**ЗМІСТ**

|  |  |
| --- | --- |
| ВСТУП …………………………………………………………...……………. | 4 |
| РОЗДІЛ 1. МЕТОДИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ… | 6 |
| 1.1 Математичні основи криптографічного захисту інформації………… | 6 |
| 1.2 Симетричні криптографічні системи………………………………….. | 14 |
| 1.3 Асиметричні криптографічні системи ……………………………….. | 18 |
| 1.4 Хеш-функції…………………………………………………………….. | 22 |
| 1.5 Порівняльний аналіз симетричних та асиметричних алгоритмів…… | 24 |
| РОЗДІЛ 2. ПРАКТИЧНЕ ЗАСТОСУВАННЯ МАТЕМАТИЧНИХ МЕТОДІВ В КРИПТОГРАФІЇ……………………………………………………………. | 28 |
| 2.1 Розробка математичної моделі криптоалгоритму…………………… | 28 |
| ВИСНОВКИ ………………………………………………………………….. | 35 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ……………………………………. | 36 |
| ДОДАТОК…………………………………………………………………….. | 38 |

**ВСТУП**

Останнім часом активно обговорюється проблема захисту інформаційних мереж.Щоб вберегти непосвячену людину від некоректних дій з інформаційними технологіями можна використати вчення про захист інформації, що називається криптографією.

Криптографія – галузь науки, яка вивчає математичні методи та технологїї забезпечення автентичності, цілісності та конфіденційності інформації. Була зумовлена практичною потребою передавати важливу інформацію більш надійним способом.Для математичного аналізу криптографія використовує інструментарій абстрактної алгебри та теорії ймовірностей. Захист досягається шифруванням, тобто перетворенням, які роблять захищені вхідні дані важкорозкриваними за вхідними даними без знання спеціальної ключової інформації ключа.

Проблема захисту інформації шляхом її перетворення, що виключає її прочитання сторонньою особою, хвилювала людський розум з давніх часів. Історія криптографії ровесниця історії людського мовлення та спілкування. Як приклад, спочатку писемність сама по собі була криптографічного технологією, оскільки в древніх суспільствах нею оволоділи лише певні версти населення. Безліч книг Стародавньої Індії та Стародавнього Єгипту тому приклади.

**Актуальність роботи**. На сучасному етапі розвитку суспільства однією з найбільших цінностей стала інформація. Важливим питанням на сьогоднішній день є захист інформації. Існує декілька напрямків та підходів до захисту інформації, і один із них це криптологія, тобто захист математичними методами. Розвиток електронно-обчислювальної техніки та математичного апарату дає зловмисникам все більше можливостей для дешифрування інформації. З плином часу до алгоритмів шифрування висуваються все більш суворі вимоги, а кожний новий запропонований метод перевіряється все більш детально на велику кількість вразливостей.

*Мета*: дослідити зв'язок шифрування і математичних методів, ознайомитись із застосуванням вказаних методів, порівняти їх, представити власний метод шифрування.

*Завдання*:

1. Вивчити сучасну літературу з теми, в тому числі англомовну.
2. Зробити порівняльну таблицю математичних методів шифрування з посиланнями на джерела опису методів.
3. Розв’язати ряд задач, що ілюструють роботу різних математичних методів.
4. Скласти математичну модель власного криптоалгоритму.

*Методи дослідження:* метод описання, метод аналізу криптосистем, порівняння, оцінювання їх стійкості.

*Об’єктом дослідження* є процес шифрування та дешифрування.

*Предметом дослідження* є використання математичних методів у. криптографії.

**РОЗДІЛ 1.** **МЕТОДИ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ**

**1.1. Математичні основи криптографічного захисту інформації**

Криптографічний захист інформації — вид захисту, що реалізовується за допомогою перетворень інформації з використанням спеціальних (ключових) даних з метою приховування (або відновлення) змісту інформації, підтвердження її справжності, цілісності, авторства тощо.

Завдання криптографії є пошук і дослідженням математичних методів перетворення інформації. Криптографія дозволяє перетворити інформацію таким чином, що її прочитання (відновлення) стає можливим тільки при знанні ключа.

Криптографія — наука про математичні методи забезпечення конфіденційності (неможливості прочитання інформації стороннім) і автентичності (цілісності і справжності) інформації. Відкритий текст (plaintext, clear text) – це інформація, що може бути прочитана, осмислена і зрозуміла без будь-яких спеціальних заходів. Шифрування дозволяє сховати інформацію від тих, для кого вона не призначається, незважаючи на те, що вони можуть бачити сам шифротекст. Протилежний процес перетворення шифротексту в його вихідний вид називається розшифруванням або дешифруванням.

Ключ – конкретний таємний стан деяких параметрів алгоритму криптографічного перетворення даних, що забезпечує вибір одного варіанту із сукупності всіляких для даного алгоритму.

Під шифром розуміється сукупність зворотних перетворень безлічі відкритих даних на безліч зашифрованих даних, заданих алгоритмом криптографічного перетворення [3].

*Криптостійкістю* називається характеристика шифру, визначальна його стійкість до дешифрування. Звичайно ця характеристика визначається періодом часу, необхідним для дешифрування.

Є дві групи криптографічних алгоритмів:

а) загальні:

• симетричні;

• асиметричні;

б) спеціальні.

Криптографічні алгоритми застосовують із метою:

• шифрування інформації;

• захисту даних і повідомлень (інформації) від модифікації або підробки.

На протидію загрозам та з метою мінімізації можливих збитків користувачів спрямовані чотири групи заходів: правові, морально-етичні, адміністративні, фізичні.

Особливу увагу заслуговують адміністративні та фізичні засоби захисту інформації.

Криптографічний захист базується на основі використання математичних методів перетворення інформації. Оскільки написання програмного продукту для захисту інформації виконано на мові програмування Python 3.10 наведемо перелік операцій над бітами (табл. 1.1).

Усі операції для перетворення інформації на основі симетричної криптографії базуються на побітових операціях. В процесі шифрування інформації найбільш часто використовується операція побітового додавання за модулем два. Дана операція в мові програмування Python 3.10 задається за допомогою операції «виключення або» ^. Дану операцію будемо позначати знаком ⊕. Операція виконує наступне перетворення бітів:

0 ⊕0 = 0, 0 ⊕ 1 = 1, 1 ⊕ 1 = 1, 1 ⊕ 0 = 1.

Дана операція використовується у шифрі Вернама (одноразового блокноту)[10].

Таблиця 1.1. Основні операції роботи з бітами

|  |  |
| --- | --- |
| Операція | Опис |
| << | Побітовий зсув в вліво |
| >> | Побітовий зсув в право |
| | | Операція диз’юнкції |
| ^ | Операція еквіваленції |
| & | Операція кон’юнкції |

Опишемо детально цей алгоритм роботи шифру одноразового блокноту. Перед шифруванням повідомлення М записується у двійковій формі. Ключем K служить довільне двійкове слово однакової з М довжини. Криптотекст С отримують побітовим додаванням повідомлення і ключа, тобто С = М ⊕ K.

Дешифрування у шифрі одноразового блокноту збігається із шифруванням — щоб отримати вихідне повідомлення М, слід додати до криптотексту С той же ключ K. Це легко обґрунтувати: оскільки С = М ⊕ К, тоді справедливе наступне співвідношення

С⊕К = (M⊕ K) ⊕К= M⊕ (K⊕ K) = M⊕ 0= М.

Алгоритм, який полягає у шифруванні повідомлення за допомогою одного шифру, а потім застосування до отриманого крипто тексту, ще одного шифру називається композицією або добутком цих двох шифрів [5].

Частковим випадком композиції двох шифрів є послідовне застосування двічі одного і того ж шифру. Інтуїтивним виглядає припущення, що подвійне шифрування збільшує надійність.

Перспективнішим видається подвійне шифрування з незалежним вибором ключа кожного разу. Тобто, за допомогою алгоритму шифрування *Е* повідомлення *М* перетворюється у криптотекст *C= EK2( EK1 (M )),* де два ключі *K1* та *K2* вибираються незалежно один від одного. Дешифрування не складає труднощів: *М= DK (DK (C* )) (належить звернути увагу на зміну порядку, в якому застосовуються ключі).

Припустимо, що ключем служить двійкове слово довжиною n. В цьому разі всі можливі ключі є *2n*, а всі можливі пари ключів є *22n*. Однак це не означає, що ламана двократного шифру займатиме *22n* кроків замість *2n*. Є спосіб оптимізації перебірної процедури, так звана зустрічна атака, що займає кілька кроків пропорційну до *2n*, але взамін вимагає машинної пам’яті такого ж порядку. Це атака з відомим відкритим текстом, але для проведення якої потрібно знати якусь пару з повідомлення *М* та відповідного йому криптотексту С.

**Алгоритм Евкліда.** Для будь-якого цілого *a* і натурального *b* однозначно визначені цілі числа q і r такі, що *a=bq*+*r* і 0≤*r*≤*b*. Число q називають часткою, а r — остачею від ділення a на b [6]. Наприклад, рівність − 20=(−1) 67+47 означає, що −20 при ділені на 67 дає остачу −1 і остачу 47. Для остачі будемо вживати таке позначення: *r = аmodb.* Число r будемо також називати (зведеним) зведеним лишком числа a за модулем b. Якщо *r* = 0 , то кажуть, що *a* ділиться на *b* (націло або без остачі) і пишеться *a* | *b* . Кажуть також, що *b* ділить *a*, називають *b* дільником числа *a,* а *a* — кратним числа *b* . Запис *a b* означатиме, що *a* не ділиться на *b* . Для цілих *a* і *b* через НСД ( *a*,*b* ) позначатимемо їх найбільший спільний дільник (НСД) — найбільше з поміж таких *c* , що одночасно *c*|*a* і *c*|*b* (хоча б одне із *a* і *b* вважається відмінним від нуля). Числа *a* і *b* називаються взаємно простими, якщо НСД(*a*,*b*) = 1 .

Алгоритм Евкліда для знаходження НСД двох натуральних чисел a і b ґрунтується на співвідношеннях

НСД (*a*,*b*) =НСД(*a*,*a* mod *b*) для *a* ≥ *b* (1)

НСД (*a*,0) = *a* (2)

Спершу продемонструємо ідею цього алгоритму на прикладі.

Щоб знайти НСД(211,79), застосуємо послідовність (1) аж доки не опинимося у ситуації, коли можна буде використати (2). Таким чином робота алгоритму зводиться до кілька разового ділення з остачею.

211 = 79· 2 + 53

79 = 53·1+ 26

53 = 26· 2 +1

26=1·26 + 0

Маємо НСД(211,79) = НСД(79,53) = НСД(53,26) = НСД(26,1) = НСД(1,0)=1. Опишемо, що відбувається в загальному випадку.

Алгоритм Евкліда обчислення НСД(a,b), a ≥ b

•Покласти r0= a, r1 = b , i = 1.

•Виконати *i*-ий крок, що полягає в ділені з остачею: r i-1 = r i ·q + r i+1

• Якщо *r i+1 >0* то збільшити *i* на 1 і перейти до виконання наступного кроку згідно з попереднім пунктом. Якщо *r i+1 =0* , то завершити роботу із результатом НСД (*a*,*b*) = *r* .

*Коректність.* Алгоритм завершує роботу, як тільки *r i+1 =0* [6]. Рано чи пізно це трапиться, оскільки *0*≤ *r i+1 <* *r* для кожного *i* . Припустимо так сталося на *m* -ому кроці, тобто  *r m +1 =0*. Результатом роботи алгоритму є значення  *r m* . Рівність *r m =* НСД(*a*,*b*) випливає з таких міркувань. Співвідношення НСД (*a*,*b*) = НСД(*r i-1*, *r i* ) для 1< *i* ≤ *m* +1 доводиться індукцією з використанням (1). При *i* = *m*+1 із врахуванням (2) отримуємо НСД(*a*,*b*) = НСД (*rm* ,0) = *rm* .

*Ефективність.*Під ефективністю алгоритму Евкліда розуміють те, що кількість кроків *m*, які виконуються на вході *a*, *b*, не є надто великою [10]. Із співвідношення 0 ≤ r*i*+1 < r*i* випливає, що *m* ≤ *b*. Однак ця оцінка є незадовільною. Скажімо, для чисел *a* і *b* з якоюсь сотнею цифр у десятковому записі кожного, нам гарантовано лише, що на знаходження їх НСД буде витрачено не більше 10100 кроків. Але ж найбільш швидкодіючій ЕОМ для виконання такої кількості кроків потрібен фантастично великий обсяг часу.

На щастя, оцінку на *m* можна суттєво поліпшити. Покажемо, *m* ≤ 2log2 *b*+1. Для цього використаємо нерівність *ri*+1 < *ri*-1 / 2. Вона доводиться розглядом двох випадків: якщо *ri* < *ri*-1 / 2, то виконується нерівність *ri*+1 < *ri*; якщо *ri* ≥ *ri*-1 / 2, то виконується рівність *ri*+1 = *ri*-1 mod*ri*. Таким чином, кожні два кроки зменшуються *ri*+1 принаймні в двічі, і не пізніше 2log2 *b* + 1 кроків ми прийдемо до *ri*+1 = 0.

*Алгоритм Евкліда дає такий наслідок.*

*Твердження*. Для кожної пари взаємно простих чисел *a* і *b* можна знайти такі цілі *u* і *v* , що *ua* + *vb* = 1 [10].

Зазначимо, що алгоритм Евкліда дає ефективний спосіб знаходження коефіцієнтів *u* і *v* для заданої пари *a*, *b*.

Нехай *a* = 211, *b* = 79. Протокол роботи алгоритму Евкліда виписаний вище. Рухаючись знизу вгору, отримаємо

1 = 1 ⋅ 53 + (–2) ⋅ (79 – 1 ⋅ 53) = (–2) ⋅ 79 + 3 ⋅ 53 = (–2) ⋅ 79 +

+ 3 ⋅ (211 – 2 ⋅ 79) = 3 ⋅ 211 + (–8) ⋅ 79.

Отже, *u* = 3 і *v* = –8.

*Конгруенція.*

Ми домовилися позначати остачу від ділення цілого *a* на натуральне *b* через *a mod b .* Позначення mod ми будемо вживати і в іншому значенні. А саме, ми будемо писати *x* ≡ *y* (mod *n*), якщо цілі *x* і *y* при діленні на натуральне *n* дають однакову остачу, (тобто *x* mod *n* = *y* mod *n*). Такі *x* і *y* називаються конгруентними або рівними за модулем *n*. Відношення між *x* і *y* називається конгруенцією або порівнянням [5].

*Твердження.*

Наступні три умови еквівалентні.

1) *x* ≡ *y* (mod *n*).

2) *x* = *y* + *kn* для деякого цілого *k*.

3) *n* | (*x* – *y*).

Для натурального *n* через *Zn* позначаємо множину {0, 1, …, n – 1}, наділену операціями додавання та множення за модулем *n*. Сумою *x* і *y* із *Zn* є

(*x + y)*mod *n*, а їх добуток є (*x ⋅ y)*mod *n*. Відносні цих операцій *Zn* є комутативним кільцем з одиницею, яке називається кільцем зведених лишків за модулем *n*. Через *Z\*n* позначатимемо мультиплікативну групу елементів, для яких в *Zn* є обернений відносно множення.

*Твердження.* складається з елементів *x*, взаємно простих з *n*, і лише з них. Елемент, обернений до *x* ∈ відносно множення, будемо позначати через x-1 mod *n* просто x-1. Ділення на *x* в *Zn* означатиме множення на x-1. Елемент, обернений до *x* ∈ відносно додавання будемо позначати через –*x*. Зокрема, –1 = *n* – 1 в *Zn*. Як звичайно, в *Zn* можна ввести операцію віднімання: x – y = x + (– y).

Нехай ми хочемо знайти елемент, обернений до *79* в . Вище було отримано рівність *1=3⋅211+(–8)⋅79*. З неї негайно випливає *(–8)⋅79≡1(mod221)*. Отже, *79-1 mod211=(–8)mod211=203.*

Значення *ш(n)* дорівнює кількості натуральних чисел, що не перевищують *n* і взаємно прості з *n*. Функцію *ш(n)* називається функцією Ейлера.

*Теорема Ейлера***.** Для взаємно простих цілих *x* і натурального *n* справедлива конгруенція *xш(n)≡1(modn).* Звідси отриманий такий наслідок теореми Ейлера.

*Мала теорема Ферма***.** Якщо ціле *x* не ділиться на просте *р*,

то *xp-1=1(modn).*

*Китайська теорема про остачі*. Для будь-якої пари взаємно простих натуральних чисел *n*1 і *n*2 та для будь-якої пари цілих чисел *x*1 і *x*2, можна знайти таке ціле *x*, де *x ≡ x1 (mod n1) і x ≡ x2 (mod n2).*

*Твердження*.

Нехай *n* = *n*1*n*2, де *n*1 і *n*2 взаємно прості. Тоді відображення *f* : *Zn* → *Zn*1 ⊕ *Zn*2 Задане співвідношенням *f*(*x)=(xmodn1, x modn2),* є ізоморфізмом кілець.

Таким чином ми навели приклад обчислення НСД двох натуральних чисел з використанням алгоритму Евкліда. Наведемо приклад функції реалізацію даного методу на мові С#:

*double NSD(int r0,int r1)*

*{*

*int riplus1=r0;*

*int qi=r1;*

*while(riplus1>0)*

*{*

*riplus1=r0%r1;*

*qi=r0/r1;*

*r0=r1;*

*r1=riplus1;*

*}*

*return r0;*

*}*

**Генерація простих чисел.** Для алгоритмів з відкритим ключем потрібні прості числа [4]. Їх потрібна множина для довільної достатньо великої мережі. Генерування випадкових чисел із спробою їх розкладу на множники — це не правильний спосіб пошуку простих чисел. Існують різноманітні імовірності перевірки простих чисел, із завданням ступеню достовірності. При умові, що ця «достовірність» досить велика, такі спроби перевірки достатньо великі. Роберт Соловей і Фолькер Штрасен розробили алгоритм імовірності перевірки простоти числа [6]. Для перевірки простоти числа *р* цей алгоритм використовує символ Якобі:

1. Вибір випадкового числа *a*, менше *p*.

2. Якщо НСД(*a*, *p*), то *p* не проходить перевірку і являється основним.

3. Обчислюємо *j*=*a* (*p*–1)/2 mod*p*.

4. Обчислюємо символ Якобі *J*(*a*,*p*).

5. Якщо *j*≠*J*(*a*,*p*), число *p* більш усього не є простим.

6. Якщо *j*=*J*(*a*,*p*), то імовірно, що число *p* не є простим, не більше 50 відсотків.

Число *a*, яке не показує, що *p* імовірно не є простим числом, називається свідком. Якщо *p* — складне число, імовірність випадкового числа *a* бути свідком не нижче 50 відсотків. Повторивши цю перевірку *t* раз з *t* різних значень *a*. Імовірність того, що залишок числа перевищить всі *t* перевірки не перевищує (1 / 2)*t*.

Інший більш простий спосіб був розроблений Лєманом [5]. Послідовність дій при перевірці простоти числа *p*:

1. Вибрати випадкове число *a*, менше *p*.

2. Обчислити *a*(*p* – 1) / 2 mod*p*.

3. Якщо *a*(*p* – 1) / 2≠1 або –1(mod*p*), то *p* не є простим.

4. Якщо *a*(*p* – 1) / 2=1, то імовірність того, що число *p* не являється простим, не більше 50 відсотків.

**Дискретні логарифми в скінченному полі.** В якості однонаправної функції в криптографії часто використовують піднесення в степінь по модулю [6]. Легко обчислити: *ax* mod*n.*

Задачею, зворотнього піднесення в степінь по модулю, є пошук дискретного логарифму, а це вже задача:

Нехай *x*, для якого *ax ≡ b (mod n).*

Наприклад: Якщо , то *x* = 6

Рішення існує не для всіх дискретних логарифмів. Легко помітити, що наступне рівняння не має рішення *3x≡7 mod13*.

Ще складніше цю задачу вирішити для 1024-бітових чисел.

**1.2. Симетричні криптографічні системи**

Залежно від наявності або відсутності ключа кодовані алгоритми діляться на тайнопис і криптографію. Залежно від відповідності ключів шифрування і дешифрування на симетричні і асиметричні. Залежно від типу використовуваних перетворень на підстановочні і перестановочні. Залежно від розміру шифрованого блоку на потокові та блокові шифри.

Відносно криптоалгоритмів існує кілька схем класифікації, кожна з яких заснована на групі характерних ознак. Таким чином, один і той же алгоритм "проходить" відразу за кількома схемами, опиняючись в кожній з них в будь-якії з підгруп [7].

Основною схемою класифікації всіх криптоалгоритмів є наступна:

• *Тайнопис.* Відправник і одержувач призводять над повідомленням перетворення, відомі тільки їм двом. Стороннім особам невідомий сам алгоритм шифрування.

• *Криптографія з ключем*. Алгоритм впливу на дані, що передаються, котрий відомий всім стороннім особам, але він залежить від деякого параметра "ключа", яким володіють тільки відправник і одержувач.

• *Симетричні криптоалгоритми*. Для кодування і розшифровки повідомлення використовується один і той же блок інформації (ключ).

• *Асиметричні криптоалгоритми*. Алгоритм такий, що для шифрування повідомлення використовується один ("відкритий") ключ, відомий всім бажаючим, а для розшифровки інший ("закритий"), що існує тільки в одержувача.

Надалі докладніше розглянемо криптографії з ключем, так як більшість фахівців саме по відношенню до цих криптоалгоритм використовують термін криптографія, що цілком виправдано.

Залежно від характеру впливів, вироблених над даними, алгоритми поділяються на:

**•** *Перестановки.* Блоки інформації (байти, біти, більші одиниці) не змінюються самі по собі, але змінюється їх порядок проходження, що робить інформацію недоступною сторонньому спостерігачеві.

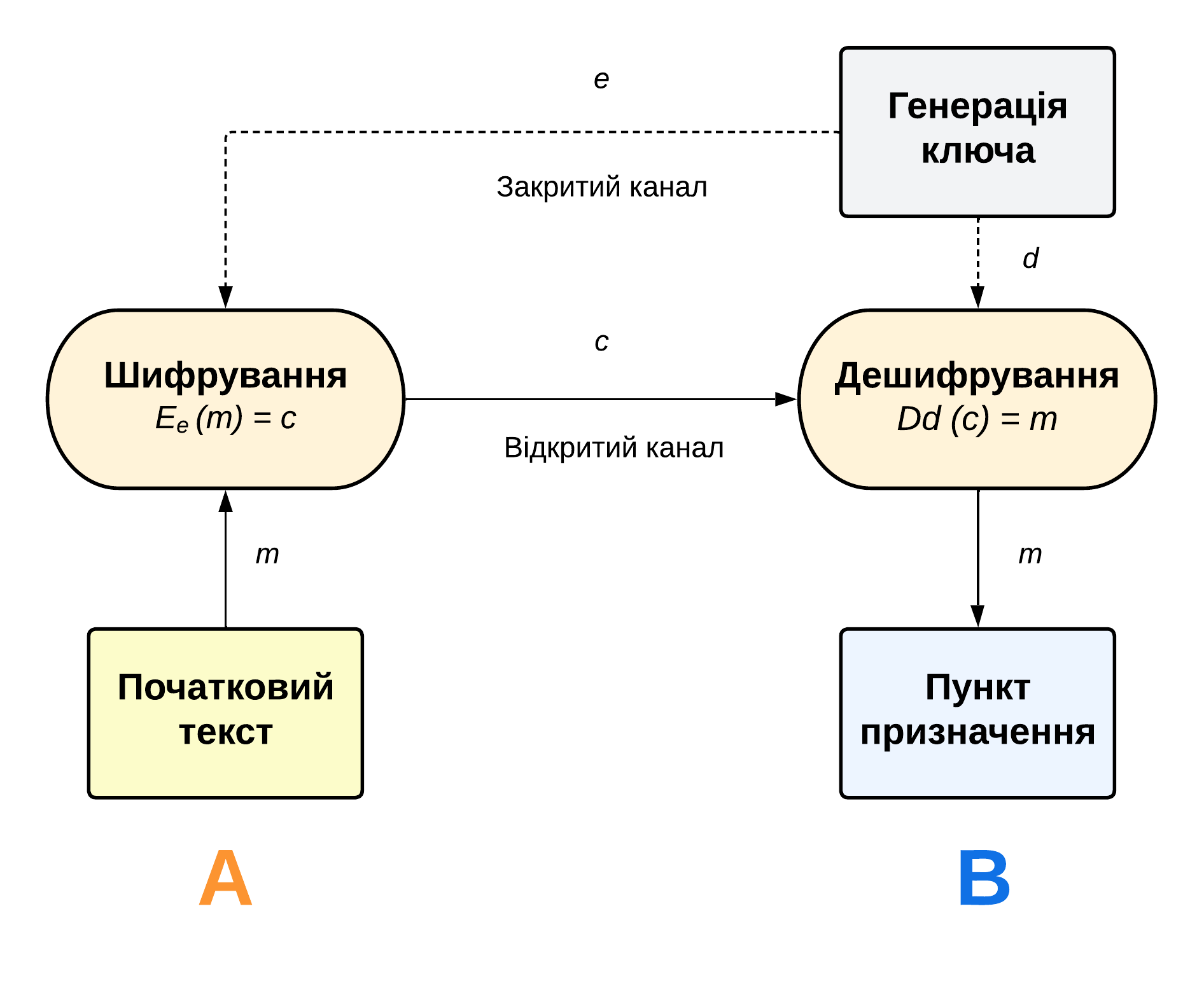
• *Символи.* Самі блоки інформації змінюються за законами криптоалгоритмів. Переважна більшість сучасних алгоритмів належить цій групі.

Залежно від розміру блоку інформації криптоалгоритми діляться на:

• *Потокові шифри*. Одиницею кодування є один біт. Результат кодування не залежить від минулого раніше вхідного потоку. Схема застосовується в системах передачі потоків інформації, тобто в тих випадках, коли передача інформації починається і закінчується в довільні моменти часу і може випадково перериватися. Найбільш поширеними представниками поточних шифрів є скремблери.

• *Блокові шифри*. Одиницею кодування є блок з декількох байтів (на сьогодні це 4-32). Результат кодування залежить від усіх вихідних байтів цього блоку. Схема застосовується при пакетної передачі інформації і кодування файлів.

*Симетричні шифрування* (також симетричні криптосистеми) спосіб шифрування, в якому застосовується один і той же криптографічний ключ для шифрування і дешифрування. До винаходу асиметричного шифрування єдиним існуючим способом було симетричне шифрування. Обидві сторони мають зберігати ключ алгоритму в секреті. Ключ алгоритму обирається сторонами ще до початку обміну повідомленнями.



*Рисунок 1.1. – Загальна схема симетричної системи шифрування*

На сьогоднішній день симетричні шифри – це поточні та блочні шифри.

**Блочний шифр** – один із двох видів симетричного шифру. Особливістю блочного шифру є обробка блоку з декількох байт за одну ітерацію (зазвичай 8 або 16). Блочні криптосистеми розбивають текст повідомлення на окремі блоки і потім здійснюють перетворення цих блоків з використанням ключа [8].

Перетворення має використовувати такі принципи:

• Розсіювання (diffusion) тобто зміна будь-якого знака відкритого тексту або ключа впливає на велику кількість знаків шифротекста, котрі приховують статистичні властивості відкритого тексту;

• Перемішування (confusion) використання перетворень, котрі ускладнюють отримання статистичних залежностей між шифротекстом і відкритим текстом.

До переваг блочних шифрів відносять схожість процедур шифрування і дешифрування, які, як правило, відрізняються лише порядком дій. Це спрощує створення пристроїв шифрування, так як дозволяє використовувати одні і ті ж блоки в ланцюгах шифрування і дешифрування.

Блоковий шифр складається з двох взаємопов'язаних алгоритмів: алгоритм шифрування *E* і алгоритм дешифрування *E-1*. Вхідними даними служать блок розміром *n* біт і *k*-бітний ключ. На виході виходить n-бітний зашифрований блок. Для будь-якого фіксованого ключа функція дешифрування є зворотною до функції шифрування *E -1 (E (M)) = M* для будь-якого блоку *M* і ключа *K*.

Для будь-якого ключа *K, EK* перестановка набору вхідних блоків. Ключ вибирається з *2n!* можливих перестановок.

Розмір блоку *n* це фіксований параметр блокового шифру, зазвичай рівний 64 або 128 бітів, хоча деякі шифри допускають кілька різних значень. Довжина 64 біта була прийнятна до середини 90-х років, потім використовувалася довжина 128 біт і більше. Різні схеми шифрування дозволяють зашифровувати відкритий текст довільної довжини. Кожна має певні характеристики: ймовірність помилки, простота доступу, вразливість до атак. Типовими розміру ключа є 40, 56, 64, 80, 128, 192 і 256 біт. У 2006 році 80-бітний ключ здатний був запобігти атаці грубою силою.

Найпростішим режимом роботи блочного шифру є ECB (Electronic CodeBook (англ.) режим простої заміни або електронної кодової книги), де всі блоки відкритого тексту зашифровуються незалежно один від одного. Однак, при використанні цього режиму статистичні властивості відкритих даних частково зберігаються, так як кожному однаковому блоку даних однозначно відповідає зашифрований блок даних. При великій кількості даних (наприклад, відео або звук) це може призвести до витоку інформації про їх зміст і дати більший простір для криптоаналізу. Видалення статистичних залежностей у відкритому тексті можливо за допомогою попереднього архівування, але воно не вирішує завдання повністю, так як в файлі залишається службова інформація архіватора, що не завжди допустимо [8].

**Поточний шифр** це симетричний шифр, в якому кожен символ відкритого тексту перетворюється в символ шифрованого тексту в залежності не тільки від використовуваного ключа, а й від його розташування в потоці відкритого тексту. Поточний шифр реалізує інший підхід до симетричного шифрування, ніж блокові шифри. При блоковому шифруванні відкритий текст розбивається на блоки однакової довжини, при цьому збігаються блоки, при тому що ключі завжди шифрується однаково, при потоковому шифруванні це не так.

У 1949 році Клод Шеннон опублікував роботу, в якій довів абсолютну стійкість шифру Вернама (також відомий, як одноразовий блокнот [one-time pad]). У шифрі Вернама ключ має довжину, рівну довжині самого переданого повідомлення. Ключ використовується в якості гами, і якщо кожен біт ключа вибирається випадково, то криптоаналітику для розтину залишається лише метод грубої сили. Але ключі, які можна порівняти за довжиною з переданими повідомленнями, важко використовувати на практиці. Тому зазвичай застосовують ключ меншої довжини (наприклад, 128 біт). За допомогою нього генерується псевдовипадкова гаммована послідовність. Природно, псевдовипадкова гама може бути використана при атаці на поточний шифр [8].

Типи потокових шифрів:

• При використанні *синхронізованих потокових шифрів* необхідна примусова синхронізація приймального і передавального шифратора (установка ідентичних внутрішніх станів і фази). При втраті бітів або вставці нових бітів відбувається втрата синхронізації і коректне дешифрування стає неможливим. Для синхронізації можуть використовуватися спеціальні марковані послідовності, що вставляються в шифротекст.

• *Самосинхронізовані потокові шифри* використовують спеціально розроблені алгоритми вироблення шифротекста, що дозволяють приймальному шифратору отримавши N неспотворених бітів шифротекста автоматично синхронізуватися з передавальному шифратором. Як наслідок, такі потокові шифри розмножують помилки в каналі зв'язку.

Існує безліч (не менше двох десятків) алгоритмів симетричних шифрів, істотними параметрами яких є:

• стійкість;

• довжина ключа;

• число раундів;

• довжина оброблюваного блоку;

• складність апаратної / програмної реалізації.

Поширені алгоритми:

• DES (Data Encryption Standard, стандарт шифрування даних);

• 3DES (Triple-DES, потрійний DES);

• AES (Advanced Encryption Standard, поліпшений стандарт шифрування);

• RC2 (Шифр Ривеста (Rivest Cipher));

• Blowfish;

• Twofish;

• NUSH;

• IDEA (International Data Encryption Algorithm, міжнародний алгоритм шифрування даних);

• CAST (за ініціалами розробників Carlisle Adams і Stafford Tavares).

**1.3. Асиметричні криптографічні системи**

Асиметричні криптографічні системи були розроблені в 1970-х рр. Принципова відмінність асиметричної криптосистеми від криптосистеми симетричного шифрування полягає в тому, що для шифрування інформації і її подальшого дешифрування використовуються різні ключі:

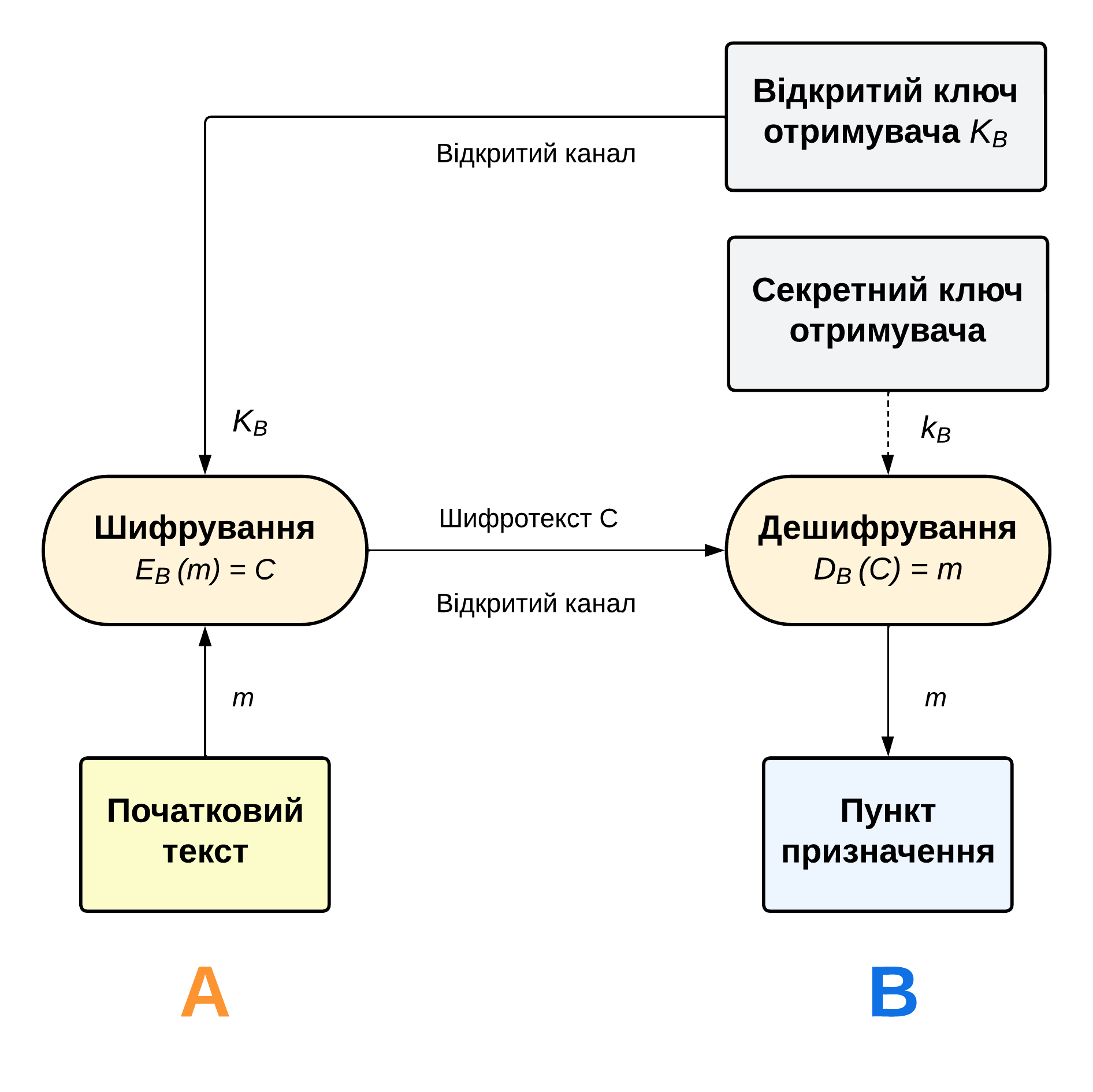
• відкритий ключ *К* використовується для шифрування інформації, обчислюється з секретного ключа до ;

• секретний ключ *k* використовується для дешифрування інформації, зашифрованої за допомогою парного йому відкритого ключа *К.*

Ці ключі розрізняються таким чином, що за допомогою обчислень не можна вивести секретний ключ до з відкритого ключа *К.* Тому відкритий ключ *k* може вільно передаватися по каналах зв'язку.

Асиметричні системи називають також двох ключовими криптографічними системами, або криптосистемами з відкритим ключем.

Узагальнена схема асиметричної криптосистеми шифрування з відкритим ключем показана на рисунку 2.1.



*Рисунок 1.2. – Загальна схема асиметричної системи шифрування*

Для криптографічного закриття і подальшого дешифрування інформації, що передається використовуються відкритий і секретний ключі одержувача В повідомлення.

Як ключ шифрування повинен використовуватися відкритий ключ одержувача, а в якості ключа дешифрування його секретний ключ.

Секретний і відкритий ключі генеруються попарно. Секретний ключ повинен залишатися у його власника і бути надійно захищений від несанкціонованого доступу (аналогічно ключу шифрування в симетричних алгоритмах). Копія відкритого ключа повинна знаходитися у кожного абонента криптографічної мережі, з яким обмінюється інформацією власник секретного ключа [9].

Процес передачі зашифрованої інформації в асиметричній криптосистемі здійснюється наступним чином.

Підготовчий етап:

• абонент *В* генерує пару ключів: секретний ключ *кв* і відкритий ключ *Кв*;

• відкритий ключ *Кв* надсилається абоненту *А* і іншим абонентам (або робиться доступним, наприклад на розділеному ресурсі).

Використання обмін інформацією між абонентами *А і В:*

• абонент *А* зашифровує повідомлення за допомогою відкритого ключа *Кв* абонента *В* і відправляє шифротекст абоненту *В*;

• абонент *В* розшифровує повідомлення за допомогою свого секретного ключа *Кв*. Ніхто інший (в тому числі абонент *А*) не може розшифрувати дане повідомлення, так як не має секретного ключа абонента *В*. Захист інформації в асиметричній криптосистемі заснований на секретності ключа *кв* одержувача повідомлення.

*Характерні особливості асиметричних криптосистем*:

• відкритий ключ *Кв* і криптограма *C* можуть бути відправлені по незахищених каналах, тобто противнику відомі *Кв і С*;

• алгоритми шифрування і дешифрування: *Ев* : М -> С; *йв* : С -> М , є відкритими.

У. Діффі і М. Хеллман сформулювали вимоги, виконання яких забезпечує безпеку асиметричної криптосистеми.

1. Обчислення пари ключів (*Кв , kв*) отримувачем В повинно бути простим.

2. Відправник A, знаючи відкритий ключ *Кв* і повідомлення М, може легко обчислити криптограму *С = Єкв (М).*

3. Одержувач В, використовуючи секретний ключ *кв* і криптограму С, може легко відновити вихідне повідомлення *М = Пкв (С).*

4. Противник, знаючи відкритий ключ *Кв* , при спробі обчислити секретний ключ *кв* наштовхується на непереборну обчислювальну проблему.

5. Противник, знаючи пару (*Кв,С),* при спробі обчислити вихідне повідомлення М наштовхується на непереборну обчислювальну проблему. Концепція асиметричних криптографічних систем з відкритим ключем заснована на застосуванні односпрямованих функцій. Односпрямованою функцією називається функція Р(Х), що володіє двома властивостями:

• існує алгоритм обчислення значень функції у = Р(х,у);

• не існує ефективного алгоритму звернення (інвертування) функції *У7* (тобто не існує рішення рівняння Р(Х)=У відносно X).

Як приклад односпрямованої функції можна вказати цілочисленне множення. Пряма задача обчислення добутку двох дуже великих цілих чисел *Р* і 0, тобто знаходження значення N=Р•3 достатньо нескладне завдання для комп'ютера.

Інший характерний приклад односпрямованої функції це модульна експонента з фіксованими підстановкою і модулем[9].

Як і в випадку симетричних криптографічних систем, за допомогою асиметричних криптосистем забезпечується не тільки конфіденційність, але також справжність і цілісність переданої інформації. Справжність і цілісність будь-якого повідомлення забезпечується формуванням цифрового підпису цього повідомлення і відправкою в зашифрованому вигляді повідомлення разом з цифровим підписом. Перевірка відповідності підпису отриманого повідомлення після його попереднього дешифрування являє собою перевірку цілісності та автентичності прийнятого повідомлення.

**1.4. Хеш-функції**

На сьогоднішній день майже жоден сервіс криптографії не може обійтись без використання хеш-функцій. Це функції, які призначені для «стиснення» будь якого набору даних або повідомлення, котрі записані, як правило, в двійковому алфавіті, в деяку бітову комбінацію фіксованої довжини, котру називають згорткою. Хеш-функції мають велику кількість застосування, при тестуванні логічних пристроїв, перевірці цілісності записів в базах даних, при проведенні статистичних експериментів та при побудові алгоритмів швидкого пошуку. Рівномірність розподілу їх значень при випадковому виборі значень аргументу є основною вимогою до хеш-функцій.

Криптографічною хеш-функцією можна назвати будь-яку хеш-функцію, яка є крипостійкою, тобто задовольняє специфічні для криптографічних додатків вимоги. Хеш-функції у криптографії застосовуються для вирішення наступних завдань:

• автентифікація джерела даних;

• побудови систем контролю цілісності даних при їх передачі або зберіганні.

Хеш-функцією називають будь-яку функцію *h:* *X -> Y*, легко обчислювану та таку, котра для будь-якого повідомлення *M* значення *h(M)=H* (згортка) має фіксовану бітову довжину. *X* безліч всіх повідомлень, Y безліч двійкових векторів фіксованої довжини.

Зазвичай, хеш-функції будують на основі так званих однокрокових стискаючих функцій *y = f(x1, x2)* двох змінних, де *x1, x2* і *y* направляючі вектори довжини *m*, *n* і *n* відповідно, причому *n* довжина згортки, а *m* довжина блоку повідомлення. Для отримання значення *h*(*M*) повідомлення спочатку розбивається на блоки довжиною *m* (але, якщо довжина повідомлення не кратна *m* то останній блок певним чином доповнюється до повного), а потім до отриманих блоків *M1*, *M2, .., MN* застосовують наступну послідовну процедуру обчислення згортки:

*H0=v1*

*Hi=f(Mi*,*Hi-1), i=1,..,N,*

*h(M)=HN*

Тут *V* певна константа, часто називається ініціалізуючим вектором. Її обирають з різних міркувань і вона може являти собою секретну константу чи набір випадкових даних (наприклад, вибірку дати і часу). При такому підході властивості хеш-функції будуть повністю визначатися властивостями однокрокового стискання функції [4].

Виділяють два основних види криптографічних хеш-функцій – безключові і ключові. Ключові хеш-функції ще називають кодами автентифікації повідомлень. Вони надають можливість без додаткових засобів гарантувати як цілісність даних в системах,де користувачі довіряють один одному так і достовірність джерела даних.

Безключовими хеш-функціями називають коди виявлення помилок. Вони дають можливість за допомогою додаткових засобів (наприклад, шифрування) гарантувати цілісність даних. Ці хеш-функції можуть застосовуватися в системах, в яких користувачі довіряють один-одному, так і не довіряють.

До хеш-функцій основною вимогою є рівномірність розподілу їх значень при випадковому виборі значень аргументу. Для криптографічних хеш-функцій також важливо, щоб значення функції сильно змінювалося при щонайменшій зміні аргументу. Це можна назвати лавинним ефектом.

До основних функцій хешування зазвичай ставлять такі вимоги:

• відсутність можливості модифікації;

• відсутність можливості фабрикації.

Перша вимога означає високу складність підбору для заданого повідомлення з відомим значенням згортки іншого повідомлення з правильним значенням згортки.

Друге високу складність підбору повідомлення з правильним значенням згортки.

До безключової функцій ставлять наступні вимоги:

• односпрямованість;

• стійкість до знаходження другого прообразу;

• стійкість до колізій.

Під односпрямованістю мається на увазі висока складність знаходження повідомлення за заданим значенням згортки. Слід зауважити, що донині немає використовуваних хеш-функцій, де була би доведена односпрямовність.

Під стійкістю до знаходження другого прообразу розуміють складність знаходження другого повідомлення з аналогічним значенням згортки для заданого повідомлення з відомим значенням згортки.

Під стійкістю до колізій розуміють складність знаходження пари повідомлень з аналогічними значеннями згортки. Частіше за все у криптоаналітиків служить першим сигналом старіння алгоритму і необхідності його швидкої заміни саме знаходження способу побудови колізій.

*Популярні хеш-алгоритми*:

• CRC16/32 контрольна сума (не є криптографічним перетворенням).

• MD2/4/5/6 -творіння одного з авторів алгоритму RSA, Рона Райвеста.

• MD5 колись був дуже популярним, але перші передумови злому з'явилися ще в кінці 1990-х, тож зараз його популярність швидко згасає.

• MD6 досить цікавий алгоритм з конструктивної точки зору. Був висунутий на конкурс SHA-3, але автори не встигли довести його до робочого прототипу, та цей алгоритм відсутній в списку кандидатів, які пройшли у другий раунд.

• SHA широко поширені зараз алгоритми. Відбувається активний перехід від SHA-1 до стандартів версії SHA-2. SHA-2 загальна назва групи алгоритмів SHA224, SHA256, SHA384 і SHA512. SHA224 та SHA384 – в цілому є аналогами для SHA256 та SHA512 відповідно, але лише після самого розрахунку згортки частина даних в ній відкидається. Використовувати їх є сенс лише для забезпечення сумісності з застарілим обладнанням.

**1.5. Порівняльний аналіз симетричних та асиметричних алгоритмів**

З метою визначення найдоцільнішого криптографічного алгоритму при проектуванні системи захисту інформації, зроблено порівняльний аналіз алгоритмів шифрування з огляду на наступні критерії:

* високий рівень захисту даних проти дешифрування та можливої модифікації;
* захищеність інформації повинна базуватися тільки на знанні ключа та не залежати від того, відомий алгоритм чи ні;
* невеликі зміни вихідного тексту або ключа має приводити до значних змін шифрованого тексту;
* область значень ключа має виключати можливість дешифрування даних шляхом перебору значень ключа;
* економічність реалізації алгоритму при достатній швидкодії;
* вартість дешифрування даних без відомого ключа має перевищувати вартість самих даних.

Відомо, що шифрування й дешифрування даних відбувається за допомогою симетричних і асиметричних криптосистем, причому до появи останніх єдиними існуючими були симетричні криптосистеми.

Більшість сучасних стійких симетричних алгоритмів використовують ключ довжини 64-256 біт ( 8-32 байта). У таблиці 1.1 наведено основні використовувані в цей час алгоритми, довжини блоку й довжини ключа.

Таблиця 1.2

|  |  |  |
| --- | --- | --- |
| Алгоритм | Довжина ключа (у бітах) | Довжина блоку (у бітах) |
| DES | 64 | 64 |
| Blowfish | Змінна, до 448 біт | 64 |
| IDEA | 128 | 64 |
| RC5 | Змінна | Змінна |
| ДЕРЖСТАНДАРТ ГОСТ 28147-89 | 256 | 64 |

Порівняння з асиметричними криптосистемами:

*Переваги:*

• швидкість (за даними Applied Cryptography на 4 порядки вище);

• простота реалізації (за рахунок більш простих операцій);

• менша необхідна довжина ключа для порівнянної стійкості;

• вивченість (за рахунок більшого віку).

*Недоліки:*

• складність управління ключами у великій мережі. Значить квадратичне зростання числа пар ключів, які треба генерувати, передавати, зберігати і 30 знищувати в мережі. Для мережі в 10 абонентів потрібно 45 ключів, для 100 вже 4950, для 1000 499500 і т. д.

• складність обміну ключами. Для застосування необхідно вирішити проблему надійної передачі ключів кожному абоненту, так як потрібен секретний канал для передачі кожного ключа обом сторонам.

Для компенсації недоліків симетричного шифрування на сьогоднішній день широко застосовується комбінована (гібридна) криптографічний схема, де за допомогою асиметричного шифрування передається сеансів ключ, який використовується сторонами для обміну даними за допомогою симетричного шифрування [8].

Важливою властивістю симетричних шифрів є неможливість їх використання для підтвердження авторства, так як ключ відомий кожній стороні.

*Переваги асиметричних* криптографічних систем перед симетричними криптосистемами:

• в асиметричних криптосистемах вирішена складна проблема розподілу ключів між користувачами, так як кожен користувач може згенерувати свою пару ключів сам, а відкриті ключі користувачів можуть вільно публікуватися і поширюватися по мережевим комунікацій;

• зникає квадратична залежність числа ключів від числа користувачів; в асиметричною криптосистемою число використовуваних ключів пов'язано з числом абонентів лінійною залежністю (в системі з N користувачів використовуються 2УУ ключів), а не квадратичні, як в симетричних системах;

• асиметричні криптосистеми дозволяють реалізувати протоколи взаємодії сторін, які не довіряють один одному, оскільки при використанні асиметричних криптосистем закритий ключ повинен бути відомий тільки його власнику.

*Недоліки асиметричних криптосистем*:

• донині немає математичного доказу незворотності використовуваних в асиметричних алгоритмах функцій;

• асиметричне шифрування істотно повільніше симетричного, оскільки при шифруванні і розшифруванні використовуються досить ресурсномісткі операції. З цієї ж причини реалізувати апаратний шифратор з асиметричним алгоритмом істотно складніше, ніж реалізувати апаратно симетричний алгоритм;

• необхідність захисту відкритих ключів від підміни.

Отже, розглянувши основні види криптографічних систем та проаналізувавши їх особливості, можна зробити висновок, що, з огляду на наведені вище критерії, доцільно використовувати асиметричні методи шифрування даних. Хоча несиметрична криптографія є досить повільною у порівнянні зі швидкими і перевіреними часом і практикою симетричних алгоритмів, все ж таки використання несиметричної криптографії радикально спрощує процедуру розподілу ключів між учасниками інформаційних відносин. До того ж, за допомогою відкритого й секретного ключів стає можливим використання електронно-цифрового підпису. Але, в той же час, хоча описані асиметричні алгоритми дозволяють обійти проблему схованої передачі ключа, необхідність аутентифікації залишається. Без додаткових засобів, один з користувачів не може бути впевнений, що він обмінявся ключами саме з тим користувачем, який йому потрібний. Небезпека імітації в цьому випадку залишається.

**РОЗДІЛ 2. ПРАКТИЧНЕ ЗАСТОСУВАННЯ МАТЕМАТИЧНИХ МЕТОДІВ В КРИПТОГРАФІЇ**

**2.1 Розробка математичної моделі криптоалгоритму**

Програмна реалізація методу шифрування VeLKoM

Алгоритм реалізації методу шифрування умовно можна розділити на кілька етапів:

1. Вибір кодування символів для вхідного тексту.
2. Вирішення, чи буде вхідний текст містити всі символи обраного кодування, чи тільки окремий діапазон (наприклад тільки символи українського алфавіту).
3. Вибір формату шифротексту – числовий або будь-які символи кодування (в такому випадку додатково потрібно **обрати кодування для шифротексту**).
4. Якщо обрано будь-які символи кодування – потрібно **визначити діапазон можливих числових значень ключів.**
5. Вибір типу ключа (Простий чи складний)
6. Організація технології обміну ключами та інформації щодо кодувань.
7. Організація процесу кодування даних шифром VeLKoM.

**Вибір кодування для шифротексту.** Кількість символів кодування має бути щонайменше в двічі більшою за числовий діапазон символів вхідного тексту, інакше при розшифруванні можуть виникнути проблеми з колізією – коли двом різним комбінаціям символів відповідає одна комбінація в шифротексті, і неможливо встановити яка саме, через що втрачається інформація. Отже, якщо числовий діапазон символів вхідного тексту не більше половини від загальної кількості обраного кодування, то для шифротексту можна обрати те ж саме кодування. Якщо діапазон більший, то потрібно обрати кодування, що має щонайменше вдвічі більшу кількість елементів. Якщо вхідний текст містить всі елементи кодування, потрібно обрати таке кодування, що містить щонайменше вдвічі більшу кількість елементів.

**Визначення діапазону можливих числових значень ключів.**

При неправильному виборі ключа, можуть виникнути проблеми з обчисленнями та з колізією. Числове значення ключа рекомендується обирати в межах від числа що є серединою діапазону вхідних значень, до чверті від загальної кількості символів у кодуванні шифротексту. Однак можливі випадки коли при такому обмеженні все одно виникають помилки, тому в таких випадках рекомендується обрати більш досконалу модифікацію шифру.

**Ключ.**

В наведеному методі шифрування можливі дві варіації ключів – простий та складний. Простий ключ є одним числом яке застосовується до всіх операцій з ключем. Складний ключ – набір чисел, в якому числа беруться по черзі парами на кожній ітерації шифрування. В цьому випадку кожне число ключа може мати будь-яке натуральне значення. Також можна використовувати в якості ключа набір символів, що переводиться в число за кодуванням. У запропонованому шифрі вхідний текст перетворюють на набор чисел за певним кодуванням символів, про який відправник та отримувач домовляються заздалегідь. Після цього текст проходить через функцію f, після якої отримується шифротекст у вигляді набору чисел. Щоб відрізнити серед написаних разом цифр окремі числа, в ході перетворень через функцію f, числа можна записувати у будь-якій системі числення з фіксованою кількістю цифр, або одразу переводити в символи за тим же самим (або іншим) кодуванням. Тобто, залежно від потреб, шифротекст може бути у вигляді цифр, або у вигляді набору символів певного кодування.

На кожній ітерації береться 2 елементи вхідного тексту, та видається 2 елементи шифротексту. Отже, вхідний текст має містити парну кількість символів. Відповідно якщо він має непарну кількість елементів, перед початком шифрування додаємо в кінець тексту будь-який символ (Пробіл або спеціальний символ про який додатково домовляються, що він не несе інформацію).

**Розглянемо два варіанти реалізації методу шифрування VeLKoM**

**1**. **Спрощений варіант.** За алгоритмом встановлюємо:

1. Кодування: utf-8. Використовувати будемо перші 1112064 символів.
2. Символи вхідного тексту в окремому діапазон – від символу «пробілу» до літер Кирилиці. В utf-8 це діапазон буде від 32 до 1111, де 32 – кодування символу «пробіл», а 1111 – кодування літери «ї».
3. Шифротекст в форматі символів кодування, кодування для шифротексту: utf-8.
4. Діапазон числових значень ключа становить від 572 до 277516.
5. Простий ключ.
6. Обмін ключами та інформацією щодо кодувань відбувається через закриті канали.
7. Процес кодування реалізовано в середовищі PyCharm версії 2022.2.3, мова програмування: Python 3.10.

Розглянемо алгоритм даної варіації методу шифрування VeLKoM.\

Це модифікована версія методу шифрування VeLKoM!

**Алгоритм функції f при шифруванні:**

1. За розташуванням у вхідному тексті беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як a та b відповідно.
2. Обчислюємо перший елемент шифротексту k за формулою: k=(a+b)/2. Якщо k не ціле, округляємо до меншого цілого.
3. Обчислюємо другий елемент шифротексту m за формулою: ((b-a)/2) +key (key додається для уникнення від'ємності m коли a<b). Якщо m не ціле, округляємо до меншого цілого.
4. Перевіряємо a та b на однакову парність, для цього значення m після округлення порівнюємо зі значенням до округлення. Якщо вони співпадають, то a та b на однакову парність, інакше потрібно це позначити. Для цього до m додаємо key\*2, тоді значення m при непарному a+b буде значно більшим ніж при парному, що можна побачити при розшифруванні.
5. Переводимо k та m у символи за кодуванням, та записуємо до шифротексту.
6. Повторюємо операцію f для всіх інших пар символів вхідного тексту.

Цей алгоритм представлено на мові програмування Python 3.10 на рис. А.1.

**Алгоритм функції f' при розшифруванні:**

1) За розташуванням у вхідному тексті беремо два символи та переводимо їх у числа за кодуванням, позначаємо їх як k та m відповідно.

2) Якщо m менше або рівне key\*2 - позначаємо x=0 та переходимо до наступного пункту, інакше від m віднімаємо key\*2, там позначаємо x=1.

3) Обчислюємо перший елемент вихідного тексту a' за формулою: a'=|k-m+key|.

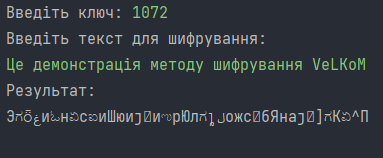
4) Обчислюємо другий елемент вихідного тексту b' за формулою: b'=k+m+x-key.

5) Переводимо a' та b' у символи за кодуванням, та записуємо до вихідного тексту.

Повторюємо операцію f' для всіх інших пар символів шифротексту.

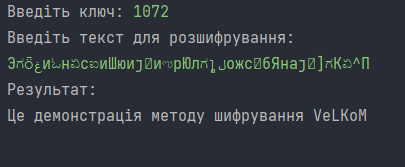
Цей алгоритм представлено на мові програмування Python 3.10 на рис. А.2.

Приклад роботи алгоритму шифрування:



*Рис. 2. 1. Демонстрація роботи програми шифрування алгоритму VeLKoM – спрощений варіант*

Приклад роботи алгоритму дешифрування:



*Рис. 2. 2. Демонстрація роботи програми дешифрування алгоритму VeLKoM – спрощений варіант*

Ця варіація методу шифрування VeLKoM є простою для обчислень комп’ютером та має меншу кількість символів у шифротексті – їх кількість рівна кількості символів вхідного тексту. Це може бути корисним коли передача шифру ускладнена за певних обставин, та вимагає якомога коротших даних.

Однак ця варіація вразлива до статистичного аналізу, оскільки однаковим парам символів вхідного тексту відповідають однакові пари шифротексту.

**2**. **Модифікований варіант**. За алгоритмом встановлюємо:

1. Кодування: utf-8. Використовувати будемо перші 1112064 символів.
2. Вхідний текст містить всі символи кодування.
3. Шифротекст в форматі чисел десяткової системи числення.
4. Будь-які значення ключа
5. Складний ключ у вигляді символів за кодуванням utf-8.
6. Обмін ключами та інформацією щодо кодувань відбувається через закриті канали.
7. Процес кодування реалізовано в середовищі PyCharm версії 2022.2.3, мова програмування: Python 3.10.

Розглянемо алгоритм даної варіації методу шифрування VeLKoM.

Перед шифруванням та дешифруванням визначаємо довжину кожного числа у шифрі. Спочатку аналізуємо введений ключ та серед всіх його елементів обираємо найбільший і записуємо його кодування. Далі до кількості символів у кодуванні помноженої на 3 додаємо найбільше кодування ключа помножене

на 2, оскільки code може використовуватись максимум тричі, а ключ двічі за одну ітерацію. Отримане число є найбільшим можливим числом в шифрі при даному ключі. Підраховуємо кількість цифр в ньому, та записуємо в змінну length.

Наступним кроком перевіряємо чи є довжина ключа меншою за довжину вхідного тексту. Якщо так, дублюємо ключ поки він не стане більшим або рівним вхідному тексту. Задаємо змінну code рівну кількості символів при кодуванні.

**Алгоритм функції f при шифруванні:**

1. За розташуванням у ключі беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як key1 та key2 відповідно, та додаємо до обох code.
2. За розташуванням у вхідному тексті беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як a та b відповідно.
3. Обчислюємо перший елемент шифротексту k за формулою: k=(a+b)/2. Якщо k не ціле, округляємо до меншого цілого.
4. Обчислюємо другий елемент шифротексту m за формулою: ((b-a)/2)+key2+code. Якщо m не ціле, округляємо до меншого цілого.
5. Перевіряємо a та b на однакову парність, для цього значення m після округлення порівнюємо зі значенням до округлення. Якщо вони співпадають, то a та b на однакову парність, інакше потрібно це позначити. Для цього до m додаємо code\*2, тоді значення m при непарному a+b буде значно більшим ніж при парному, що можна побачити при розшифруванні. (code\*2 додається для повного уникнення проблем колізії)
6. Додаємо до k та m значення key1 та key2 відповідно, та переводимо у символи за кодуванням.
7. Якщо довжина k або m менша за length, дописуємо перед відповідним числом 0 в кількості якої недостатньо для рівності з length.

Повторюємо операцію f для всіх інших пар символів вхідного тексту.

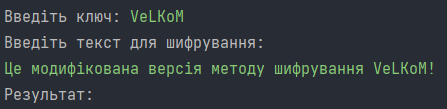
Цей алгоритм представлено на мові програмування Python 3.10 на рис. А.3.

**Алгоритм функції f' при розшифруванні:**

1. За розташуванням у ключі беремо два символи, переводимо їх у числа відповідно до обраного кодування, та позначаємо їх як key1 та key2 відповідно, та додаємо до обох code.
2. Розбиваємо шифротекст на числа довжини length, позначаємо їх як k та m відповідно, прибираємо зайві 0. Віднімаємо key1 та key2 від k та m відповідно. Від m додатково віднімається code.
3. Якщо m менше або рівне code\*2 - позначаємо x=0 та переходимо до наступного пункту, інакше від m віднімаємо code\*2, та позначаємо x=1.
4. Обчислюємо перший елемент вихідного тексту a' за формулою: a'=|k-m+key2|.
5. Обчислюємо другий елемент вихідного тексту b' за формулою: b'=k+m+x-key2.
6. Переводимо a' та b' у символи за кодуванням, та записуємо до вихідного тексту.
7. Повторюємо операцію f' для всіх інших пар символів шифротексту.

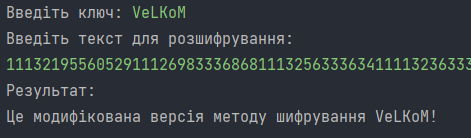
Цей алгоритм представлено на мові програмування Python 3.10 на рис. А.4.

Приклад роботи алгоритму шифрування:



*Рис. 2. 3. Демонстрація роботи програми шифрування алгоритму VeLKoM – спрощений варіант*

111321955605291112698333686811132563336341111323633364001113236333632811132553336340111322855605281112692333582211132505560475111323855605221113246556046611127333336872111323355605281113221333633711127365559944111323833363861113230333634011132575560465111322855605281113234333635111122343336373111223855605091112233333636011122303336324



*Рис. 2. 4. Демонстрація роботи програми шифрування алгоритму VeLKoM – спрощений варіант*

Дана варіація методу шифрування VeLKoM є набагато стійкішою до криптоаналізу, зокрема якщо використовувати ключ, довжина якого рівна довжині вхідного тексту, та елементи якого не повторюються – має зникнути вразливість до статистичного аналізу.

В цій варіації можна використовувати всі елементи кодування.

Також в ній повністю усунено проблеми колізії за рахунок додавання числа code, яке є максимальною кількістю елементів.

**ВИСНОВКИ**

В результаті розв'язання поставлених задач, було досягнуто таких результатів:

1. Вивчено сучасну літературу з теми, в тому числі англомовну. Огляд показав, що дана тема є актуальною на сьогоднішній день. Актуальність полягає у необхідності захисту всіх сфер життєдіяльності від небажаного доступу до інформації.

2. Досліджено базові методи криптографічного захисту інформації, такі як симетричні та асиметричні шифри, методи заміни та перестановки. Розроблено порівняльну таблицю математичних методів шифрування з посиланнями на джерела опису методів.

Перевагами симетричних криптосистем є швидкість (на три порядки вище асиметричних), простота реалізації, менша необхідна довжина ключа, вивченість. До недоліків можна віднести складність управління ключами у великій мережі, складність обміну ключами.

Перевагами асиметричних криптографічних систем перед симетричними криптосистемами є вирішена проблема складного розподілу ключів між користувачами, відсутність квадратичної залежності, можливість застосування протоколів взаємодії сторін, що не довіряють одне одному. До недоліків можна віднести те, що нема математичного доведення незворотності, набагато менша швидкість в порівнянні з симетричним шифруванням та необхідність захисту відкритих ключів від підміни.

3. Проілюстровано роботу різних математичних методів, складено математичну модель власного криптоалгоритму.

На сьогоднішній день криптографія широко використовується не лише хакерами, але і борцями за свободу інформації, фінансовим сектором, воєнними структурами та простими користувачами, охочими захистити свої дані в мережі.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Дробиш Р. С. Аналіз існуючих методів захисту та шифрування інформації / Р. С. Дробиш, М. В. Люта // Інноватика в освіті, науці та бізнесі: виклики та можливості : матеріали II Всеукраїнської конференції здобувачів вищої освіти і молодих учених, м. Київ, 18 листопада 2021 року. – Т. 1. – Київ : КНУТД, 2021. 250-253 с. URL:   
<https://er.knutd.edu.ua/handle/123456789/19525>

2. Худякова К. Ю. Захист конфіденційної інформації в офісній мережі із застосуванням криптографічного методу : магістерська дис. : 171 Електроніка. КПІ ім. Ігоря Сікорського. Київ, 2019. 87 с. URL: <https://ela.kpi.ua/handle/123456789/30851>

3. Миронюк Т. В. Методи та засоби синтезу операцій перестановок, керованих інформацією, для комп’ютерних криптографічних систем : дис. … канд. технічних наук : 05.13.05 / Черкаський держ. техн. універ. Черкаси, 2017. 181 с.

URL:   
<https://www.slideshare.net/sitecdtu/i-75720925>

4. Лужецький В. А., Горбенко І. С. Метод формування перестановок довільної кількості елементів. Т. 15, № 3 Захист інформації. 2013. 262-267 с. URL: <http://ir.lib.vntu.edu.ua/handle/123456789/14580>

5. Симетричні криптосистеми – це. URL: <http://jak.bono.odessa.ua/articles/simetrichni-kriptosistemi-ce.php>

6. Остапенко Н. В., Кінзерявий В. М., Кириченко К. С., Грицак А. В. Удосконалена функція гешування MD4. Безпека інформації. 2018. Т. 28 № 2. 118–123 с. URL:   
<https://jrnl.nau.edu.ua/index.php/Infosecurity/article/view/12955>

7. Лужецький В. А., Горбенко І. С. Шифр заміни на основі псевдонедетермінованого генератора гами. Безпека інформації. 2014. Т. 20 № 2. 159–163 с. URL: <https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwjKy7HV_ND8AhWLuosKHWaaCmgQFnoECBAQAQ&url=https%3A%2F%2Fjrnl.nau.edu.ua%2Findex.php%2FInfosecurity%2Farticle%2Fview%2F7302%2F8196&usg=AOvVaw3IkZdjDIoLe6gKJurTC5dO>

8. Шифрування методом заміни. URL: <http://ni.biz.ua/11/11_4/11_46236_shifrovanie-metodom-zameni-podstanovki.html>

9. Mohammad Ubaidullah Bokhari, Qahtan M. Shallal. A Review on Symmetric Key Encryption Techniques in Cryptography. International Journal of Computer Applications (0975 – 8887) Volume 147 – No.10, August 2016. p. 43-48. URL: <https://www.researchgate.net/publication/333118027_A_Review_on_Symmetric_Key_Encryption_Techniques_in_Cryptography>

10. Swathi Edem, G. Vivek, G. Sandhya Rani. Role of Hash Function in Cryptography. National Conference on Computer Security, Image Processing, Graphics, Mobility and Analytics (NCCSIGMA) International Journal of Advanced Engineering Research and Science (IJAERS). 2016. p. 10-13. URL: <https://www.researchgate.net/publication/312242372_Role_of_Hash_Function_in_Cryptography>

11. Use of Transposition Cipher and its Types. International Journal of Research in Engineering, Science and Management Volume 4, Issue 11, November 2021. p. 164-165. URL:   
<https://journals.resaim.com/ijresm/article/view/1556/1499>

**ДОДАТОК А**

Рисунок А.1.

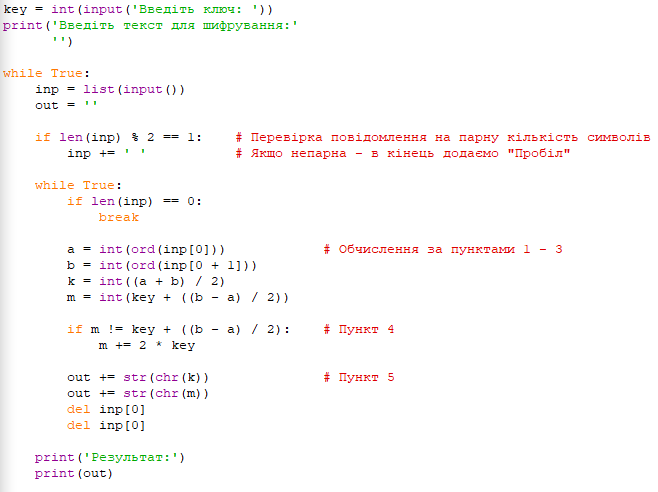


Рисунок А.2.

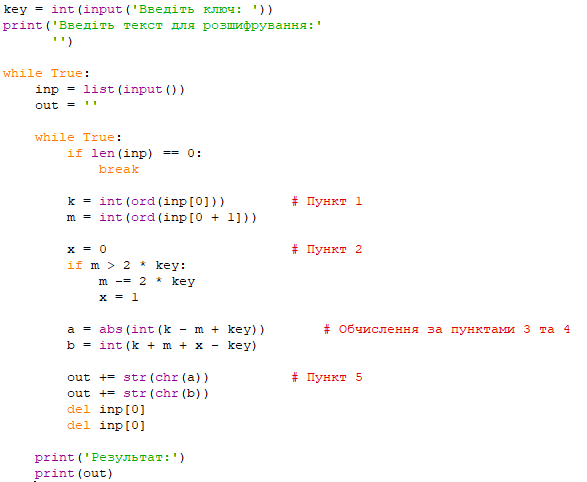


Рисунок А.3.

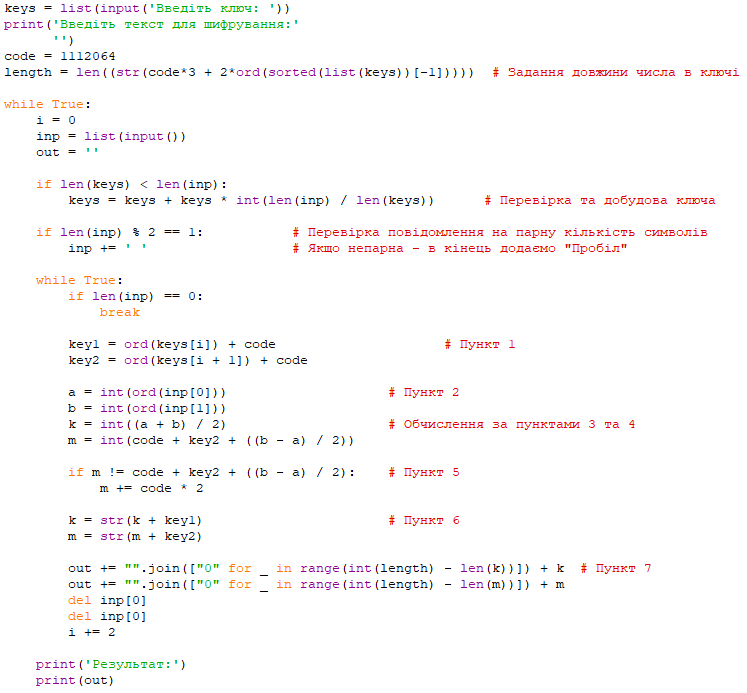
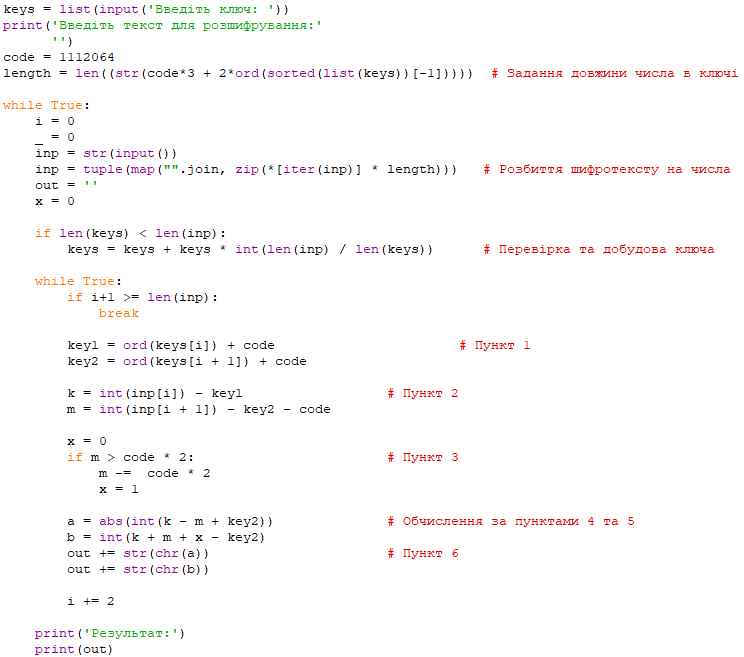


Рисунок А.4.

****