МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
КАФЕДРА КОМП'ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____ С.В. Казмірчук
«_____»_____ 20__ р.

На правах рукопису
УДК 004.056.55

ДИПЛОМНА РОБОТА

ЗДОБУВАЧА ВИЩОЇ ОСВІТИ
ОСВІТНЬОГО СТУПЕНЯ «БАКАЛАВР»

Тема: Криптографічний модуль на базі генератора псевдовипадкових чисел

Виконавець: В.О. Лапенко
Керівник: к.т.н., доцент А.В. Ільєнко
Нормоконтролер: к.т.н., доцент А.В. Ільєнко

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет: Кібербезпеки, комп'ютерної та програмної інженерії

Кафедра: Комп'ютеризованих систем захисту інформації

Освітній ступінь: Бакалавр

Спеціальність: 125 «Кібербезпека»

Освітньо-професійна програма: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ С.В. Казмірчук

«__» _____ 20__ р.

ЗАВДАННЯ

на виконання дипломної роботи

здобувача вищої освіти Лапенко Валентини Олександрівни

1. Тема: *Криптографічний модуль на базі генератора псевдовипадкових чисел* затверджена наказом ректора від «26» квітня 2021 р. № 652/ст.

2. Термін виконання: з 10.05.2021 р. по 20.06.2021 р.

3. Вихідні дані: провести дослідження сучасних принципів побудови та реалізації генераторів псевдовипадкових чисел; розробити авторський криптографічний модуль забезпечення конфіденційності інформації на базі генератора псевдовипадкових чисел; провести дослідження працездатності власного програмного модуля.

4. Зміст пояснювальної записки: дослідження сучасних принципів побудови та реалізації генераторів псевдовипадкових чисел; порівняльний аналіз використання сучасних генераторів псевдовипадкових чисел; розробка та дослідження криптографічного модуля на базі Вихору Мерсенна з використанням синхронного потокового шифрування.

КАЛЕНДАРНИЙ ПЛАН

виконання дипломної роботи

№ п/п	Етапи виконання дипломної роботи	Термін виконання етапів	Примітка
1	Уточнення постановки задачі	19.04.2021	*Виконано*
2	Аналіз літературних джерел	20.04.2021 – 27.04.2021	*Виконано*
3	Обґрунтування рішення	28.04.2021 – 30.04.2021	*Виконано*
4	Дослідження сучасних принципів побудови та реалізації генераторів псевдовипадкових чисел	01.05.2021 – 15.05.2021	*Виконано*
5	Розробка авторського криптографічного модуля забезпечення конфіденційності інформації на базі генератора псевдовипадкових чисел	16.05.2021 – 25.05.21	*Виконано*
6	Дослідження працездатності власного програмного модуля	26.05.2021 – 30.05.2021	*Виконано*
7	Оформлення і друк пояснювальної записки	31.05.2021 – 05.06.2021	*Виконано*
8	Оформлення презентації	06.06.21 – 08.06.2021	*Виконано*
9	Отримання рецензій від рецензентів	08.06.2021 – 13.06.2021	*Виконано*

Здобувач вищої освіти В.О. Лапенко

(підпис, дата)

Керівник дипломної роботи А.В. Ільєнко

(підпис, дата)

РЕФЕРАТ

Дипломна робота складається зі вступу, двох розділів, загальних висновків, списку використаних джерел, додатків, загальним обсягом робота складає 63 сторінки, має 14 рисунків, 2 таблиці та 25 використаних джерел.

Метою роботи є розробка авторського криптографічного модуля забезпечення конфіденційності інформації на базі генератора псевдовипадкових чисел.

Об'єкт дослідження: процедура реалізації генератора псевдовипадкових чисел та синхронного потокового шифру.

Предмет дослідження - методи функціонування захищених систем та мереж на основі реалізації адитивного потокового шифру, методики тестування генератора псевдовипадкових чисел.

Методи дослідження: Проведені дослідження базуються на сучасних методах побудови захищених інформаційних мереж, методах та підходах реалізації генератора псевдовипадкових чисел

Практична цінність полягає у створенні власного криптографічного модуля на базі генератора псевдовипадкових чисел Вихор Мерсенна мовою програмування Python, що дозволяє наглядно продемонструвати етапи генерування псевдовипадкової послідовності, провести процедуру статистичного тесту згідно FIPS – 140-1 та реалізувати синхронний потоковий шифр для забезпечення конфіденційності інформації.

Ключові слова: псевдовипадкова послідовність, генератор псевдовипадкової послідовності, захист інформації, криптографічний модуль, синхронне потокове шифрування, тестування послідовності.

ЗМІСТ

ВСТУП

Нині паралельно з розвитком інформаційних технологій загострюється питання захисту даних, що зберігаються в інформаційних ресурсах. Проводиться дослідження вже існуючих та створення нових способів та методів захисту інформації, багато вчених та інженерів працюють над створеннями ще невідомих засобів захисту даних. Проблема захисту інформації включає у себе багато факторів, а, отже, може вважатися розв'язаною тільки з використанням комплексу методів.

Одним з найважливіших елементів безпеки інформації є випадкові числа, вони відіграють важливу роль при реалізації розмежованого та обмеженого доступу до інформації. Випадкові числа є основою генерування паролів, створення електронно-цифрового підпису та протоколів безпеки. Все це є причиною значного інтересу науковців до області випадкових та псевдовипадкових чисел, ведуться активні роботи над удосконаленням якості вже існуючих типів генераторів, а також над розробкою більшої теоретичної бази.

Істинно випадкову послідовність вдається згенерувати тільки за допомогою апаратних генераторів, так як за початкові значення вони можуть використовувати фізичні явище, оскільки шум, оцифровані звукові бути та механічні переривання. Оскільки комп'ютери є детермінованими машинами, згенерувати істинно випадкову послідовність вони не можуть, саме тому так широко використовуються генератори псевдовипадкових чисел. Існують певні труднощі при генеруванні псевдовипадкових чисел, адже отримана послідовність має бути дуже близькою до істинно випадкової, але незважаючи на це генератори ПВЧ використовуються практично у всіх сучасних комп'ютерних програмах. Вони застосовуються у таких сферах як криптографія для шифрування та розшифрування інформації, генерації ключів і паролів, імітації модулювання, що підходять як для розв'язання завдань статистики так проведення наукових експериментів, використовуються при створенні

вимірювальної техніки, а також у галузі створення комп’ютерних ігор. Всі ці фактори зумовлюють **актуальність** обраної теми дипломного проекту.

Метою даної роботи є створення криптографічного модуля, заснованого на генерації псевдовипадкових чисел, результати перевірки якого будуть задовольняти статистичні тести на перевірку схожості створеної послідовності до істинно випадкової та подальше використання створеної послідовності для шифрування повідомлення.

Виходячи з мети **завданням** дипломної роботи є:

- провести дослідження сучасних принципів побудови та реалізації генераторів псевдовипадкових чисел;
- розробити авторський криптографічний модуль забезпечення конфіденційності інформації на базі генератора псевдовипадкових чисел;
- провести дослідження працездатності власного програмного модуля.

Об’єктом дослідження даної роботи є процедура створення генератора псевдовипадкових чисел, що зможе на виході своєї роботи дати статистично хорошу послідовність, що відповідатиме всім вимогам, а також реалізація синхронного потокового шифрування А **предметом** дослідження є методи функціонування захищених систем та мереж на основі реалізації адитивного потокового шифру, методики тестування генератора псевдовипадкових чисел.

В якості **методів** було проведено дослідження, що базується на сучасних методах побудови захищених інформаційних мереж, методах та підходах реалізації генератора псевдовипадкових чисел

Практична цінність полягає у створенні власного криптографічного модуля на базі генератора псевдовипадкових чисел Вихор Мерсенна мовою програмування Python, що дозволяє наглядно продемонструвати етапи генерування псевдовипадкової послідовності, провести процедуру статистичного тесту згідно FIPS – 140-1 та реалізувати синхронний потоковий шифр для забезпечення конфіденційності інформації.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ ЗАХИСТУ ІНФОРМАЦІЇ ТА ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

1.1 Сучасні методи та засоби захисту інформаційних ресурсів

У наш час нас скрізь оточує інформація, всі наші персональні дані, всі банківські рахунки, документація та навіть новини, які ми щодня дізнаємось, чи реклама, яку щодня бачимо – все це інформація. З розвитком інформаційних технологій майже всі дані перейшли у цифровий вигляд і зберігаються на електронних носіях. Зі збільшенням їх використання постало і питання захисту даних, що містяться в електронних ресурсах, унеможливлення несанкціонованого доступу до інформації, перевірка осіб, які намагаються отримати доступ до даних.

Інформація використовується у кожній галузі життя як і окремої особи, так і цілої держави і її викрадення може принести ті чи інші наслідки залежно від сфери її використання. У час глобальної діджиталізації гостро постає проблема захисту даних, що містяться в інформаційних ресурсах. Концентрація інформації в цифровому вигляді змушує одних підсилювати пошуки шляхів доступу до даних, а інших, відповідно, підсилювати контроль над нею з метою захисту.

Інформаційна безпека – це набір практик, призначених захистити дані від несанкціонованого доступу чи змін, як коли вони зберігаються, так і коли вони передаються з однієї машини чи фізичного місця в іншу. Оскільки знання стали одним із найважливіших надбань 21 століття, зусилля щодо захисту інформації, відповідно, стають все більш важливими.

Для опису критеріїв інформаційної безпеки використовують модель тріади CIA. Дана модель виокремлює такі основні властивості інформації як конфіденційність, цілісність та доступність, на захист яких направлені основні сили системи захисту.

Конфіденційність полягає у неможливості несанкціонованого доступу до даних користувачами, у яких нема відповідних прав. Дані є конфіденційними, коли користуватися ними можуть лише ті люди, яким надано до них доступ; для забезпечення конфіденційності, необхідно мати можливість визначити, хто намагається отримати доступ до даних, і блокувати спроби тих, хто не має дозволу. Паролі, шифрування, автентифікація та захист від атак проникнення - це всі методи, призначені для забезпечення конфіденційності.

Цілісність означає підтримку даних у немодифікованому стані та запобігання їх неправомірній зміні, випадково чи зловмисно. Багато методів, що забезпечують конфіденційність, також захищають цілісність, але існують інші інструменти, які допомагають забезпечити глибокий захист цілісності: контрольні суми можуть допомогти перевірити дані на цілісність, наприклад, програмне забезпечення для контролю версій та часті резервні копії можуть допомогти відновити дані до початкового стану, якщо це необхідно.

Доступність - це дзеркальне відображення конфіденційності, тобто властивість, що характеризує можливість авторизованого користувача отримати доступ до необхідних йому даних у будь-який зручний для нього час. Хоча потрібно переконатись, що неавторизовані користувачі не можуть отримати доступ до даних, також потрібно забезпечити доступ до них тим, хто має належні дозволи. Забезпечення доступності даних означає узгодження мережевих та обчислювальних ресурсів з обсягом доступу до даних та впровадження належної політики резервного копіювання для цілей аварійного відновлення[1].

Для захисту інформаційних ресурсів використовуються методи та засоби захисту інформації, які у свою чергу можна поділити на такі категорії:

- Правові методи та засоби захисту. До цієї категорії відносяться усі чинні закони, укази, нормативно-правові документи, що регулюють правила користування інформацією, встановлюють відповідальність за комп’ютерні злочини, захищають авторські права програмістів. Також до цієї групи відноситься питання контролю за розробкою комп’ютерних систем та

прийняття національних та міжнародних договорів про обмеження , якщо дані системи впливають або можуть вплинути на військові, економічні та соціальні аспекти країни.

Широке використання комп’ютеризованих систем призводить до нових можливостей для правопорушень, що не зустрічались раніше, або ж нових методів проведення злочинів, що були відомі раніше. Це зумовило необхідність розвитку правових засобів для регулювання суспільних відносин у сфері інформаційних відносин. Останнім часом спостерігається посилення мір покарання стосовно комп’ютерних злочинів та інформаційних правопорушників. Так відповідно до Кримінального кодексу України «Незаконне втручання в роботу автоматизованих електронно-обчислювальних машин, їх систем чи комп'ютерних мереж, що призвело до перекручення чи знищення комп'ютерної інформації або носіїв такої інформації, а також розповсюдження комп'ютерного вірусу шляхом застосування програмних і технічних засобів, призначених для незаконного проникнення в ці машини, системи чи комп'ютерні мережі і здатних спричинити перекручення або знищення комп'ютерної інформації чи носіїв такої інформації,- караються штрафом до сімдесяти неоподатковуваних мінімумів доходів громадян, або виправними роботами на строк до двох років, або обмеженням волі на той самий строк» [Доповнення до Статті 361 Кримінального кодексу України][2]. Для порівняння, У Гонконгу за злочин, що призвів до виведення з ладу інформаційної системи або Web-сайту, покаранням є позбавлення волі на 10 років.

- Організаційно-адміністративні методи та засоби захисту. До даної категорії належать підбір персоналу, охорона комп’ютеризованих систем, розподіл навантаження особливо важливих робіт між працівниками, план відновлення системи після збоїв, розподіл відповідальності між особами, що забезпечують безпеку системи, розподіл ресурсів, контроль за дотриманням політики безпеки. Адміністративні засоби захисту включають ряд заходів, які контролюють функціонування системи, до них відносяться:

a. Заходи, що виконуються під час проектування системи та розроблення плану її охорони. До таких заходів відносяться врахування можливості стихійної небезпеки, складання протипожежних планів, планування пропускного режиму та контролю роботи працівників, а також облік захисних мір при проектуванні та реалізації.

b. Заходи пов’язані з використання програмного та апаратного забезпечення у системі. Сюди відносяться перевірка ліцензій ПЗ та сертифікація всіх технічних засобів, підтримка правильної конфігурації операційної системи.

c. Заходи спрямовані на роботу з персоналом, такі як проведення інструктажів безпеки, перевірка даних співробітників, що вступають на роботу, ознайомлення з порядком доступу до конфіденційної інформації та її використання.

d. Заходи пов’язані з розробкою правил користування інформацією та планом її захисту, а саме створенням порядку обліку, зберігання та знищення документації та носії, що зберігають критично важливі дані, створенням парольного захисту та розподілу повноважень між працівниками.

- Інженерно-технічні методи та засоби захисту. Да цієї категорії належать всі засоби і заходи призначені для захисту системи від несанкціонованого доступу, розкрадання даних, що містяться у системі, резервного копіювання даних, пристрої, що забезпечують безперебійну роботу електроживлення, також до даної групи належить розробка і використання апаратних та програмних засобів, що забезпечують безпеку системи. Дану категорію можна розділити на 4 групи засобів:

1. Фізичні засоби. Ця група являє собою сукупність механічних, електронних та електронно-механічних пристроїв, будівель та інженерних засобів, призначених для унеможливлення несанкціонованого доступу до об’єкта, захищають персонал та дані, що містять у системі.

2. Апаратні засоби. До даної групи належать технічні прилади та пристрої, призначені для захисту інформації, засоби розроблені та реалізовані

для пошуку закладних пристроїв, що несуть загрозу даним. Завданням засобів, що відносяться до цієї групи є створення стійкої системи захисту інформації, що забезпечує безпеку від розголошення або витоку даних, або несанкціонованого доступу до них з використанням технічних каналів.

3. Програмні засоби. Програми та програмні комплекси, призначені для зберігання, обробки або знищення даних, вони забезпечують можливість ідентифікації та аутентифікації користувачів системи, здійснюють контроль за правами доступу до критично важливих даних, що містяться в інформаційній системі.

4. Криптографічні засоби. До даної групи належать математичні та алгоритмічні засоби захисту інформації, що зберігається і обробляється в системі, апаратно-програмні засоби, частину криптографічних функцій яких реалізовано на спеціальних апаратних пристроях до електронно-обчислювальної техніки, засоби призначені для виготовлення ключових таких, та управління ними. Метою використання криптографічних засобів є приховування або відновлення вмісту ключових даних, підтвердження цілісності інформації та її авторства[3].

Безпека сучасної інформації та зв’язку системи нерозривно пов’язані з алгоритмами, що забезпечують конфіденційність і цілісність збереженої та криптографічну стабільність переданої інформації змішаного типу, а також функцій ідентифікації та автентифікації[4].

1.2 Дослідження сучасних принципів побудови та реалізації генераторів псевдовипадкових чисел

Генератори псевдовипадкових чисел (ГПВЧ) – це детерміновані алгоритми для створення унікальної послідовності випадкових чисел шляхом введення початкової умови (початкової точки). Якість ГПВЧ може характеризуватись його здатністю представляти рівномірний розподіл,

непередбачуваність і тривалу періодичність ряду вихідних чисел, некорельовані послідовні результати та портативність алгоритму. ГПВЧ використовуються в різних додатках, таких як:

- Статистика та теорія ймовірності для вибірки даних;
- Теорія прийняття рішень як стратегія оптимізації обчислювальних алгоритмів;
- Чисельний розрахунок, наприклад, методом інтегрування Монте-Карло;
- Моделювання систем для імітації стохастичної поведінки, як у явищах природи;
- Індустрія розваг, наприклад, для рендерингу сценаріїв та текстур у фільмах;
- Мови програмування та будь-які алгоритми чи евристики, що вимагають вибірки, наприклад, розпізнавання зразків, штучний інтелект, генетичні алгоритми, глибоке навчання;
- Криптографія, наприклад, як джерело випадковості, вектор ініціалізації або генерація пароля, де криптографічно захищенні генератори випадкових чисел мають першочергове значення для алгоритмів шифрування, що діють у військових комунікаціях, розвідувальних інструментах, банківських системах, електронній комерції тощо[5].

Розглядаючи детальніше використання генераторів псевдовипадкових чисел в системах забезпечення інформаційної безпеки можна виокремити такі їх функції:

- Формування тестових взаємодій на входи компонентів системи, що перевіряються, при автономній чи вбудованій діагностиці.
- Реалізація лічильників команд або адрес комп’ютерних систем.
- Визначення відповідності між адресою порту входу-виходу та функцією, що запитується, при реалізації плаваючих протоколів взаємодій програмного забезпечення і пристроїв введення-виведення, механізму прихованих функцій введення-виведення.

- Формування елементів ймовірнісного простору при внесенні невизначеності в результат роботи алгоритмів захисту інформації.
- Задання послідовності виконання при внесенні невизначеності у послідовність виконання окремих кроків алгоритму.
- Задання тривалості виконання при внесенні невизначеності в тривалість, виконання окремих кроків алгоритму для захисту від витоку побічними каналами.
- Формування елементів ймовірнісного простору при внесенні невизначеності в механізм роботи програмних засобів.
- Формування гами при шифруванні інформації в режимах гамування і гамування зі зворотнім зв’язком.
- Формування ключів та паролів користувачів.
- Формування випадкових запитів при аутентифікації віддалених абонентів за принципом «запит-відповідь».
- Формування прекурсорів для захисту прав власників інформації (протокол сліпого підпису).

Вимоги, що висуваються до генератора ПВЧ, що орієнтований на використання з метою захисту інформації:

1. Непередбачуваність (криптографічна стійкість).
2. Хороші статистичні властивості, згенерована послідовність не повинна відрізнятись від істинно випадкової послідовності за своїми властивостями.
3. Великий період послідовності, що формується, враховуючи факт, що при перетворенні великих масивів даних кожному елементу вхідної послідовності необхідно поставити у відповідність свій елемент псевдовипадкової послідовності.
4. Ефективна програмна і апаратна реалізація[6].

Статично безпечний генератор ПВЧ має задовольняти наступні вимоги:

- Жоден статичний тест не виявляє в ПВЧ будь-яких закономірностей (не відрізняє послідовність від істинно випадкової).

- Нелінійне перетворення Fk, залежне від секретної інформації (ключа k), що використовується для побудови генератора, має властивість «розмноження» спотворень – всі вихідні (перетворені) вектори e’ можливі і однаково ймовірні незалежно від початкового вектора e.

- При ініціалізації випадковими значеннями генератор породжує статично незалежні псевдовипадкові числа.

За типом функції, що використовується, генератори ПВЧ можна розділити на некриптографічні і криптографічні. До некриптографічних відносяться конгруентні генератори, генератори, що функціонують в кінцевих полях, і генератори на регістрах зсуву з нелінійними зворотними зв'язками. До криптографічних - блокові і потокові генератори та генератори на основі односторонніх функцій

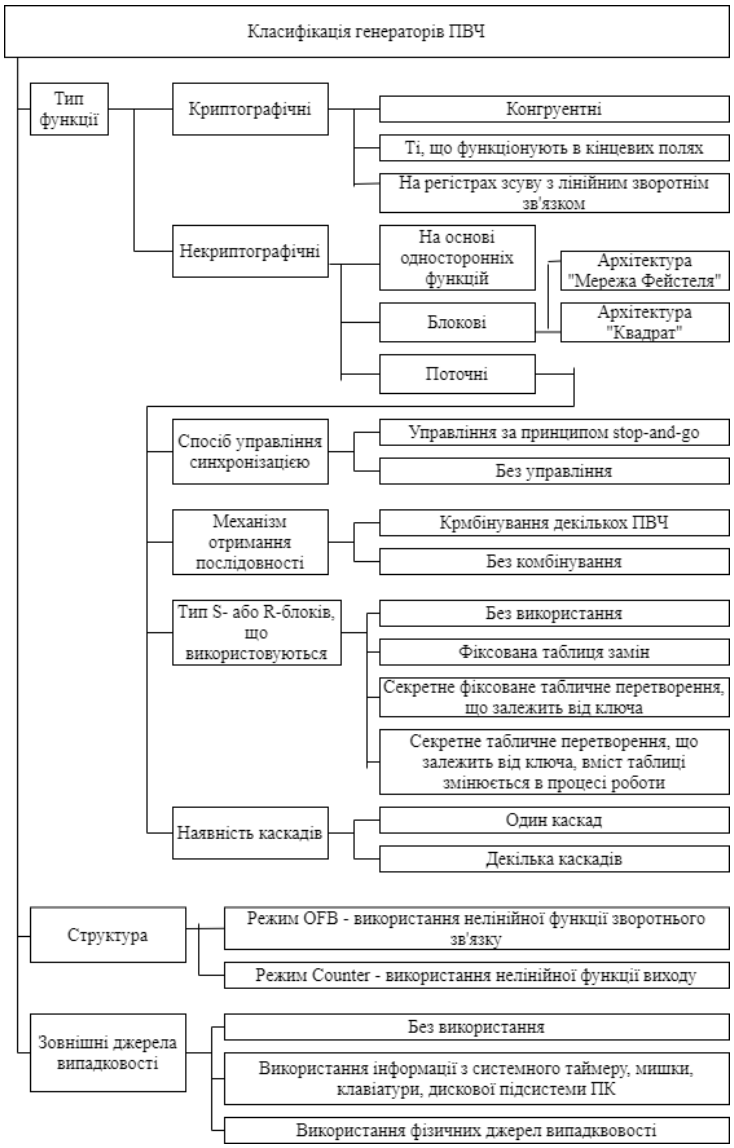


Рис. 1.1. Класифікація генераторів ПВЧ

Серед некриптографічних методів генерування псевдовипадкових послідовностей найбільш відомими є:

1. Лінійний конгруентний метод[7] - це клас алгоритмів генератора псевдовипадкових чисел (PRNG), який використовується для генерації послідовностей випадкових чисел у певному діапазоні. Відповідно до цього методу послідовність псевдовипадкових чисел формується за правилом:

$$X_{i+1} = (a \cdot X_i + c) \bmod m \qquad (1.1)$$

де X – послідовність псевдовипадкових чисел,

m (> 0) – модуль порівняння,

a (0, m) – множник,

c (0, m) – приріст,

X_0 [0, m) – початкове число.

При a = 1 це буде адитивний метод конгруентності.

При c = 0 це буде мультиплікативний метод конгруентності.

Даний метод генерує статистично хорошу псевдовипадкову послідовність чисел, однак він не є криптографічно стійким, адже є досить передбачуваним.

2. Алгоритм «Mother-of-All» розроблений професором університету Флориди Джорджем Марсалією. Даний алгоритм базується на множенні з піднесенням і має період 2^{250}. Обчислювальна формула:

$$\begin{cases} \boldsymbol{S = 21111111111x_{n-4} + 1439x_{n-3} + 1776x_{n-2} + 5115x_{n-1} + } \\ \boldsymbol{x_n = \frac{S}{2^{32}}} \\ \boldsymbol{C = \left[\frac{S}{2^{32}}\right]} \end{cases} \qquad (1.2)$$

Цей алгоритм є узагальненням лінійного конгруентного методі і позбавлений його головного недоліку – короткого періоду.

При реалізації даного алгоритму використовують два масиви, що зберігають значення в їх першому елементі та випадкові 16-бітні числа в елементах від 1 до 8. Ці випадкові числа переміщуються до елементів від 2 до 9, а новий носій і номер створюються і поміщаються в елементи 0 і 1. 32-бітове

випадкове число отримується шляхом об’єднання виводу двох генераторів і повернення rnd. Обидва масиви заповнюються випадковими 16-бітними значеннями при першому виклику функції Next().

3. Вихор Мерсенна[8] – генератор псевдовипадкових чисел. На сьогоднішній день це найпоширеніший ГПВЧ загального призначення. Його назва походить від того, що тривалість періоду обрана як просте число Мерсенна.

Цей генератор псевдовипадкових чисел має період 2 у степені 19937-1, і його вихід не має довгострокових кореляцій при розгляді з точки зору 623 вимірів. Для порівняння, навіть коли використовується двовимірний графік послідовних членів пар чисел, що створюються звичайним лінійним конгруентним генератором псевдовипадкових чисел, очевидні закономірності видно.

Алгоритм працює на початковому значенні, яке становить 19 937 біт; це значення зберігається в масиві з 624 елементами, які місять 32-розрядні значення. У масиві є 31 невикористаний біт, який мігрує через масив під час обчислення наступного насіння, тому жоден окремий елемент масиву не може бути замінений однобітовою коміркою зберігання. Він базується на тому, що 2 у степені 19937-1 є простим числом (і, оскільки, це на один менше степеня двійки, також є простим числом Мерсенна). Як зазначено про шифри, засновані на регістрах зсуву, якщо регістр зсуву має кілька комірок, які відповідають показнику для простого числа Мерсенна, процес тестування цього регістру зсуву протягом максимального періоду спрощується. Більш досконале спрощення цього процесу тестування, відоме як метод інверсної децимації, використовується, щоб дозволити побудову регістру зсуву максимального періоду конкретної форми. Саме розробка цього методу розробниками вихору Мерсенна дозволила створити такий великий регістр зсуву максимального періоду.

Незважаючи на той факт, що вихор Мерсенна[9] є хорошим генератором, він сам по собі не є криптографічно безпечним. Причиною цього є те, що

можна визначити всі наступні стани генератора за станом, який він має у будь-який момент часу, і для забезпечення цього стану достатньо або 624 32-розрядних виходи, або 19 937 однобітових виходи. Використання криптографічно захищеної хеш-функції, такої як SHA-1, на виході вихору Мерсенна рекомендується як один з ключових способів отримання ключового потоку, корисного в криптографії.

4. Xorshift[10] є класом генераторів псевдовипадкових чисел, що був відкритий Джорджем Марсалієм. Генератори такого типу є підмножиною регістрів зсуву з лінійним зворотнім зв'язком, що дозволяє ефективно реалізувати їх без надмірного використання розріджених многочленів. Генерація наступного числа в послідовності відбувається шляхом багаторазового обчислення виключаючого «АБО» поточного числа і його бітового зсуву, що робить xorshift надзвичайно швидким на сучасних комп'ютерних архітектурах.

Стан генератора xorshift є вектором бітів. На кожному кроці наступний стан отримується шляхом застосування певної кількості операцій xorshift до w-бітових блоків у поточному стані, де w = 32 або 64, і операція xorshift визначається наступним чином: заміняється w-бітовий блок побітовим XOR (виключаючим або) оригінального блоку зі зміщеною копією самого себе на a позицій вправо або вліво, де 0 < a < w.

Хоча у простому вигляді генератори xorshift не проходять всі статистичні тести випадковості, цей недолік добре відомий і легко виправляється шляхом додавання в їх структуру нелінійної функції, в результаті чого виходять такі генератори як xorshift + або xorshift *.

Випадкові числа в прикладній криптографії можуть використовуватись для генерації ключів, в якості одноразових випадкових чисел в протоколах аутентифікації, генерації одноразових шифроблоків, а також у схемах електронно-цифрового підпису.

Існують три великих класи алгоритмів, що використовуються у криптостійких генераторах ПВС:

1. Методи на основі криптографічних алгоритмів.

2. Методі на основі односторонньої функції (математично складних задач).

3. Спеціальні реалізації.

У першому випадку генерація ПВЧ може бути реалізована одним з наступних методів:

- Використанням безпечного блокового шифру в режимі лічильника (гамування) до якогось випадкового числа.

- Використанням криптографічно стійкої хеш-функції до початкового секретного випадкового числа.

- Використанням потокових шифрів, які у свою чергу, самі працюють на основі генератора псевдовипадкових бітів.

У другому класі алгоритмів для генерації використовується односторонні функції з ключем, в якості якого може слугувати функція піднесення до квадрату, функція експоненціонування, функція множення простих чисел у скінченному полі.

Третій клас алгоритмів заснований на використанні стохастичних методів перетворення чисел.

Блокові генератори ПВЧ признані найкращими за двома критеріями – непередбачуваності та швидкодії. Блоковий шифр – різновид шифру, основними особливостями якого є:

- Шифрування вихідного тексту блоками.

- Зміст кожного блоку ніяк не впливає на результат шифрування інших блоків.

Блоковий шифр можна представити функцією

$$E:\{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \qquad (1.3)$$

У цих позначеннях функція має два вхідні параметри: перший k-бітовий рядок, що носить назву ключ, і позначається $K \in \{0,1\}^k$, і k-бітовий рядок повідомлення, який позначається $M \in \{0,1\}^n$. Результат функції називається шифротекстом і позначається $C \in \{0,1\}^n$.

Для конкретного ключа *K*, функція блокового шифру позначається

$$E_K(M) = C \qquad (1.4)$$

Для кожного визначеного ключа K, функція блочного шифру повинна бути взаємно однозначним відображення. Це означає, що E_K завжди має зворотню функцію E_K^{-1} таку, що $E_K(E_K^{-1}(C)) = C$ і $E_K^{-1}(E_K(M)) = M$ для будь-яких $C, M \in \{0,1\}^n$.

Стандартний спосіб застосування блочного шифру:

- вибір деякого ключа K, який відомий двом сторонам інформаційного обміну, але тримається в секреті від інших;
- генерація шифротекста шляхом застосування функції шифрування до повідомленням, перед його відправкою партнеру $C = E_K(M)$.
- відправка шифротекста партнеру;
- отримання і розшифровка партнером шифротекста шляхом використання зворотньої функції $M = E_K^{-1}(C)$.

Режим лічильника блокового шифру (CTR mode) - такий режим шифрування вхідної послідовності, при якому на вхід відповідного алгоритму блокового шифрування подаються значення, так званого лічильника, накопиченого з моменту старту. Змінюючи значення лічильника, алгоритм блокового шифрування утворює рядок бітів, який використовується в якості ключа, що біжить, шифру Вернама, тобто ключа, що біжить, і блокам результуючого повідомлення застосовуються операції побітового складання по модулю 2.

По суті, режим лічильника робить з блочного шифру потоковий.

Приклади алгоритмів на основі блокових шифрів для генерації псевдовипадкових послідовностей:

- Алгоритм ANSI X9.17[11] схвалений FIPS і затверджений стандартом для використання з метою генерації псевдовипадкових ключів і ініціалізації векторів в алгоритмі шифрування DES. Так як алгоритм DES зараз вже не вважається безпечним, то ГПСЧ на його основі також не може

вважатися безпечним. Однак цей ГПВЧ отримав велике поширення, зокрема, застосовується в PGP

Опис алгоритму. На вхід подаються: секретний початковий 64-бітний вектор s, ціле число m і секретний 112-бітний ключ для DES шифрування k.

На виході: m псевдовипадкових чисел $x_1, x_2, ..., x_m$.

1. Вираховується проміжне значення $I = E_k(D)$, де D – 64-бітне представлення часу з максимально можливою точністю.

2. *For i = 1 to m*

   a. $x_i \leftarrow E_k (I \oplus s)$.

   b. $s \leftarrow E_k (x_i \oplus I)$.

3. Повертається $(x_1, x_2, ..., x_m)$.

Будь-який рядок бітів x_i може бути використаний в якості ініціалізаційного вектору (*IV*) для одного з режимів шифрування DES.

- FIPS 186[12]. Алгоритм генерації секретного ключа для ЕЦП. Даний алгоритм представлений в рекомендованих методах для генерації псевдовипадкових параметрів електронно-цифрового підпису. Цей алгоритм генерує секретний ключ a, використовуючи для цього початковий вектор s, який повинен бути згенерований випадковим чином і перетворений хеш-функцією SHA-1.

Опис алгоритму. На вхід подаються: ціле число m і 160-бітне просте число q.

На виході: m псевдовипадкових чисел $a_1, a_2, ..., a_m$. в інтервалі $[0, q-1]$, які можуть бути використані в якості секретних ключів для ЕЦП.

1. Вибирається 160-бітне число b.

2. Визначити 160-бітний рядок t=0x67452301 EFCDAB89 98BADCFE 10325476 C3D2E1F0.

3. *For i = 1 to m*

   a. $y_i \leftarrow 0$.

   b. $z_i \leftarrow (s + y_i) \bmod 2^b$.

c. $a_i \leftarrow G(t, z_i) \bmod 2^b$.

d. $s \leftarrow (1 + s + a_i) \bmod 2^b$.

4. Повертається $(a_1, a_2, ..., a_m)$.

В якості функції G можна обрати або функцію SHA-1, або функцію алгоритму DES.

Використання хеш-функції для генерації криптографічної послідовності псевдовипадкових чисел побідне використанню блокового шифру з тією ж метою. Вибравши деяке ініціалізуюче натуральне число S_i, поступово застосовується перетворення $S_{i+1} = H(S_i)$. Алгоритм може бути модифікований, але головна його вразливість залишиться – безпека повністю залежить від ініціалізуючого значення. Дізнавшись ініціалізуюче значення і алгоритм генератора ПВЧ, зловмисник може побудувати всю послідовність ПВЧ.

Крім того, для побудови криптостійких генераторів ПВЧ[13] рекомендується використовувати хеш-функцію для генерації початкового вектору, реініціалізації, генерації внутрішнього стану і генерації чергового числа послідовності. Всі ці вимоги підвищують надійність алгоритму і знижують ймовірність успішної атаки на ГПВЧ.

Блокові генератори ПВЧ варто признати кращими за сукупністю двох критеріїв – непередбачуваності і швидкодії.

Потокові генератори ПВЧ, будучи найбільш швидкодіючим типом генераторів, застосовуються в тих випадках, коли необхідна передача великих інформаційних потоків в реальному масштабі часу або близькому до нього.

На відміну від блокових генераторів ПВЧ, чиї нелінійні функції будуються по ітеративному принципом, при проектуванні потокових генераторів використовується безліч прийомів і методів, класифікувати які дуже складно. Можна виділити все ж такі параметри:

- спосіб управління синхронізацією;
- механізм отримання вихідної послідовності;
- тип використовуваних S- або R-блоків;
- наявність або відсутність каскадів.

1. Спосіб управління синхронізацією. За цим параметром генератори діляться на дві групи: що працюють за принципом stop-and-go і не використовують управління синхронізацією. Класичні приклади генераторів ПВЧ першої групи – генератори ПВЧ A5 і PIKE.

2. Механізм отримання вихідної послідовності. З цього параметру генератори можна розділити на дві групи - відповідно ті, що використовують, і ті, що не використовують, для отримання результуючої послідовності комбінування декількох псевдовипадкових послідовностей (ПВП). Класичний приклад першого підходу - перемішування двох ПВП під керуванням третьої. Прикладами генераторів першого типу є згадані вище генератори A5 і PIKE, в яких комбінування здійснюється з використанням операції додавання по модулю 2.

3. Тип використовуваних S- або R-блоків. За цим параметром генератори можна розділити на чотири групи:

- взагалі без табличних перетворень;
- використовують фіксовані несекретні таблиці;
- використовують фіксовані секретні (ключові або залежать від ключа) таблиці;
- використовують секретні таблиці, що безперервно змінюються в процесі функціонування генератора.

Найбільшу перевагу надають останньому варіанту, так як він робить позбавленими особливого сенсу дії аналітика, пов'язані з відновленням ключової таблиці за наявним фрагментом ПВП. Класичним прикладом такого підходу є один з кращих поточних генераторів ПВЧ, генератор RC4, розроблений Р. Рівестом.

4. Наявність або відсутність каскадів. За цим параметром генератори можна розділити на дві групи - за відсутністю або наявністю каскадів відповідно. В останньому випадку принцип побудови якісного генератора ПВЧ є каскадне включення декількох генераторів відносно невисокої якості.

Генератора на основі односторонніх функцій варто признати найбільш математично обгрунтованими. Тим не менш ці генератори не знайшли широкого розповсюдження в першу чергу через надзвичайно низьку швидкодію. Вони працюють у сотні разів повільніше блокових генераторів, як самі по собі вважаються повільними.

1.3 Порівняльний аналіз використання сучасних генераторів псевдовипадкових чисел

Сьогодні генератори псевдовипадкових чисел широко використовуються у багатьох сферах життя для виконання різних завдань. Сфера використання генераторів ПВЧ не обмежується лише галуззю розробкою програмного забезпечення чи забезпеченням інформаційної безпеки, вона набагато ширша.

Розглядаючи генератори псевдовипадкових чисел на основі блокових шифрів можна побачити такі сфери їх використання:

- ANSI X9.17. Створюється прямий захищений ANSI генератор ПВЧ, який включає вдосконалений блок-шифр, який не є інвертованим, навіть якщо ключ і функція блочного шифру, що використовується в існуючому генераторі , стають відомими. Крім того, захищений ANSI ГПВЧ включає вдосконалений наступний стан, що дозволяє попереднім станам залишатись у секреті, навіть коли ключ і поточний стан стають відомими. Генератор ПВЧ ANSI є частиною популярного банківського стандарту і був запропонований як механізм генерування DES. У число додатків, що використовуються даний генератор входять додатки фінансової безпеки і PGP – комп'ютерна програма, а також бібліотечні функції, що дозволяють виконувати операції шифрування та цифрового підпису повідомлень, файлів та іншої інформації, даних в електронному вікні.
- FIPS 186 є алгоритмом, описаним у стандарті електронного підпису. Стандарт визначає набір алгоритмів, які можна використовувати для

формування цифрового підпису. Цифрові підписи використовуються для виявлення несанкціонованих модифікацій даних та автентифікації особи підписанта. Крім того, одержувач підписаних даних може використовувати цифровий підпис як доказ, демонструючи третій стороні, що підпис фактично був сформований заявником, на якого заявлено право. Це відоме як відмова від відмови, оскільки підписант не може легко відмовитись від підпису пізніше. Цей стандарт визначає три методи для генерації та перевірки цифрових підписів: DSA, ECDSA та RSA. Цей перегляд збільшує довжину ключів, дозволених для DSA, забезпечує додаткові вимоги до використання ECDSA та RSA, а також включає вимоги щодо отримання гарантій, необхідних для дійсних цифрових підписів.

Розглядаючи генератори псевдовипадкових чисел на основі хеш-функцій можна побачити такі сфери їх використання:

- CRC32[14] – функція виявлення помилок, яка використовує алгоритм CRC32 для виявлення змін між вихідними та цільовими даними. Функція CRC32 перетворює рядок змінної довжини у 8-символьний рядок, який є текстовим поданням шістнадцяткового значення 32-бітової двійкової послідовності. CRC (циклічний надлишковий код) не є криптографічно стійкою хеш-функцією, бо не має стійкості до колізій. Проте, ця функція широко застосовується в різних додатках для захисту від випадкових (ненавмисних) змін даних, виявлення помилок при передачі даних.

- MD4[15] – алгоритм генерування хеш-функцій. Хеш-значення представляє шістнадцяткове число з 32 4-бітових символів. Цей алгоритм не є криптографічно стійким, проте на базі цього алгоритму було розроблено ціле сімейство інших більш складних алгоритмів хешування, - MD5, SHA-1, RIPEMD-160, SHA-256, SHA-512, SHA-384.

- MD5[16] є вдосконаленою версією алгоритму MD4. За допомогою MD5 перевіряли цілісність і автентичність завантажених файлів - так, деякі програми поставляються разом зі значенням контрольної суми, наприклад, пакети для інсталяції вільного програмного забезпечення.

Розглядаючи генератори псевдовипадкових чисел на основі потокових шифрів можна побачити такі сфери їх використання:

- RC4[17] - це потоковий шифр, який широко застосовується в різних системах захисту інформації в комп'ютерних мережах (наприклад, в протоколі SSL і для шифрування паролів в Windows NT).

Алгоритм RC4 будується як і будь-який потоковий шифр на основі параметризованих ключем генератора псевдовипадкових бітів з рівномірним розподілом. Основні переваги шифру - висока швидкість роботи і змінний розмір ключа. Типова реалізація виконує 19 машинних команд на кожен байт тексту.

- Генератор Шаміра і RSA[18]. Алгоритми ГПВЧ на основі складних математичних задач використовують складність вирішення деяких завдань для отримання псевдовипадкових чисел, захищених від криптоаналізу. Іншими словами, криптограф, що створює ГПВЧ, намагається використовувати теорію складності так, щоб рішення задачі криптоаналізу було б еквівалентно рішенню важкою теоретичної задачі.

Еді Шамір запропонував використовувати алгоритм шифрування з відкритим ключем RSA для генерації ПВЧ. У його роботі показано, що пророкування виходу генератора псевдовипадкових чисел рівносильно злому RSA. Очевидним недоліком такого алгоритму є низька швидкість і громіздкість реалізації. Подальшою модифікацією цього алгоритму є ГПСЧ RSA, який також заснований на складності рішення проблеми RSA.

Висновки до розділу 1

Нині життя без доступу до необхідної інформації у будь-який час з будь-якого місця стало немислимим. Однак, разом зі зростанням попиту на доступ до інформації, зросла і важливість захисту даних, які ми хочемо отримати.

Проведемо порівняльний аналіз (табл. 1.1) методів генерування псевдовипадкових послідовностей.

Таблиця 1.1.

Порівняльний аналіз основних методів генерування псевдовипадкових послідовностей

	Лінійний конгруентний метод	**Mother of All**	**Xorshift**	**Вихор Мерсена**
Період послідовності	< m, де m – довільне ціле число, за винятком 0	2^{250}	$2^{128} – 1$	$2^{19937} – 1$
Галузь застосування	Сучасні системи програмування	Прикладні науки	Сучасні комп'ютерні архітектури	Додатки, що вимагають висококласного генератора ПВЧ
Швидкодія	+	-	+	+
Переваги	Породжує статистично хорошу псевдовипадкову послідовність	Має більший період ніж лінійний конгруентний метод	Одні з найбільш швидких генераторів	Відсутні недоліки інших генераторів ПВЧ, такі як малий період, передбачуваність, статистичні закономірності, що легко виявляються
Недоліки	Передбачуваність Не є криптографічно стійким	Не є криптостійким Значно повільніший за інші алгоритми	В «сирому» (немодифікованому)вигляді не проходять всі статистичні тести Не є	Не є криптостійким, але можна збільшити захист використавши

			криптостій-ким	додаткове хешування послідовності

Останнім часом ми все менше чуємо про нововведення щодо зберігання, використання, обробки інформації в електронному вигляді і її передачі, ніж ми чуємо про несанкціонований доступ, кібератаки, злом, порушення конфіденційності тощо. Ці явища існують вже не на рівні окремих випадків, компаній чи підприємств, а на глобальному рівні, що викликає занепокоєння, адже дані проблеми є актуальними на державному та міжнародному рівнях.

Задля створення системи захисту інформаційних ресурсів розроблено методи та засоби їх захисту, які включають у себе різні аспекти, а саме правовий, адміністративно-організаційний та інженерно-технічний. Кожен з методів включає відповідні заходи, положення, та план дій необхідних для забезпечення тієї чи іншої ланки безпеки.

Щоб забезпечити безпеку даних, їх конфіденційність і цілісність використовують різноманітні криптографічні засоби, серед яких і генератори псевдовипадкових чисел. Генератор псевдовипадкових чисел – алгоритм, який породжує послідовність чисел, елементи якої майже незалежні один від одного і підкоряються заданому розподілу, дана послідовність має властивості випадкової.

Генератори ГПВ використовуються у багатьох сферах, у тому числі і системах забезпечення інформаційної безпеки. Існують криптографічно стійкі і нестійкі генератори, на які покладено різні функції. Багато прикладних завдань криптографії вимагають випадкових чисел, наприклад, генерація ключів, одноразові випадкові числа, одноразові шифроблокноти, сіль в схемах цифрового підпису. Для кожного завдання варто підібрати відповідний тип генератору ПВЧ.

РОЗДІЛ 2 ПРОГРАМНИЙ МОДУЛЬ НА БАЗІ ГЕНЕРАТОРА ПСЕВДОВИПАДКОВИХ ЧИСЕЛ

2.1. Опис середовища розробки

Для написання програмного модуля використовується редактор тексту Atom[19]. Atom – це безкоштовний редактор тексту та вихідного коду з відкритим кодом, розроблений GitHub (Atom - Hackable Text and Source Code Editor for Linux). Atom дозволяє користувачам встановлювати сторонні пакети та теми для налаштування функцій та зовнішнього вигляду редактора. Atom дозволяє працювати з R та Python послідовно. Ці дві мови не є єдиними мовами, що підтримуються, за замовчуванням пакети Atom можуть застосовувати підсвічування синтаксису для таких мов програмування та форматів файлів: C, C ++, C #, COBOL, CSS, CoffeeScript, Go, HTML, Java, JavaScript, JSON, Perl , PHP, Ruby, Scala, SQL та багато інших.

Atom сам по собі має основні функціональні можливості, але також можна використовувати ряд корисних пакетів, які додають нові функції. Пакети неймовірно потужні і можуть змінювати все - від зовнішнього вигляду до відчуття інтерфейсу Atom і навіть базової роботи навіть основних функціональних можливостей Atom.

Рис. 2.1. Зовнішній вигляд робочої області в редакторі Atom

В якості мови програмування, що використовується для програмування модуля було обрано Python[20]. Python – об’єктно-орієнтована мова програмування високого рівня. Філософія дизайну Python наголошує на читабельності коду завдяки помітному використанню значних відступів. Його мовні конструкції, а також об’єктно-орієнтований підхід мають на меті допомогти програмістам писати чіткий логічний код для малих та великих проектів. Python використовує динамічне введення тексту та комбінацію підрахунку посилань та збирач сміття для виявлення циклів для управління пам’яттю. Він також має динамічну роздільну здатність імен, яка зв'язує імена методів та змінних під час виконання програми.

Основна філософія мови узагальнена у документі Zen of Python (PEP 20), який включає такі афоризми як:

- Красиве краще ніж потворне.
- Явне краще ніж неявне.
- Просте краще ніж складніше
- Комплекс краще ніж ускладнення
- Читабельність враховується.

Замість того, щоб усі його функціональні можливості були вбудовані в його ядро, Python був розроблений таким чином, щоб бути дуже розширюваним (з модулями). Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих програм.

Переваги[21] мови програмування Python:

1. Легко читати, вивчати та писати. Python - це мова програмування високого рівня, яка має синтаксис подібний до англійської мови, це полегшує читання та розуміння коду.

2. Підвищена продуктивність Python. Завдяки простоті Python розробники можуть зосередитись на вирішенні проблеми, їм не потрібно витрачати занадто багато часу на розуміння синтаксису чи поведінки мови програмування.

3. Python - це інтерпретована мова, що означає, що Python безпосередньо виконує код рядок за рядком. У разі будь-якої помилки він зупиняє подальше виконання та повідомляє про помилку, що сталася. Python показує лише одну помилку, навіть якщо програма має кілька помилок, це полегшує налагодження коду.

4. Python не знає тип змінної, поки код не буде запущено. Він автоматично призначає тип даних під час виконання, тобто програмісту не потрібно турбуватися про оголошення змінних та їх типів даних.

5. Python підпадає під ліцензію з відкритим кодом, затверджену OSI. Це дозволяє використовувати та розповсюджувати його безкоштовно. Ви можете завантажити вихідний код, змінити його і навіть поширити свою версію Python.

6. Підтримка величезних бібліотек. Стандартна бібліотека Python величезна, там можна знайти майже всі функції, необхідні для виконання поставленого завдання.

7. Переносимість. У багатьох мовах, таких як C / C ++, вам потрібно змінити код, щоб запускати програму на різних платформах, але для Python це інакше. Код пишеться один раз і запускається його де завгодно.

Сфери застосування[22] мови програмування Python:

1. Системне адміністрування та автоматизація задач. Дана мова програмування є досить простою, потужною і підтримує спеціальні пакети, також перевагою даної мови є те, що вона за замовчуванням встановлена на всі сервери з операційною системою Linux.

2. Наукові випробування. Python має ряд бібліотек, які використовуються для проведення досліджень і обрахунків:

- SciPy – бібліотека з науковими інструментами.

- NumPy – розширення, що додає підтримку матриць і багатовимірних масивів, а також математичні функції для роботи з ними.

- Matplotlib – бібліотека для роботи з 2D- і 3D-графікою.

3. Data Science. Мовою програмування Python пишуть алгоритми програм з машинним навчанням і аналітичні додатки. За допомогою даної мови проводиться обслуговування сховища даних і хмарні сервіси.

2.2 Опис програмного забезпечення.

Відповідно до порівняльного аналізу методів генерування псевдовипадкових послідовностей, проведеному у Розділі 1, за основу криптографічного модуля було вирішено взяти вихор Мерсенна, адже він позбавлений недоліків інших методів і проходить тести для перевірки на випадковість, що буде продемонстровано далі у цьому розділі . Оскільки сам по собі Вихор Мерсенна не є криптографічно стійким, його було використано для генерування ключової послідовності для симетричного потокового шифрування відкритого повідомлення.

Метод Вихор Мерсенна був запропонований в 1997 році японськими вченими Макото Мацумото і Такудзі Нисимура. Метод заснований на властивості простих чисел Мерсенна і має ряд переваг щодо багатьох інших генераторів ПВЧ. «Вихор» - це перетворення, яке забезпечує рівномірний розподіл ПВЧ.

Алгоритм Вихор Мерсенна складається з почергового виконання процедур рекурсивної генерації і «загартування». Рекурсивна генерація вдає з себе регістр зсуву з лінійним зворотнім зв’язком з додатковою рекурсивною функцією для потоку вихідних бітів. Операція «загартування» є процедурою, що підсилює рівномірність розподілу на великих розмірностях бітових векторів.

Існує декілька варіантів реалізації Вихора Мерсенна. У даній роботі було реалізовано генератор MT19937, фактично даний генератор ПВС є регістром зсуву з лінійним зворотнім зв’язком, що складається з 624 комірок по 32 біти. Метод Вихор Мерсенна дозволяє генерувати послідовність двійкових

псевдовипадкових цілих w-бітових чисел у відповідності з наступною формулою:

$$\boldsymbol{X_{n+p} = X_{n+p} \oplus \left(X_n^r \middle| X_{n+1}^l\right) A} \quad (n = 0, 1, 2, \ldots) \qquad (2.1)$$

де p, q, r – цілі константи, p – степінь рекурентності, 1≤ q ≤ p;

X_n - w-бітове двійкове ціле число;

$\left(X_n^r \middle| X_{n+1}^l\right)$ – двійкове ціле число, отримане в результаті конкатенації чисел X_n^r і X_{n+1}^l у тому ж порядку.

Алгоритм складається з почергового виконання процедур рекурсивної генерації і «загартування». Рекурсивна генерація являє собою регістр зсуву з лінійним зворотнім зв'язком з додатковою рекурсивною функцією для потоку вихідних бітів. Операція «загартування» є процедурою, що підсилює розподіл на великих розмірностях бітів.

Переваги даного алгоритму:

1. Дозволена ліцензія та вільний від патенту для всіх варіантів, крім CryptMT.
2. Проходить численні перевірки на випадковість, включаючи тести Дієрда та більшість, але не всі тести TestU01.
3. Має дуже довгий період.
4. k-розподілений до 32-бітової точності для кожного 1 ≤ k ≤ 623.
5. Реалізація генерує псевдовипадкові числа швидше ніж справжні випадкові методи.
6. Дослідження показали, що Вихор Мерсенна створює 64-розрядні випадкові числа з плаваючою комою приблизно у двадцять разів швидше ніж апаратно реалізований набір інструкцій RDRAND на основі процесора.

Недоліки алгоритму Вихор Мерсенна:

1. Посередня пропускна здатність за сучасними стандартами, якщо використовується варіант SFMT.

2. Дослідження виявляє дві явні помилки як у Crush, так і в BigCrush у наборі тестів TestU01.

3. Кілька екземплярів, що відрізняються лише початковим значенням не підходять для моделювання Монте-Карло, що вимагає незалежних генераторів випадкових чисел, хоча існує метод вибору декількох наборів значень параметрів.

Розглянемо схему роботи створеного криптографічного модуля.

Рис. 2.2. Схема роботи програми

Етап 1. Введення початкового значення генератора. Початкове значення може бути довільним додатнім цілим числом, кращі послідовності генеруються за умови, що початкове значення генератора – просте число. Початкове значення генератора є ключем подальшого шифрування. Всі інші параметри алгоритму є завчасно підібраними.

Етап 2. Відбувається генерування послідовності відповідно до алгоритму.

Крок 1а. Ініціалізація значення u, h, a за формулою:

u := (1,0,...,0) – всього w-r біт, h := (0,1,...,1) – всього r біт, a := (a_{w-1}, a_{w-2},…, a_0) – останній рядок матриці A.

Крок 1б. X_0, X_1,…, X_{p-1} заповнюються початковими значеннями.

Крок 2. Обчислюється $Y := (y_0, y_1, \ldots, y_{w-1}) := \left(X_n^r \middle| X_{n+1}^l\right)$

Крок 3. Обчислюється ноче значення X_i:

$$X_n := X_{(n+q)\ mod\ p} \oplus (\mathrm{Y} \gg 1) \oplus a, \text{якщо молодший біт } y_0 = 1$$

$$X_n := X_{(n+q)\ mod\ p} \oplus (\mathrm{Y} \gg 1) \oplus 0, \text{якщо молодший біт } y_0 = 0$$

Крок 4. Обчислюється X_iT (проводиться операція «загартування»)

$$\boldsymbol{Y} := \boldsymbol{X_n} \tag{2.2}$$

$$\boldsymbol{Y} := \boldsymbol{Y} \oplus (\boldsymbol{Y} \gg \boldsymbol{u})$$

$$\boldsymbol{Y} := \boldsymbol{Y} \oplus ((\boldsymbol{Y} \ll \boldsymbol{s}) \cdot \boldsymbol{b})$$

$$\boldsymbol{Y} := \boldsymbol{Y} \oplus ((\boldsymbol{Y} \ll \boldsymbol{t}) \cdot \boldsymbol{c})$$

$$\boldsymbol{Z} := \boldsymbol{Y} \oplus (\boldsymbol{Y} \gg \boldsymbol{l})$$

Z подається на вихід, як результат.

Крок 5. n:= (n +1) mod p. Перехід на крок 2.

Після проведення всіх необхідних обрахунків користувачу виводиться згенерована послідовність.

Етап 3. Задля того, щоб згенеровану послідовність можна було перевірити за допомогою тестів вона переводиться у двійковий вигляд і виводиться на екран.

Етап 4. Проводиться тестування отриманої послідовності за допомогою статистичних тестів FIPS – 140-1[23].

Якість генератора ПВЧ та створеної ним послідовності перевіряють на схожість до істинно випадкової послідовності за допомогою певних тестів. Існують графічні та статистичні тести. У графічних тестах статистичні властивості рядка відображаються у вигляді графічних залежностей, на основі яких робляться висновки про властивості досліджуваної послідовності. Ця група включає такі способи представлення тестувань, як гістограма розподілу елементів, розподіл елементів плану, верифікація серії, перевірка монотонності, функція автокореляції, графічний спектральний тест. При тестуванні У статистичних тестах статистичні властивості послідовностей визначаються числовими характеристиками. На основі критеріїв оцінки робляться висновки про ступінь близькості аналізованих і справді випадкових послідовностей.

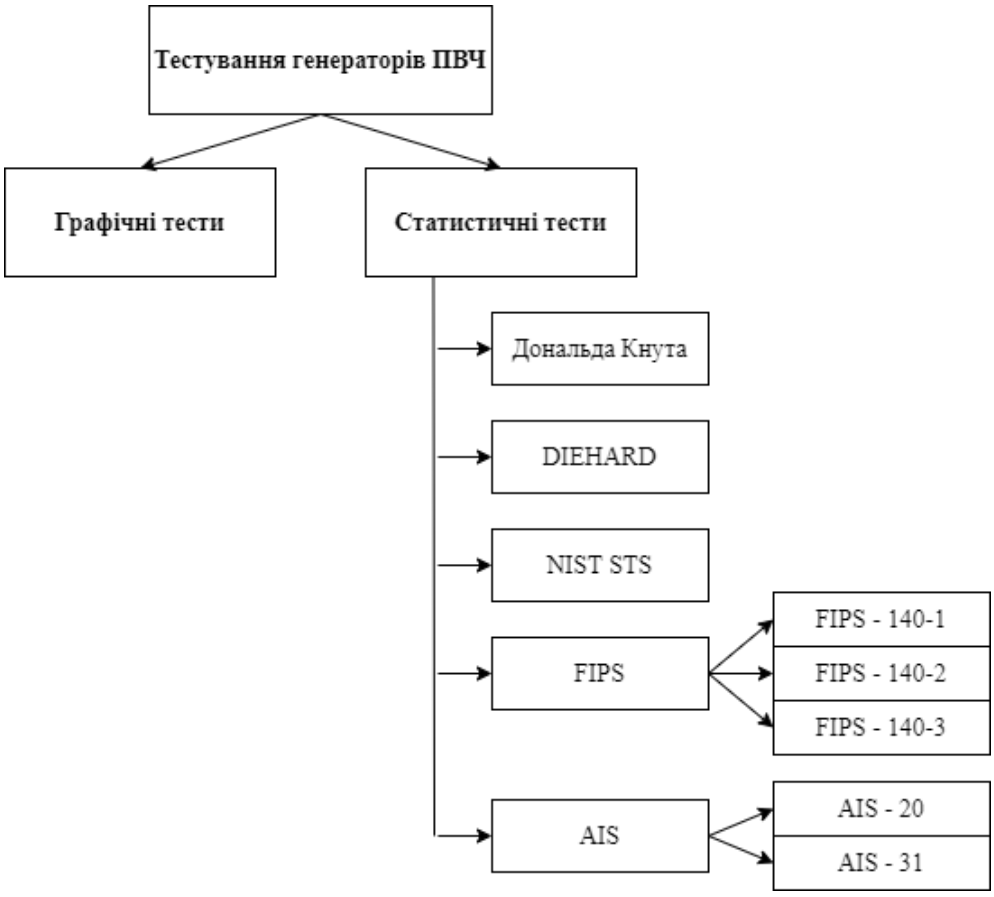


Рис. 2.3. Класифікація методик тестування генераторів ПВЧ

В даній роботі було використано тести FIPS – 140-1. Американський федеральний стандарт FIPS 140-1 визначає чотири статистичних тести на випадковість: монобітний тест, блоковий тест, тест серій, тест довжини серій. Для цих тестів задаються межі для задовільних статистичних параметрів.

Окремий бітовий рядок довжиною 20000 бітів, що отримується з генератора, піддається кожному з чотирьох наведених тестів. Якщо який-небудь тест не буде пройдено, то вважається, послідовність не пройшла увесь комплекс перевірок.

1. Монобітний тест[24]

Суть тесту полягає в підрахунку кількості нулів і одиниць на відрізку послідовності довжиною 20000 біт. Нехай n1 і n2 позначають число нулів і одиниць у послідовності. Якщо послідовність випадкова, то значення n1 і n2 мають задовольняти умові 9654 <n1 (n2) <10346.

2. Блоковий тест

Нехай m додатне ціле число, таке, що $\left[\frac{n}{m}\right] \geq 5 \cdot (2^m)$ і нехай $k = \left[\frac{n}{m}\right]$. Розіб'ємо послідовність x на k послідовностей, що не перетинаються, кожна довжиною m, і нехай n_i буде числом появ i-го типу послідовності довжиною m. Блоковий тест визначає, чи дійсно послідовність довжиною m, з'являється приблизно стільки ж разів у послідовності x, скільки очікується для випадкової послідовності. Для застосування критерію використовується розрахунок параметра

$$\boldsymbol{X_3 = \frac{2^m}{k} \cdot (\sum_{i=1}^{2^m} n_i{}^2) - k,} \qquad (2.3)$$

який узгоджується з розподілом $\chi 2$ з $2^m - 1$ ступенями свободи. Статичний пареметр, що задається рівнянням, обчислюється для m = 4. Статистика повинна задовольняти умов1, що 1,03 <X_3 <57,4.

3. Тест серій

Під серією розуміється послідовність однакових символів, тобто з послідовних одиниць і нулів. Суть тесту полягає у тому, що на заданій довжині послідовності, що тестується, проводиться підрахунок серій довжиною 1, 2, 3, 4, 5, 6 елементів (серії довжиною більше ніж 6 елементів розглядаються як серії довжиною 6). Якщо послідовність випадкова, то кількість серій кожної довжини має знаходитись в певних інтервалах.

4. Тест довжини серій.

Суть тесту полягає у перевірці максимальної довжини серії однакових елементів. Якщо послідовність випадкова, то максимальна довжина серії не повинна перевищувати значення 34. (Ймовірність події, що заключається у появі серії такої довжини дуже мала).

Відповідно до Федерального стандарту у програмному модулі було передбачено умову, що якщо послідовність не пройде принаймні один тест, то вважатиметься, що перевірку не пройдено. Проведення даного тестування дає змогу визначити, чи придатною для подальшого використання є отримана послідовність.

Етап 5. На даному етапі відбувається перехід до синхронного потокового шифрування[25]. Потокові шифри представляють собою різновид гамування та перетворюють відкритий текст в шифрований послідовно, по одному біту. Генератор ключової послідовності, видає послідовність біт k_1, k_2, ..., k_i. Ця ключова послідовність складається по модулю 2 з послідовністю біт вихідного тексту p_1, p_2, ..., p_i для отримання шифрованого тексту $c= p_i \oplus k_i$. На приймальній стороні текст складається по модулю 2 з ідентичною ключовою послідовністю для отримання вихідного тексту. Таке перетворення називається гамуванням за допомогою операції XOR.

У синхронному потоковому шифрі потік псевдовипадкової послідовності генерується незалежно від відкритого та зашифрованого повідомлень, а потім поєднується із відкритим текстом (для шифрування) або зашифрованим текстом (для дешифрування).

В основу потокового шифрування покладено ідею використання операції додавання за модулем два (що виключає «або», xor) вихідного тексту з деякою гамою. Гама створюється за допомогою генераторів псевдовипадкових чисел.

Процес дешифрування даних зводиться до повторної генерації гами шифру при відомому ключі і накладення такої гами на зашифровані дані.

Отриманий зашифрований текст є досить важким для розкриття в тому випадку, якщо гама шифру не містить бітових послідовностей, що повторюються. По суті гама шифру повинна змінюватися випадковим чином

для кожного слова, що шифрується. Фактично ж, якщо період гами перевищує довжину всього зашифрованого тексту і не відома ніяка частина вихідного тексту, то шифр можна розкрити тільки прямим перебором (пробою на ключ). Крипостійкість в цьому випадку визначається розміром ключа.

Потокові шифри підходять для швидких реалізацій з низьким споживанням ресурсів. Ці дві функції допомагають захиснику застосовувати стратегії опору на пристроях, які можуть не мати ресурсів для реалізації блочного шифру. Потокові шифри також корисні для шифрування бездротових сигналів, які більш природно відповідають потоковій моделі, ніж передача даних більшими фрагментами фіксованого розміру.

Першим кроком реалізації шифрування є завантаження відкритого тексту з файлу.

Етап 6. На даному етапі відбувається шифрування відкритого тесту шляхом виконання операції додавання за модулем два двійкових представлень відкритого повідомлення та ключової послідовності.

Етап 7. Повідомлення, отримане в результаті шифрування, виводиться на екран, а також записується у файл для його подальшої передачі отримувачу.

Етап 8. На наступному етапі було розглянуто сторону отримувача зашифрованого повідомлення. Спочатку відбувається завантаження шифротексту з файлу у програму для подальшої роботи з ним.

Етап 9. На даному етапі відбувається розшифрування повідомлення шляхом додавання за модулем два двійкового представлення шифротексту, а також ключової гами, згенерованої отримувачем з використанням такого ж ключа, як і відправник. Використання однакового ключа призводить до створення однакової ключової послідовності, завдяки чому повідомлення вдається розшифрувати.

Етап 10. Розшифроване повідомлення виводиться на екран для демонстрації правильності виконання розшифрування.

2.3.Дослідження розробленого програмного модуля.

Першим кроком після запуску програми є введення початкового значення генератора (ключа шифрування). Вважається, що чим довшим є ключ, тим надійнішою є згенерована послідовність. Інші параметри, необхідні для роботи генератора, за замовчування внесенні у код програми, адже є завчасно підібраними для даного варіанту реалізації алгоритму. Після введення відповідного значення на екран виводяться усі параметри генератора (табл. 2.1.).

Таблиця 2.1.

Параметри 32-бітного генератора

Параметр	**Назва параметру**	**Значення**	**Примітка**
1	2	3	4
w	Розмір слова	32	Кількість бітів кожного слова в послідовності стану
n	Розмір стану	624	Кількість елементів у послідовності станів (ступінь повторюваності)
q	Розмір зсуву	397	Розмір зсуву, який використовується у вихорі для перетворення значень
r	Маска бітів	31	Кількість бітів, що позначають розділення слів на кожному повороті
f(a)	Маска XOR (множник ініціалізації)	0x9908B0DF	Маска XOR, що застосовується як лінійна функція при кожному повороті.
u	Параметр u	11	Розмір зсуву параметра u, що застосовується в процесі «загартування» алгоритму генерації
d	Параметр d	0xFFFFFFFF	Маска XOR, що використовується як параметр d у процесі «загартування» алгоритму генерації

Продовження таблиці 2.1

1	2	3	4
s	Параметр s	7	Розмір зсуву параметра s, що застосовується в процесі «загартування» алгоритму генерації
b	Параметр b	0x9D2C5680	Маска XOR, що використовується як параметр b у процесі «загартування» алгоритму генерації
t	Параметр t	15	Розмір зсуву параметра t, що застосовується в процесі «загартування» алгоритму генерації
c	Параметр c	0xEFC60000	Маска XOR, що використовується як параметр c у процесі «загартування» алгоритму генерації
l	Параметр l	18	Розмір зсуву параметра l, що застосовується в процесі «загартування» алгоритму генерації

Параметри n і r підібрані так, що характеристичний многочлен (nw – r) був рівним числу Мерсенна 19937. w - розмір слова комп'ютера, для реалізації даного алгоритму використовується 32 біти. Параметри «загартування» підібрані так, щоб отримати рівномірний розподіл.

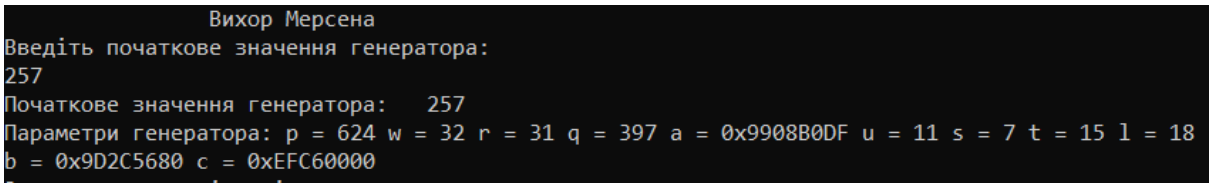


Рис. 2.4. Параметри генератора

Після цього на екран виводиться згенерована послідовність у десятковому вигляді. Елементами послідовності є 32-розлядні елементи.

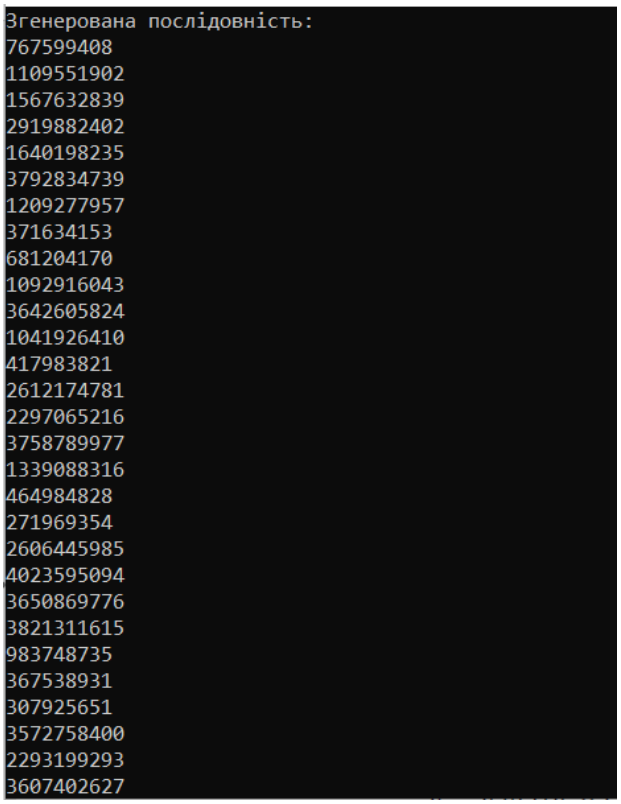


Рис. 2.5. Виведення згенерованої послідовності

Далі відбувається переведення послідовності у двійковий вигляд для її подальшого тестування.


Послідовність у двійковому представленні:
['10110111000000101000110011000010000100010001001101011000111101011101011100000010110111000111
10101110000010011101111010100010111000011100001101110000010110111100010000100100001000010110011
11001000000101000001111000000101101100010011010101111111010011010001001101001011001110010100
00010010010010010011010010111101100100011101110000010000000011111000011010100010010000101011100
01110100111101101010011011001101110110010100111111011110110001000111010100110111100000000111100
00000000101010010101010110011001111110100001101110110111100110111011011100011010111111001000000
11010111101100010010100110110101101110011010110100001111011111101001100110000010101101101100 1
10011011110110100001000011100011110001001001011001111111111101010100010110100000111111101011111
01000001100101111001110010010110101001001010010011110101001111001111101111000000100010001010
11110111000110111011101011100000100100110001000001110110011100111011010000101000111111101000 0
11101000111000010110010100000101001001011100011010011100101000011001101110011101110010111101 00
00111111000001001010010001100001101000101000110010011110110011011101101011110010110111000100 00
10101011111100011100111010001100010010001110101011010100011010101100011101000111000010001111100000110
10011100100111011110010000011110000001100111000011101110011101010001110000010010010100011110 00
00001101100001101011010001111000001000110101111101101110001011011010010100100111011011000000 01
11101000110001101000010000001111011010010100011111010101111110111010100000011011010110011010 1
11010011011010110001000001101111111111111010110101110001010000011010000000100111111001100010010
00001110001010110001101011010101000001001111100101111000110010000101001000110100100010110011101
00010110101101110011100000101001101000001101110010001100100101100101101101011110101111010000011000010
11010110011011000001100111100101001100100111110100010001110100100000000011011001101000101100 01
10101000000011111111000100111100111000101011100100000001100001111101010001111001110001111010 10
11010101011111010011110010000010001101100011101100100110000110111011110001110100010100101010 10
10000000000111001011111101000100011111111111000101010000001001110101101001100111100110010110 1
10111110111101011101001001001100000111001010010011101101111001001011100111100001011000101101 11
10011111010011001010111101001100011111010011110000011101111101101101011011111000000000011111 1
11011101011001011001011011000111011110011101001100011101000010111100111011011111011010011000111
01000010000110000001110111001001011011111001011001010010101100010101110110111110010000101110 10
01100100000010100000100110100000111111101000101101100110001111101000101111110111101110100111 11
01001001010100000111100001101001110101100110110110101000111101011111010000001010110101000011110

Рис. 2.6. Виведення послідовності у двійковому вигляді

Проводиться тестування послідовності, якщо вона пройшла усі перевірки, то виводиться результат, показаний на рис 2.7, після чого відбувається зчитування з файлу відкритого тексту, який буде далі шифруватися.

```
        Тести FIPS 140-1:
        Монобітний тест:
Кількість бітів у двійковій послідовності: 19974
Кількість одиниць: 10274
Тест пройдено!
        Блоковий тест:
Значення x =  21.824000000000524
Тест пройдено!
        Тест серій:
Тест пройдено!
        Тест довжини серій:
Тест пройдено!
Всі тести FIPS 140-1 пройдено!
```

Рис. 2.7. Приклад успішного проходження послідовністю всіх тестів

Якщо послідовність не пройшла принаймні один тест, то на екран будуть виведені результати та повідомлення про те, що дана послідовність не підходить для подальшого її використання. Після цього роботу програми буде припинено.

```
        Тести FIPS 140-1:
        Монобітний тест:
Кількість бітів у двійковій послідовності: 20050
Кількість одиниць: 10338
Тест пройдено!
        Блоковий тест:
Значення x =  64.84160000000065
Тест не пройдено! Значення x має бути в межах від 1.03 до 57.4
        Тест серій:
Тест пройдено!
        Тест довжини серій:
Тест пройдено!
Тести не пройдено! Результати: Монобітний тест: True. Блоковий тест: False. Тест серій: True. Тест довжини серій: True
Згенерована послідовність не проходить комплекс статистичних тестів FIPS - 140-1, а, отже, не є близькою до істинно випадкової.
 Дану послідовність не варто використовувати для шифрування відкритого тексту!
Для виходу з програми натисніть будь-яку клавішу ...
```

Рис. 2.8. Приклад випадку, коли послідовність не пройшла перевірку

Після зчитування відкритий текст виводиться на екран

```
        Відкритий текст:
National Aviation University is one of the most powerful aviation-related higher educational facilities in the world, with
about 25,000 students enrolled, including nearly 1,500 foreigners from 55 countries. Rector of National Aviation University
 is a Candidate of Technical Sciences,  Rimvidas Khrashchevskyi.

The origin of the university dates back to aviation courses organized by Kyiv Polytechnic Institute in the late XIX century
; while the independent history began in 1933 by the Resolution of the Council of People's Commissars of the USSR; accordin
g to it Kyiv Aviation Institute was founded at the aviation department of Kyiv Polytechnic Institute.

Later its name was being changed many times: Kyiv Institute of Civil Aviation (1947), Kyiv Institute of Civil Aviation (196
5), Kyiv International University of Civil Aviation (1994), National Aviation University (2000).

For  85 years of its history more than 200,000 highly skilled professionals has been trained in this higher aviation educat
ional institution. Among them there are well-known scientists, heads of aviation companies, enterprises, organizations and
institutions that provide aircraft flights, their maintenance and repair, transportation of passengers and cargo.
```

Рис. 2.9. Виведення відкритого повідомлення

Для подальшого шифрування текст переводиться у двійковий вигляд і виводиться на екран.

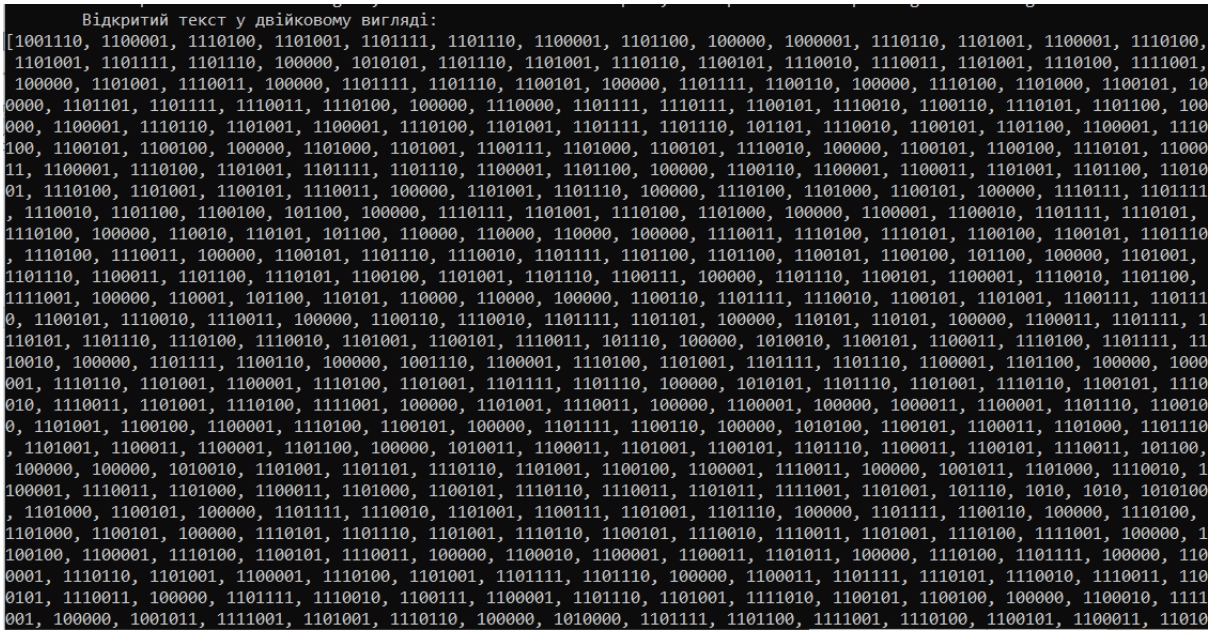


Рис. 2.10. Виведення відкритого тексту у двійковому вигляді

Шифрування повідомлення відбувається шляхом виконання операції XOR відкритого повідомлення у двійковому представленні та згенерованої послідовності у двійковому представленні. На екран виводиться результат шифрування, а також записується у файл.

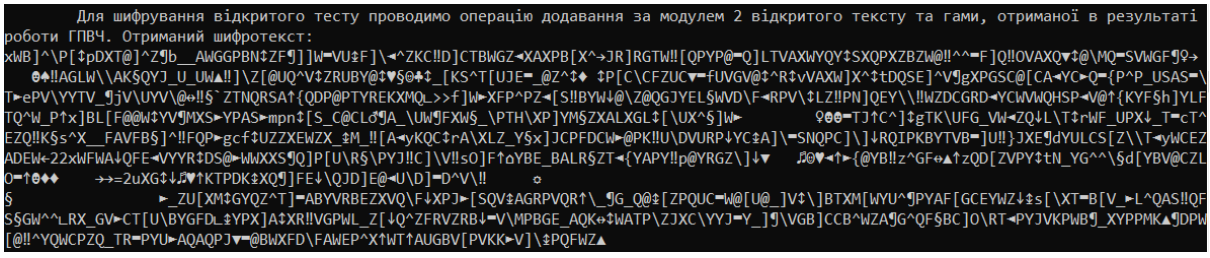


Рис. 2.11. Зашифроване повідомлення

Для розшифрування повідомлення шифротекст спочатку зчитується з файлу для подальшої роботи з ним.

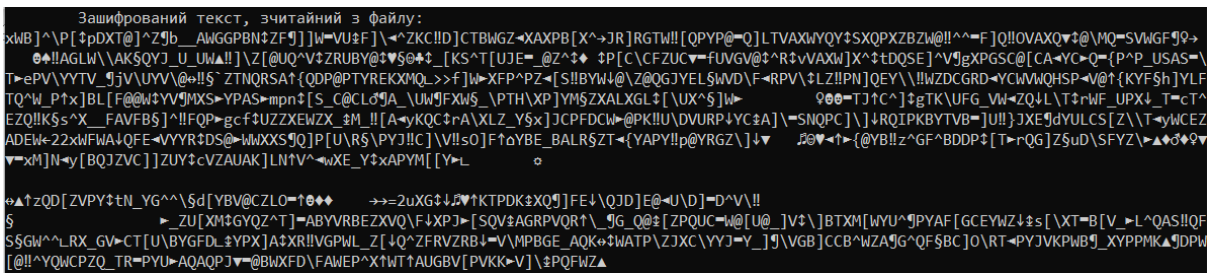


Рис. 2.12. Зашифрований текст, зчитаний з файлу

Розшифрування відбувається виконанням операції додавання за модулем два шифротексту та послідовності згенерованої з ключем, ідентичним до ключа за допомогою якого було зашифровано повідомлення. Як видно з рис. 2.13. розшифрування пройшло успішно, так як отриманий текст повністю відповідає відкритому тексу, що був використий.

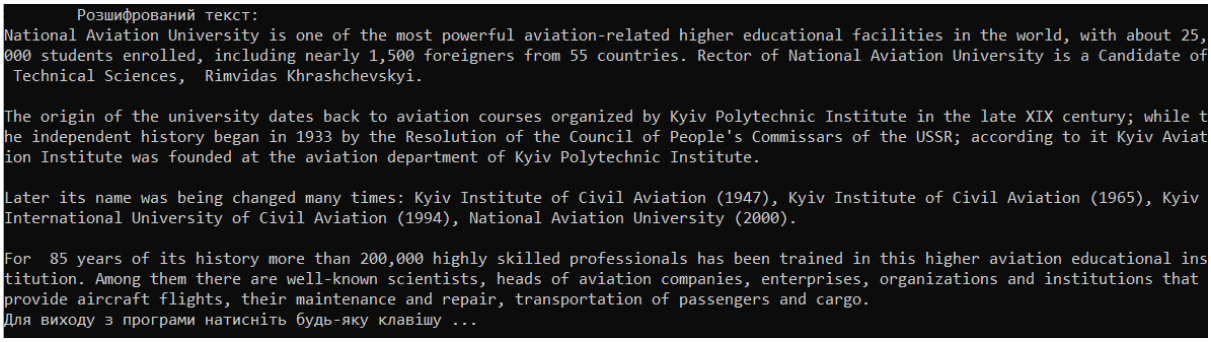


Рис. 2.13. Виведення розшифрованого тексту

Висновки до розділу 2

Відповідно до поставленої прикладної задачі проводиться вибір середовища розробки та мови програмування для її реалізації. У даній роботі було зроблено вибір на користь редактора вихідного коду Atom та мову програмування Python за їх доступність, простоту використання та наявність всіх необхідних для реалізації алгоритму функцій.

Відповідно до порівняльного аналізу проведеного у попередньому розділі для реалізації було обрано метод генерування псевдовипадкових послідовностей Вихор Мерсенна. Даний метод позбавлений недоліків, присутніх в інших алгоритмах, він має довгий період, не є абсолютно передбачуваним і породжує статистично хорошу послідовність, яку можна використовувати у багатьох галузях. У даному розділі було продемонстровано етапи генерування послідовності та їх програмне відображення.

Якість згенерованої послідовності визначається її успішним чи неуспішним проходженням тестів. Для перевірки було обрано статистичні

тести FIPS – 140-1, засновані на американському федеральному стандарті, вони включають в себе чотири тести, такі як монобітний тест, блоковий тест, тест серій та тест довжини серій. Стандарт передбачає умову, що для того, щоб послідовність могла бути ідентифікована як близька до істинно випадкової, вона має успішно пройти усі наведені вище тести. Якщо принаймні один тест не вдасться пройти, вважатиметься, що псевдовипадкова послідовність провалила увесь комплекс тестів. Саме тому було продемонстровано результат програми для обох варіантів згенерованих псевдовипадкових послідовностей.

Оскільки Вихор Мерсенна сам по собі не є криптографічно стійким генератором, то його самостійне використання не є рекомендованим. Задля забезпечення конфіденційності повідомлення, що передається було використано синхронне потокове шифрування, яке використовує обраний алгоритм генерування псевдовипадкових чисел для формування ключової послідовності. Шифротекст отримується в результаті виконання операції XOR відкритого тексту та згенерованої послідовності.

Перевагами синхронного потокового шифрування є те, у в ньому відсутнє поширення помилок, оскільки тільки спотворений біт розшифрується неправильно, також дане шифрування створює захист від модифікації тексту, адже відповідні зміни буде виявлено. Отже використання даного шифру дозволило забезпечити конфіденційність, адже користувач, що не знає ключа, не зможе розшифрувати повідомлення, та цілісність інформації.

ВИСНОВКИ

У процесі виконання дипломної роботи було проведено дослідження сучасних методів захисту інформації та принципів побудови та реалізації генераторів псевдовипадкових чисел. Існують криптографічно стійкі і нестійкі генератори, на які покладено різні функції. Багато прикладних завдань криптографії вимагають випадкових чисел, наприклад, генерація ключів, одноразові випадкові числа, одноразові шифроблокноти, сіль в схемах цифрового підпису. Відповідно до поставленої прикладної задачі змінюється вибір генератора для її вирішення. Було проведено порівняльний аналіз сучасних методів генерування псевдовипадкових чисел.

Результатом виконання роботи є розроблений криптографічний модуль для шифрування інформації методом синхронного потокового шифрування з використанням Вихору Мерсенна для генерування ключової послідовності. Згенеровану послідовність було протестовано за допомогою комплексу статистичних тестів FIPS – 140-1. Перевірка результату дає змогу оцінити схожість псевдовипадкової послідовності з істинно випадковою та зробити висновки про можливість її подальшого використання.

Оскільки обраний генератор псевдовипадкових чисел сам по собі не є криптографічно стійким, тобто самостійно його використовувати не рекомендується, він був взятий за основу генерування ключової послідовності потокового шифру. Синхронний потоковий шифр дає змогу забезпечити конфіденційність даних, оскільки користувач, що не знає ключа (початкового значення генератора) не зможе розшифрувати повідомлення, що передається. Також даний метод шифрування дає змогу побачити випадки, коли зміст тексту намагались змінити, адже спотворені біти буде одразу видно.

Створений криптографічний модуль дає змогу користувачу створити власну послідовність, провести її тестування для визначення можливості її використання. У випадку, коли послідовність проходить всі тести, користувач може зашифрувати відкритий текст та передати уже зашифрований. Даний

програмний модуль можна використовувати з метою забезпечення конфіденційності інформації, що передається, адже тільки адресату і одержувачу відомий ключ шифрування.

Відповідно до поставленого завдання у результаті виконання дипломної роботи було отримано такі результати:

- проведено дослідження сучасних принципів побудови та реалізації генераторів псевдовипадкових чисел, що дозволило вибрати для подальшої реалізації Вихор Мерсенна для генерування ключової послідовності;
- розроблено авторський криптографічний модуль забезпечення конфіденційності інформації на базі синхронного потокового шифру з використанням генератора псевдовипадкових Вихор Мерсенна мовою програмування Python;
- проведено дослідження працездатності власного програмного модуля, що дозволяє наглядно продемонструвати етапи генерування псевдовипадкової послідовності, провести процедуру статистичного тесту згідно FIPS–140-1 та реалізувати синхронний потоковий шифр для забезпечення конфіденційності інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Josh Fruhlinger What is information security? Definition, principles, and jobs. (17 January 2020) [Електронний ресурс] – Режим доступу до ресурсу: https://www.csoonline.com/article/3513899/what-is-information-security-definition-principles-and-jobs.html

2. Закон України «Про внесення змін до Кримінального кодексу України щодо відповідальності за незаконне втручання в роботу мереж електрозв'язку», 5 червня 2003 року.

3. Положення про порядок розроблення, виробництва та експлуатації засобів криптографічного захисту конфіденційної інформації та відкритої інформації з використанням електронного цифрового підпису [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0868-99#Text

4. Bayram G. Ibrahinov Cryptographic Methods And Means Protection Transmitted Information in Telecommunication Systems/ Bayram G. Ibrahinov, Ramiz T. Humbatov., Rufat F. Ibrahimov (2018) [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S2405896318328519

5. Jeaneth Machicao A visual analysis method of randomness for classifying and ranking pseudo-random number generators/ Jeaneth Machicao, Quynh Quang Ngo, Vladimir Molchanov, Lars Linsen, Odemir Bruno (2020) [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0020025520310264

6. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях/ Иванов М.А ,Чугунков И.В. Москва 2012 – 400с.

7. Linear Congruence method for generating Pseudo Random Numbers (2021)[Електронний ресурс] – Режим доступу до файлу:

https://www.geeksforgeeks.org/linear-congruence-method-for-generating-pseudo-random-numbers/

8. Ying Tan GPU-Based Random Number Generators [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/B9780128093627500108

9. The Mersenne Twister [Електронний ресурс] – Режим доступу до ресурсу: http://www.quadibloc.com/crypto/co4814.htm

10. George Marsaglia Xorshift RNGs. 2003

11. Cryptographic Services ICSF Application Programmer's Guide [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/docs/en/zos/2.1.0?topic=keys-ansi-x917-key-management-services

12. FIPS 186-3 FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION Digital Signature Standard (DSS) (2009) [Електронний ресурс] – Режим доступу до ресурсу: https://csrc.nist.gov/CSRC/media/Publications/fips/186/3/archive/2009-06-25/documents/fips_186-3.pdf

13. Слеповичев И.И. Генераторы псевдослучайных чисел. 2017. – 118с.

14. CRC32 function [Електронний ресурс] – Режим доступу до ресурсу: https://docs.aws.amazon.com/redshift/latest/dg/crc32-function.html

15. MD4 [Електронний ресурс] – Режим доступу до ресурсу: http://kriptografea.narod.ru/MD4.html

16. MD5 [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/MD5

17. RC4 [Електронний ресурс] – Режим доступу до ресурсу: http://kriptografea.narod.ru/RC4.html

18. RSA: от простых чисел до электронной подписи [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/post/534014/

19. Overview of Atom IDE [Електронний ресурс] – Режим доступу до ресурсу: https://atom.io/

20. Python [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python

21. Python Advantages and Disadvantages – Step in the right direction [Електронний ресурс] – Режим доступу до ресурсу: https://techvidvan.com/tutorials/python-advantages-and-disadvantages

22. Язык программирования Python: преимущества, недостатки и область применения [Електронний ресурс] – Режим доступу до ресурсу: https://skillbox.ru/media/code/dlya_chego_nuzhen_python/

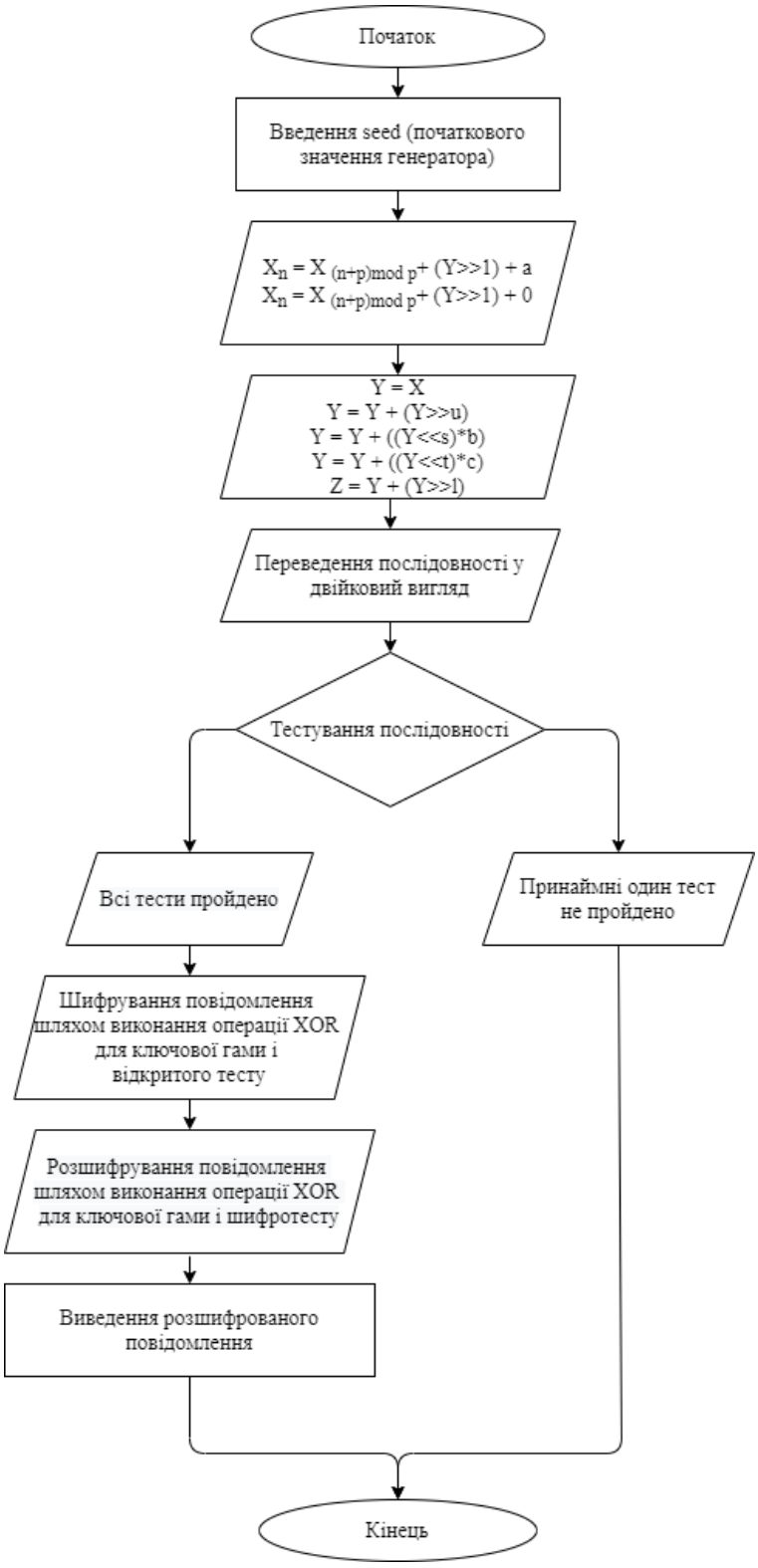
23. Federal Information Processing Standard (FIPS) 140-1 [Електронний ресурс] – Режим доступу до ресурсу: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=917970

24. Тестування джерел пвп на основі методики fips-140-1 [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/4494505/page:8/

25. Jeff Gilchrist Encyclopedia of Information Systems. 2003, Pages 87-100 [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/B012227240400054X

ДОДАТОК А

БЛОК-СХЕМА ПРОГРАМНОГО МОДУЛЯ



ДОДАТОК Б

ПРИКЛАД ПСЕВДОВИПАДКОВОЇ ПОСЛІДОВНОСТІ

Приклад псевдовипадкової бінарної послідовності:

10011110111000101001001000110010010000111001000101010001001111100011
10110001000101010010100011010011001000111001111010011001100110100111
01001111111100001110100011111010011110011011011101011000101010011011
00000111001111111000101011111011000100101011011100111011111001101110
01010101110101010000010010110011100011110110110001000000100010000101
00110100000011010110010011111001101101001001000000110101111011100101
00100101001101111101110010110001110101011001100010100001011001111000
00011110011111100110000101011000101010101101001010101110010110000100
10010001010101001101011001110000100011001101010011111001010001101011
01010001011000000001001010001100101010001010110010111010010110011001
01101110000011111110110101100010011110111010011100110100000111110101
10110110111010101001111100001110110100101110000101011000110000100111
11001011010100111000101111111110000010001011001110110001011101000001
01100100110110100010011010000100110111111100000001110101001110111011
10100111000101111101100000000100100110011010100111100000110010110111
11101111111011100010101001100011000111110001100111100001100100001001
01111011101000110101010001000001101101010100001000000100010000010101
10000001100101010111010011101011100000000000101101010010111000101100
10100100111111000110110101111001101100011110010011000111

ДОДАТОК В

ВМІСТ ФАЙЛУ З ВІДКРИТИМ ТЕКСТОМ

National Aviation University is one of the most powerful aviation-related higher educational facilities in the world, with about 25,000 students enrolled, including nearly 1,500 foreigners from 55 countries. Rector of National Aviation University is a Candidate of Technical Sciences, Rimvidas Khrashchevskyi.

The origin of the university dates back to aviation courses organized by Kyiv Polytechnic Institute in the late XIX century; while the independent history began in 1933 by the Resolution of the Council of People's Commissars of the USSR; according to it Kyiv Aviation Institute was founded at the aviation department of Kyiv Polytechnic Institute.

Later its name was being changed many times: Kyiv Institute of Civil Aviation (1947), Kyiv Institute of Civil Aviation (1965), Kyiv International University of Civil Aviation (1994), National Aviation University (2000).

For 85 years of its history more than 200,000 highly skilled professionals has been trained in this higher aviation educational institution. Among them there are well-known scientists, heads of aviation companies, enterprises, organizations and institutions that provide aircraft flights, their maintenance and repair, transportation of passengers and cargo.

ДОДАТОК Г

КОД ПРОГРАМИ

```
import sys
from collections import Counter

class mersenne_rng(object):
    def __init__(self, seed = 5489): #5489 початкове значення за замовчуванням
        self.state = [0]*624
        self.f = 0x9908B0DF #Множник ініціалізації
        self.m = 397 #     Розмір зсуву
        self.u = 11  #Розмір зсуву параметра и
        self.s = 7  #Розмір зсуву параметра s
        self.b = 0x9D2C5680  #Маска XOR, що використовується як параметр b
        self.t = 15  #Розмір зсуву параметра t
        self.c = 0xEFC60000  #Маска XOR, що використовується як параметр c
        self.l = 18  #Розмір зсуву параметра l
        self.index = 624  #Кількість елементів у послідовності станів
        self.lower_mask = 0xFFFFFFFF
        self.upper_mask = 0x00000000

        # update state
        self.state[0] = seed
        for i in range(1,624):
            self.state[i] = self.int_32(self.f*(self.state[i-1]^(self.state[i-1]>>30)) + i)

    def twist(self):
        for i in range(624):
```

Продовження додатку Г

```
                temp =
self.int_32((self.state[i]&self.upper_mask)+(self.state[(i+1)%624]&self.lower_mask))
                temp_shift = temp>>1
                if temp%2 != 0:
                    temp_shift = temp_shift^0x9908b0df
                self.state[i] = self.state[(i+self.m)%624]^temp_shift
            self.index = 0

        def get_random_number(self):
            if self.index >= 624:
                self.twist()
            y = self.state[self.index]
            y = y^(y>>self.u)
            y = y^((y<<self.s)&self.b)
            y = y^((y<<self.t)&self.c)
            y = y^(y>>self.l)
            self.index+=1

            return self.int_32(y)
        def int_32(self, number):
            return int(0xFFFFFFFF & number)

    print ("\t\tВихор Мерсена")
    print("Введіть початкове значення генератора:")
    seed = input()  #Початкове значення генератора
    if (len(sys.argv)>1):
            seed=int(sys.argv[1])

    def dec_to_bin(x):
```

Продовження додатку Г

```
    return int(bin(x)[2:])


f = open('sequence.txt', 'w')
a = open('decimal_sequence.txt','w')
binary = []
if __name__ == "__main__":
    rng = mersenne_rng(int(seed))
    print ("Початкове значення генератора:\t",seed)
    print ("Параметри генератора: p = 624 w = 32 r = 31 q = 397 a = 0x9908B0DF u = 11
s = 7 t = 15 l = 18 b = 0x9D2C5680 c = 0xEFC60000")
    print ("Згенерована послідовність: ")
    for i in range(645): #вказується розмір масиву
        sequence = rng.get_random_number()
        a.write(str(sequence))
        print (sequence)
        binary = str(dec_to_bin(sequence))
        f.write(binary)
    f.close()
    a.close()



def monobit_test(binary):
    number_of_ones = 0
    number = 0
    print ("\tМонобітний тест:")
    for bit in binary:
        number += 1
        if bit == '1':
            number_of_ones += 1
```

Продовження додатку Г

```
    if (number_of_ones > 9654) and (number_of_ones < 10346):
        print ("Кількість бітів у двійковій послідовності:", number)
        print ("Кількість одиниць:", number_of_ones )
        print ("Тест пройдено!")
        return True
    else:
        print("Кількість бітів у двійковій послідовності:", number)
        print("Кількість одиниць:", number_of_ones )
        print("Тест не пройдено! Кількість одиниць повинна бути в межах від 9654 до
10346!")
        return False


def poker_test(binary):
    print ("\tБлоковий тест:")
    contiguous_segments = [binary[i:i+4] for i in range(0, len(binary), 4)]
    counter = Counter(contiguous_segments)
    f_i = 0
    for string, amount in counter.items():
        f_i += amount*amount
        x = ((16/5000) * f_i) - 5000

    if (x > 1.03) and (x < 57.4):
        print ("Значення x = ", x)
        print ("Тест пройдено!")
        return True
    else:
        print ("Значення x = ", x)
        print ("Тест не пройдено! Значення x має бути в межах від 1.03 до 57.4")
```

Продовження додатку Г

```
    return False


def runs_test(bin_string):
    print ("\tТест серій:")
    count = 1
    repetitions = []
    counter_0 = [0, 0, 0, 0, 0, 0]
    counter_1 = [0, 0, 0, 0, 0, 0]
    if len(bin_string) > 1:
        for i in range(1, len(bin_string)):
            if bin_string[i - 1] == bin_string[i]:
                count += 1
            else:
                repetitions.append([bin_string[i - 1], count])
                count = 1
        repetitions.append([bin_string[i], count])

    for rep in repetitions:
        if rep[0] == '0':
            if rep[1] >= 6:
                counter_0[5] += 1
            else:
                counter_0[rep[1]-1] += 1
        if rep[0] == '1':
            if rep[1] >= 6:
                counter_1[5] += 1
            else:
                counter_1[rep[1]-1] += 1
```

Продовження додатку Г

```
        zero_ok = False
        one_ok = False
        if (counter_0[0] > 2267) and (counter_0[0] < 2733) and (counter_0[1] > 1079) and
(counter_0[1] < 1421) and \
            (counter_0[2] > 502) and (counter_0[2] < 748) and (counter_0[3] > 223) and
(counter_0[3] < 402) and \
            (counter_0[4] > 90) and (counter_0[4] < 223) and (counter_0[5] > 90) and
(counter_0[5] < 223):
            zero_ok = True

        if (counter_1[0] > 2267) and (counter_1[0] < 2733) and (counter_1[1] > 1079) and
(counter_1[1] < 1421) and \
            (counter_1[2] > 502) and (counter_1[2] < 748) and (counter_1[3] > 223) and
(counter_1[3] < 402) and \
            (counter_1[4] > 90) and (counter_1[4] < 223) and (counter_1[5] > 90) and
(counter_1[5] < 223):
            one_ok = True

        if zero_ok and one_ok:
            print ("Тест пройдено!")
            return True
        else:
            print ("Тест не пройдено! Кількість серій кожної довжини не належать
інтервалам необхідних значень!")
            return False


    def long_run(bin_string):
        print ("\tТест довжини серій:")
```

Продовження додатку Г

```
    count = 1
    if len(bin_string) > 1:
        for i in range(1, len(bin_string)):
            if bin_string[i - 1] == bin_string[i]:
                count += 1
            else:
                if count >= 34:
                    print ("Тест не пройдено! Довжина серії не має перевищувати 34!")
                    return False
                count = 1
    print ("Тест пройдено!")
    return True

if __name__ == '__main__':
    keys = open('sequence.txt', 'r')
    binary_keys = []
    text = open ('text.txt', 'r')
    decimal_text = str(text.read())
    binary_text = []

    for key in keys:
        temp = key[:]
        binary_keys.append(temp)
    print ("Послідовність у двійковому представленні:")
    print (binary_keys)

    for i in range (0, len(decimal_text)):
            binary_text.append(dec_to_bin(ord(decimal_text[i])))
```

Продовження додатку Г

```
binarytext = open('binarytext.txt','w')
binarytext.write(str(binary_text))
binarytext.close()

def xor_strings(s,t):
    return "".join(chr(ord(a)^ord(b)) for a,b in zip(s,t))

encryption = open('decimal_sequence.txt','r')
cipher1 = open('cipher.txt','w')
ciphertext = xor_strings(str(encryption.read()),decimal_text)
cipher1.write(ciphertext)
encryption.close()
cipher1.close()

cipher = open('cipher.txt','r')
decryption = open('decimal_sequence.txt','r')
decrypted_text = xor_strings(str(decryption.read()),ciphertext)
decryption.close()

print("\tТести FIPS 140-1:")
for binary in binary_keys:
    answer_monobit = monobit_test(binary)
    answer_poker = poker_test(binary)
    answer_runs = runs_test(binary)
    answer_long = long_run(binary)
    if answer_monobit and answer_poker and answer_runs and answer_long:
        print ("Всі тести FIPS 140-1 пройдено!")
        print ("\tВідкритий текст:")
        print (decimal_text)
```

Продовження додатку Г

```
            print ("\tВідкритий текст у двійковому вигляді:")
            print (binary_text)
            print ("\tДля шифрування відкритого тесту проводимо операцію додавання за модулем 2 відкритого тексту та гами, отриманої в результаті роботи ГПВЧ. Отриманий шифротекст:")
            print (ciphertext)
            print ("\tЗашифрований текст, зчитайний з файлу:")
            print (cipher.read())
            print ("\tРозшифрований текст:")
            print (decrypted_text)
        else:
            print ("Тести не пройдено! Результати: Монобітний тест: " + str(answer_monobit) + ". Блоковий тест: " +
                   str(answer_poker) + ". Тест серій: " + str(answer_runs) + ". Тест довжини серій: " + str(answer_long))
            print ("Згенерована послідовність не проходить комплекс статистичних тестів FIPS - 140-1, а, отже, не є близькою до істинно випадкової. Дану послідовність не варто використовувати для шифрування відкритого тексту!")

    print ("Для виходу з програми натисніть будь-яку клавішу ...")
    end = input()
```