з найпростіших прикладів застосування текстової стеганографії є розташування різної кількості пробілів між словами. Наприклад, один пробіл «*» буде означати «0», подвійний пробіл «**» - «1». Такі додаткові пробіли будуть непомітними в тексті, але зможуть передати інформацію. Якщо ми візьмемо звичайне речення, наприклад з 4-х слів:

«Привіт, мене звати Петро»,

то детальний аналіз пробілів «*» дає нам наступну картину:

«Привіт,мене**звати*Петро.»**

Таким чином, в такий спосіб можемо передати приховане число в бінарній формі, наприклад «110», що відповідає числу «6» в десятковій системі. Аналогічно можна передавати двійкові коди літер, різних знаків і т.п.

Існують різні модифікації даного підходу, які роблять таку приховану передачу даних ще більш непомітною і стійкою до криптоаналізу.

В літературі розглядають три групи методів, що зазнали найбільшого поширення при вбудовуванні приховуваних даних до тексту:

– *методи довільного інтервалу*, що здійснюють вбудовування шляхом маніпуляції з пробільними символами;

– *синтаксичні методи*, які працюють з пунктуацією;

– *семантичні методи*, до основи яких покладене залежне від приховуваних бітів даних маніпулювання словами.

Комп'ютерна стеганографія — напрям класичної стеганографії, заснований на особливостях комп'ютерної платформи. Наприклад,

приховування даних в невикористовуваних областях форматів файлів, підміна символів в назвах файлів, текстова стеганографія і т.д. Класифікація методів комп'ютерної стеганографії наведена на рис.14.8 [44].

Рис.14.8. Класифікація методів комп'ютерної стеганографії

Як зазначається в [44], за способом **обрання контейнера** розрізнюють сурогатні, селективні та конструюючі методи стеганографії. В *сурогатних* (*безальтернативних*) методах стеганографії повністю відсутня можливість вибору контейнера і для приховування повідомлення обирається перший контейнер, що трапився. У *селективних* методах комп'ютерної стеганографії передбачається, що приховане повідомлення повинно від- творювати спеціальні статистичні характеристики шуму контейнера. У *конструюючих* методах стеганографії контейнер генерується самою стеганосистемою. При цьому існують декілька варіантів реалізації. Так, наприклад, шум контейнера може імітуватися приховуваним повідомленням.

За способом **доступу до приховуваної інформації** розрізняють методи для *потокових* (*безперервних*) контейнерів і методи для фіксованих (*обмеженої довжини*) контейнерів.

За **способом організації** контейнери, подібно завадозахищеним кодам, можуть бути *систематичними* і *несистематичними*. У перших можна вказати конкретні місця стегано-грами, де знаходяться інформаційні біти власне контейнера, а де шумові біти, призначені для приховування інформації (як, наприклад, у широко поширеному методі найменшого значущого біту). У випадку несистематичної організації контейнера такий поділ не можливий. У цьому разі для виділення прихованої інформації необхідно обробляти вміст усієї стеганограми.

За **принципом приховування** методи комп’ютерної стеганографії поділяють на два основних класи: методи *безпосередньої заміни* і *спектральні* методи. Якщо перші, використовуючи надлишковість інформаційного середовища в просторовій (для зображення) або часовій (для звуку) області,

полягають в заміні малозначимої частини контейнера бітами секретного повідомлення, то другі для приховування даних використовують спектральне представлення елементів середовища, куди вбудовуються приховувані дані (наприклад, до різних коефіцієнтів масивів дикретно-косинусних перетворень, перетворень Фур'є тощо).

За **призначенням** розрізнюють стеганометоди для *прихованого передавання* (або *прихованого збереження*) даних і методи для приховування даних у цифрових об'єктах з метою захисту авторських прав на них.

За **типами контейнера** виділяють стеганографічні методи із контейнерами у вигляді тексту, аудіофайлу, зображення та відео.

Наведемо деякі приклади використання комп'ютерної стеганографії. Наприклад, використання зарезервованих полів комп'ютерних форматів файлів — суть методу полягає в тому, що частина поля розширень не заповнена інформацією про розширення, за замовчуванням заповнюється нулями. Відповідно, ми можемо використовувати цю «нульову» частину для запису прихованих даних. Очевидним недоліком такого методу є низька скритність і малий обсяг повідомлення, яке можна передати.

Іншими прикладми стеганографії з застуванням комп'ютерної техніки може бути відтворення звукової доріжки назад, щоб відкрити секретне повідомлення. Відтворення відео з вищою частотою кадрів (FPS), щоб виявити приховане зображення та ін.

Використання особливостей файлових систем — при зберіганні на жорсткому диску файл завжди займає ціле число кластерів. Нпаприклад, для зберігання 1 Кб інформації на диску виділяється 4Кб інформації, з яких 1Кб потрібен для зберігання файлу, а інші 3Кб не використовуються. В цьому випадку їх можна використовувати для зберігання додаткової інформації. Недоліком такого підходу є легкість їх виявлення.

Розвиток засобів цифрової обчислювальної техніки дав поштовх для розвитку комп'ютерної стеганографії, яка ґрунтується на вбудовуванні секретного повідомлення в цифрові дані, що, як правило, мають аналогову природу (аудіозаписи, зображення, відео). Можливе також вбудовування інформації в текстові та скомпресовані файли.

Цифрова стеганографія — напрям класичної стеганографії, заснований на приховуванні додаткової інформації в цифрові об'єкти при внесенні певних спотворень до цих цифрових об'єктів (рис.14.9).

Рис.14.9. Класифікація цифрової стеганографії

В більшості випадків такими об'єктами є мультимедіа-об'єкти (зображення, відео, аудіо та ін.). Внесення змін до таких об'єктів не повинно привертати уваги і бути помітними для середньостатистичної людини.

Більш детально про один із методів цифрової стеганографії – кодування найменш значуших біт (Least Significant Bit) буде показано в п.14.4.

Мережева стеганографія. Останнім часом стало поширеним використання властивостей протоколів передачі даних для передачі прихованої інформації через комп'ютерні мережі. Такий метод отримав назву «мережева стеганографія». Цей термін вперше ввів Кжиштоф Щипьорский (Krzysztof Szczypiorski) в 2003 році. Поширений метод мережевої стеганографії передбачає зміну однієї з властивостей мережевого протоколу. Крім того, взаємозв'язок між двома або більше різними протоколами можна використовувати для кращого приховування передачі секретних повідомлень. Мережева стеганографія охоплює широкий спектр методів, зокрема:

WLAN стеганографія грунтується на методах, які використовуються для передачі стеганограм в бездротових локальних мережах (Wireless Local Area Networks). Практичний приклад WLAN стеганографії — система HICCUPS (Hidden Communication System for Corrupted Networks);

- *LACK стеганографія* — приховування повідомлень під час розмов з використанням IP-телефонії. Наприклад: використання пакетів, що затримуються, або навмисно пошкоджуються та ігноруються приймачем (цей метод називають LACK — Lost Audio Packets Steganography), або приховування інформації в полях заголовку, які не використовуються;
- *DAB- і DVB стеганографія* — стеганографічна обробка інформації, що циркулює у мережах цифрового звукового і/або телевізійного мовлення;

VoIP (англ. *voice over IP*) — технологія передачі медіа даних в реальному часі за допомогою протоколів TCP/IP. IP-телефонія — система зв'язку, при якій аналоговий звуковий сигнал від одного абонента дискретизується (кодується в цифровий вигляд), компресується і пересилається по цифрових каналах зв'язку до іншого абонента, де проводиться зворотна операція — декомпресія, декодування і відтворення.

Розмова відбувається у формі аудіо-потоків за допомогою протоколів RTP (Real-Time Transport Protocol);

LACK — це метод стеганографії для IP-телефонії, який модифікує пакети з голосовим потоком. Він використовує те, що в типових мультимедійних комунікаційних протоколах, таких як RTP, надмірно затримані пакети вважаються приймачем марними і відкидаються.

Принцип функціонування LACK виглядає наступним чином. Передавач (Аліса) вибирає один з пакетів з голосового потоку і його корисне навантаження замінює бітами таємного повідомлення — стеганограмою, яка вбудовується в пакет. Потім обраний пакет навмисно затримується. Кожного разу, коли надмірно затриманий пакет досягає отримувача, незнайомого з стеганографічною процедурою, він відкидається. Однак, якщо отримувач (Боб) знає про прихований зв'язок, то замість видалення отриманих RTP пакетів, він вилучає приховану інформацію.

14.2. Основні властивості стеганосистем

Приховання інформації тільки на основі факту невідомості зловмисникові методу або методів, закладених в основу приховання, на сьогоднішній день є малоефективним. Ще в 1883 р. фламандський криптограф А.Керкгоффз (A.Kerckhoffs) указував на той факт, що система захисту інформації повинна виконувати покладені на неї функції навіть при **повній інформованості противника** про її структуру та алгоритм функціонування.

Принцип Керкгоффза - правило, згідно з яким стійкість криптографічного алгоритму не має залежати від архітектури алгоритму, а має залежати тільки від ключів. Іншими словами, при оцінці надійності шифрування необхідно вважати, що супротивник знає все про систему шифрування, що використовується, крім ключів. Цей принцип так само розповсюджується і на побудову стеганографічних систем та визначає їх криптостійкість.

Таким чином, **безпека системи повинна повністю визначатися таємністю ключа.** Це означає, що зловмисник може повністю знати всі алгоритми роботи стеганосистеми та статистичні характеристики множин повідомлень і контейнерів, і це **НЕ дасть** йому ніякої додаткової інформації про наявність або відсутність повідомлення в даному контейнері.

Знання зловмисником факту наявності повідомлення в якому- небудь контейнері не повинне допомогти йому при виявленні повідомлень в інших контейнерах.

Заповнений контейнер повинен візуально не відрізнятися від незаповненого. Для задоволення цієї вимоги треба, здавалося б, впроваджувати приховане повідомлення у **візуально незначущі множини** сигналу. Однак ці ж множини використовують і алгоритми стиску. Тому, якщо зображення буде надалі піддаватися стиску, то приховане повідомлення може зруйнуватися. Отже, біти повинні **вбудовуватися у візуально значущі множини**, а відносна непомітність може бути досягнута за рахунок

використання спеціальних методів, наприклад, модуляції з розширенням спектра.

При побудові стеганосистем повинні враховуватися наступні правила:

– стеганосистема повинна мати прийнятну складність обчислення реалізації;

– методи приховування повинні забезпечити цілісність вбудованої інформації для одержувача;

– при виявленні факту існування вбудованого повідомлення, порушник не може його відтворити як відкритий текст;

– необхідно забезпечувати задану пропускну здатність стеганоконтейнера та каналу зв'язку;

– при побудові стеганосистеми необхідно використовувати таку модель потенційного порушника, при я якій він може мати повне уявлення про існування і функціонування стеганосистеми. Разом зтим йому не повинно бути відомо про місце знаходження, вид і розмір ключа, за допомогою якого можна визначити наявність повідомлення та його зміст;

– – порушник має бути позбавлений будь-яких технічних, організаційних та і інших переваг.

Основними поняттями у стеганографії є **повідомлення та контейнер**.

Повідомлення $\mathbf{m} \in \mathbf{M}$ – певна закрита інформація, яку необхідно приховати. Нехай $M = \{m_1, m_2, ..., m_n\}$- набір інформаційних повідомлень, що формуються в стеганосистемі.

Контейнер (стегоконтейнер) $\mathbf{c} \in \mathbf{C}$ – набір відкритих даних, які використовується для вбудовування закритої інформації; $C = \{c_1, c_2, ..., c_q\}$ – набір всіх контейнерів, де $q >> n$.

Порожній контейнер *(стегоконтейнер)* – контейнер $\mathbf{c_i}$, який не містить закритої інформації.

Заповнений контейнер – такий контейнер, який містить приховану інформацію ($\mathbf{C_m}$). Заповнений контейнер не повинен візуально відрізнятися від порожнього.

Окрім того вводять ще такі поняття, як: ***стегоканал*** - канал для передачі стегоконтейнера; ***ключ*** - ключ для додавання прихованого повідомлення до контейнера та вилучення (використовується не завжди); ***стеганосистема*** - методи і засоби, що використовуються для створення прихованого каналу для передачі інформації. На рис.14.10. наведена узагальнена модель стеганосистеми.

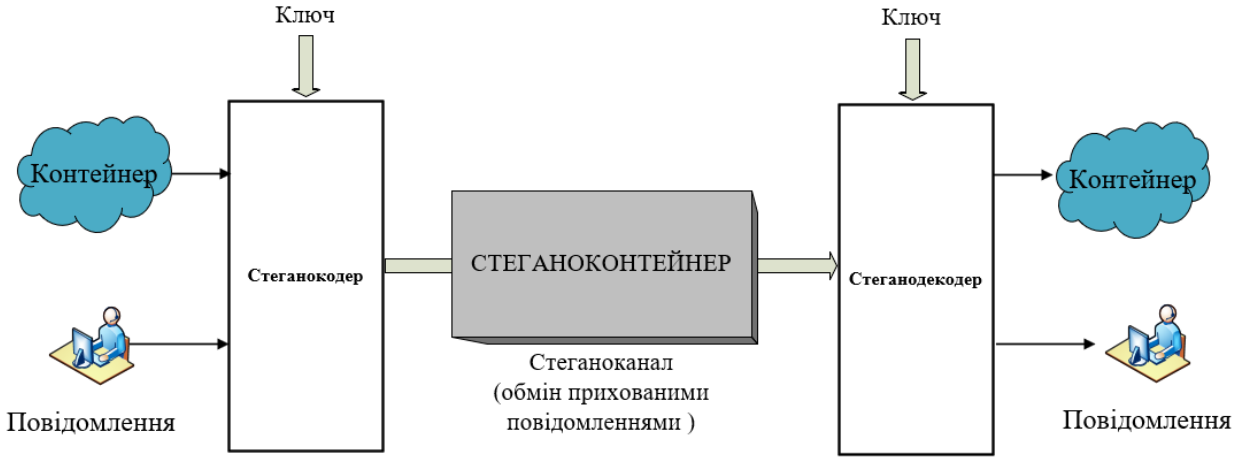


Рис.14.10. Узагальнена модель стеганосистеми

Математичну модель стеганосистеми можна рписати наступними залежностями:

$$E:\ C \times M \rightarrow S;$$

$$D:\ S \rightarrow M,$$

де $S = \{(c1, m1), (c2, m2), ..., (cn, mn), ..., (cq, mq)\} = \{s1, s2, ..., sq\}$ – множина контейнерів-результатів (стеганограм). Залежність E описує процес приховування інформації, залежність D – видобування прихованої інформації.

Розширену модель стеганосистеми можна представти у вигляді структури, яка наведена на рис.14.11 [39].

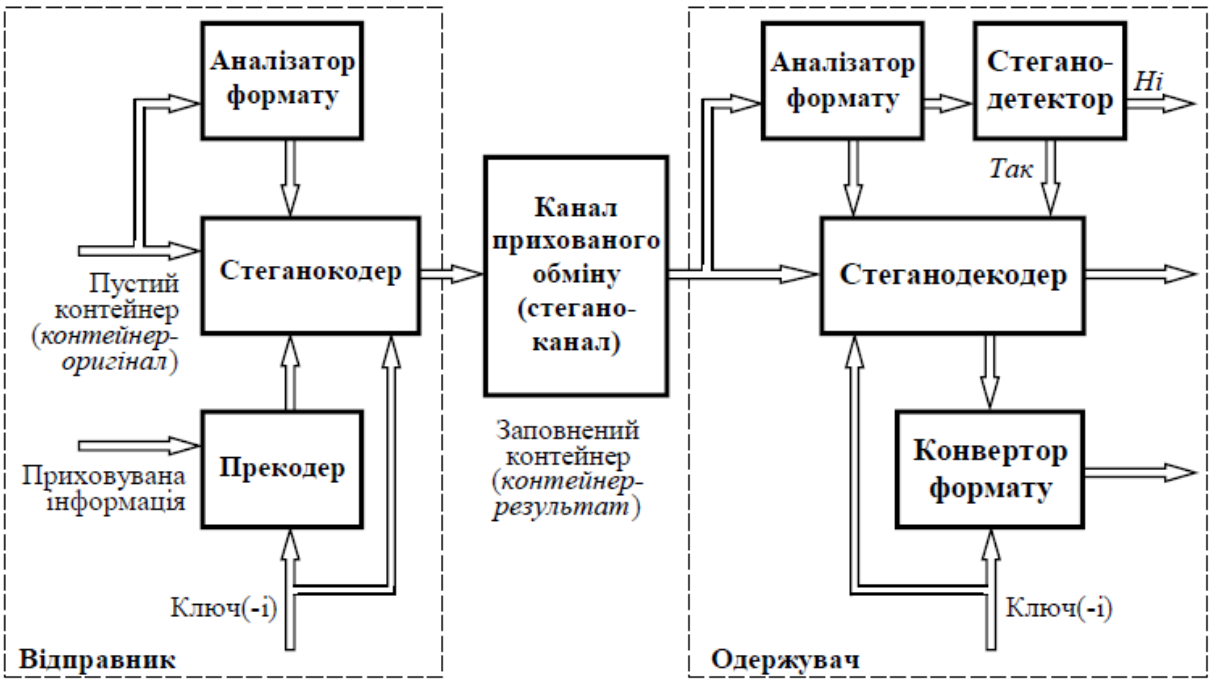


Рис.14.11. Структурна схема стеганосистеми як системи зв’язку

Під час передачі аудіо, відезображення або іншої інформації, яка використовується як контейнер, може відбутися її трансформація під дією різноманітних перетворень, у тому числі з використанням алгоритмів із

втратою даних (наприклад, стиснення даних). Через зміну розміру, перетворення в інший формат тощо можуть знадобитися додаткові алгоритми для виправлення помилок, щоб щоб зберегти цілісність вбудованого повідомлення. Іншими словами, може додатково використовуватися **завадостійке кодування**.

Прекодер виконує початкову обробку прихованої інформації. Одним із найважливіших етапів попередньої обробки повідомлень (і контейнерів) є обчислення узагальненого перетворення Фур'є. Це дає можливість вбудовувати дані в спектральний множині, що значно підвищує стійкість до спотворень.

Крім того, для підвищення конфіденційності вбудовування часто виконується попередня обробка за допомогою **ключа**.

Упакування повідомлення в контейнер (з урахуванням формату даних, що представляють контейнер) виконуються за допомогою **стеганокодера**. Вкладення відбувається, наприклад, шляхом модифікації найменших значущих бітів контейнера. Взагалі, саме алгоритм (стратегія) внесення елементів повідомлення в контейнер визначає методи стеганографії, які, у свою чергу, діляться на визначені групи, наприклад, залежно від того, файл якого формату був обраний як контейнер.

Перед упакуванням у контейнер, для підвищення захищеності секретної інформації останню можна зашифрувати досить стійким **криптографічним кодом.**

Спосіб формування стеганоконтейнера за допомогою стеганографії передбачає наявність певного розміру повідомлення та надійність його приховування. Очевидно, що при збільшенні об'єму повідомлення розмір стеганоконтейнера також буде збільшуватися і може статися ситуація, коли таке повідомлення буде помітним, а стеганоконтейнер викликати підозри. Таким чином, надійність приховування інформаціїї залежить від обсягу повідомлень в стеганоконтейнері, як показано на рис.14.12 [39].

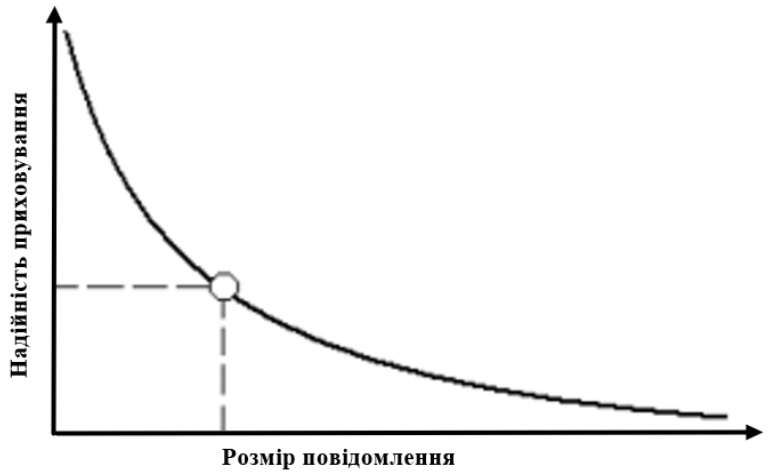


Рис.14.12. Залежність надійності приховування інформації від об'єму повідомлення

Змінюючи певні властивості стеганоконтейнера можна досягти високої надійності приховування повідомлень або їх великого обсягу. Але збільшення будь-якого з цих значень неминуче призводить до зниження іншого. Таким

чином, існує оптимальне значення між об’ємом даних та стійкістю системи до стеганоаналізу.

При проектуванні стеганосистм важливу роль відіграє пропускна здатність. Під пропускною здатністю каналів передачі приховуваних даних (**КППД**) або просто прихованою пропускною здатністю (ППЗ) розуміють максимальну кількість інформації, яка може бути вбудована до одного елементу (наприклад, пікселя чи відліку) контейнера. Обов’язковою умовою при цьому є безпомилковість передачі приховуваних даних одержувачеві, а також їх захищеність від таких атак порушника, як спроби виявлення факту наявності каналу прихованого зв’язку, одержання змісту приховуваних повідомлень, навмисне введення сфальсифікованих даних або ж руйнування вбудованої до контейнера інформації.

Розглядають два основних підходи до оцінки пропускної здатності КППД. Перший підхід - орієнтований на стеганографічні системи, в яких повідомлення, що підлягають захистові, повинні бути **безпомилково передані** в умовах активної протидії порушника. Порушник крім **пасивних дій** аналізу може використати й **активні дії** (атакуючий порушник) - руйнування приховуваної інформації. Даний підхід описує сценарій приховання так званих **безнадлишкових повідомлень** у даних контейнера, і, що найголовніше, дозволяє врахувати той факт, що крім спотворювань структури контейнера при вбудовуванні до нього приховуваних даних, можливі його навмисні спотворювання з боку порушника, крім того, існує ще й імовірність спотворень випадкового характеру, викликаних ненавмисними завадами каналу зв’язку.

Другий підхід - дає оцінки прихованої пропускної здатності безпосередньо в процесі вбудовування приховуваних повідомлень у **надлишкові дані контейнера**. Такий підхід враховує, що контейнери формуються реальними надлишковими джерелами з істотною пам’яттю, такими як джерела зображень або аудіосигналів.

В стеганографії розрізнюють системи із **секретним ключем і системи з відкритим ключем**. Системи із секретним ключем застосовують такий ключ, який має бути заздалегідь відомий авторизованим учасникам (або переданий через захищений канал під час певного обміну) перед початком секретного обміну секретними повідомленнями. Система відкритих ключів використовує різноманітні закриті ключі (відкриті та закриті) для вбудовування або вилучення прихованої інформації.

Таким чином, різноманіття стеганографічних систем можна прокласифікувати з огляду на відомості про ключ [44]: *безключові стеганосистеми*, *системи із секретним ключем*, *системи з відкритим ключем* та *змішані стеганосистеми*.

14.3. Цифрові водяні знаки

Забезпечення захисту авторського права, промислової власності або конфіденційних даних (зазвичай у цифровому вигляді) від несанкціонованого доступу - є однією з найдавніших і наразі невирішених проблем. У зв’язку з

інтенсивним розвитком і поширенням цифрових технологій, які дозволяють інтегрувати, зберігати, модифікувати та відтворювати різні типи даних (так звані мультимедійні технології), питання захисту інформації є надзвичайно актуальним. Методи стеганографії, які призначені для захисту авторських прав – називаються цифровими водяними знаками (**ЦВЗ - Digital Watermark**). На відміну від звичайних знаків, які можна інтегрувати з обєктом, їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення. ЦВЗ записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Розробляються різні засоби захисту інформації, організаційного та технічного характеру. Один з найбільш ефективних технічних засобів захисту мультимедійної інформації полягає у вбудовуванні в захисті об’єкта **невидимих міток** – ЦВЗ.

ЦВЗ — технологія, створена для захисту авторських прав мультимедійних файлів та інтелектуальної власності контейнера (*Intellectual Property*). Зазвичай ЦВЗ є невидимі без спеціальної обробки чи пристроїв. Разом з тим, ЦВЗ можуть бути видимими на зображенні або відео. Зазвичай, ця інформація являє собою логотип, текст, картинка, що ідентифікує автора (рис.14.13).

Рис.14.13. Приклад реалізації ЦВЗ (https://www.online-tech-tips.com/web-site-tips/how-to-easily-add-watermarks-to-your-online-images-before-uploading/)

Системи ЦВЗ актуальні для широкого ряду практичних застосувань, таких як завадостійка автентифікація аудіо та візуальних даних (зокрема контроль цілісності знімків камер спостереження, записів телефонних розмов и т.п.), автентифікація власника даних (захист авторських прав), автентифікація джерела даних, контроль телевізійного та радіомовлення, контроль копіювання і т.д. Стеганосистеми ЦВЗ мають на увазі два об’єкти інформаційного інтересу – вкраплене повідомлення та контейнер. Їх основна мета – зберегти цілісність вкрапленого повідомлення після певного ряду

можливих модифікацій стеганоконтейнера. Характерними атаками виступають активні та зловмисні атаки порушника, а також ненавмисні атаки, спричинені обробкою контейнера. ЦВЗ, як правило, має невеликий розмір, що дозволяє вкрапити його так, щоб забезпечити стійкість до ненавмисних та активних атак.

До основних характеристик ЦВЗ можнна віднести наступні.

Невідчутність (imperceptibility): вкраплення ЦВЗ повинно зберігати якість оригінального контейнера. Для аудіосигналів ЦВЗ повинен бути невідчутним на слух, для зображень ЦВЗ повинен бути візуально непомітним.

Стійкість (robustness): вкраплена інформація повинна коректно вилучатися легальним користувачем після типових операцій обробки в каналі зберігання і передачі сигналу-контейнера та цілеспрямованих атак порушника на ЦВЗ.

Безпека (security): несанкціонований користувач не повинен мати можливості оцінити секретні параметри системи ЦВЗ. При прийнятті принципу Кергоффза рівень безпеки системи оцінюється зусиллями, які необхідні для визначення стеганоаналітиком секретного ключа.

Місткість (capacity): захищуваний сигнал повинен мати деяку міру надлишковості, достатню вля вкраплення в нього ЦВЗ певного діапазону розмірів.

Швидкість (speed): системи ЦВЗ можуть застосовуватися у режимі реального часу, наприклад, потокове аудіо. Вкраплення та вилученя ЦВЗ повинно відбуватися досить швидко, щоб задовільнити вимоги даних застосувань.

Стеганографія застосовує ЦВЗ, коли сторони обмінюються секретними повідомленнями, впровадженими в цифровий сигнал. ЦВЗ використовується як засіб захисту документів з фотографіями — паспортів, кредитних карт, водійських посвідчень тощо.

ЦВЗ також можна використовувати для виявлення потенційного піратства. Наприклад, в зображення вбудовується інформація про час продажу та інформація про покупця. Головною відмінністю ЦВЗ від звичайного приховування інформації є наявність активного супротивника, що передбачає внесення змін в докумени (обєкт). Наприклад, якщо ЦВЗ використовується для захисту авторських прав, активні зловмисники намагатимуться видалити або змінити вбудований ЦВЗ. Тому головною вимогою до ЦВЗ є стійкість вбудованих даних до атак. Таємність не є настільки важливою, як у прихованій комунікації.

14.4. Алгоритм LSB стеганографії

LSB (Least Significant Bit, найменший значущий біт) — суть цього методу полягає в заміні останніх значущих бітів у контейнері (зображення, аудіо або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

Принцип цього методу продемонструємо на наступному прикладі.

Приклад 14.2. Припустимо, що є 8-бітне зображення в градаціях сірого. 00h (00000000b) позначає чорний колір, FFh (11111111b) — білий. Усього є 256 градацій (2^8). Також припустимо, що секретне повідомлення складається з 1 байта (8 біт) — наприклад, **01 10 10 11**b. При використанні 2 молодших біт в описах пікселів, нам буде потрібно 4 різних пікселя контейнера.

Нехай всі пікселі чорного кольору (мають 8-ми бітні нульові значення) і розташовані один за одним. Тоді пікселі, що містять приховане повідомлення, будуть виглядати наступним чином: 00000001 00000010 00000010 00000011. В цьому випадку колір пікселів зміниться і буде відрізнятися від дійсно чорного. Проведемо розрахунок змін: перший піксель змінить свою градацію на 1/255, другий і третій піксель — на 2/255, четвертий — на 3/255 градацій сірого. Зміни рівнів градацій таких значень є непомітними для людини. Таким чином, застосування такого способу передачі прихованих повідомлень залишиться непомітним на пристроях відображення (рис.14.14).

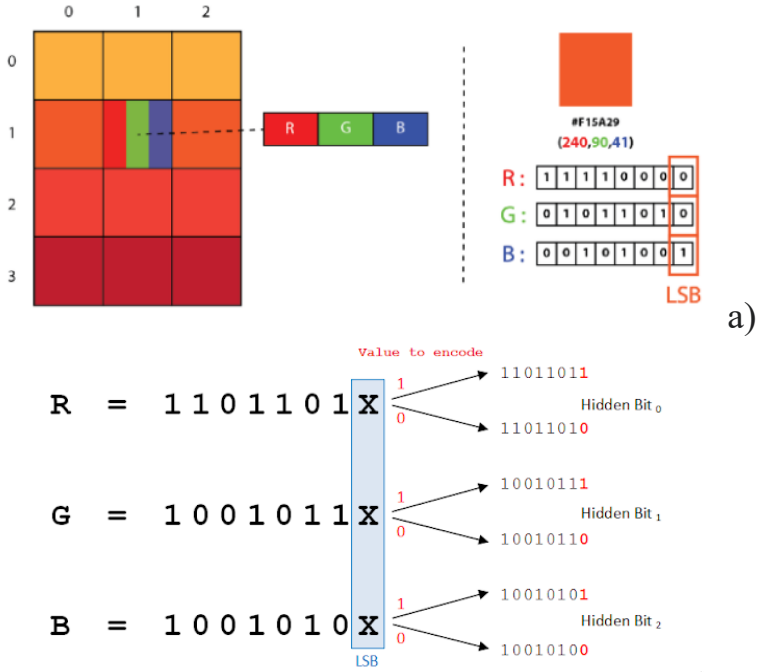


Рис.14.14. Принцип приховування інформації в молодших бітах бінарного представлення пікселя у форматі RGB (https://medium.com/swlh/lsb-image-steganography-using-python-2bbbee2c69a2)

В ролі базового контейнера пропонується використовувати файли BMP-зображень високої роздільності з глибиною кольору 24 та 32 біти, таємне зображення може мати розширення .BMP, .GIF, .PNG, .JPEG.

Приклад 14.3. Визначимо значення яскравості другого пікселю контейнеру-зображення в градаціях сірого кольору після приховування **секретного числа 233** методом заміни **двох** найменш значущих бітів, якщо його початкове значення рівне **235.**

Розпочати рішення такої задачі потрібно з представлення цих двох чисел: значення секретного повідомлення (233) і початкового значення контейнера (235) в бінарній формі:

233 --→ «11 10 10 01» 235 --→ «11 10 10 11».

Тоді перша пара чисел секретного повідомлення буде розташована в першому пікселі (11) на місці молодших двох значень біта, друга (10) – в другому пікселі і т.д. Таким чином значення другого пікселя перетвориться з «**11 10 10 11**» в «**11 10 10 10**», що буде відповідати числу 234. Такі зміни контейнера залишаться непомітними для ока людини, але це дасть можливість передати приховане повідомлення (рис.14.15).

Рис.14.15. Ілюстрація застосування методу **LSB** на прикладі передачі секретного числа.

Недоліком методу LSB є чутливість до розміру зображення, тобто чим менший розмір зображення, тим більше будуть відрізнятися два сусідні пікселі. Тому пропонується використовувати зображення з великою роздільністю. Також метод «видає себе» при побітовому перегляді зображення, де чітко видно області зображення в які «вбудовано» таємну інформацію. Попри це, метод запису **Least Significant Bit** є досить популярним, стійким та простим в реалізації.

Підвиди LSB-алгоритмів для растрових зображень без палітри

BlindHide (приховування наосліп). Найпростіший алгоритм, в якому дані записуються піксель за пікселем від верхнього лівого кута до нижнього правого кута зображення. Програма записує приховані дані в останній біт пікселя. Приховані дані нерівномірно розподілені всередині контейнера. Якщо приховані дані не заповнюють контейнер повністю, то лише верхня частина зображення буде містити приховані дані.

HideSee (заховати-знайти). Цей алгоритм псевдовипадково розподіляє приховані повідомлення по контейнеру, де для такого розподілу потрібен пароль. При такій реалізації не враховується властивість контейнера.

FilterFirst (попередня фільтрація). Алгоритм попередньо аналізує зображення на предмет виявлення тих місць (пікселів), розміщення в яких

прихованого повідомлення (в наймолодших бітах) буде найменш помітно для людини.

BattleSteg (стеганографія морської битви). Цей алгоритм буде найдосконалішим, де прихована інформація розміщується в таких місцях контейнера випадковим чином (як у HideSeek), щоб це буде найменш помітно для людини.

Інші методи приховування інформації в графічних файлах орієнтовані на формати файлів з втратою, наприклад, JPEG. На відміну від LSB вони більш стійкі до геометричних перетворень. Це виходить за рахунок варіювання в широкому діапазоні якості зображення, що призводить до неможливості визначення джерела зображення.

14.5. Підготовка до завдання

Ознайомитися з теоретичними положеннями предмету стеганографії, її класифікацією, метою застосування, основними механізмами реалізації та властивостями.

14.6. Практичне завдання

1. Практична робота полягає у створенні і порівнянні двох стеганографічних методів передавання інформації в графічних зображеннях.

ПЕРШИЙ МЕТОД базується на заміні ОДНОГО **випадкового** пікселя зображення НОВИМ значенням, яке відповідає символьному коду ОДНОГО знака текстового повідомлення. Тобто, якщо буде передано повідомлення з 5 символів, то в графічному зображенні будуть замінені випадковим чином також 5 пікселів. При цьому формується текстовий файл з координатами змінених пікселів (наприклад, **keys.txt - ключі**), або ключова інформація.

Для реалізації даної задачі пропонується готова функція **«stega_encrypt»**, яка наведена на рис.14.16 та рис.14.17. При її запуску запитується про назву файла – контейнера, (він вже має існувати, розширення файла ***.png**). При цьому формується новий файл (контейнер), в якому буде зберігатися прихована інформація (в програмі фігурує файл «**new.png**»). Для дешифрування інформації запускається функція **«stega_decrypt» (**рис.14.16, 14.17**),** яка запитує про файл-контейнер (**new.png**) та інформацію про ключ (**keys.txt**).

Задачею даного пункта є реалізація даного першого методу **(можете запропонувати свою власну програмну реалізацію)** і аналіз змісту контейнерів.

На рис.14.18. наведений результат приховування повідомлення в різні контейнери для проведення аналізу щодо візуального сприйняття.

Функція кодування повідомлення до зображення

```
from PIL import Image, ImageDraw
from random import randint
```

```
def stega_encrypt():
    keys = []#сюди будуть розташовані ключі
    img = Image.open(input("path to image: "))#створюємо файл для розміщення інформації
    draw = ImageDraw.Draw(img)#ообєкт для вставки
    width = img.size[0]#ширина
    height = img.size[1]#висота
    pix = img.load()#всі пікселі
    f = open('keys.txt','w')#текстовий файл для ключів

    for elem in ([ord(elem) for elem in input("text here: ")]):
        key = (randint(1,width-10),randint(1,height-10))
        g, b = pix[key][1:3]
        draw.point(key, (elem*1, g , b))
        f.write(str(key)+'\n')

    print('keys were written to the keys.txt file')
    img.save("new.png", "PNG")
    f.close()

stega_encrypt()
```

```
path to image: black.png
text here: Hello BI81 + CK
keys were written to the keys.txt file
```

Функція декодування повідомлення з зображення

```
from PIL import Image
from re import findall

def stega_decrypt():

    a = []
    keys = []
    img = Image.open(input("path to image: "))
    pix = img.load()
    f = open(input('path to keys: '),'r')
    y = str([line.strip() for line in f])

    for i in range(len(findall(r'\((\d+)\,',y))):
        keys.append((int(findall(r'\((\d+)\,',y)[i]),int(findall(r'\,\s(\d+)\)',y)[i])))
    for key in keys:
        a.append(pix[tuple(key)][0])
    return ''.join([chr(int(elem/1)) for elem in a])

print("you message: ",stega_decrypt())
```

```
path to image: new.png
path to keys: keys.txt
you message:  Hello BI81+CK
```

Рис.14.16. Скрін програми для реалізації заміни значення пікселя в растровому зображенні

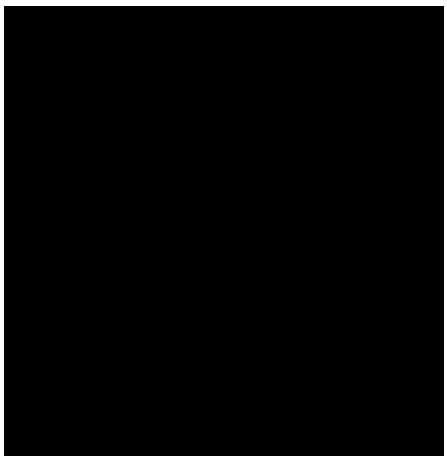
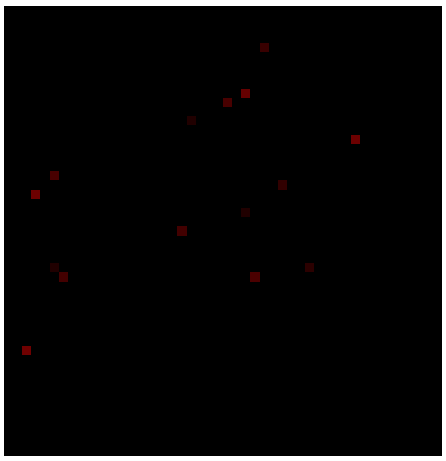
Функція додавання повідомлення до зображення

```
from PIL import Image, ImageDraw
from random import randint

def stega_encrypt():
    keys = []#сюди будуть розташовані ключі
    img = Image.open(input("path to image: "))#створюємо файл для розміщення інформації
    draw = ImageDraw.Draw(img)#ообєкт для вставки
    width = img.size[0]#ширина
    height = img.size[1]#висота
    pix = img.load()#всі пікселі
    f = open('keys.txt','w')#текстовий файл для ключів
    for elem in ([ord(elem) for elem in input("text here: ")]):
        key = (randint(1,width-10),randint(1,height-10))
        g, b = pix[key][1:3]
        draw.point(key, (elem*1, g , b))
        f.write(str(key)+'\n')
    print('keys were written to the keys.txt file')
    img.save("new.png", "PNG")
    f.close()
stega_encrypt()
```

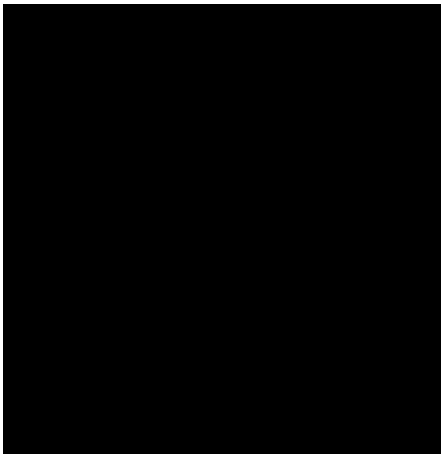
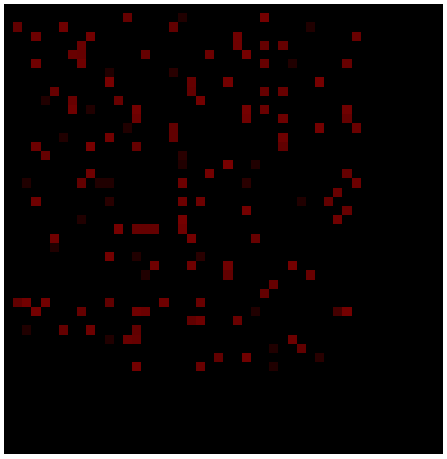
Функція видобування повідомлення з зображення

```
from PIL import Image
from re import findall

def stega_decrypt():
    a = []
    keys = []
    img = Image.open(input("path to image: "))
    pix = img.load()
    f = open(input('path to keys: '),'r')
    y = str([line.strip() for line in f])
    for i in range(len(findall(r'\((\d+)\,',y))):
        keys.append((int(findall(r'\((\d+)\,',y)[i]),int(findall(r'\,\s(\d+)\)',y)[i])))
    for key in keys:
        a.append(pix[tuple(key)][0])
    return ''.join([chr(int(elem/1)) for elem in a])
print("you message: ", stega_decrypt())
```

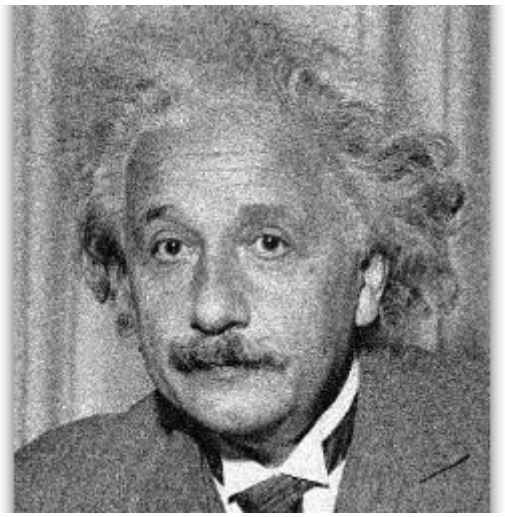
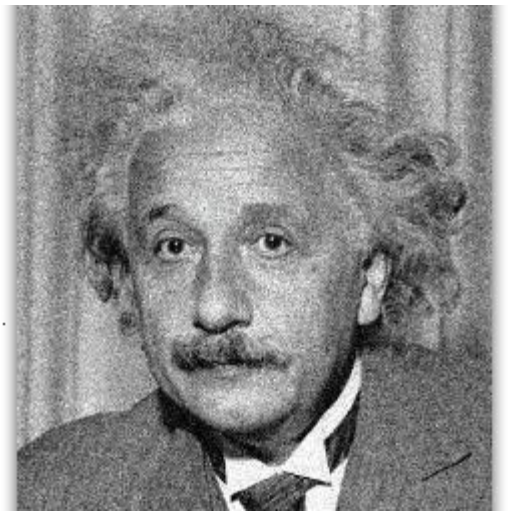
Рис.14.17. Код програми (Python) для реалізації заміни значення пікселя в растровому зображенні на значення прихованого повідомлення


а)

б)

Повідомлення: «***Hello BI81 + CK***»


а)

б)

Повідомлення: «***Example of visual observation of the contents of the original container (model example of a solid black image) (a) and with the posted message (b)***»


а)

б)

Повідомлення: «***Example of visual observation of the contents of the original container (model example of a solid black image) (a) and with the posted message (b)***»

Рис.14.18. Приклад візуального аналізу модельних стеганоконтейнерів:
а) порожній контейнер; б) – контейнер з прихованим повідомленням.

2. Дослідити принципи програмної реалізації (рис.14.16) методу вбудування повідомлення в растрове зображення та її функціонування.
3. Проаналізувати відмінність новоствореного файла **new.png** у порівнянні з початковим контейнером. В якості пустого контейнера візьміть однотонне (наприклад біле і чорне) растрове зображення невеликих розмірів (наприклад, 10*10, 100*100 пікселів) для зручності їх візуального аналізу.
4. Провести дослідження і зясувати, як загальний фон контейнера (білий і чорний), його розміри (10*10 та 100*100), розмір інформаційного повідомлення (коротке - одне слово, довге - одне речення) впливають на новостворений файл **new.png** і його візуальну відмінність від оригіналу.
5. Провести візуальний порівняльний аналіз новостворених файлів з прихованим повідомленням (**new.png** - стеганоконтейнер) у порівнянні з оригіналом (пустий контейнер). Результати візуального порівняння занести в таблицю 14.1 та зробити висновки.

Таблиця 14.1.

Візуальний аналіз спостереження за прихованим повідомленням

№	Тип контейнера (довжина повідомлення)	Розмір контейнера 10*10 (пікс)	Розмір контейнера 100*100 (пікс)	Візуальний аналіз (непомітно, слабо помітно, помітно)
1	Білий (коротке)			
2	Білий (довге)			
3	Чорний (коротке)			
4	Чорний (довге)			
5	Відтінки сірого (довге)			
6	Кольорове зображення (довге)			

6. Зясувати, ЩО ПОТРІБНО ЗМІНИТИ в запропонованому програмному рішенні, щоб додавання нових пікселів в стеганоконтейнер (їх колір, яскравість) була: а) по можливості менш помітною; б)відповідала б контекстному фону зображення; в)взагалі не була б помітною?
7. Зясувати, чи впливає зміна формату (наприклад, перехід від *.png до *.jpeg та ін.), архівування з подальши відновленнмя, інші методи впливу на відновлення прихованої інформації зі стеганоконтейнера за ПЕРШИМ методом?
8. Проаналізувати програмну реалізацію ДРУГОГО МЕТОДУ, акий реалізує алгоритм LSB в растровому зображенні (рис.14.19). Реалізувати даний метод.
9. Пояснити, чим відрізняється перший метод приховування повідомлення від другого. Яким чином відбувається інтегрування

повідомлення до зображення за першим і другим методом? Для якого методу приховане повідомлення буде менш помітним? В якому випадку інтегрування повідоблення до зображення за другим методом може стати візуально помітним?

10. Зясувати, чи впливає зміна формату (наприклад, перехід від *.png до *.jpeg та ін.), архівування з подальши відновленнмя, інші методи впливу на відновлення прихованої інформації зі стеганоконтейнера, отриманим за ДРУГИМ методом?

14.7. Зміст протоколу роботи

Протокол до виконаної пактичної роботи оформлюється у вигляді файлу та містить титульний лист з назвою роботи та її автора. Наступна сторінка містить мету роботи, короткі теоретичні відомості, які складають методологію виконання роботи, завдання, експериментальні результати та їх аналіз, висновки.

Робота захищається індивідуально.

14.8. Контрольні питання для самоперевірки

1. Дайте визначення стеганографії, галузі застосування.
2. Наведіть класифікацію стеганографічних методів та дайте їм порівняльну характеристику з точки зору криптостійкості.
3. Надайте класифікацію класичної стеганографії. Наведіть приклади класичних методів приховування інформації.
4. Що таке лінгвістична стеганографія, її різновиди. Наведіть приклади.
5. Що таке соціальна стеганографія, наведіть приклади.
6. Що таке комп’ютерна стеганографія. Класифікація комп’ютерної стеганографії.
7. Що таке мережева стеганографія. Наведіть приклади.
8. Заст осування принципу Керкгоффза при побудові стеганографічних систем.
9. Охарактеризуйте узагальнену модель стеганосистеми та її математичну модель.
10. Поняття пропускної здатності каналу передачі приховуваних даних.
11. Класифікація стеганосистем з огляду на відомості про ключ. Основні всластивості таких стеганосистем.
12. Поняття цифрових водяних знаків та їх застосування.
13. Яка основна задача систем цифрових водяних знаків?
14. Які основні властивості цифрових водяних знаків?
15. Назвіть методи формування цифрових водяних знаків.
16. Дайте якісну оцінку залежності надійності приховування інформації від об’єму повідомлення.
17. Охарактеризуйте алгоритм LSB стеганографії. Наведіть приклад.
18. Які існують різновиди алгоритму LSB та їх криптостійкість.
19. Чим визначається безпека стеганосистеми?

20. На чому базується найпростіша стеганографія на прикладі передачі зображень?
21. Який основний недолік алгоритму LSB?
22. Що є основною характеристикою стеганографічних систем?
23. Яке призначення стеганокодера, стеганодекодера?
24. Охарактеризуйте пасивного та активного порушника для стеганографічних систем?
25. Охарактеризуйте метод псевдовипадкового інтервалу.
26. Чим відрізняються прихований та відкритий стеганоконтейнери?
27. Охарактеризуйте основні вимоги до стеганосистем з секретним ключем.
28. Охарактеризуйте основні вимоги до стеганосистем з відкритим ключем

14.9. Задачі до практикуму №14

1. Текстовий документ містить 11 слів, між якими знаходяться пробіли для передачі секретного повідомлення. Звичайний пробіл призначений для кодування «0», а подвійний пробіл – «1». Яке найбільше десяткове ціле число може бути передано за допомогою цього документа? Відповідь обгрунтуйте.
2. Чому буде рівне значення яскравості четвертого пікселя контейнера растрового зображення в градаціях сірого після приховування секретного числа 177 методом заміни одного найменш значущого біта, якщо його початкове значення відповідало білому кольору? Чорному кольору? Відповідь обгрунтуйте.
3. Скільки приблизно можна додати інформації (кбайт) до растрового зображення розміром 512*512 пікселів методом заміни одного наймолодшого біту в градаціях сірого? В форматі RGB? Відповідь обгрунтуйте.
4. Нехай дано растрове зображення розміром 128*128 пікселів. Який максимальний розмір повідомлення, яке складається з малих і великих букв українського алфавіту та десяти цифр можна приховати методом заміни двох наймолодших бітів?

14.10. Завдання до самостійної роботи

1. Особливості зорової системи людини та їх врахування при реалізації стеганографічних систем.
2. Принципи побудови стеганографічних систем із секретним та відкритими ключем. Особливості побудови, криптостійкість.
3. Методи приховування даних у частотній множині зображень.
4. Методи приховування даних в аудіосигналах.
5. Методи текстової стеганографії.
6. Атаки на стеганосистеми та протидія атакам.

Дослідження методу Least Significant Bit на прикладі растрового зображення

```
#import libraries
import sys
import numpy as np
from PIL import Image
np.set_printoptions(threshold=sys.maxsize)

#encoding function
password = ""
def Encode(src, message, dest,password):
    img = Image.open(src, 'r')
    width, height = img.size
    array = np.array(list(img.getdata()))
    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4
    total_pixels = array.size//n
    message += password
    b_message = ''.join([format(ord(i), "08b") for i in message])
    req_pixels = len(b_message)

    if req_pixels > (total_pixels * 3):
        print("ERROR: Need larger file size")
    else:
        index=0
        for p in range(total_pixels):
            for q in range(0, 3):
                if index < req_pixels:
                    array[p][q] = int(bin(array[p][q])[2:9] + b_message[index], 2)
                    index += 1

        array=array.reshape(height, width, n)
        enc_img = Image.fromarray(array.astype('uint8'), img.mode)
        enc_img.save(dest)
        print("Image Encoded Successfully")

#decoding function
def Decode(src, password):
    img = Image.open(src, 'r')
    array = np.array(list(img.getdata()))
    if img.mode == 'RGB':
        n = 3
    elif img.mode == 'RGBA':
        n = 4
    total_pixels = array.size//n
    hidden_bits = ""
    for p in range(total_pixels):
        for q in range(0, 3):
            hidden_bits += (bin(array[p][q])[2:][-1])
```

```
    hidden_bits = [hidden_bits[i:i+8] for i in range(0, len(hidden_bits), 8)]
    message = ""
    hiddenmessage = ""
    for i in range(len(hidden_bits)):
        x = len(password)
        if message[-x:] == password:
            break
        else:
            message += chr(int(hidden_bits[i], 2))
            message = f'{message}'
            hiddenmessage = message
    #verifying the password
    if password in message:
        print("Hidden Message:", hiddenmessage[:-x])
    else:
        print("You entered the wrong password: Please Try Again")

#main function
def Stego():
    print("--Welcome to $t3g0--")
    print("1: Encode")
    print("2: Decode")
    func = input()
    if func == '1':
        print("Enter Source Image Path")
        src = input()
        print("Enter Message to Hide")
        message = input()
        print("Enter Destination Image Path")
        dest = input()
        print("Enter password")
        password =  input()
        print("Encoding...")
        Encode(src, message, dest,password)

    elif func == '2':
        print("Enter Source Image Path")
        src = input()
        print("Enter Password")
        password = input()
        print("Decoding...")
        Decode(src,password)

    else:
        print("ERROR: Invalid option chosen")

Stego()
```

Рис.14.19. Скрін програми для реалізації методу Least Significant Bit

РЕКОМЕНДОВАНА ЛІТЕРАТУРА ТА ПОСИЛАННЯ

1. Бобало Ю. Я. Інформаційна безпека: навч. посібник / Ю. Я. Бобало, І. В. Горбатий, М. Д. Кіселичник та ін.; за заг. ред. д-ра техн. наук, проф. Ю. Я. Бобала та д-ра техн. наук, доц. І. В. Горбатого. – Львів : Видавництво Львівської політехніки, 2019. – 580 с.
2. Інформаційна безпека держави : навчальний посібник / В.М.Рудницький, С.О.Гнатюк, Н.В.Лада, Р.В.Бреус. – Харків : ТОВ «ДІСА ПЛЮС», 2018. – 359 с.
3. Лужецький В. А. Основи інформаційної безпеки / В. А. Лужецький, А. Д. Кожухівський, О. П. Войтович. – Вінниця: ВНТУ, 2013. – 267 с.
4. Грайворонський М. В. Безпека інформаційно-комунікаційних систем. / М. В. Грайворонський, О. М. Новіков. – К. : Видавнича група BHV, 2009. – 608 с.
5. Корченко О.Г. Прикладна криптологія : системи шифрування : підручник / О. Г. Корченко, В. П. Сіденко, Ю. О. Дрейс. – К. : ДУТ, 2014. – 448 с.: іл.
6. Про затвердження переліку стандартів та технічних специфікацій, дозволених для реалізації в засобах криптографічного захисту інформації. Режим доступу: – https://zakon.rada.gov.ua/laws/show/z1272-20#Text , 22.05.2023.
7. Bruce Schneier. Applied Cryptography. Protocols, Algorithms, and Source Code in C John Wiley & Sons. 1996, 784 p.
8. Козіна Г.Л. Криптографія від історії до сучасних стандартів: навч.посібник / Г. Л. Козіна. – Запоріжжя : НУ «Запорізька політехніка», 2020. – 192 с.
9. Горбенко І. Д. Прикладна криптологія. Теорія. Практика. Застосування : Монографія / І. Д. Горбенко, Ю. І. Горбенко. – Харків : Видавницт-во “Форт”, 2012. – 880 с.: іл.
10. Горбенко І. Д. Захист інформації в інформаційно-телекомунікаційних системах: Навч. посіб. для студ. Ч. 1. Криптографічний захист інформації / І. Д. Горбенко, Т. О. Гріненко. – Х. : Харк. нац. ун-т радіоелектрон., 2004. – 368 с.: іл.
11. Щур Н.О., Покотило О.А. Основи криптології: навч. посібник. – Житомир: Державний університет «Житомирська політехніка», 2021. 120 с.
12. Франчук В.М. Захист інформаційних ресурсів: криптографічні та стеганографічні методи захисту даних. Посібник для викладачів, вчителів та студентів інформатичних спеціальностей. – К.: НПУ імені М.П. Драгоманова, 2012. – 120 с.
13. Глинчук Л.Я. Криптологія: навч.-метод. посіб. / Людмила Ярославівна Глинчук – Луцьк: Вежа-Друк, 2014. – 164 с.

14. Безпека інформаційних систем і технологій: Навч. посібник / В. І. Єсін, О. О. Кузнецов, Л. С. Сорока. – Х. : ХНУ імені В. Н. Каразіна, 2013. – 632 с.
15. Кузнецов О. О. Захист інформації в інформаційних системах.Методи традиційної криптографії : навчальний посібник / О. О. Кузнецов, С. П. Євсєєв, О. Г. Король. – Харків : Вид. ХНЕУ, 2010. – 316 с.
16. Захист інформації в мережах передачі даних: Підручник / Юдін О.К., Корченко О.Г., Конахович Г.Ф. – К.: Вид-во ТОВ "НВП" ІНТЕРСЕРВІС", 2009. – 716 с.
17. Хорошко В.А., Методы и средства защиты информации / Хорошко В.А., Чекатков А.А.. — К.:Изд-во Юниор, 2003. – 504 с.
18. Задірака В.К. Компьютерная криптологія. Підручник. К, 2002 ,504с.
19. Задірака В.К. Комп'ютерні технології криптографічного захисту інформації на спеціальних цифрових носіях: навч. посіб. / В. К. Задірака, А. М. Кудін, В. О. Людвиченко, О. С.Олексюк. – К. -Тернопіль: Підручники і посібники, 2007. – 272 с.
20. Тарнавський Ю.А. Технології захисту інформації [Електронний ресурс]: підручник для студ. спеціальності 122 «Комп'ютерні науки». КПІ ім. Ігоря Сікорського. — Київ : 2018. – 162 с. Режим доступу: – https://ela.kpi.ua/bitstream/123456789/23896/1/TZI_book.pdf, 22.05.2023.
21. Гребенніков В.В. Історія криптології & секретного звязку. [Електронний ресурс] / Режим доступу: - https://www.academia.edu/39113887/Історія_криптології_and_секретного_звязку, 22.05.2023.
22. Handbook of Applied Cryptography [Електронний ресурс] / Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. Режим доступу: – http://cacr.uwaterloo.ca/hac/about/chap1.pdf, 22.05.2023.
23. Криптографічне кодування: обробка та захист інформації: колективна монографія / під ред. В.М.Рудницького. — Харків: ТОВ «ДІСА ПЛЮС», 2018. – 139 с.
24. Криптографічне кодування: синтез операцій потокового шифрування з точністю до перестановки: монографія / В.М.Рудницький, Н.В. Лада, В.Г. Бабенко. – Харків: ТОВ «ДІСА ПЛЮС», 2018. – 184 с.
25. Richard E. Klima, Neil P. Sigmon. Cryptology: Classical and Modern with Maplets: // CRC Press. — 2012.
26. Захарченко М.В. Асиметричні методи шифрування в телекомунікаціях: навч. посіб. / М.В.Захарченко, О.В.Онацький, Л.Г.Йона, Т.М.Шинкарчук. – Одеса: ОНАЗ ім. О.С.Попова, 2011. – 184 с.
27. Богуш В. М. Криптографічні застосування елементарної теорії чисел: Навч. посібник / В. М. Богуш, В. А. Мухачов. – К.: Державний ун-т інформаційно-комунікаційних технологій, 2006. – 126 с.: іл.
28. Кузнецов Г.В. Математичні основи криптографії / Г.В. Кузнецов, В.В. Фомичов, С.О. Сушко, Л.Я. Фомичова. – Дніпропетровськ: НГУ, 2006. – 391 с.

29. Сушко С.О. Математичні основи криптоаналізу: навч. посібник для студ. вищ. навч. закл.: рек. МОНУ / С.О. Сушко, Г.В. Кузнецов, Л.Я. Фомичова, А.В. Корабльов. – Дніпропетровськ: НГУ, 2010.-466с.
30. Криптологія у прикладах, тестах і задачах: навч. посібник / Т.В.Бабенко, Г.М.Гулак, С.О.Сушко, Л.Я.Фомичова. - Д.: Національний гірничий університет, 2013. - 318 с
31. RSA — криптографічний алгоритм. Режим доступу: – https://uk.wikipedia.org/wiki/RSA, 22.05.2023.
32. Бессалов А., Телиженко А. Криптосистемы на эллиптических кривых. – К.: «Політехніка», 2004. – 224 с.
33. Методи та засоби швидкого багатоканального гешування даних в комп'ютерних системах : монографія / Ю. В. Баришев, В. А. Лужецький; за заг. ред. В. А. Лужецького – Вінниця : ВНТУ, 2016. – 144 с.
34. Preneel B. Analysis and Design of Cryptographic Hash Functions. PhD thesis / Bart Preneel. – Katholieke Universiteit Leuven, 1993. – 338 с. – Режим доступу: http://homes.esat.kuleuven.be/~preneel/phd_preneel_feb1993.pdf, 22.05.2023.
35. Козіна Г.Л. Криптопротоколи: схеми цифрового підпису: навч.посіб./ Г.Л. Козіна, М.А. Молдов'ян, Г.В. Неласа. – Запоріжжя: ЗНТУ, 2014. – 170 с.
36. Горбенко Ю.І. Інфраструктури відкритих ключів. Електронний цифровий підпис. Теорія та практика : монографія. / Горбенко Ю.І. Горбенко І.Д. – Харків : Видавництво «Форт», 2010. – 608 с.
37. ISO/IEC 9798. [Електронний ресурс]. – Режим доступу: https://www.iso.org/standards.html, 22.05.2023.
38. Стеганографія : навчальний посібник / О. О. Кузнецов, С. П. Єв-сеєв, О. Г. Король. – Х. : Вид. ХНЕУ, 2011. – 232 с.
39. Компютерна стеганографічна обробка й аналіз мультимедійних даних. Підручник. / Г.Ф.Конахович, Д.О.Прогонов, О.Ю.Пузиренко. – Київ: «Центр учбової літератури». 2018. – 558с.
40. Конахович Г.Ф., Пузыренко А.Ю. Компьютерная стеганография. Теория и практика. К.: «МК-Пресс», 2006. — 288 с.
41. Хорошко В.О., Азаров О.Д., Шелест М.Є., Яремчук Ю.Є. Основи комп'ютерної стеганографії: Навч. посіб. — Вінниця: ВДТУ, 2003. – 143с.
42. Wayner P. Strong Theoretical Steganography. – Berlin: Cryptologia, 2005. 410 p.
43. Лукічов В.В. Методи та засоби стеганографічного захисту інформації на основі вейвлет-перетворень: монографія / В.В.Лукічов, В.А.Лужецький, А.С.Васюра. – Вінниця : ВНТУ, 2014. – 160 с.
44. Комп'ютерна стеганографія: навчальний посібник / В.О.Хорошко, Ю.Є.Яремчук, В.В.Карпінець – Вінниця: ВНТУ, 2017. – 155с.

Нормативні інформаційні ресурси

1. Закон України від 15 грудня 2005 року № 3200-IV "Про основи національної безпеки України".
2. Закон України "Про інформацію". Із змінами, внесеними згідно із Законом від 07.02.2002 № 3047-III-ВР.
3. Закон України "Про Національну програму інформатизації" Із змінами, внесеними згідно із Законом від 13.09.2001 № 2684-III-ВР.
4. Закон України «Про захист інформації в інформаційно-телекомунікаційних системах» від 31.05.2005 № 2594-IV.
5. Закон України «Про електронний цифровий підпис» від 22.05.2003 № 852-IX.
6. Закон України «Про електронний документ та електронний документообіг» від 22.05.2003 № 851-IV.
7. Директива 1999/93/ЄС Європейського парламенту та Ради від 13 грудня 1999 року про систему електронних підписів, що застосовується в межах Співтовариства.
8. Правила посиленої сертифікації, затверджені наказом Департаменту спеціальних телекомунікаційних систем та захисту інформації Служби безпеки України від 13.01.2005 № 3, зареєстровані в Міністерстві юстиції України 27.01.2005 за № 104/10384 (у редакції наказу Департаменту спеціальних телекомунікаційних систем та захисту інформації Служби безпеки України від 10.05.2006 № 50).
9. ДСТУ ETSI TS 101 456:2015 Електронні підписи та інфраструктура (ESI). Вимоги до політики органів сертифікації, які видають кваліфіковані сертифікати (ETSI TS 101 456:2007, IDT).
10. ДСТУ ISO/IEC 13888-1:2015 Інформаційні технології. Методи захисту. Неспростовність. Частина 1: Загальні положення (ISO/IEC 13888-1:2009, IDT).
11. ДСТУ ISO/IEC 13888-3:2015 Інформаційні технології. Методи захисту. Неспростовність. Частина 3. Механізми з використанням асиметричних методів (ISO/IEC 13888-3:2009, IDT).
12. ДСТУ ISO/IEC 11770-3:2015 Інформаційні технології. Методи захисту. Керування ключами. Частина 3: Механізми з використанням асиметричних методів (ISO/IEC 11770-3:2008, Cor 1:2009, IDT).. .
13. ДСТУ ISO/IEC 9798-3:2021 Інформаційні технології. Методи безпеки ІТ. Автентифікація об’єктів. Частина 3. Механізми з використанням методу цифрового підпису (ISO/IEC 9798-3:2019, IDT).

14. ДСТУ ISO/IEC 18033-2:2015 Інформаційні технології. Методи захисту. Алгоритми шифрування. Частина 2. Асиметричні шифри (ISO/IEC 18033-2:2006, IDT).
15. ДСТУ ISO/IEC 9796-3:2015 Інформаційні технології. Методи захисту. Схеми цифрового підпису, які забезпечують відновлення повідомлення. Частина 3. Механізми, що грунтуються на дискретному логарифмі (ISO/IEC 9796-3:2006, IDT).
16. ДСТУ ISO/IEC 15946-1:2019 Інформаційні технології. Методи захисту. Криптографічні методи на основі еліптичних кривих. Частина 1. Загальні положення (ISO/IEC 15946-1:2016, IDT).
17. ДСТУ ISO/IEC 15946-5:2019 Інформаційні технології. Методи захисту. Криптографічні методи на основі еліптичних кривих. Частина 5. Генерування еліптичних кривих (ISO/IEC 15946-5:2017, IDT).
18. ДСТУ ISO/IEC 14888-3:2019 Інформаційні технології. Методи захисту. Цифрові підписи з доповненням. Частина 3. Механізми на основі дискретного логарифмування (ISO/IEC 14888-3:2018, IDT).
19. ДСТУ 4145-2002 «Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння».

Інформаційні ресурси

1. https://cip.gov.ua
2. https://zakon.rada.gov.ua/laws/show/2163-19#Text
3. https://cybersec.net.ua/pro-nas.html
4. https://attack.mitre.org
5. www.nist.gov
6. www.ansi.org
7. www.cryptography.org
8. www.iso.org
9. www.linuxiso.org
10. www.cryptography.com
11. www.cacr.math.uwaterloo.ca
12. www.financialcryptography.com
13. www.cryptonessie.org
14. The CrypTool Portal [Електронний ресурс]. — Режим доступу : http://www.cryptool.org/en, 22.05.2023.