Одеський національний університет імені І. І. Мечникова
(повне найменування закладу вищої освіти)

Факультет математики , фізики та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної алгебри
(повна назва кафедри)

Кваліфікаційна робота

на здобуття ступеня вищої освіти «магістр»

«Псевдовипадкові числа на еліптичних кривих»
(тема кваліфікаційної роботи українською мовою)
«Pseudorandom numbers on elliptic curves»
(тема кваліфікаційної роботи англійською мовою)

Виконав(ла): здобувач(ка) денної/заочної форми навчання
спеціальності 123, комп'ютерна інженерія
(код, назва спеціальності)
Борисов Демид Сергійович
(прізвище, ім'я, по-батькові здобувача)

Керівник д. ф-м н , проф Варбанець П.Д
(науковий ступінь, вчене звання,прізвище, ініціали) (підпис)

Рецензент канд ф-м н, Савастру О.В.
(науковий ступінь, вчене звання,прізвище, ініціали)

Рекомендовано до захисту:
Протокол засідання кафедри
№ ___ від ____.____. 20___ р.

Завідувач(ка) кафедри

_____ _____
(підпис) (прізвище,ім'я)

Захищено на засіданні ЕК № _____
протокол № __від ____.____.20___ р.

Оцінка_____/___ ___/_____
(за національною шкалою/шкалою ECTS/ бали)

Голова ЕК

_____ _____
(підпис) (прізвище, ім'я)

Одеса 2023

АНОТАЦІЯ

Дана дипломна робота присвячена дослідженню використання псевдовипадкових чисел на еліптичних кривих у різноманітних застосуваннях. Еліптичні криві широко використовуються в сучасній криптографії, і використання псевдовипадкових чисел на цих кривих є важливою областю досліджень. Робота містить огляд основних понять, пов’язаних з еліптичними кривими та псевдовипадковими числами, а також розглядає існуючі методи генерації псевдовипадкових чисел. Основна увага приділяється розробці та аналізу методів генерації псевдовипадкових чисел, спеціально оптимізованих для використання з еліптичними кривими. Під час дослідження були розглянуті різні аспекти безпеки та ефективності запропонованого способу. Результати експериментального та порівняльного аналізів дозволяють зробити висновки про придатність та ефективність розробленого алгоритму генерації псевдовипадкових чисел.

ABSTRACT

This thesis is devoted to the study of the use of pseudorandom numbers on elliptic curves in various applications. Elliptic curves are widely used in modern cryptography, and the use of pseudorandom numbers on these curves is an important area of research. This paper provides an overview of the basic concepts related to elliptic curves and pseudorandom numbers, and discusses existing methods for generating pseudorandom numbers. The main focus is on the development and analysis of pseudorandom number generation methods specifically optimised for use with elliptic curves. Various aspects of security and efficiency of the proposed method were considered in the study. The results of the experimental and comparative analyses allow us to draw conclusions about the suitability and effectiveness of the developed pseudorandom number generation algorithm.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ , УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

IoT – Internet of Things , концепція передачі даних між фізичними об’єктами

ЕК – еліптична крива

HMAC – Hash-based message authentication code , конструкція у криптографії , яка використовує хеш-функцію

CTR – Counter Mode , режим роботи блочного шифрування лічильника.

DRBG – Deterministic random bit generator , генератор випадкових бітів , який визначається детермінованим способом.

ВСТУП

Актуальність теми дослідження завжди залежить від її важливості та застосування в різних галузях наукової, виробничої та освітньої діяльності. Криптографія еліптичної кривої є однією з ключових областей, яка значно зросла за останні десятиліття. У цьому контексті приділення особливої уваги проблемі генерації псевдовипадкових чисел на еліптичних кривих має велике теоретичне і практичне значення.

Поточний стан проблеми зумовлений потребою в ефективних і безпечних алгоритмах генерації псевдовипадкових чисел для захисту інформації в електронних системах зв’язку, фінансових транзакціях, інтернет-протоколах та інших сферах, де конфіденційність і безпека даних визначаються як критичні. Водночас варто враховувати тенденції сучасного технологічного розвитку, оскільки зростання обчислювальної потужності та розширення можливостей атак вимагають постійного вдосконалення алгоритмів для забезпечення конфіденційності та цілісності інформації.

Враховуючи вищесказане, актуальним завданням стає подальше вдосконалення алгоритмів генерації псевдовипадкових чисел на еліптичних кривих з урахуванням сучасних вимог безпеки та ефективності. У даній магістерській роботі розглядаються сучасні вимоги та тенденції розвитку криптографії та досліджуються, аналізуються та розробляються нові методи генерації псевдовипадкових чисел на еліптичних кривих. Тому проведення даного дослідження є актуальним і важливим завданням, яке сприятиме подальшому розвитку сучасних методів захисту інформації та забезпеченню високого рівня безпеки в усіх сферах еліптичної криптографії.

Метою цієї роботи є оцінка якості алгоритмів генерації псевдовипадкових чисел на еліптичній кривій з метою подальшого вдосконалення та розробки нових методів, які відповідають сучасним вимогам безпеки та ефективності в галузі криптографії.

Завдання для досягнення мети:

1) Аналіз стану проблеми – огляд існуючих методів генерації псевдовипадкових чисел на еліптичних кривих і визначення їхніх переваг та недоліків.
2) Розробка нових алгоритмів – виявлення проблем із існуючими методами та розробка нових алгоритмів, призначених для підвищення безпеки та ефективності генерації псевдовипадкових чисел.
3) Математичне моделювання – за допомогою математичних методів моделюється розроблений алгоритм для визначення його ефективності та стійкості до потенційних атак
4) Експериментальні дослідження – розроблені алгоритми експериментально перевіряються в реальних умовах, щоб забезпечити практичне використання та оцінити їх продуктивність.
5) Порівняння з існуючими методами – розроблений алгоритм порівнюється з існуючими методами генерації псевдовипадкових чисел на еліптичних кривих для визначення його конкурентоспроможності та переваг.

Об’єкт дослідження: Методи генерації псевдовипадкових чисел на еліптичних кривих.

Предмет дослідження: Застосування алгоритмів генерації псевдовипадкових чисел у криптографії

Матеріальні джерела та методи дослідження:

Вихідні матеріали включають літературу, наукові статті та програмне забезпечення , пов’язане з криптографією еліптичної кривої. Методи дослідження включають математичне моделювання, аналітичний огляд, експереминтальне тестування та порівняльний аналіз.

1 ТЕОРЕТИЧНІ ОСНОВИ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ ТА ЕЛІПТИЧНИХ КРИВИХ

1.1 Визначення еліптичної кривої

Еліптична крива – це математичний об’єкт , який описується наступним рівнянням:

$$y^2 = x^3 + ax + b \tag{1.1}$$

де a і b – константи , що належать до області , що визначає криву. Еліптичні криві широко використовуються в різних розділах математики, включаючи алгебру , топологію та криптографію.

Формальне визначення:

Нехай K – поле , розглянемо рівняння в проективних координатах:

$$Y^2Z = X^3 + aXZ^2 + bZ^2 \tag{1.2}$$

Еліптичну криву також можна визначити в афінних координатах як рівняння у вигляді “див. вираз (1.1)” , де – x та y – афінні координати точки.

Приклад:

Розглянемо еліптичну криву в афінних координатах:

$$y^2 = x^3 - 4x + 4.7$$

У цьому випадку:

- $a = -4$ (коефіцієнт при x)
- $b = 4.7$ (вільний член рівняння)

Ключові аспекти визначення:

а) Рівняння кривої:

1) Рівняння $y^2 = x^3 - 4x + 4.7$ визначає форму кривої.

б) Константи a та b:

1) $a = -4$, $b = 4.7$ визначають параметром кривої

в) Поле K – рівняння визначене над деяким поле K

г) Проективні або афінні координати:

1) Рівняння представлене в афінних координатах , як $y^2 = x^3 - 4x + + 4.7$, або в проективних координатах за допомогою проективних змінних X, Y, Z.

д) Графічне представлення:

1) Визначимо графік кривої на декартовій площині , побудуємо точки , що задовольняють рівняння.

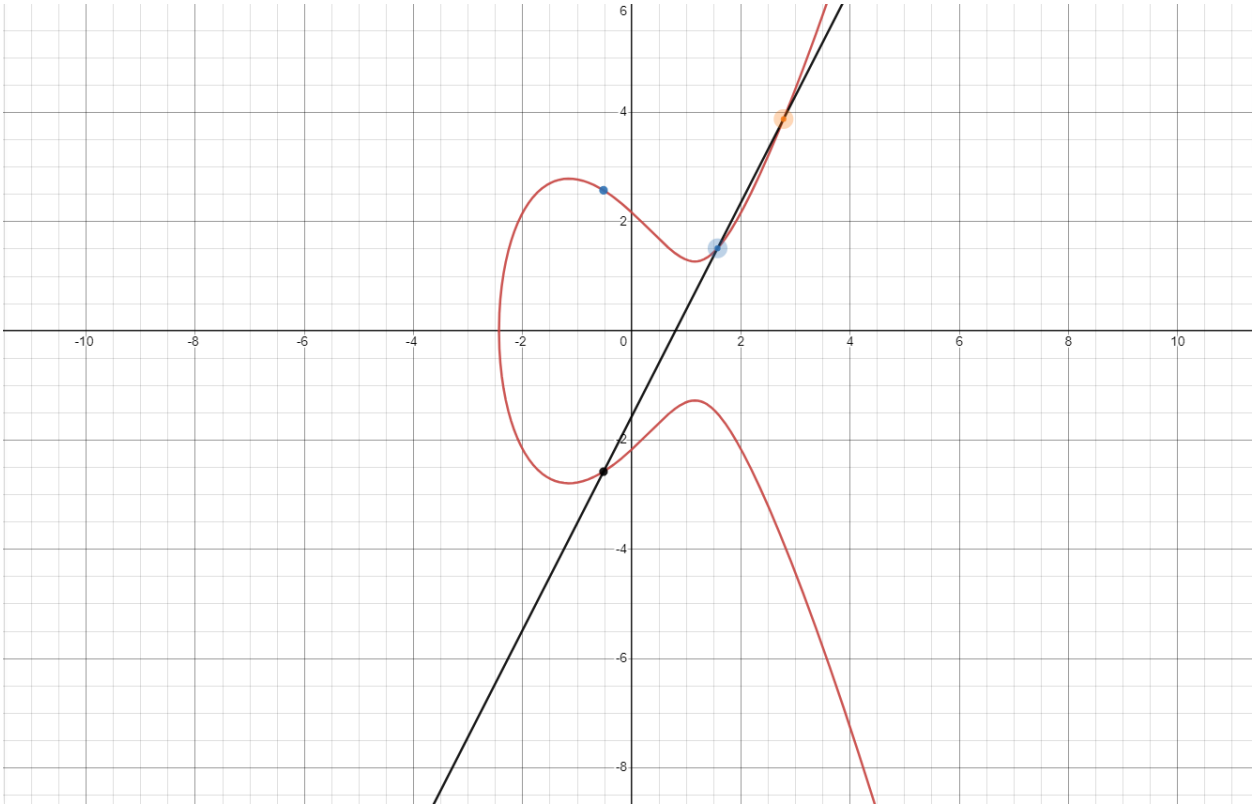

Рисунок 1.1 – Графік кривої на декартовій площині.

Цей графік показує , як рівняння $y^2 = x^3 - 4x + 4.7$ визначають криву з характерною формою еліптчиної кривої. Точки на цій кривій визначаються значенням координат , які задовольняють рівнянню. Такий графічний підхід допомогає візуалізувати абстрактні математичні об’єкти , продемонструвати основні особливості еліптичних кривих.

Математичне представлення еліптичної кривої. Еліптичні криві можуть бути математично представленні різними способами, включаючи афінні та проективні координати, параметризацію точки та рівняння кривої. Розглянемо ці аспекти докладніше.

- Рівняння кривої:

Узагальнене рівняння еліптичної кривої в афінних координатах виглядає як “див. вираз 1.1” , де a і b – константи , які визначають форму кривої. Це рівняння визначає множину точок (x, y) , які задовільняють рівнянню.

- Параматеризація точок:

  Точки на еліптичній кривій можна параметризувати за допомогою параметра t:

$$x(t) = t^2 - a \tag{1.3}$$

$$y(t) = t \times (t^2 - a) - b \tag{1.4}$$

  де t – параметр , а a та b – константи кривої.

- Представлення в проективних координатах:

  У проективних координатах рівняння еліптичної кривої має вигляд “див. вираз 1.2” . Проективні координати дозволяють легше взаємодіяти з алгебраїчними структурами та використовувати їх у криптографії.

Приклад :

Розглянемо еліптичну криву з рівнянням $y^2 = x^3 - 2x + 2$ та параметрами:

$$x(t) = t^2 - 2$$

$$y(t) = t \times (t^2 - 2) - 2$$

Така параметризація дозволяє отримати точки на кривій з конкретними значеннями параметрів t. Таким чином математичне представлення еліптичних кривих забезпечує спосіб зрозуміти цей абстрактний математичний об’єкт і взаємодіяти з ним.

Геометрична інтерпретація еліптичних кривих. Еліптичні криві можна інтерпретувати геометрично через їх графічне представлення, що дозволяє інтуїтивно зрозуміти їх структуру та характеристики. Розглянемо деякі ключові аспекти геометричної інтерпретації еліптичних кривих:

- Кривизна та форма:

Еліптичні криві мають характерну форму витягнутої кривої, яка може нагадувати форму еліпса або кола. Їх кривизна змінюється вздовж кривої, визначаючи вигляд і структуру точок на кривій.

- Нейтральний елемент та безкінечні точки:
  Геометрично нейтральні елементи на еліптичній кривій можуть відповідати нескінченно віддаленим точкам. Тобто з певної точки кривої можна йти до нескінченності , а потім до іншої точки , створюючи замкнуту структуру.
- Комутативність та асоціативність точок:
  Операція додавання точок до еліптичної кривої демонструє комутативність і асоціативність, які визначаються геометричною інтерпретацією. Сума трьох точок однакова незалежно від порядку додавання.
- Точкові операції:
  Операції з точками , такі як додавання точок і подвоєння точок, представляють геометричні операції над точками на кривій . Наприклад, подвоєння точки буде виглядати дотичною до кривої у відповідній точці та віддзеркалюватиме дотичну.
- Афінні та проективні координати:
  Геометрично афінні та проективні координати використовуються для зображення точок на еліптичних кривих. Проективні координати полегшують взаємодію з іншими алгебраїчними структрурами.

Приклад:

Розглянемо графік еліптичної кривої $y^2 = x^3 - 7x + 5.2$

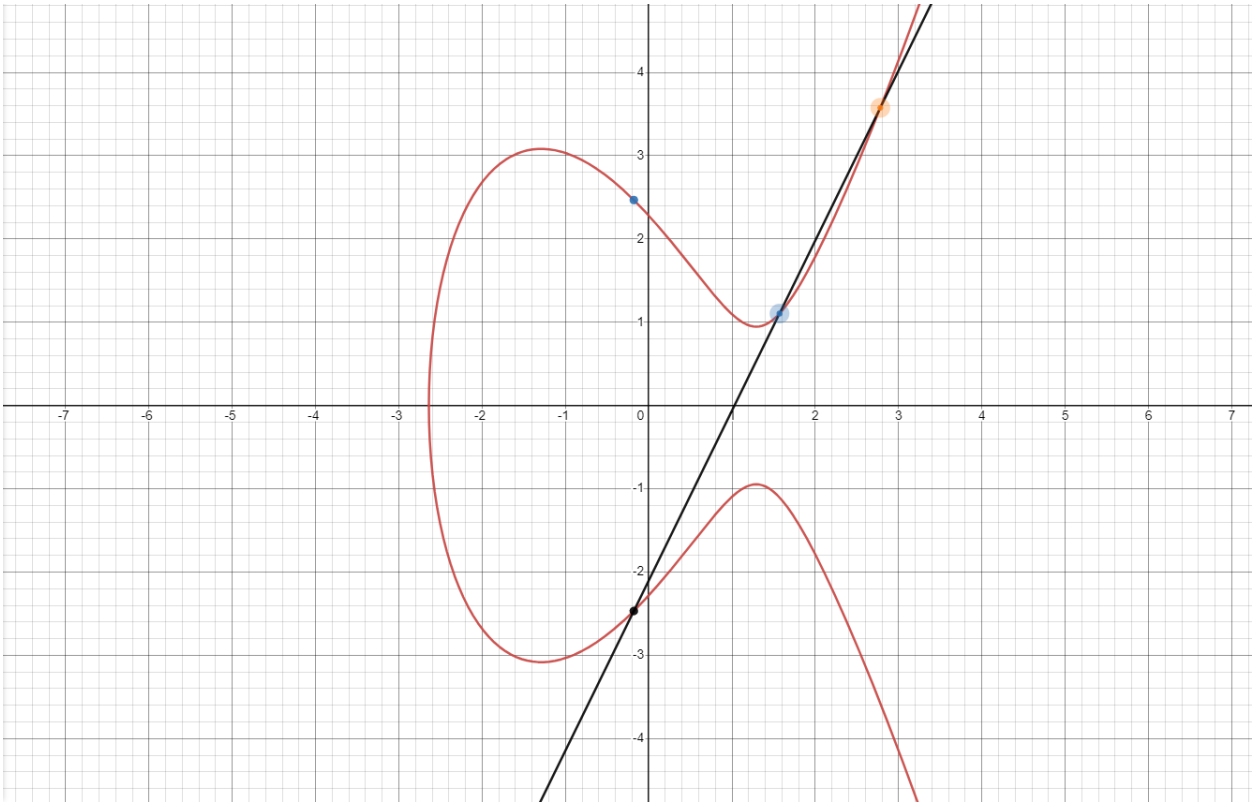

Рисунок 1.2 – Графік еліптичної кривої

На цьому графіку показано, як точки, що відповідають рівнянню, розташовані уздовж фігури еліптичної кривої. Важливо звернути увагу на деякі геометричні аспекти:

- Форма кривої:

  Цей графік показує текстуру , яку мають еліптичні криві. Форма може бути овальною або круглою, залежно від конкретного рівняння.
- Точки перетину з осями:

  Точка, де крива перетинає вісь X, представляє розв’язок рівняння $y = 0$. Їх можна використовувати для знаходження коренів кривих.
- Точки дотику:

  Точки, де крива торкається осі X або Y, представляють рішення з умовами дотику.
- Специфічні геометричні операції:

  Робота з графіком допомогає візуалізувати такі геометричні операції, як додавання точок на кривій або подвоєння точки.
- Властивості безкінечно віддалених точок:

Нескінченна точка може бути визначена горизонтальною лінією, яка перетинає криву одночасно в кількох точках.

Цей приклад демонструє, як графічне представлення еліптичної кривої може допомогти людям інтуїтивно зрозуміти її геометричні властивості. Основні поняття, такі як фігури, перетини та геометричні операції, можна легко зрозуміти за допомогою зображень.

1.2 Основні характеристики еліптичної кривої

Точки на еліптичній кривій. Еліптична крива визначається рівнянням “див. вираз 1.1”. Точка (*x, y)* на цій кривій задовольняє цьому рівнянню. Основні елементи визначення точок:

- Координати точок:
  Точка (*x, y*) є точкою на еліптичній кривій, якщо її координати задовольняють рівнянню кривої.
- Константи *a* та *b*:
  Координати точок залежать від констант *a* та *b,* які визначають форму кривої.
- Задане поле K:
  Точки визначаються в полі K, яке може бути комплексним або кінечним полем

Приклад:

Розглянемо еліптичну криву з рівнянням $y^2 = x^3 - 2x + 2$. Для знаходження на цій кривій можна вибрати конкретні значення *x* та обчислити відповідні значення *y* з рівняння:

При *x* = 1:

$y^2 = 1^3 - 2 \times 1 + 2$

$y^2 = 1 - 2 + 2 = 1$

y = 1 або *y* = -1

Таким чином , точки (1,1) та (1, -1) належать еліптичній кривій

При *x* = 2:

$y^2 = 2^3 - 2 \times 2 + 2$

$y^2 = 8 - 4 + 2 = 6$

Тут точки можуть бути представлені у вигляді (2, √6) та (2, -√6).

Ці значення *x* і відповідні їм значення *y* являють собою точки на еліптичній кривій, які задовольняють умовам рівняння. Загалом кажучи, точки на еліптичній кривій можна знайти шляхом підключення різних значень *x* і обчислення відповідних значень *y*.

Нейтральний елемент і віддалені точки. На еліптичній кривій є нейтральний елемент , роль якого подібна до ролі нуля в адитивній групі. Цей елемент представлений як **0** і розташований на “нескінченності”. Нейтральні елементи мають такі властивості:

- Нейтральність відносно додавання:
  Для будь-якої точки *P* на еліптичній кривій виконується *O* + *P* = *P* + *O* = *P*. Іншими словами , додавання точки до нейтрального елемента не змінює точку.
- Ідентичність для додавання:
  O + *O* = *O*. Тобто , додавання бескінечно віддаленої точки до самої себе дає бескінчено віддалену точку.

Геометрична інтерпретація. Нескінчену точку на еліптичній кривій можна вважати такою, що лежить на вертикальній лінії, яка ніколи не перетинає криву. Нейтральний елемент *O* можна розглядати як точку перетину цієї вертикальної лінії та еліптичної кривої.

Групова структура та операція додавання точок на еліптичній кривій. Еліптичні криві утворюють адитивні комутативні групи точок. Операція додавання точок на еліптичний кривих визначається геометрично та алгебраїчно. Розглянемо цю операцію докладніше:

Геометрична інтерпретація. Нехай *P* і *Q* – дві точки на еліптичній кривій. Тоді результат операції додавання *P* і *Q* може бути геометрично інтерпретованим як третя точка *R* , яка знаходиться на лінії, яка проходить через *P* і *Q* , і перетинає еліптичну криву. Якщо *P* і *Q* різні, лінія проходить

через дві точки і її перетин з еліптичною точкою додає нову точку R . Якщо $P = Q$, лінія стає дотичною до кривої в точці P , і вона знову перетинає криву в третій точці R .

Алгебраїчне визначення операції додавання. Нехай $P = (x_p , y_p)$ і $Q = (x_Q , y_Q)$ - дві точки на еліптичній кривій. Тоді операція додавання визначається наступним чином:

- Визначення лінії L :

  Якщо $P \neq Q$, лінія L через P і Q задається рівнянням:

$$L : s = \frac{y_Q - y_P}{x_Q - x_P} \times (x - x_P) - y_P \quad (1.5)$$

  Якщо $P = Q$, лінія L через P задається рівнянням:

$$L : s = \frac{3x_P^2 + a}{2y_P} \times (x - x_P) - y_P \quad (1.6)$$

- Знаходження точки R :

  Третя точка R знаходиться перетином лінії L і еліптичної кривої.

$$R = (x_R , y_R) = (s^2 - x_P - x_Q , -s \times (x_R - x_P) - y_P) \quad (1.7)$$

Приклад :
Розглянемо еліптичну криву $y^2 = x^3 - 2x + 2$ та дві точки P = (1, -1) і Q = (2 , -√6). Знайдемо $P + Q$:

- Визначення лінії L:

  $L : s = \frac{-\sqrt{6}-1}{2-1} \times (x - 1) - 1$

- Знаходження точки R:

  $R = (s^2 - 1 - 2 , -s \times (x_R - 1) - 1)$

Отже , результат операції додавання $P + Q$ на еліптичній кривій дорівнює

$R = (8 + 2\sqrt{6}, 7\sqrt{6} - 14)$

Існування оберненого елемента для кожної точки. У груповій структурі еліптичних кривих кожна точка на кривій відповідає інверсній точці, тобто іншій точці на тій же кривій. Обернена точка задовольняє властивості , якщо їх сума дає нейтральний елемент(віддалену точку). Нехай $P = (x , y)$ – точка на еліптичний кривій. Тоді оберненою точкою буде $-P = (x , -y)$. Іншими

словами, обернена точка має ті самі *x*-координати, але з протилежним знаком *y*-координати.

Приклад :

Розглянемо точку $P = (2,3)$ на еліптичній кривій. Її обернена точка буде дорівнювати $-P = (2, -3)$. Перевіримо властивості оберненого елемента:

- $P + (-P) = (2,3) + (2,-3) = \boldsymbol{O}$(нейтральний елемент)
- $-(-P) = -(2, -3) = (2,3)$

Отже , обернена точка P – це $(-P) = (2, -3)$, і вони утворюють пару, для якої сума дає нейтральний елемент.

Ранг еліптичної кривої – це важливий параметр , який характеризує структуру точок на еліптичній кривій у контексті теорії чисел та криптографії. Ранг еліптичної кривої E , позначається як r , визначається як максимальне число лінійно незалежних точок у наборі точок кривої E. Формально ранг еліптичної кривої – це порядок набору точок , визначений наступним чином:

$$r = ord(E) \tag{1.8}$$

Вплив рангу на структуру групи точок :

- Ранг 0:

  Якщо ранг $r = 0$, то група точок складається тільки з нейтрального елемента. Група є тривіальною , і крива не має інший раціональних точок.
- Ранг 1:

  Якщо ранг $r = 1$, то група має структуру циклічної групи порядку r.

  Тобто група складається з елементів 0 , P , $2P$, ... , $(r – 1)P$, де P – базова точка. Ранг 1 є досить звичайним для багатьох еліптичних кривих.
- Ранг більше 1:

  Якщо ранг r більше 1, група має більш складну структуру. Вона може бути виражена як пряма сума циклічних груп різних порядків і може містити нескінчену кількість раціональних точок.

У криптографії використання еліптичних кривих у криптографічних алгоритмах , таких як еліптична криптографія, часто залежить від розміру

рангу. Велике значення рангу може забезпечити достатню безпеку для криптографічних програм.

Визначення рангу еліптичної кривої є складним завданням, і для будь-якої еліптичної кривої не існує точного методу обчислення рангу. Однак існують алгоритми та методи , які дозволяють наближено ранжувати певні класи еліптичних кривих. Важливо враховувати , що ці методи не гарантують точного визначення рангу , а лише надають оцінку рейтингу. Ось деякі з цих алгоритмів:

- Алгоритм Куффа:

  Алгоритм заснований на розв’язуванні діофантового рівняння за допомогою операцій з раціональними координатами точок і раціональних чисел. По-перше, алгоритм використовує метод Паріса-Саг’є для пошуку обмежень на раціональні точки на кривій. Після цього методи теорії чисел використовуються для аналізу множини раціональних точок і визначення їх рангу.
- Алгоритм Сатоха-Арасакі:

  Цей алгоритм використовує комбінацію спеціальних точок для апроксимації раціональних точок на еліптичний кривій. Алгоритм використовує ітераційний підхід для пошуку таких точок і отримання оцінок ранжирування.
- Метод бравіссімо:

  Цей метод використовує квадратичну сітку раціональних точок для апроксимації раціональних точок на еліптичний кривій. Далі для аналізу цих балів і визначення рейтингу застосовано теорію Гегі-Пітерсона.

Варто зазначити , що ці алгоритми не є точними і вимагають певних геометричних і арифметичних прийомів для вирішення проблеми визначення рангу еліптичних кривих. Крім того , у деяких випадках вони можуть потребувати значних обчислювальних ресурсів.

1.3 Особливі випадки еліптичних кривих

Еліптичні криві над скінченними полями. Еліптичні криві над кінцевими полями (еліптичні криві над полями Галуа) мають широкий спектр застосувань , включаючи криптографію , алгебраїчну геометрію та теорію чисел. Такі криві можна визначити за допомогою рівнянь над скінченними полями , а їхні властивості можна вивчати в контексті скінченних полів. Еліптична крива над скінченним полем F_q визначається рівнянням:

$$E: y^2 = x^3 + ax + b \ mod \ p \quad (1.9)$$

де a , b F_q – коефіцієнти рівняння , а p – порядок кінцевого поля. Рівняння має виконуватися для кожної пари координат (x , y) $Fq \times Fq$.

Особливості еліптичних кривих над кінцевими полями:

- Порядок групи точок:

  Група точок на еліптичній кривій над кінцевим полем завжди скінченна. Порядок груп можна знайти за допомогою таких алгоритмів , як алгоритм Шенкса або алгоритм Поліга-Хелмана.

- Теорема Хасе:

  Теорема Хасе для еліптичних кривих над кінцевими полями встановлює зв’язок між порядком точкової групи та розміром поля. У ньому йдеться про кількість раціональних точок N . N задовільняє умову:

$$|N - (p + 1)| \ \leq 2 \sqrt{p} \quad (1.10)$$

- Криві над бінарними полями

Еліптичні криві над бінарними полями (поля Галуа порядку 2^m) мають певні характеристики , які використовуються в криптографії . Їх властивості є основою для ефективного шифрування та алгоритмів цифрового підпису.

Застосування:

- Криптографія

  Еліптична криптографія використовує властивості еліптичних кривих над кінцевими полями для створення ефективних алгоритмів шифрування та цифрового підпису.

- Кодування та корекція помилок:

Еліптичні коди базуються на еліптичних кривих над кінцевими полями і використовуються для виправлення помилок у передачі даних.

- Теорія чисел

Еліптичні криві над кінцевими полями відіграють важливу роль у теорії чисел , особливо в доведенні теореми Ферма.

Сингулярність еліптичної кривої. Особливість еліптичної кривої пов’язана з властивостями її рівняння, тобто зі значеннями параметрів, які використовуються в цьому рівнянні. Еліптична крива вважається особливою, якщо вона має точку, де графік рівняння втрачає свою гладкість, тобто якщо похідне рівняння не існує або дорівнює нулю.

Зв’язок із рівнянням еліптичної кривої. Рівняння еліптичної кривої в загальному вигляді має форму:

$$E: y^2 = x^3 + ax + b \quad (1.11)$$

де a та b – константи. Якщо дискримінант рівняння, що визначається як $4a^3 + 27b^2$, дорівнює нулю, то крива вважається сингулярною. Тобто, еліптична крива сингулярна, якщо $4a^3 + 27b^2 = 0$. Властивості сингулярних і несингулярних кривих:

- Сингулярні криві:

Сингулярні криві можуть мати точки , де гладкість графіка порушується.

Властивості набору точок на сингулярній кривій можуть бути менш очевидними або невизначеними взагалі.

Особливості можуть виникати при великих значеннях параметрів a та b в рівняннях еліптичної кривої.

- Несингулярні криві:

Несингулярні криві гладкі по всьому графіку, без точок , де гладкість втрачається.

Властивості множини точок на неособливій кривій можна визначити більш чітко і правильно.

Більшість еліптичних кривих, які використовуються в криптографії та інших програмах вибираються несингулярними.

Приклад:

Розглянемо еліптичну криву $E: y^2 = x^3 - 3x + 5$. Її дискримінант $4a^3 + +27b^2$ рівний -783 , що не дорівнює нулю. Отже , ця крива є несингулярною.

1.4 Застосування еліптичних кривих у криптографії

Роль еліптичних кривих у сучасній криптографії. Еліптичні криві відіграють ключову роль у сучасній криптографії, особливо в області еліптичної криптографії. Використання еліптичних кривих у криптографії забезпечує високий рівень безпеки та дозволяє більш ефективно використовувати ресурси порівняно з традиційними криптографічними системами, такими як RSA або DSA. Декілька криптографічних застосувань еліптичних кривих:

- Еліптичне шифрування:
  Еліптичне шифрування використовує математичні властивості еліптичних кривих для забезпечення безпеки передачі даних. Він забезпечує такий самий рівень безпеки, як і традиційні криптосистеми, але з меншими розмірами ключів, що робить його більш ефективними там , де ресурси обмежені.
- Еліптичні цифрові підписи(ECDSA)
  ECDSA – це алгоритм цифрового підпису, який використовує еліптичні криві. Він забезпечує високий рівень безпеки за допомогою коротких ключів і знижує навантаження на обчислювальні ресурси та мережевий трафік.
- Еліптичний обмін шифрами(ECDH):
  Обмін еліптичним шифром використовується для забезпечення обміну ключами між сторонами. Він заснований на складності задачі обчислення дискретного логарифма на еліптичних кривих і забезпечує безпеку обміну ключами.
- Ідентифікація та аутентифікація:

Еліптичні криві також використовуються в протоколах ідентифікації та аутентифікації , таких як еліптичні криптосистеми з відкритим ключем(ECKEY). Вони забезпечують безпечний обмін даними для аутентифікації та перевірки автентичності.

- Конфіденційні обчислення:
  Еліптичні криві також використовуються в конфіденційних обчислювальних протоколах, де дві сторони можуть спільно обчислювати функцію , не розкриваючи свої вхідні дані.
- Стандарти безпеки:
  Еліптичні криві тепер є частиною багатьох стандартів безпеки, таких як стандарти шифрування та цифрового підпису NIST(Національний інститут стандартів і технологій)

Використання еліптичних кривих у криптографії забезпечує баланс між безпекою та ефективністю , що особливо важливо в середовищах з обмеженими ресурсами , таких як мобільні пристрої та Інтернет речей(IoT)

Переваги еліптичних кривих в порівнянні з іншими криптографічними методами:

- Менший розмір ключів:
  Еліптичні криві дозволяють досягти такого ж рівня безпеки , як і інші криптосистеми , але з більш короткими ключами. Це особливо важливо в сучасному світі, де обмін ключами через мережу вимагає менше обчислювальних ресурсів і зменшує обсяг даних, що передаються.
- Ефективність з точки зору ресурсів:
  Криптографічні операції (шифрування, підписання, обмін ключами) вимагають менше обчислювальних ресурсів , щоб бути ефективними , що робить еліптичні криві особливо придатними для обмежених пристроїв, таких як мобільні телефони або пристрої Інтернету речей (IoT).
- Мінімізація ресурсів у мережі:

Коротші ключі еліптичної кривої зменшують навантаження на мережу , особливо кількість даних, що передаються під час обміну ключами та інших криптографічних операцій.

- Високий рівень безпеки:
  Еліптичні криві забезпечують високий рівень безпеки. Оскільки складність обчислення дискретних логарифмів на еліптичних кривих забезпечує надійну криптографічну стабільність, вони вважаються одним із найбезпечніших методів шифрування.
- Оптимальність з точки зору розмірів ключів та швидкості:
  Еліптичні криві забезпечують найкращий баланс між розміром ключа та швидкістю криптографічних операцій. Це особливо важливо для використання сучасних криптографічних протоколів.
- Спрощення управління ключами:
  Менший розмір ключів з еліптичною кривою полегшує керування ключами та зменшує обсяг збереженої ключової інформації.
- Застосування в області інтернету речей (IoT)
  Еліптичні криві особливо підходять для додатків IoT завдяки їх ресурсоефективності та здатності працювати на пристроях з обмеженими властивостями.

Загалом еліптичні криві мають значні переваги в сучасних криптосистемах , де ключову роль відіграють такі важливі аспекти , як швидкість , безпека та ефективність використання ресурсів.

Еліптичні криві у схемах електронного підпису(ЄЦП). Еліптичні криві широко використовуються в схемах електронного підпису завдяки їх ефективності та високому рівню безпеки. Нижче описано дві основні схеми електронного підпису з використанням еліптичних кривих:

a) ECDSA(Elliptic Curve Digital Signature Algorithm)
   ECDSA – це алгоритм електронного підпису, заснований на математичних властивостях еліптичних кривих. Він використовує ключі

на основі еліптичної кривої для створення та перевірки електронних підписів. Кроки алгоритму:

1) Генерація ключів

   Аліса генерує пару ключів – закритий ключ d і відповідний йому відкритий ключ Q , де $Q = d \times G$, G – базова точка(твірна) еліптичної кривої.

2) Підпис повідомлення

   Аліса хешує повідомлення та створює підпис, використовуючи свій закритий ключ і вказані параметри еліптичної кривої.

3) Перевірка підпису

   Боб отримує повідомлення , хеш якого збігається з хешем у підписі, і використовує відкритий ключ Аліси для перевірки підпису

Переваги ECDSA – розмір ключа менший порівняно з традиційними схемами , що зменшує обчислювальну складність та високий рівень безпеки через складність задачі обчислення дискретних логарифмів на еліптичних кривих.

б) EdDSA(Edwards-curve Digital Signature Algorithm)

EdDSA – це алгоритм електронного підпису , який використовує еліптичні криві у формі Едвардса. Цей алгоритм забезпечує високий рівень безпеки та ефективності. Кроки алгоритму:

1) Генерація ключів:

   Такий же , як і в ECDSA, генерація пари ключів – приватного та публічного.

2) Підпис повідомлення:

   Створення підпису, використовуючи закритий ключ, хеш повідомлення та параметри еліптичної кривої у формі Едвардса.

3) Перевірка підпису:

   Перевірка підпису аналогічна ECDSA, де використовуються публічний ключ для верифікації.

Переваги – запобігання певним атакам , таким як атаки з бічних каналів та ефективний і простий у реалізації.

Загальні переваги використання еліптичних кривих в схемах електронного підпису:

- Для підписання та перевірки використовуються менше даних.
- Ефективність при роботі на пристроях з обмеженим ресурсом.
- Високий рівень безпеки.

Ці схеми відображають широке використання еліптичних кривих у сучасній криптографії , де забезпечення безпеки при мінімізації обчислювальних ресурсів стає ключовим фактором.

Безпека та ефективність використання еліптичних кривих в схемах електронного підпису:

а) Безпека:

1) Сложність атак:

   Використання еліптичних кривих у схемах електронного підпису базується на розв’язуванні задачі дискретного логарифмування на еліптичних кривих. Це завдання важливе для безпеки системи. Складність розв’язання дискретних логарифмів на еліптичних кривих робить їх чудовим вибором для безпеки.

2) Стійкість до атак:

   Порівняно з класичними методами шифрування, еліптичні криві можуть забезпечити високий рівень стійкості до атак, наприклад тих, що використовують квантові обчислення.

3) Розмір ключів:

   Розмір ключа в схемах електронного підпису з еліптичною кривою менший порівняно з традиційними методами, що робить його менш вразливим до атак і зменшує обчислювальні зусилля.

б) Ефективність:

1) Менший обсяг даних:

Використання еліптичних кривих дозволяє створювати та перевіряти підписи з меншою кількістю даних , що полегшує передачу та зберігання цифрових підписів.

2) Швидкість виконання операцій:
   Операції, пов’язані з електронними підписами, виконуються швидше на еліптичних кривих порівняно з традіційними методами шифрування. Це робить їх дуже ефективними для програм реального часу.
3) Ефективність ресурсів:
   Зменшення розміру ключа та обчислень може покращити продуктивність пристроїв з обмеженими ресурсами , таких як мобільні телефони та пристрої IoT.
4) Висока ступінь паралелізації:
   Еліптична криві дозволяють виконувати операції паралельно, що робить обчислення більш ефективними на сучасних багатоядерних пристроях.

Загальний висновок:

Використання еліптичних кривих у схемах електронного підпису забезпечує високий рівень безпеки завдяки ефективному використанню ресурсів. Їх переваги в обчислювальній складності, малий розмір ключа і висока стійкість до різних атак роблять їх привабливими для сучасних криптографічних додатків.

1.5 Визначення псевдовипадкового числа

Псевдовипадкові числа - це числові значення, які на перший погляд здаються випадковими, але насправді генеруються за допомогою певного алгоритму. Вони використовуються в різних сферах, включаючи інформаційну безпеку, статистику, моделювання, тестування тощо. Генерація псевдовипадкових чисел є детермінованим процесом, але в той же час послідовності чисел намагаються мати властивості, подібні до випадкових

чисел, що дозволяє використовувати їх у різноманітних програмах. Основна мета – створити послідовність, яка намагається імітувати статистичні властивості випадкових чисел, хоча вони визначаються певним алгоритмом. Псевдовипадкові числа мають кілька основних характеристик, які визначають їхні властивості та застосування:

- Детермінованість:

  Генерація псевдовипадкових чисел є детермінованим процесом, оскільки базується на певному алгоритмі. Це означає, що та сама послідовність чисел буде згенерована на основі початкового стану, заданого алгоритмом.
- Періодичність:

  Кожен генератор псевдовипадкових чисел має кінцевий період, після якого послідовність повторюється. Це означає, що при тривалому використанні одна і та ж послідовність чисел може бути відновлена.
- Ініціалізація:

  Для багатьох генераторів потрібне початкове значення, яке називається “засіб” , яке використовується для запуску процесу генерації. Однак однакові частки призводять до однакової послідовності чисел.
- Статистичні властивості:

  Псевдовипадкові числа повинні мати подібні статистичні властивості до тих, які ми очікуємо від випадкових чисел. Це означає, що вони мають бути рівномірно розподілені та добре проходити різні статистичні тести на випадковість.
- Ефективність:

  Генератори мають бути достатньо швидкими та вимагати обмежених обчислювальних ресурсів для використання в режимі реального часу в різноманітних програмах.

Ці характеристики визначають використання та надійність генераторів псевдовипадкових чисел у різних областях, включаючи криптографію, моделювання, тестування тощо.

1.6 Методи генерації псевдовипадкових чисел

Методи генерації псевдовипадкових чисел. Метод лінійної конгруенції є одним із найпоширеніших і найпростіших методів генерації випадкових чисел. Їх основна ідея полягає у використанні рекурсивних лінійних виразів для створення послідовностей чисел. Формула методу лінійної конгруенції має такий вигляд:

$$X_{n+1} = (aX_n + c) mod\ m \tag{1.12}$$

де:

X_n – поточний член послідовності

a – множник

c – приріст

m – модуль(величина, якою обмежується генерована послідовність)

Основні характеристики лінійних конгруентних методів включають:

- Періодичність:
  Період цього методу може бути дуже великим, але обмежений максимально можливою кількістю унікальних значень у послідовності, яка дорівнює по модулю m .
- Спроектована випадковість:
  Якщо параметри вибрано правильно (особливо a і m взаємно прості), то цифрова послідовність може демонструвати випадкові характеристики.
- Залежність від початкового значення:
  Методи лінійної конгруенції чутливі до початкових значень(зернистості). Тобто за однакових параметрів різні частинки призведуть до різних числових послідовностей.
- Використання у криптографії:

Лінійні конгруентні методи не підходять для криптографічних додатків, оскільки їхні властивості передбачувані та піддаються атаці.

Методи на основі регістру зсуву популярні та ефективні для генерації псевдовипадкових чисел. Основна ідея полягає у використанні бітових регістрів і операцій зсуву для створення нових чисел у вигляді послідовності. Завдяки своїй простоті та ефективності вони особливо використовуються в апаратних реалізаціях.

Основні характеристики методів на основі зсуву:

- Ініціалізація:
  Початкове значення регістра (seed) визначається як вхідний параметр генератора
- Генерація чисел:
  Операції зсуву та побітові операції використовуються для зміни стану регістра та отримання нових чисел.
- Періодичність та період:
  Коли використовуються певні комбінації операцій регістра та зсуву, у послідовності чисел може виникати періодчиність. Період – це кількість унікальних значень, які генератор може згенерувати, перш ніж він почне повторюватися.
- Лінійна та спроектована випадковість:
  Хороший регістр зсуву, який використовується в поєднанні з певними операціями, може гарантувати лінійність та інженерну випадковість у згенерованих числах.

Недоліки:

- Короткі періоди:
  Деякі комбінації регістрів і операцій можуть мати коротші періоди, що обмежує кількість унікальних значень.
- Залежність від початкового значення:

Якщо параметри погані, може бути велика залежність від початкового значення.

- Нестійкість у криптографії:

  Багато регістрів зсуву не підходять для криптографічних програм через їх певну структуру та передбачуваність.

Однак при правильному виборі параметрів регістр зсуву може бути дуже хорошим і ефективним інструментом для генерації псевдовипадкових чисел. Алгоритми , засновані на хеш-функціях, використовують властивості хешування для генерації псевдовипадкових чисел. Основна ідея полягає в тому , що вхідні дані (можливо, попередньо визначені) обробляються хеш-функцією, а отримане значення розглядається як псевдовипадкове число. Щоб забезпечити непередбачуваність і розподілені значення, важливо вибрати хеш-функцію без колізій (колізія – це ситуація , коли різні вхідні дані дають однакове хеш-значення)

Основні етапи алгоритмів на основі хеш-функцій:

- Вибір хеш-функції:

  Виберіть хеш-функцію без колізій, яка може приймати великі обсяги вхідних даних і створювати коротки хеш-коди.

- Ініціалізація:

  За бажанням генератор можна ініціалізувати початковим значенням або початковим значенням.

- Генерація чисел:

  Вхідні дані (або їх комбінація) обробляються хеш-функцією, а отримане значення використовується як псевдовипадкове число.

- Непередбачуваність:

  Використання хеш-функції без колізій дозволяє отримати непередбачувані псевдовипадкові значення.

- Розподіленість значень:

Якщо хеш-функція має хороші властивості розподілу, то ймовірність отримання певного числа повинна бути однаковою для всіх можливих значень.

Недоліки:

- Обмежена довжина чисел:
  Розмір хеш-функції обмежує довжину псевдовипадкових чисел , які можуть бути згенеровані.
- Залежність від хеш-функцій:
  Розмір хеш-функції обмежує довжину псевдовипадкових чисел , які можуть бути згенеровані
- Нестійкість у криптографії:
  Хоча існують криптографічно стабільні хеш-функції, не всі хеш-функції підходять для криптографічного використання.

Алгоритми, засновані на хеш-функціях, зазвичай використовуються в різноманітних програмах, таких як генерація випадкового ключа в криптографії, але важливо вибрати хеш-функцію на основі конкретного контексту та вимог програми.

1.7 Статистичні тести

Оцінка якості псевдовипадкових чисел. Оцінка якості псевдовипадкових чисел передбачає виконання статистичних тестів для перевірки їх випадку. Ці тести допомогають визначити, наскільки послідовність чисел відповідає властивостям випадковості та чи виявляють вони будь-які порушення чи закономірності. Нижче приведені деякі загальні статистичні тести:

а) Тест на рівномірність розподілу:

Перевірка рівномірності розподілу перевіряє, наскільки кожне значення в послідовності псевдовипадкових чисел зустрічається приблизно з однаковою частотою. Це важливо для забезпечення рівномірності розподілу чисел у

заданому інтервалі. Основна ідея полягає в тому , щоб порівняти фактичну частоту появи значення з очікуваною частотою появи.

Кроки тесту на рівномірність розподілу:

- Обрання інтервалів:
  Визначає задану кількість рівних інтервалів у діапазоні можливих значень (наприклад , від 0 до 1)
- Підрахунок частот:
  Підрахування фактичної кількості чисел , які потрапляють у кожен інтервал.
- Обчислення очікуваних частот:
  Обчислення того , скільки в середньому чисел повинно потрапити в кожен інтервал, якщо розподіл буде рівномірним
- Хі-квадрат тест:
  Використання хі-квадрат тесту для порівняння фактичних та очікуваних частот. Формула хі-квадрат:

$$x^2 = \sum \frac{(O_i - E_i)}{E_i} \quad (1.12)$$

  де – O_i фактична частота в і-му інтервалі, E_i – очікувана частота в інтервалі.

Порівняння з критичним значенням – порівняйте розраховане значення хі-квадрат із критичним значенням, яке визначає рівень значущості (зазвичай використовується рівень значущості 0.05)

Прийняття рішення – якщо розраховане значення хі-квадрат менше за критичне значення, немає доказів того , що розподіл є нерівномірним. Інакше можуть виникнути питання щодо якості цифрової генерації.

б) Тест серій:

Тест серії або перевірка послідовності перевіряє, чи існує довгий ряд ідентичних або збіжних значень у послідовності псевдовипадкових чисел. Мета тесту – визначити , чи відбувається подія, коли числа в послідовності

схожі або ідентичні одне одному протягом тривалого періоду часу, що може вказувати на відсутність випадковості.

Кроки алгоритму:

- Визначення довжини серій:
  Визначення мінімальної тривалості серії, яку слід вважати значущою для аналізу
- Пошук серій:
  Перебирання ряду чисел і визначення довгих серій подій, коли значення збігаються або мають незначні відмінності.
- Обчислення частот:
  Розрахунок фактичної кількості довгих серій у вибірці.
- Тест на статистичну значущість:
  Визначення , чи є частота послідовності значущою з точки зору випадковості, використовуючи статистичні методи , такі як перевірки ознак.
- Порівняння з критичним значенням:
  Порівняння результатів тесту з критичним значенням для визначеного рівня значущості (зазвичай використовується рівень значущості 0.05)

Цей тест допомогає виявити аномалії, пов’язані з послідовностію подій у необроблених даних , і визначити , наскільки добре вони відповідають очікуванням випадкових чисел.

в) Тест частот:

Частотний тест перевіряє, чи трапляються певні значення приблизно однаково часто в послідовності псевдовипадкових чисел. Це важливо для забезпечення рівномірності розподілу чисел у великому діапазоні та перевірки того , що генератор псевдовипадкових чисел досягає рівномірного розподілу. Кроки ті самі , як у тесті на рівномірність розподілу.

г) Тест покращеної однорідності:

Модифікований тест на однорідність використовується для виявлення аномалій у частоті появи чисел і виявлення закономірностей, які можуть вказувати на відсутність випадковості в послідовності псевдовипадкових чисел. Цей тест може виявити вибіркові аномалії в розподілі, які не виявляються іншими стандартними темами.

Кроки алгоритму:

- Розбиття послідовності на блоки:
  Послідовність чисел розбивається на блоки заданого розміру
- Підрахунок частот:
  Для кожного блоку обчислюється фактична кількість входжень кожного числа.
- Очікувані частоти:
  Для кожного числа обчислюється очікувана частота , яка вважається приблизно рівною для всіх можливих чисел у діапазоні генерації.
- Визначення відхилень:
  Для кожного блоку порівнюється фактична частота з очікуваною частотою та визначається відхилення

Цей тест дозволяє виявляти аномалії , які не завжди виявляються стандартними тестами , наприклад неоднорідності розподілу, і може використовуватися як доповнення до інших методів оцінки якості генераторів псевдовипадкових чисел.

д) Тест на самоврядованість:

Тест на автокореляцію визначає, чи існують кореляції та зв’язки між числами в послідовності псевдовипадкових чисел . Цей тест допомогає визначити структурні моделі, які можуть вказувати на відсутність випадковості та повтюрюваності згенерованих чисел.

Кроки алгоритму:

- Вибір затримки:

Вибір значення затримки (зазвичай маленьке ціле число) , яке визначає , з якою кількістю позицій у послідовності чисел ці значення будуть порівнюватися

- Обчислення коефіцієнта автокореляції:
  Обчислює коефіцієнт автокореляції для вибраних значень затримки. Це може включати обчислення коефіцієнта кореляції Пірсона або іншу відповідну міру.
- Статистична оцінка:
  Оцінюється статистична значущість отриманих коефіцієнтів автокореляції. Як правило, критичні значення використовуються для визначення того, чи є результат статистично значущим.
- Порівняння з критичним значенням:
  Результат порівнюється з критичними значеннями для визначеного рівня значущості.
- Прийняття рішення:
  Якщо отриманий коефіцієнт автокореляції значний, можна підозрювати кореляцію або зв’язок у послідовності, що може вказувати на невипадкову природу генерації чисел.

Цей тест важливий для виявлення взаємозв’язків і кореляцій між числами в послідовності, оскільки вони можуть вказувати на неявні структури та шаблони генерування чисел. Випадкові послідовності повинні бути некорельованими, а виявлення кореляцій може вказувати на проблеми з якістю генератора.

е) Тест на довжину серії:

Перевірка довжини смуги перевіряє , чи послідовність псевдовипадкових чисел має смуги однакового значення, а її період не надто короткий чи надто довгий. Це важливо для виявлення порушень і визначення ступеня, до якого числа в послідовності утворюють групи ідентичних або подібних значень.

Кроки алгоритму:

- Обрання допустимої довжини серії:
  Визначення мінімальної та максимальної допустимої довжини серії для аналізу.
- Пошук серій в послідовності:
  Проходження послідовності та розпізнавання довгих послідовностей, які мають однакові чи схожі номери.
- Визначення довжини серії:
  Визначення фактичної довжини кожної знайденої серії
- Порівняння з допустимими межами:
  Порівняння фактичної довжини ряду з передбачуваними межами. Якщо довжина ряду не в допустимому діапазоні, це може свідчити про проблеми з генерацією чисел.
- Статистична оцінка:
  Результати оцінюються на предмет статистичної значущості, щоб визначити, наскільки вони відрізняються від випадковості.
- Прийняття рішення:
  Якщо фактична довжина смуг виходить за межі прийнятного діапазону, це може свідчити про нерівності та потребу у вдосконаленому генераторі псевдовипадкових чисел.

Ці тести є лише деякими з багатьох способів оцінки статистичних випадків. Для псевдовипадкових чисел важливо використовувати генератор, який успішно пройшов різні подібні тести, оскільки вони гарантують високу якість і випадковість згенерованих чисел. Криптографічні застосування , зокрема, вимагають особливо високої якості та випадковості.

1.8 Криптографічні застосування і безпека псевдовипадкових чисел

Псевдовипадкові числа відіграють ключову роль у криптографії та важливі для генерації криптографічних ключів та інших параметрів. Це стосується як симетричної, так і асиметричної криптографії, а також різноманітних

протоколів і систем захисту інформації. Нижче приведені деякі застосування псевдовипадкових чисел у криптографії:

- Генерація криптографічних ключів:
  Псевдовипадкові числа використовуються для генерації ключів у симетричних криптосистемах. Вони допомогають забезпечити непередбачуваність і випадковість ключового матеріалу, що має вирішальне значення для безпеки криптографічних алгоритмів.
- Ініціалізація векторів Ініціалізації (IV):
  У криптографії, особливо при використанні режиму блокового шифрування, використовуються вектори ініціалізації. Використання псевдовипадкових чисел для генерації непередбачуваних IV забезпечує унікальність і безпеку шифрування.
- Ключові розгортки
  Псевдовипадкові числа використовуються для генерації похідних ключів для різних криптографічних цілей , таких як генерація ключів для цифрових підписів або протоколів автентифікації.
- Стійкість до атак на основі випадковості:
  Забезпечення випадковості та непередбачуваності псевдовипадкових чисел є важливим для захисту від різних типів криптографічних атак , включаючи атаки на основі випадковості.
- Протоколи обміну ключами:
  Протоколи обміну ключами , такі як протокол Діффі-Хелмана, використовують псевдовипадкові числа для створення та обміну випадковими змінними , які використовуються для створення загального секретного ключа.
- Криптографічні хеш-функції
  В криптографічних хеш-функціях псевдовипадкові числа можна використовувати для створення “солі” , які підвищують унікальність і ускладнюють атаки на основі попередніх знань.
- Захист генераторів випадкових чисел:

Самі генераторі псевдовипадкових чисел є важливим об’єктом захисту в криптографії. Їх параметри та статус повинні бути добре захищені від атак.

Безпека псевдовипадкових чисел в криптографії та її вимоги. Нижче приведені загальні вимоги до безпеки псевдовипадкового числа:

- Непередбачуваність:
  Генератор псевдовипадкових чисел має бути непередбачуваним, тобто неможливо передбачити можливе наступне значення.
- Рівномірність розподілу:
  Числа , які генеруються, повинні рівномірно розподілятися в заданому діапазоні без виділених піддіапазонів
- Стійкість до внутрішніх і зовнішних атак:
  Генератор має бути стійким до спроб внутрішніх атак(наприклад, атак, які отримують внутрішні параметри) і зовнішніх атак(наприклад, атак, які використовують статистичні властивості числових послідовностей).
- Нестабільність при змінах умов:
  Зміни в умовах роботи генератора, такі як зміна початкових умов або параметрів, повинні викликати значні зміни в згенерованих числах.
- Відсутність кореляцій і патернів:
  У послідовності чисел не повинно бути жодних кореляцій або закономірностей , які можна використовувати для прогнозування майбутніх значень.
- Безпека при використанні в криптографічних протоколах:
  Генератори повинні відповідати вимогам безпеки певних криптографічних протоколів , включаючи стандарти та вимоги до обміну ключами, цифрових підписів та інших криптографічних операцій.
- Стійкість до криптоаналізу:

Генератор повинен бути стійким до різних типів криптоаналізу, включаючи атаки на основі внутрішньої структруи, лінійного та диференціального криптоаналізу.

Вимоги до безпеки генераторів псевдовипадкових чисел визначаються потребами криптографічних систем і протоколів та їх стійкістю до різних типів атак і криптоаналізу. Дефекти генераторів можуть призвести до серйозних уразливостей у криптографічних програмах.

Роль вбудованих генераторів у мовах програмування і операційних системах. Нижче буде приведено роль у деяких напрямках:

- Випадкові числа в мовах програмування:

  Генератори , вбудовані в мови програмування (наприклад, random) використовуються для генерації псевдовипадкових чисел у програмах. Це важливо для широкого спектру програм, включаючи ігри, симуляції, криптографічні операції та тестування програм.
- Генерація випадкових ключів

  У криптографії вбудовані генератори використовуються для генерації випадкових ключів, які використовуються в симетричній та асиметричній криптографії. Надійність цих ключів залежить від якості вбудованого генератора
- Моделювання та симуляція

  У великих системах моделювання та симуляції (наприклад, фізичні експерименти, фінансові моделі) вбудовані генератори використовуються для створення стохастичних аспектів для відтворення випадкових подій.
- Тестування програм

  Генератори випадкових чисел корисні для тестування додатків, які потребують генерації різних випадкових вхідних даних для тестування продуктивності системи за різних умов.
- Алгоритми машинного навчання

У деяких алгоритмах машинного навчання випадкові числа використовуються для ініціалізації ваг і параметрів моделі та генерування випадкових зміщень під час навчання

- Операційні системи

  Вбудовані генератори в операційній системі використовуються для генерації випадкових значень, які відповідають різним потребам, таким як надійність або конфіденційність серед інших процесів.

- Інтерфейси користувача

  Деякі інтерфейси користувача використовують випадкові числа для створення унікальних ідентифікаторів, маркерів автентифікації або випадкових послідовностей для користувачів.

- Ігрова розробка

  У розробці ігор випадкові числа використовуються для створення різних гравців, об’єктів або ситуацій, щоб забезпечити різноманітність і інтерес до гри.

Усі ці додатки демонструють важливу роль вбудованих генераторів у середовищах програмування, що забезпечують випадкові та непередбачувані елементи в різних областях інформатики та програмування.

Переваги та обмеження використання вбудованих засобів. Нижче буде приведені деякі переваги та обмеження таких засобів:

- Простота використання

  Вбудовані генератори, як правило, прості у використанні та не вимагають додаткового налаштування чи конфігурації

- Інтеграція з мовою програмування

  Вони безпосередньо інтегровані з мовами програмування, що дозволяє легко використовувати їх у різних програмах і аспектах

- Широке застосування

  Вони використовуються в різних сферах, включаючи криптографію, розробку ігор, статистику, тестування програмного забезпечення тощо.

- Висока швидкість

  Вбудовані генератори , як правило, працюють дуже швидко, що дозволяє використовувати практичні програми у реальному часі.
- Легка доступність

  Немає необхідності завантажувати додаткові бібліотеки чи інструменти, оскільки вони вже включені в мову програмування чи операційну систему.

Обмеження:

- Низька випадковість

  Вбудовані генератори можуть мати обмежену періодичність або непередбачуваність , що робить їх менш придатними для певних криптографічних програм.
- Недостатня стійкість

  У деяких випадках вбудовані генератори можуть виявляти шаблони або кореляції , що робить їх менш стійкими до криптоаналізу.
- Обмеженість функціональності

  Певну функціональність (наприклад, генерування послідовностей із певним розподілом) може бути важко досягти за допомгою вбудованих генераторів.
- Залежність від мови програмування

  Властивості та якість вбудованих генераторів можуть відрізнятися від мови програмування до мови програмування , що обмежує їх переносимість між різними середовищами.
- Відсутність розширеного контролю

  Вбудовані генератори можуть обмежити доступний для програміста рівень керування параметрами генерації.
- Відсутність криптографічної гарантії

Деякі вбудовані генератори можуть бути менш придатними для криптографічних програм через відсутність гарантій належних криптографічних властивостей.

Враховуючи ці переваги та обмеження, програмісти повинні ретельно обирати між вбудованими генераторами та іншими рішеннями на основі конкретних потреб їх застосування.

Приклади реальних використань псевдовипадкових чисел:

- Криптографія

  Генерувати ключі для симетричних і асиметричних алгоритмів шифрування, генерувати випадкові вектори ініціалізації та солі, генерувати випадкові числа в протоколах автентифікації та підпису.
- Ігрова розробка

  Створювання випадкових ігрових сцен, положення об’єктів у віртуальному світі та генеруйте випадкові події та результати, щоб забезпечити різноманітність ігрового процесу.
- Симуляції та моделювання

  Моделюйте експерименти у фізичних процесах, науці та інженерії для створення стохастичних сценаріїв для аналізу систем і вирішення проблем реального світу.
- Тестування програм

  Генерування тестових даних для перевірки роботами програми та генерування випадкових вхідних даних для аналізу реакції системи на різні сценарії.
- Інтернет-безпека

  Використання псевдовипадкових чисел при генерації токенів сесії, ключів доступу та генерації випадкових значень ускладнює атаки автентифікації та авторизації
- Фінанси та бізнес

Випадкові числа використовуються для створення унікальних ідентифікаторів транзакцій і генерування випадкових параметрів для фінансових моделей і аналізу даних.

- Генетичний та алгоритмічний дизайн

  Випадкові числа використовуються в генетичних алгоритмах для еволюції параметрів для створення випадкових початкових умов для оптимиізації.

- Маркетинг та аналітика

  Створюйте випадкові вибірки для опитувань і маркетингових досліджень, а також створюйте різні сценарії для аналізу ефективності маркетингових стратегій

- Біометрика

  Генерування випадкових шаблонів для біометричних систем, генерувати випадкові значення для тестування алгоритмів розпізнавання обличчя.

2 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ

2.1 HMAC_DRBG

HMAC_DRBG – це генератор псевдовипадкових чисел, розроблений на основі HMAC. Це описано в документі NIST SP 800-90A “Рекомендації щодо генерації випадкових чисел за допомогою детермінованих генераторів випадкових бітів” . Розглянемо докладний опис алгоритму:

- Ініціалізація:

  HMAC_DRBG приймає як вхідні дані вектор ініціалізації (початкове число) і додаткові входи , які можна використовувати для додаткового посилення безпеки. Ініціалізація передбачає обчислення внутрішнього ключа за допомогою HMAC на основі початкового числа вхідних даних.
- Актуалізація

  HMAC_DRBG можна оновити (перезаповнити) , щоб змінити внутрішній ключ і почати генерувати нову послідовність псевдовипадкових чисел. Як правило , оновлення виконуються через певний час або після того , як було згенеровано певну кількість бітів.
- Генерація

  Генерація передбачає використання HMAC для обчислення нових блоків псевдовипадкових бітів. Внутрішній ключ використовується для визначення ключа HMAC. Отримані біти виводяться як псевдовипадкова послідовність.
- Контекст

  HMAC_DRBG зберігає внутрішній стан (включаючи внутрішні ключі) для забезпечення детермінізму та надійності. Збереження контексту дозволяє при необхідності відновити стан генератора.
- Безпека

  HMAC_DRBG розроблено відповідно до високих стандартів безпеки для криптографічних генераторів псевдовипадкових чисел. Стабільність

згенерованих бітів залежить від стабільності функціональності HMAC і правильного вибору внутрішнього ключа.

- Додатковий ввід

  HMAC_DRBG може приймати додаткові входи для покращеної стабільності генератора. Для підвищення безпеки в процес генерації можна включити додаткові дані.

Базові компоненти алгоритма та механізм генерації псевдовипадкових чисел:

- Ключі та параметри

  HMAC_DRBG використовує три ключі:

  Key: Ключ для функції HMAC

  V : Внутрішній стан генератора

  C : Змішаний постійний зв’язок

  Значення цих ключів ініціалізуються під час запуску генератора

- Ініціалізація

  Генератори ініціалізуютсья вхідними параметрами, такими як початкові значення та додаткові входи. Ключ і V ініціалізуються за допомогою функції HMAC із початковим значення і значенням C

- Генерація псевдовипадкових біт

  Генерація передбачає повторення кроків HMAC для створення нових блоків псевдовипадкових бітів. Вхідні дані для кожного виклику HMAC складаються з ключа та попереднього внутрішнього стану

  Псевдовипадкових біт отримується з вихідного значення HMAC.

- Актуалізація

  Генератори можна оновлювати (перезадавати) після того, як буде згенеровано певну кількість бітів або через певний проміжок часу. Оновлення передбачають зміну початкового значення та повторення ініціалізації.

- Безпека та резистентність до атак:

Дуже важливо зберігати ключ і V в секреті. Безпека генератора значною мірою залежить від надійності HMAC і правильної реалізації ключів і параметрів.

- Додатковий ввід

  Для підвищення безпеки генератора можуть бути введені додаткові входи. Використання функції HMAC , поєднує цей додаткових вхід.

- Контекст та відновлення

  Генератори можуть зберігати контекст, щоб його можна було відновити в майбутньому для продовження створення. Збереження внутрішніх ключів генератора та іншого стану гарантує детермінованість і стабільність.

HMAC_DRBG – це стандарт NIST для генерації псевдовипадкових чисел у криптографічних програмах. Точні значення параметрів і додаткові властивості можуть відрізнятися в залежності від конкретної реалізації і стандартів.

Використання еліптичних кривих у HMAC_DRBG пропонує кілька переваг завдяки своїм унікальним математичним і криптографічним властивостям. Ось деякі можливі переваги:

- Ефективність у великих розмірах

  Еліптичні криві дозволяють досягти такого ж рівня безпеки, використовуючи ключі меншого розміру порівняно з традиційними криптосистемами. Це особливо важливо для HMAC_DRBG, оскільки менші розміри ключа можуть призвести до більш ефективної генерації псевдовипадкових бітів.

- Обчислювальні витрати

  Операції на еліптичних кривих обчислювально дешевші порівняно з іншими математичними структурами. Це зменшує втрати на обчислювальні ресурси при генерації псевдовипадкових чисел.

- Стійкість до деяких атак

Властивості еліптичних кривих можуть забезпечити певну стійкість до атак, таких як атаки факторизації, які можна застосувати до інших криптографічних схем.

- Підтримка ефективного генерування ключів:
  Генерація ключів на еліптичних кривих менш витратна , що важливо для HMAC_DRBG, оскільки повторне заповнення або інші події можуть спричинити часті зміни ключів.

Недоліки:

- Специфічність вибору кривої:
  Важливим аспектом є вибір конкретної еліптичної кривої . Неправильний вибір може призвести до вразливості. Важливо підібрати еліптичну криву, яка відповідає сучасним стандартам безпеки.
- Можливі вразливості
  Еліптичні криві можуть бути вразливими , якщо їх вибрати неправильно або якщо присутні нестандартні вектори атаки.
- Необхідність криптографічних експертів
  Робота з еліптичними кривими вимагає глибокого розуміння криптографії, що може бути складним для менш досвідчених розробників.

Важливо порівнювати HMAC_DRBG з іншими генераторами випадкових чисел враховуючи різні критерії, такі як безпека, ефективність, стійкість до атак, обчислювальна вартість та інші фактори . Ось кілька порівнянь:

- Стійкість:
  HMAC_DRBG – Використовує HMAC на еліптичних кривих , який є стійким до багатьох криптографічних атак
  Інши генератори – залежать від конкретного алгоритму. Якщо вони використовують стандартні криптографічні функції, такі як SHA-256 , то також можуть бути стійкими
- Ефективність:

HMAC_DRBG – Залежно від вибору еліптичної кривої, це може бути ефективнішим , ніж деякі альтернативи , оскільки для досягнення того самого рівня безпеки потрібно менше бітів.

- Витрати обчислень

  Залежить від обчислювальної складності еліптичних кривих , але вони можуть бути меншими порівняно з іншими криптографічними схемами.

- Безпека генерації ключів

  Він використовується для генерації ключів шифрування, і вибір еліптичної кривої може вплинути на безпеку генерації ключа

Загальна оцінка вибору між HMAC_DRBG та іншими генераторами залежить від конкретних вимог безпеки та середовища використання. Важливо зазначити, що ці аспекти можуть змінюватися з часом, тому слід дотримуватися чинних стандартів безпеки.

2.2 DUAL_EC_DRBG

DUAL_EC_DRBG – це алгоритм генерації псевдовипадкових бітів із використанням еліптичних кривих. Нижче наведемо опис алгоритму та його основних компонентів:

- DUAL_EC_DRBG використовує дві еліптичні криві, позначені як $E\ (\ F_p\ )$, де p – просте число. Вибір кривої контролюється константами, які є частиною специфікації алгоритму. Однак ці константи можуть стати потенційними вразливими місцями , якщо їх не вибрати ретельно.
- Генерація ключів :

  Алгоритм Dual_EC_DRBG можна використовувати для генерації ключів шифрування. Ініціалізація передбачає вибір початкової точки на еліптичній кривій, яка визначає інший параметр.

- Обчислення вихідних бітів:

  Вихідні біти генеруються шляхом змішування (множення) точок на еліптичній кривій.

- Вектор ініціалізації та актуалізація:

DUAL_EC_DRBG може бути ініціалізований вектором ініціалізації , а також піддаватися актуалізації для зміни свого внутрішнього стану.

- Проблема бекдору

  Основним недоліком DUAL_EC_DRBG є теоретична можливість використання бекдорів при виборі параметрів кривої , що може призвести до атак , що порушують безпеку генератора.

Важливо підкреслити , що використання цього алгоритму у зашифрованих системах не рекомендується через виявлення значних проблем безпеки. Розробникам криптографічних систем слід уникати використання цього алгоритму та обирати більш безпечні альтернативи.

DUAL_EC_DRBG визначається параметрами еліптичних кривих та іншими константами. Ось огляд параметрів та їх впливу на безпеку генерації випадкових чисел:

- Параметри еліптичної кривої:

  P (просте число) – визначає порядок еліптичної кривої , тобто кількість точок на криві.

  a , b (коефіцієнти) – визначають конкретну форму еліптичної кривої відповідно до рівняння .

  G (генератор) – початкова точка на еліптичній кривій , яка використовується для генерації псевдовипадкових чисел

  n (порядок генератора) – визначає порядок точки G , тобто найменше ціле число , при якому $nG = O$ (нейтральний елемент)

- Вплив параметрів на безпеку:

  Довжина ключа – для безпеки генерації випадкових чисел наявність достатньо великого порядку n є дуже важливою , оскільки це визначає безпеку від атак на основі дискретного логарифму.

  Вибір коефіцієнтів a і b – неконтрольований вибір цих параметрів може поставити під загрозу безпеку та може бути використаний для створення слабких місць у кривих.

Вибір генератора G – неправильній вибір може призвести до проблем із безпекою та підозри бекдору.

Вибір констант – якщо вибрані константи вибрані заздалегідь та атакуються, можливій вибір параметрів з більшим знаменником ніж чисельником, що може створити слабкі місця на кривій.

Загальний вплив параметрів на безпеку полягає в їх правильному створенні та виборі. Неправильний вибір може призвести до вразливості та створити можливості для атак . Враховуючи сумніви щодо DUAL_EC_DRBG, рекомендується використовувати інші більш безпечні генератори псевдовипадкових чисел у криптосистемах.

Огляд можливих атак на DUAL_EC_DRBG. DUAL_EC_DRBG критикували за потенційну вразливість і атаки. Нижче наведено огляд можливих атак на DUAL_EC_DRBG:

- Backdoor Attack:

  Ця атака передбачає ймовірне включення бекдору в алгоритм. Іншими словами, якщо зловмисник має спеціальні параметри, він може передбачити випадкові числа, згенеровані алгоритмом. Якщо атака успішна, зловмисник може передбачити всі випадкові числа, згенеровані DUAL_EC_DRBG, що спричинить серйозні проблеми з безпекою.
- Parameter Choice Attack:

  Якщо параметри еліптичної кривої (наприклад , точки G , P , a , b) попередньо вибрані та контрольовані зловмисником, на кривій можуть бути створені слабкі місця , які можна використовувати для атак. Атаки, які використовують слабкі місця, можуть підірвати безпеку генерації випадкових чисел.
- Reduction Attack:

  Цей вид атаки стосується здатності відновити розв’язок задачі дискретного логарифмування до розв’язку інших задач. Якщо атака буде

успішною, це може значно полегшити зловмиснику завдання вирішення складних обчислювальних задач.

- Statistical Attack:

  Атаки , засновані на статистичних властивостях згенерованих випадкових потоків, можна використовувати для виявлення слабких місць у випадковому виході алгоритму. Статистичні атаки можуть допомогти зловмисникам зрозуміти та використати властивості випадкового результату Dual_EC_DRBG.

Враховуючи проблеми безпеки DUAL_EC_DRBG , важливо порівняти DUAL_EC_DRBG з іншими поточними генераторами псевдовипадкових чисел. У порівнянні з безпечними та ефективними генераторами, DUAL_EC_DRBG, як правило , є менш популярним і небезпечним вибором через історію вразливостей. Порівняємо його із загальними вимогами та іншими відомими генераторами:

- Безпека та вразливості:

  Можливість вищезазначених проблем з бекдором і вразливості вибору параметрів ставить під сумнів його безпеку. Багато сучасних генераторів , як HMAC_DRBG , Fortuna або генератори на основі хешу мають потужні функції безпеки та не страждають від тих самих проблем , що й DUAL_EC_DRBG.

- Швидкість та ефективність:

  Через використання еліптичних кривих він може бути менш ефективним , ніж інші алгоритми . Інші алгоритми , такі як HMAC_DRBG або Fortuna, можуть забезпечити вищу продуктивність з точки зору швидкості та використання ресурсів.

- Стійкість до криптоаналізу:

  Через теоретичну можливість бекдор-атак і вибір параметрів, його надійність була поставлена під сумнів, і тестування та аналіз широко використовуються для забезпечення надійності проти різних атак.

Загалом DUAL_EC_DRBG не є рекомендованим вибором через проблеми з безпекою та застарілість. Багато інших генераторів (таких як HMAC_DRBG або CTR_DRBG) є більш безпечними та ефективними альтернативами, які використовуються в криптографічних програмах.

2.3 Метод Fortuna

Fortuna – це генератор псевдовипадкових чисел , розроблений Брюсом Шнайєром і Нілом Фергюсоном. Метод Fortuna базується на ідеях багатошаровості та гнучкості , що забезпечує високий ступінь безпеки та випадковості. Основні принципи Fortuna включають:

- Багатошаровість:
  Fortuna використовує кілька незалежних пулів псевдовипадкових чисел. Кожен пул працює незалежно та генерує власний вихідний потік.
- Резервуари та Хеш-Функції:
  Використовуйте репозиторій для зберігання ентропії та додаткових вхідних значень. Хеш-функції використовуються для хешування та вирішення колізій.
- Reseeding:
  Переосвіження використовується для регулярного оновлення ентропії для запобігання атакам.
- Синхронізація та оновлення:
  Механізм синхронізації та оновлення забезпечує безпеку та стабільність генерації номерів. Розмір фрагментів можна динамічно змінювати від ситуації та потреб.

Огляд роботи Fortuna. Кожен пул генерує певну кількість байтів, які потім об’єднуються у вихідний потік. Резервуари дозволяють зберігати ентропію, яка потім використовується для оновлення пулів. Хеш-функції використовуються для збурення ентропії та додавання нових значень. Переосвіження відбувається періодично, щоб уникнути витоку ентропії та підвищити стійкість.

Основні характеристики Fortuna. Кожен пул ПВЧ у Fortuna має свою роль та робочий принцип:

- Перший пул (Pool-0):
  Використовується для збору вхідних значень, таких як дані миші, або натискання клавіш, час події тощо. Надає базову ентропію, яка використовується для створення вихідного потоку.
- Другий пул (Pool-1):
  Використовується для випадкової вибірки даних, таких як таймери виконання. Забезпечує другий рівень безпеки та додаткову ентропію.
- Третій пул (Pool-2):
  Використовується для даних, створених розширеними алгоритмами, наприклад алгоритмами шифрування. Забезпечує третій рівень безпеки та доповнює вихідний потік

Роль кожного пула в генерації псевдовипадкового вихідного потоку:

- Кожен пул накопичує власний внутрішній стан і генерує частковий вихідний потік.
- Резервуари використовуються для зберігання великої кількості ентропії та стабілізації вихідного потоку.
- Вихідні потоки з різних пулів і резервуарів об’єднуються для формування кінцевого вихідного потоку.

Процес генерації псевдовипадкового вихідного потоку:

- Збір ентропії:
  Кожен пул ПВЧ збирає вхідні значення та накопичує ентропію.
- Генерація часткових потоків:
  Кожен пул генерує свою частину вихідного потоку на основі накопиченої ентропії.
- Комбінування та фінальний вихід:

Вихідні потоки кожного пулу об’єднуються та передаються через хеш-функцію. Резервуари використовуються для збільшення ентропії та забезпечення стабільності.

Fortuna створює надзвичайно випадкові вихідні потоки та враховує ентропію з різних джерел, забезпечуючи високий рівень безпеки та випадковості для додатків у криптографії та інших областях.

Fortuna виявляється ефективним і безпечним для застосувань в криптографії з наступними перевагами:

- Висока випадковість:
  Fortuna використовує багаторівневий підхід і збирає ентропію з різних джерел, що забезпечує високий ступінь випадковості у згенерованому вихідному потоці.
- Безпека в криптографії:
  Використання хеш-функцій і репозиторіїв дозволяє протистояти ентропійним атакам і забезпечувати стабільність використання пароля.
- Багатошаровість та гнучкість:
  Fortuna працює, полегшуючи додавання нових джерел ентропії та динамічне керування розмірами блоків для адаптації до різних умов.
- Стійкість до сучасних загроз:
  Fortuna захищає від сучасних атак, таких як ентропійні атаки та атаки на ключі, що робить його чудовим вибором для криптографічним програм.

2.4 Порівняльний аналіз методів

Порівняльний аналіз методів HMAC_DRBG та CTR_DRBG може бути наступним:

- Безпека:
  Обидва методи забезпечують високий рівень безпеки , але вибір між ними може залежити від конкретних вимог програми
- Ефективність:

CTR_DRBG може бути більш ефективним у використанні ресурсів порівняно з HMAC_DRBG, оскільки він використовує блочний шифр для генерації бітів

Нижче представлена таблиця тестів , які допомагають ліпше зрозуміти ці два методи:

Таблиця 2.1 – Тести алгоритмів

Назва тесту	**HMAC_DRBG**	**CTR_DRBG**
Швидкодія	Шифрування : 239200 нс Генерація хеш-функції : 20554 нс Дешифрування : 15061 нс	Шифрування : 43733 нс Дешифрування : 3670 нс
Безпека (атака BruttForce)	Атака без успіху	Атака без успіху
Безпека (атака Келсі)	Атака без успіху	Атака з частковим успіхом
Безпека (атака інтерполяції) **Тест випадковості (тест частоти бітів)**	Атака без успіху 10000 незалежних бітів (пройдено) 50000 незалежних бітів(пройдено)	Атака без успіху 10000 незалежних бітів(пройдено) 50000 незалежних бітів(не пройдено)

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вибір технологій та інструментів

У якості мови програмування для свого проекту я обрав C# . Нижче я виділяю декілька аспектів , які відповідають специфиці теми дослідження:

- Об’єктивно-орієнтоване програмування(ООП):
  C# - це мова,яка повністю підтримує парадигму об’єктивно-орієнтованого програмування. Еліптичні криві можути бути ефективно змодельовані та представлені об’єктами. Об’єктно-орієнтовані концепції, такі як успадкування, інкапсуляція та поліформізм, допомогають зробити програмування більш зрозумілим і структурованим.
- Математичні операції та обчислення:
  Синтаксис C# дозволяє легко виражати математичні формули для еліптичних кривих. Такі оператори, як “+” , “-“ , “*” і “/” , забезпечують чітку та зрозуміла мову для вираження алгоритмів, пов’язаних з еліптичними кривими та псевдовипадковими числами.
- Бібліотеки та фреймворки .NET:
  Мова C# інтегрована в платформу .NET, надаючи доступ до різноманітних бібліотек і фреймворків. Наявність математичних бібліотек, криптографічних інструментів і різноманітних алгоритмів полегшила впровадження та дослідження , пов’язані з еліптичними кривими.
- Вбудована підтримка генерації псевдовипадкових чисел:
  Мова C# надає вбудовані методи для генерації псевдовипадкових чисел, що робить їх використання на еліптичних кривих зручним. Це може бути важливим аспектом дослідження та впровадження криптографічних алгоритмів з використанням псевдовипадкових чисел.
- Налагоджуваність та інтерактивний режим розробки:
  Використання середовища розробки Visual Studio дозволяє швидко налагоджувати код, що важливо при розробці та тестуванні алгоритмів на

еліптичних кривих. Можливість використовувати інтерактивні інструменти для експериментів з кодом також корисна для дослідницької роботи.

Ці характеристики мови C# роблять її придатним вибором для вивчення псевдовипадкових чисел на еліптичних кривих і розробки відповідних криптографічних алгоритмів.

Значний вплив на реалізацію та дослідження теми “Псевдовипадкові числа на еліптичних кривих” мала інтеграція з платформою .NET . Різні аспекти, які доводять важливість обраної платформи для цієї теми , обговорюються нижче:

- Підтримка еліптичних кривих в .NET:
  .NET Framework забезпечує вбудовану підтримку для роботи з еліптичними кривими. Такі , як ECDSA, дозволяють виконувати криптографічні операції з використанням еліптичних кривих. Обрана мова C# інтегрована в платформу, що дозволяє легко використовувати ці функції.
- Безпека та криптографія:
  Платформа .NET надає широкі можливості для криптографічних операцій, особливо для роботи з псевдовипадковими числами на еліптичних кривих. Вбудовані бібліотеки платформи дозволяють безпечно з криптоалгоритмами та забезпечують високий рівень безпеки досліджуваних об’єктів.
- .NET Standard та кросплатформеність:
  Використання .NET Standard забезпечує кросплатформну сумісність вашої реалізації . Це означає , що код, написаний на C# для .NET Framework , може працювати на різних платформах, включаючи Windows, Linux і macOS. Це важливо для забезпечення доступності та використання ваших досліджень.
- Високий рівень інтеграції з Visual Studio:

Visual Studio, інтегроване середовище розробки для мови C# і платформи .NET , надає розширені можливості для розробки , тестування та налагодження коду. Це дуже важливо при реалізації складних криптографічних алгоритмів і експериментів з псевдовипадковими числами на еліптичних кривих.

- Можливості оптимізації та продуктивності:
  Використання .NET Framework дозволяє впроваджувати оптимізацію для підвищення продуктивності коду. Застосування відомих оптимізацій має вирішальне значення для ефективної генерації псевдовипадкових чисел на еліптичних кривих.

Обрана платформа .NET у поєднанні з мовою C# пропонує багато переваг, які допомогають зробити реалізацію та дослідження теми “Псевдовипадкові числа на еліптичних кривих” більш ефективними, безпечними та доступними

3.2 Реалізація передачі повідомлення з використанням HMAC_DRBG

У цій програмній реалізації я відзначив наступні кроки:

- Генерація ключів:

```
Byte[] senderKey = GenerateKey();
Byte[] recipientKey = GenerateKey();
```

У цих строках коду створюються випадкові ключі для відправника та отримувача з використанням методу GenerateKey() .

```
Static byte[] GenerateKey()
{
  Using (var rng = new RNGCryptoServiceProvider())
  {
       Byte[]key = new byte[32];
       Rng.GetBytes(key);
       Return key;
  }
}
```

GenerateKey() використовує ‘RNGCryptoServiceProvider’ для генерації криптографічного безпечного випадкового ключа довжиною 32 біт.

- Шифрування повідомлення

```
var (ciphertext, tag) = Encrypt(messageToSend, recipientKey);
```

Відправка повідомлення шифруеться з використанням ключа отримувача і повертається у вигляді шифротексту з тегом.

```
static (byte[], byte[]) Encrypt(string message, byte[] key)
{
 using (Aes aesAlg = Aes.Create())
 {
     aesAlg.Key = key;
     aesAlg.GenerateIV();

     using (var encryptor = aesAlg.CreateEncryptor())
     using (var msEncrypt = new MemoryStream())
     {
               using       (var       csEncrypt       =       new
  CryptoStream(msEncrypt, encryptor, CryptoStreamMode.Write))
         using (var swEncrypt = new StreamWriter(csEncrypt))
         {
             swEncrypt.Write(message);
         }

         return (msEncrypt.ToArray(), aesAlg.IV);
     }
 }
}
```

Encrypt() використовує алгоритм AES для шифрування повідомлення. IV(вектор ініціалізації) генерується автоматично та додається до шифротексту.

- Генерація та перевірка HMAC

```
byte[] hmacKey = GenerateKey();
byte[] hmacValue = GenerateHMAC(hmacKey, messageToSend);
```

У цій ділянці коду генерується випадковий ключ для HMAC та обчислюється HMAC для повідомлення

```
static byte[] GenerateHMAC(byte[] key, string data)
{
 using (var hmac = new HMACSHA256(key))
 {
     return hmac.ComputeHash(Encoding.UTF8.GetBytes(data));
 }
  }
```

GenerateHMAC() використовує HMAC-SHA256 для обчислення коду аутентичності повідомлення з використанням ключа HMAC

- Перевірка цілосності повідомлення з використанням HMAC

```
static bool CompareByteArrays(byte[] array1, byte[] array2)
{
    if (array1.Length != array2.Length)
        return false;

    for (int i = 0; i < array1.Length; i++)
    {
        if (array1[i] != array2[i])
            return false;
    }

    return true;
}
```

Тут порівнюються обчислювальний HMAC з очікуваним HMAC для перевірки цілісності повідомлення . CompareByteArrays() використовується для безпечного порівняння двух массивів байтів.

- Дешифрування повідомлення

```
static string Decrypt(byte[] ciphertext, byte[] tag, byte[]
key)
{
    using (Aes aesAlg = Aes.Create())
    {
        aesAlg.Key = key;
        aesAlg.IV = tag;

        using (var decryptor = aesAlg.CreateDecryptor())
        using (var msDecrypt = new MemoryStream(ciphertext))
        using (var csDecrypt = new CryptoStream(msDecrypt,
decryptor, CryptoStreamMode.Read))
        using (var srDecrypt = new StreamReader(csDecrypt))
        {
            return srDecrypt.ReadToEnd();
        }
    }
}
```

Шифротекст дешифрується з використанням ключа отримувача та повертається дешифроване повідомлення. Decrypt() використовує алгоритм AES задля дешифрування повідомлення з використанням ключа та вектора ініціалізації(IV).

- Результати

```
Console.WriteLine($"Отправитель:
{BitConverter.ToString(senderKey)}");
Console.WriteLine($"Получатель:
{BitConverter.ToString(recipientKey)}");
Console.WriteLine($"Шифрованное сообщение:
{Convert.ToBase64String(ciphertext)}");
Console.WriteLine($"HMAC:
{Convert.ToBase64String(hmacValue)}");
Console.WriteLine($"Расшифрованное сообщение:
{decryptedMessage}");
```

У цій ділянці коду виводяться результати наших операцій , такі як ключі відправника та отримувача , шифротекст, HMAC , та дешифроване повідомлення .

3.3 Реалізація передачі повідомлення за допомогою CTR_DRBG

У цьому методі я відзначив наступні кроки:

- Генерація ключа та вектора ініціалізації

```
byte[] key = GenerateRandomBytes(16); // 128 біт
byte[] iv = GenerateRandomBytes(16); // 128 біт
```

Тут генеруються та створюються випадкові послідовності байтів для ключа (128 біт) та вектора ініціалізації для використання у алгоритмі шифрування AES у режимі CTR(Counter Mode)

- Шифрування повідомлення

```
string encryptedMessage = EncryptMessage(messageToSend, key,
iv);
```

Вихідне повідомлення шифрується за допомогою алгоритма AES у режимі CTR (Counter Mode) , використовує сгенерований ключ та вектор ініціалізації

```
static string EncryptMessage(string message, byte[] key, byte[]
iv)
{
 IBufferedCipher                    cipher                    =
CipherUtilities.GetCipher("AES/CTR/NoPadding");
 cipher.Init(true,                                          new
ParametersWithIV(ParameterUtilities.CreateKeyParameter("AES",
key), iv));

 byte[] messageBytes = Encoding.UTF8.GetBytes(message);
 byte[] encryptedBytes = cipher.DoFinal(messageBytes);
```

```
    return Convert.ToBase64String(encryptedBytes);
}
```

- Дешифрування повідомлення

```
string decryptedMessage = DecryptMessage(encryptedMessage,
key, iv);
```

Зашифроване повідомлення дешифруеться за допомогою алгоритма AES у режимі CTR (Counter Mode) та використанням того ж ключа та IV.

Ця програма демонструє процес шифрування та дешифрування повідомлення з використанням AES у режимі CTR та генерації ключа та IV з використанням генератора випадкових чисел.

ВИСНОВКИ

Висновки щодо моєї роботи “Псевдовипадкові числа на еліптичних кривих” можна сформулювати наступним чином:

- Огляд теми:

  Ця робота містить огляд псевдовипадкових чисел на еліптичних кривих , важливої області криптографії.

  Дослідження включає аналіз основних властивостей еліптичних кривих та їх використання в генерації псевдовипадкових чисел.
- Програмна реалізація:

  Створена програма використовує алгоритми HMAC_DRBG і CTR_DRBG для генерації псевдовипадкових чисел.

  У цьому випадку використання еліптичних кривих забезпечує високий ступінь безпеки генерації випадкових значень.
- Функціональність програми:

  Ця програма демонструє здатність генерувати псевдовипадкові числа за допомогою вибраного алгоритму. Він також містить можливість надсилати повідомлення, шифрувати, дешифрувати та перевіряти цілісність , розширюючи функціональні можливості та покращуючи додатки еліптичної кривої.
- Перспективи та дальший розвиток:

  Вказано можливості подальшого розвитку програми , включаючи оптимізацію продуктивності та розширення функціональності.
- Висновки:

  Ця робота успішно аналізує , вивчає та реалізує генерацію псевдовипадкових чисел на еліптичних кривих за допомогою HMAC_DRBG та CTR_DRBG. Результати демонструють можливість ефективного використання еліптичних кривих у криптографічних програмах.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

Книга :

1. Сингх С. Книга шифрів. Історія шифрів та їх дешифровки. М. Аст, Астрель, 2006. 447с.

2. О.Н.Жданов. Еліптичні криві : Основи теорії та криптографічні прилади. 2013 р . 200с.

3. Бессалов А.В. Еліптичні криві у формі Єдвардса і криптографія. 2017р. 266с.

Стаття та тези:

- **2-4 автори:**

4. О.І.Гарасимчук. В.М.Максимович . Генератори псевдовипадкових чисел , їх застосування , класифікація , основні методи побудови і оцінка якості // Науково – технічний журнал “Захист інформації” №3, 2003р.

5. Микола Карпінський , Ігор Васильцов , Ігор Якименко , Ярослав Кінах . Метод генерування параметрів еліптичних кривих. // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні, вип.6 2003 р.

6. Даниленко Д.А. , Іванчук Ю.Р. Оптимізація еліптичних кривих // Вісник Київського національного університету імені Тараса Шевченка Серія фізико-математичні науки. 2014р.

ДОДАТОК А

Програмний код алгоритму HMAC_DRBG

```
using System;
using System.Diagnostics;
using System.IO;
using System.Security.Cryptography;
using System.Text;

class Program
{
    static void Main()
    {
        byte[] senderKey = GenerateKey();
        byte[] recipientKey = GenerateKey();

        string messageToSend = "Доброго дня!";


        var encryptStopwatch = Stopwatch.StartNew();
        var (ciphertext, tag) = Encrypt(messageToSend, recipientKey);
        encryptStopwatch.Stop();

        byte[] hmacKey = GenerateKey();


        var hmacStopwatch = Stopwatch.StartNew();
        byte[] hmacValue = GenerateHMAC(hmacKey, messageToSend);
        hmacStopwatch.Stop();

        if (CompareByteArrays(hmacValue, GenerateHMAC(hmacKey, messageToSend)))
        {
            Console.WriteLine("Цілісність повідомлення підтверджена");
        }
        else
        {
            Console.WriteLine("Можливе змінення повідомлення");
        }


        var decryptStopwatch = Stopwatch.StartNew();
        string decryptedMessage = Decrypt(ciphertext, tag, recipientKey);
        decryptStopwatch.Stop();

        Console.WriteLine($"Відправник: {BitConverter.ToString(senderKey)}");
        Console.WriteLine($"Отримувач: {BitConverter.ToString(recipientKey)}");
        Console.WriteLine($"Зашифроване                                    повідомлення:
{Convert.ToBase64String(ciphertext)}");
        Console.WriteLine($"HMAC: {Convert.ToBase64String(hmacValue)}");
        Console.WriteLine($"Розшифроване повідомлення: {decryptedMessage}");


        Console.WriteLine($"Шифрування: {encryptStopwatch.Elapsed} мс");
        Console.WriteLine($"Генерація HMAC: {hmacStopwatch.Elapsed} мс");
        Console.WriteLine($"Дешифрування: {decryptStopwatch.Elapsed} мс");
    }

    static byte[] GenerateKey()
    {
        using (var rng = new RNGCryptoServiceProvider())
        {
            byte[] key = new byte[32];
            rng.GetBytes(key);
```

```
            return key;
        }
    }

    static (byte[], byte[]) Encrypt(string message, byte[] key)
    {
        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = key;
            aesAlg.GenerateIV();

            using (var encryptor = aesAlg.CreateEncryptor())
            using (var msEncrypt = new MemoryStream())
            {
                using (var csEncrypt = new CryptoStream(msEncrypt, encryptor,
CryptoStreamMode.Write))
                using (var swEncrypt = new StreamWriter(csEncrypt))
                {
                    swEncrypt.Write(message);
                }

                return (msEncrypt.ToArray(), aesAlg.IV);
            }
        }
    }

    static string Decrypt(byte[] ciphertext, byte[] tag, byte[] key)
    {
        using (Aes aesAlg = Aes.Create())
        {
            aesAlg.Key = key;
            aesAlg.IV = tag;

            using (var decryptor = aesAlg.CreateDecryptor())
            using (var msDecrypt = new MemoryStream(ciphertext))
            using (var csDecrypt = new CryptoStream(msDecrypt, decryptor,
CryptoStreamMode.Read))
            using (var srDecrypt = new StreamReader(csDecrypt))
            {
                return srDecrypt.ReadToEnd();
            }
        }
    }

    static byte[] GenerateHMAC(byte[] key, string data)
    {
        using (var hmac = new HMACSHA256(key))
        {
            return hmac.ComputeHash(Encoding.UTF8.GetBytes(data));
        }
    }

    static bool CompareByteArrays(byte[] array1, byte[] array2)
    {
        if (array1.Length != array2.Length)
            return false;

        for (int i = 0; i < array1.Length; i++)
        {
            if (array1[i] != array2[i])
                return false;
        }

        return true;
    }
```

ДОДАТОК Б

Програмний код алгоритму CTR_DRBG

```
using System;
using System.Diagnostics;
using System.Text;
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.Crypto.Generators;
using Org.BouncyCastle.Crypto.Parameters;
using Org.BouncyCastle.Security;
using Org.BouncyCastle.Utilities;

class Program
{
    static void Main()
    {
        byte[] key = GenerateRandomBytes(16);
        byte[] iv = GenerateRandomBytes(16);

        Console.WriteLine("Сгенерований ключ: " +
BitConverter.ToString(key).Replace("-", ""));
        Console.WriteLine("Сгенерований IV: " + BitConverter.ToString(iv).Replace("-
", ""));
        Console.WriteLine();

        string messageToSend = "Доброго дня!";
        Console.WriteLine("Вихідне повідомлення: " + messageToSend);
        Console.WriteLine();


        Stopwatch encryptionStopwatch = Stopwatch.StartNew();
        string encryptedMessage = EncryptMessage(messageToSend, key, iv);
        encryptionStopwatch.Stop();
        Console.WriteLine("Зашифроване повідомлення: " + encryptedMessage);
        Console.WriteLine("Час шифрування: " + encryptionStopwatch.Elapsed + " мс");
        Console.WriteLine();


        Stopwatch decryptionStopwatch = Stopwatch.StartNew();
        string decryptedMessage = DecryptMessage(encryptedMessage, key, iv);
        decryptionStopwatch.Stop();
        Console.WriteLine("Розшифроване повідомлення: " + decryptedMessage);
        Console.WriteLine("Час дешифрування: " + decryptionStopwatch.Elapsed + "
мс");
    }

    static byte[] GenerateRandomBytes(int length)
    {
        SecureRandom random = new SecureRandom();
        byte[] bytes = new byte[length];
        random.NextBytes(bytes);
        return bytes;
    }

    static string EncryptMessage(string message, byte[] key, byte[] iv)
    {
        IBufferedCipher cipher = CipherUtilities.GetCipher("AES/CTR/NoPadding");
        cipher.Init(true, new
ParametersWithIV(ParameterUtilities.CreateKeyParameter("AES", key), iv));

        byte[] messageBytes = Encoding.UTF8.GetBytes(message);
```

```
        Stopwatch encryptionStopwatch = Stopwatch.StartNew();
        byte[] encryptedBytes = cipher.DoFinal(messageBytes);
        encryptionStopwatch.Stop();

        Console.WriteLine("Час шифрування в середньому: " +
encryptionStopwatch.Elapsed + " мс");

        return Convert.ToBase64String(encryptedBytes);
    }

    static string DecryptMessage(string encryptedMessage, byte[] key, byte[] iv)
    {
        IBufferedCipher cipher = CipherUtilities.GetCipher("AES/CTR/NoPadding");
        cipher.Init(false, new
ParametersWithIV(ParameterUtilities.CreateKeyParameter("AES", key), iv));

        byte[] encryptedBytes = Convert.FromBase64String(encryptedMessage);

        Stopwatch decryptionStopwatch = Stopwatch.StartNew();
        byte[] decryptedBytes = cipher.DoFinal(encryptedBytes);
        decryptionStopwatch.Stop();

        Console.WriteLine("Час дешифрування в середньому: " +
decryptionStopwatch.Elapsed + " мс");

        return Encoding.UTF8.GetString(decryptedBytes);
    }
}
```