Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Підвищення захищеності веб-ресурсів стійкими криптографічними алгоритмами на основі генераторів випадкових чисел, що враховують ентропію поведінки користувача у багатокористувацькому середовищі

Виконав: ст. 2-го курсу, групи УБ-20м
спеціальності 125– Кібербезпека
Освітня програма – Управління інформаційною безпекою
(шифр і назва напряму підготовки, спеціальності)

Пархоменко Р. М.
(прізвище та ініціали)

Керівник: к.т.н., доцент, завідувач кафедри МБІС
Карпінець В.В.
(прізвище та ініціали)
«____» _____ 2021 р.

Опонент: к.т.н., доц., доцент каф. ОТ
Савицька Л. А.
(прізвище та ініціали)
«____» _____ 2021 р.

Допущено до захисту
Голова секції УБ кафедри МБІС
д.т.н., проф. Яремчук Ю.Є.
«____» _____ 2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет
Факультет менеджменту та інформаційної безпеки
Кафедра менеджменту та безпеки інформаційних систем

Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 125 – Кібербезпека
Освітньо-професійна програма - Управління інформаційною безпекою

ЗАТВЕРДЖУЮ
Голова секції УБ, кафедра МБІС

_____д.т.н., проф. Яремчук Ю.Є.
“ 24 ” вересня 2021 р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу студенту
Пархоменко Руслану Михайловичу
(прізвище, ім`я, по-батькові)

1. Тема роботи Підвищення захищеності веб-ресурсів стійкими криптографічними алгоритмами на основі генераторів випадкових чисел, що враховують ентропію поведінки користувача у багатокористувацькому середовищі

Керівник роботи к.т.н., доцент, завідувач кафедри МБІС Карпінець В.В.
(прізвище, ім`я, по-батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 24.09.2021 р. №277.

2. Строк подання студентом роботи 14.12.2021 р.

3. Вихідні дані до роботи: монографії та наукові статті по темі, існуюче програмне забезпечення по темі.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити). Для досягнення мети необхідно: В першому розділі дослідити та проаналізувати залежність стійкості криптографічних алгоритмів від якості генераторів випадкових послідовностей які в них використовуються. В другому розділі розробити табличний генератора випадкових чисел на основі ентропії поведінки користувача в багатокористувацькому веб-ресурсі, сформувати

алгоритм роботи програмного продукту. В третьому розділі розробити програмний засіб для реалізації розробленого генератора випадкових чисел.

5. Перелік графічного матеріалу (з точним зазначенням обов`язкових креслень): у першому розділі наведено 4 рис.; у другому розділі наведено 9 рис. та 1 табл.; у третьому розділі наведено 10 рис. та 1 табл.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	к.т.н., доцент, завідувач кафедри МБІС Карпінець В.В.		
Економічна частина	к.т.н., доц. Адлер Оксана Олександрівна		

7. Дата видачі завдання 24 вересня 2021 р.

1 КАЛЕНДАРНИЙ ПЛАН

№	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області обраної теми	27.09.2021	10.10.2021	
2	Розробка алгоритму роботи	11.10.2021	31.10.2021	
3	Написання магістерської кваліфікаційної роботи на основі розробленої теми	01.11.2021	24.11.2021	
4	Передзахист магістерської кваліфікаційної роботи	25.11.2021	26.11.2021	
5	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2021	16.12.2021	
6	Захист магістерської кваліфікаційної роботи	20.12.2021	20.12.2021	

Студент _____ Пархоменко Р.М.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Карпінець В.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У роботі детально розглянуто та описано необхідність використання криптографічно якісних генераторів випадкових послідовностей для підвищення криптографічної стійкості веб-ресурсів .

Другий розділ присвячений пошуку можливості використання табличних генераторів істино випадкових послідовностей, а саме розроблено алгоритм отримання ентропії поведінки користувача для використання її в якості Seed данних для табличного ГВЧ.

Третій розділ стосується програмної реалізації підсистеми генерації випадкових чисел.

Результатом роботи є підвищення стійкості криптографічних алгоритмів, за рахунок використання в них розробленого генератора випадкових чисел на основі ентропії поведінки користувача в багатокористувацькому веб ресурсі.

Ключеві слова: веб-ресурс, генератора випадкових чисел, табличний генератор випадкових чисел, ентропія, випадкова послідовність чисел, NIST.

ABSTRACT

The paper considers in detail and describes the need to use cryptographically high-quality random sequence generators to increase the cryptographic stability of web resources.

The second section is devoted to finding the possibility of using tabular generators of truly random sequences, namely, developed an algorithm for obtaining the entropy of user behavior to use it as Seed data for tabular HF.

The third section deals with the software implementation of the random number generation subsystem.

The result is to increase the stability of cryptographic algorithms, through the use of a random number generator based on the entropy of user behavior in a multi-user web resource.

Keywords: web resource, random number generator, tabular random number generator, entropy, random sequence of numbers, NIST.

ЗМІСТ

ВСТУП 7
1 ГЕНЕРАТОРИ ВИПАДКОВИХ ЧИСЕЛ ЯК СКЛАДОВА СТІЙКОСТІ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ 9
1.1 Аналіз залежності стійкості криптографічних алгоритмів багатокористувацького веб-ресурсу від якості використовуваних в генераторів випадкових чисел 9
1.2 Випадкові числа та способи генерації їх послідовностей 13
1.3 Аналіз існуючих програмних генератора випадкових чисел 19
1.4 Висновки та постановка задачі 25
2 РОЗРОБКА АЛГОРИТМУ РОБОТИ ТАБЛИЧНОГО ГЕНЕРАТОРА ВИПАДКОВИХ чисел НА ОСНОВІ ЕНТРОПІЇ ПОВЕДІНКИ КОРИСТУВАЧА 26
2.1 Аналіз можливості підвищення стійкості криптографічних алгоритмів за рахунок використання табличних генераторів випадкових чисел 26
2.2 Розробка алгоритму надійного джерела ентропії на основі поведінки користувача веб-ресурсу 29
2.3 Розробка алгоритму роботи табличного генератора випадкових чисел ... 35
3 РОЗРОБКА РОЗРОБЛЕНОГО ТАБЛИЧНОГО ГЕНЕРАТОРА ВИПАДКОВИХ ЧИСЕЛ 38
3.1 Обґрунтування вибору мови програмування 38
3.2 Програмна реалізація розробленого алгоритму генерації випадкових чисел 41
3.3 Аналіз підвищення стійкості криптографічних алгоритмів за рахунок використання в них розробленого ГВЧ 46
3.4 Висновок 50
4 ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДВИЩЕННЯ СТІЙКОСТІ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ 51
4.1 Проведення наукового аудиту науково-дослідної роботи 51
4.2 Проведення комерційного та технологічного аудиту науково-технічної розробки 54

4.3 Розрахунок витрат на здійснення науково-дослідної роботи 57
4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності ... 64
4.5 Висновки до розділу 68
ВИСНОВОК 69
ПЕРЕЛІК ВИКОРИСТАНИХ ПОСИЛАНЬ 70
ДОДАТКИ 75
Додаток А. Технічне завдання 76
Додаток Б. Лістинг програми 80
Додаток В. Ілюстративний матеріал 84
Додаток Г. ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ 92

ВСТУП

Актуальність.

Оскільки зловмисні криптоаналітики постій шукають вразливості систем захисту, а також враховуючи той факт, що комп'ютеризація сучасного суспільства досягла рівня коли майже вся інформація обробляється, зберігається та передається за допомогою комп'ютерних технологій, удосконалення алгоритмів захисту є вкрай важливою метою.

Якщо розглядати криптографічну стійкість алгоритм як стійкість всіх модулів які в них використовуються то вході їх удосконалення необхідно звертати увагу на всі, навіть найменш значущі на перший погляд, складові цих алгоритмів.

Такою складовою можуть виступати генератори випадкових послідовностей.

Не завжди є можливість використовувати апаратні генератори які можуть задовольняти потреби під час їх використання в криптографії, оскільки вони потребують розроби на впровадження апаратних модулів у систему, що не завжди є фінансово та практично доцільним.

Цей факт змушує нас шукати нові алгоритми для генерації випадкових чисел для захисту та підвищувати стійкість криптографічних алгоритмів.

Мета і задачі дослідження. Метою роботи підвищення стійкості криптографічних алгоритмів в багатокористувацьких веб-ресурсах.

Для досягнення мети роботи поставлено такі задачі:

1) Дослідити залежність стійкості криптографічних алгоритмів від стійкості використовуваних в них генераторів випадкових чисел.

2) Дослідити існуючі алгоритми реалізації генераторів випадкових чисел;

3) Розробити алгоритм роботи табличного генератора випадкових чисел на основі ентропії поведінки користувача;

4) Розробити програмну реалізацію;

5) Провести статистичні тести розробленого генератора.

Об'єкт дослідження – підвищення криптографічної стійкості криптографічних алгоритмів

Предмет дослідження – залежність стійкості криптографічних алгоритмів від криптографічної якості генераторів випадкових чисел які в них використовуються.

Новизна. Отримані результати дають підґрунтя для використання ентропії поведінки користувача в якості Seed даних для табличного генератора випадкових чисел.

Практичне значення одержаних результатів. Розроблено пакет програм криптографічної підсистеми захисту інформації, зокрема програмна реалізація серверної частини алгоритму табличного генератора випадкових чисел для платформи ASP.NET Core, та клієнтської для Angular2.

1 ГЕНЕРАТОРИ ВИПАДКОВИХ ЧИСЕЛ ЯК СКЛАДОВА СТІЙКОСТІ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ

Роберта Р. Кавью «Генерація випадкових чисел занадто важлива, щоб залишати її на волю випадку».

1.1 Аналіз залежності стійкості криптографічних алгоритмів багатокористувацького веб-ресурсу від якості використовуваних в генераторів випадкових чисел

Терміном багатокористувацький веб-ресурс можна описати будь-який веб-додаток в якому n-на кількість користувачів можуть взаємодіяти один з одним або з самою системою для виконання будь-яких бізнес завдань або для розваг чи спілкування.

В якості прикладу можуть виступати:

1) CRM системи - системи управління взаємовідносинами з клієнтами та керування життєвим циклом бізнесу;

2) соціальні мережі;

3) поштові клієнти;

4) хмарні середовища збереження даних;

5) та інші.

На даному етапі розвитку інформаційних технологій веб-ресурси є дуже популярним різновидом побудови додатків який швидко розвивається збільшуючи кількість користувачів та інформації яку потрібно захищати.

Основною перевагою такого підходу є клієнт-серверна архітектура. Вона дозволяє працювати з системою використовуючи дешеві та малопотужні комп’ютерні станції.

Яскравим прикладом є використання великою компанією CRM. До появи можливості використання клієнт-серверної архітектури веб-ресурсів на кожній комп’ютерній станції було необхідно встановлювати повнозначну версію ПЗ, що вимагало використовувати потужні та дорогі ПК. Якщо таж CRM буде реалізована у вигляді веб-ресурсу в якості робочої комп’ютерної станції для

користувача можна буде використовувати дешеві та малопотужні ПК, а основну роботу буде виконувати сервер розміщений в локальній або віддаленій мережі, що зменшить витрати та спростить обслуговування системи.

В якості користувача в даній роботі буде розглядатися ідентифікована та авторизована персона яка взаємодіє через користувацький інтерфейс з серверною частиною веб-ресурсу.

Оскільки користувач постійно взаємодіє з серверною частиною веб-додатку, під час такої взаємодії на сервер може передаватись конфіденційна інформація, цю взаємодію необхідно захищати.

Для забезпечення безпечного обміну даними використовується протокол HTTPS – криптографічно стійкий протокол передачі даних, що використовується в комп'ютерних мережах.

Даний протокол в представляє собою розширену версію протоколу HTTP за рахунок використання криптографічний протокол SSL або TLS, що забезпечує безпечне спілкування користувача та сервера в небезпечній мережі.

В межах теми даної роботи протоколи цікаві тим, що в своїй роботі використовують ГВЧ.

Найбільш яскравим та широко використовуваним прикладом використання ГВЧ є протокол Діффі-Геллмана.

Алгоритм Діффі-Хеллмана (DH) — це протокол обміну ключами, який дає змогу двом сторонам, що спілкуються через публічний канал, встановити взаємну таємницю без її передачі через Інтернет. Діффі-Хеллман дозволяє двом користувачам використовувати відкритий ключ для шифрування та дешифрування їх розмови або даних за допомогою симетричної криптографії.

Якщо Аліса і Боб бажають спілкуватися один з одним, вони спочатку узгоджують між собою велике просте число p і генератор (або основу) g (де $0 < g < p$). Аліса вибирає секретне ціле число a (її закритий ключ), а потім обчислює $g^a \bmod p$ (яке є її відкритим ключем). Боб вибирає свій закритий ключ b і таким же чином обчислює свій відкритий ключ.

Боб знає b і g^a, тому він може обчислити $(g^a)^b \bmod p = g^{ab} \bmod p$. Тому і Аліса, і Боб знають загальний секрет $g^{ab} \bmod p$. Підслуховувач Єва, яка

прослуховувала повідомлення, знає p, g, відкритий ключ Аліси і відкритий ключ Боба. Вона не може обчислити загальний секрет із цих значень.

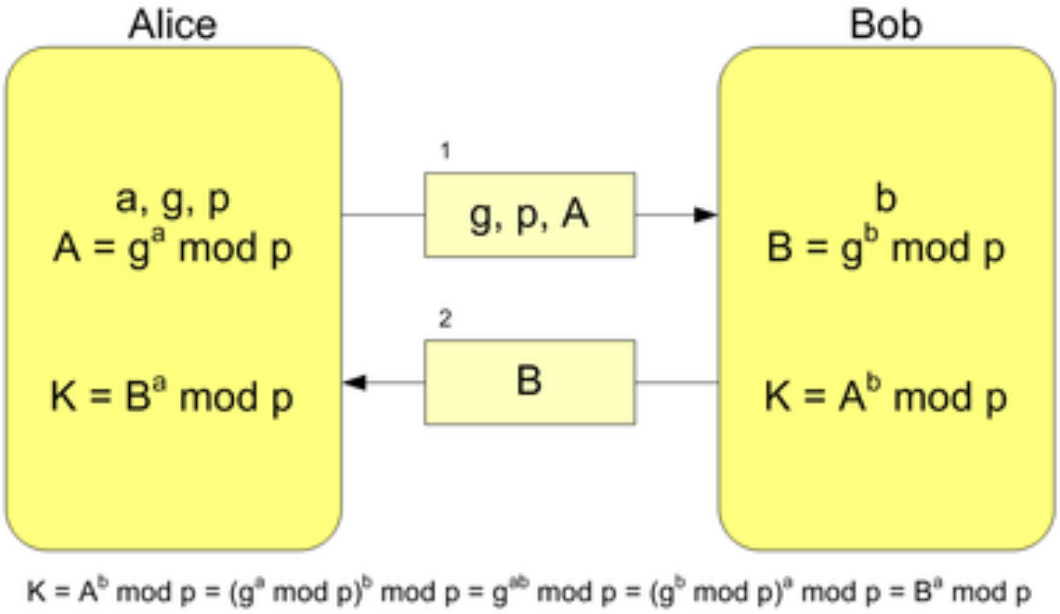


Рисунок 1.1 – алгоритм Діффі-Хеллмана

При роботі алгоритму кожна сторона:

1) генерує випадкове натуральне число a - закритий ключ

2) спільно з віддаленою стороною встановлює відкриті параметри p і g (зазвичай значення p і g генеруються на одній стороні та передаються іншій), де

p є випадковим простим числом

g є первісним коренем по модулю p

3) обчислює відкритий ключ A, використовуючи перетворення над закритим ключем

4) обмінюється відкритими ключами із віддаленою стороною

5) обчислює загальний секретний ключ K, використовуючи відкритий ключ віддаленої сторони B і закритий ключ a

У практичних реалізаціях, для a і b застосовуються числа порядку 10100 і p порядку 10300. Число g має бути великим і зазвичай має значення не більше першого десятка.

Оскільки кожна сторона генерує випадкове натуральне число a, яке є закритим ключем, необхідність якісного ГВЧ на даному етапі є дуже важливою.

Якщо зловмисник який хоче отримати доступ до інформації якою обмінюються користувачі зможе використати якийсь з недоліків генераторів псевдовипадкових чисел то він зможе згенерувати для себе ключ який буде ідентичний секретному ключу користувача.

Розглянемо цю можливість більш детально.

Наприклад зловмисник знаючи який конкретний математичний апарат для генерації псевдовипадких послідовностей використовується в досліджуваній ним криптосистемі та початкові її значення зможе дублювати випадкову послідовність.

Лінійний конгруентний метод є однією з найпростіших і найбільш уживаних в даний час процедур, що імітують випадкові числа.

Для початкових коефіцієнтів цього методу зазвичай вибирають істино випадкові значення які отримують з системи але ці значення не є секретними і маючи доступ до системи зловмисник може запустити аналогічний ГПВЧ з аналогічними початковими коефіцієнтами.

Альтернативою може бути використання фізичних генераторів які видаються істино випадкові послідовності чисел. Вони є більш безпечною альтернативою генераторам псевдовипадкових чисел (ГПВЧ). ГПВЧ використовують детермінований алгоритм для створення числових послідовностей. Хоча ці псевдовипадкові послідовності проходять тести статистичних шаблонів на випадковість, знаючи алгоритм і умови, які використовуються для його ініціалізації, так зване «початкове число», вихідні дані можна передбачити. Оскільки послідовність чисел, вироблена ГПВЧ, в принципі передбачувана, дані, зашифровані за допомогою псевдовипадкових чисел, потенційно вразливі для криптоаналізу.

Апаратні генератори випадкових чисел створюють послідовності чисел, які, як передбачається, не можна піддати криптоаналізу, і тому забезпечують максимальну безпеку при використанні для шифрування даних.

Але використання фізичних генераторів не завжди є можливим оскільки вони мають багато недоліків. Їх використання потребує додаткових фінансових витрат, також необхідність фізичної наявності апаратного модуля не завжди є

можливе з врахуванням розмірів та апаратного забезпечення системи в якій буде працювати ГВЧ. Також даний вид ГВЧ є повільним в порівняні з програмними реалізаціями, а ті розробки фізичних ГВЧ які могли б задовільнити всі потреби, засекречені і використовуються у військових розробках, що унеможливлює їх використання для широкого колу користувачів.

Отже, проаналізувавши тенденції розвитку веб-ресурсів та необхідності захищати інформацію яка передається з клієнтської сторони на серверну можна зробити висновок про необхідність розвитку та підвищення криптографічної стійкості таких протоколів як SSL та TLS.

Якщо розглядати стійкість системи до криптоаналізу як стійкість найменш захищеного її модуля то розробка та вдосконалення такого модуля як генератор випадкових чисел є важливою частиною комплексної системи захисту веб ресурсу. Також потрібно взяти до уваги, що використання ГВЧ можна знайти не тільки в протоколі обміну ключами, навіть генератор тимчасових паролів для доступу користувачів, будучи скомпрометованим може дати зловмисник отримати доступ до інформації яка зберігається на веб-ресурсі.

Враховуючи вище вказане, доцільним буде розглянути та дослідити природу випадковості чисел, а також розглянути способи її генерації.

1.2 Випадкові числа та способи генерації їх послідовностей

Випадкові числа — це числа, послідовність яких рівномірно розподілена, і неможливо передбачити подальші майбутні значення на основі поточної послідовності. Ці числа генеруються за допомогою певних математичних алгоритмів, які рівномірно розподіляють усі попередньо встановлені числа в послідовності [1].

Випадкові числа мають багато застосувань, але найважливіше застосування випадкових чисел — у криптографії, де вони є основною частиною ключів шифрування. Якість випадкових чисел, що використовуються в криптографії, визначає надійність системи безпеки. Якщо якість генератора випадкових чисел хороша, то зловмисникам буде дуже важко проникнути в

систему. Випадковість і непередбачуваність є двома основними вимогами до послідовності випадкових чисел [2].

Сучасна інформатика широко використовує псевдовипадкові числа в самих різних додатках. Перерахуємо деякі області їх застосування:

1) Соціологічні та наукові дослідження. Підготовка випадкових вибірок при зборі даних, опитуванні думок або в дослідженні фізичних явищ з випадковим вибором результатів експериментів.

2) Моделювання. У комп'ютерному моделюванні фізичних явищ. Крім того, математичне моделювання використовує випадкові числа як один з інструментів чисельного аналізу.

3) Криптографія та інформаційна безпека. Випадкові числа можуть використовуватися в тестуванні коректності або ефективності алгоритмів і програм. Багато алгоритми використовують генерацію псевдовипадкових чисел для вирішення прикладних завдань (наприклад, криптографічні алгоритми шифрування, генерація унікальних ідентифікаторів і ін.).

4) Прийняття рішень в автоматизованих експертних системах. Використання випадкових чисел є частиною стратегій прийняття рішень. Наприклад, для неупередженості вибору екзаменаційного білета студентом на іспиті. Випадковість також використовується в теорії матричних ігор.

5) Оптимізація функціональних залежностей. Деякі математичні методи оптимізації використовують стохастичні методи для пошуку екстремумів функцій.

6) Розваги та ігри. Випадковість в іграх має значну роль. У комп'ютерних або настільних іграх випадковість допомагає урізноманітнити ігровий процес.

При більш глибокому знайомстві з природою випадковості і випадкових чисел виникають питання про те, що таке «істинно» випадкове число, якою має бути послідовність випадкових чисел, яким може бути розподіл випадкових чисел на деякому інтервалі (наприклад, тимчасовому).

З точки зору криптографії важливо гарантувати, що секретні ключі є випадковими та абсолютно непередбачуваними, вони мають відповідати правилам випадковості [3].

В даний час існує велика кількість способів генерації послідовностей, що володіють тим чи іншим ступенем випадковості [1-4, 7-9, 14-15]. Однак на практиці більшість таких генераторів виробляють послідовності, властивості яких не відповідають вимогам випадковості. Один із найпоширеніших прикладів цього – генератори псевдовипадкових чисел, вбудовані в стандартні бібліотеки багатьох мов програмування, наприклад функція стандартної бібліотеки мови C rand().

Часто в числах, що згенеровані за допомогою подібних функцій, простежуються очевидні закономірності. Наприклад, отримані числа в одному та тому ж сеансі з часом монотонно зростають, що прямо суперечить вимогам до властивостей випадкових (і псевдовипадкових) послідовностей.

Для широко відомих та поширених лінійних конгруентних генераторів по чотирьох відомих згенерованих числах також можна передбачити подальші значення.

Більшість криптографічних додатків використовують генератори випадкових чисел для створення ключів, за допомогою яких шифрується та розшифровується необхідна інформація. Однак часто саме застосовувані в них генератори є найслабшим місцем у системах шифрування. Справа в тому, що програмні генератори повністю детерміновані. Зазвичай вони використовують різні складні функції для обчислення псевдовипадкових чисел. Відповідно, послідовності, отримані внаслідок роботи таких генераторів, є в тій чи іншій мірі передбачуваними та відтворюваними і не підходять, наприклад, для використання в криптографічних додатків.

Слід зазначити, що у деяких випадках можливість відтворити випадкову послідовність є корисною наприклад при тестуванні алгоритмів розробником.

Проте, послідовність не повинна володіти властивостями, які б дозволили зловмисному криптоаналітику відновити її у процесі аналізу роботи захищеної програми або протоколу.

Існує табличний спосіб генерації випадкових послідовностей. Він полягає в тому, що випадкові числа оформлені у вигляді таблиці, паперової або електронної, яка зберігається в оперативній пам'яті або на зовнішньому носії.

Перевага цього способу полягає в тому, що з його допомогою можна відтворювати неодноразово одну і ту ж послідовність псевдовипадкових чисел. Однак серйозним недоліком, фактично не допускає застосування таких генераторів при вирішенні Практичним завданням є те, що запас доступних чисел обмежений. Також при такому підході можливе неефективне використання обчислювальних ресурсів комп'ютера (наприклад, через необхідність зберігати таблицю або її частини в оперативній пам'яті або звертатися до зовнішньої пам'яті). В даний час такий спосіб генерації використовується досить рідко.

Серед генераторів псевдовипадкових послідовностей (ГПВЧ), отримавших широке поширення і застосовних при вирішенні задач з серйозними вимогами до якості згенерованої послідовності, розрізняють апаратні, програмні та програмно-апаратні (змішані).

Апаратний генератор випадкових чисел – це пристрій, який генерує послідовність випадкових чисел на основі вимірюваних, хаотично змінюваних параметрів фізичного процесу. При апаратному способі генерації випадкові числа є прямим або побічним продуктом вимірів деякої фізичної величини, що служить надійним джерелом ентропії. Зазвичай це процеси, які протікаю у неживій природі. Теоретично такі процеси абсолютно непередбачувані, проте на практиці отримані таким чином випадкові числа доводиться піддавати перевірці з допомогою спеціальних статистичних тестів. Незважаючи на найкращі статистичні властивості і, відповідно, більш високий ступінь випадковості, апаратним генераторам притаманні такі недоліки:

- потенційно високі часові та матеріальні витрати на конструювання, встановлення та налаштування в порівнянні з програмними ГПВЧ;
- нижча швидкість генерації випадкових чисел, ніж за програмної реалізації ГПВЧ [14, 15];
- неможливість відтворення раніше згенерованої послідовності чисел (що в деяких випадках є небажаним).

Програмні (алгоритмічні) генератори засновані на детермінованих алгоритмах. У отриманих таким чином послідовностей завжди існує період

нехай іноді і дуже великий, а також спостерігаються й інші відхилення від випадковості. Будь-який ГПВЧ з обмеженими ресурсами рано чи пізно зациклюється – починає повторювати ту саму послідовність чисел.

Період ГПВЧ залежить від типу генератора та його параметрів [1, 4, 7, 9]. Якщо Породжувана послідовність ГПВЧ має дуже короткий період, то такий ГПВЧ стає непридатним для багатьох практичних програм. Більшість простих арифметичних генераторів хоч і мають великою швидкістю, але страждають від багатьох серйозних недоліків:

- надто короткий період;
- послідовні значення не є незалежними;
- деякі біти «менш випадкові», ніж інші;
- нерівномірний розподіл;
- оборотність.

Фактично результат роботи таких генераторів не є випадковою послідовністю. Тим не менш, до послідовностей, вироблених програмними генераторами, пред'являються певні вимоги, оскільки вони мають якоюсь мірою імітувати випадкові послідовність. Зокрема, період таких послідовностей має бути достатньо великим, щоб при генерації необхідної послідовності довжини не виникало повторень. На відміну від апаратних генераторів, програмні генератори здатні відтворити раніше згенеровану послідовність, що в деяких випадках є безперечною перевагою.

Програмно-апаратні генератори- такий генератор може формувати потік випадкових шумів, які потім перетворюються на числа. Також можливий варіант, коли «зерно», тобто деякі вхідні дані алгоритму шифрування, генерується за допомогою апаратного генератора оскільки її розмір досить невеликий, і відповідно її отримання не вимагає великих витрат часу та ресурсів, а підсумкова послідовність – за допомогою програмного.

До програмно-апаратних генераторів випадкових чисел можуть належати, наприклад, пристрої комп'ютера. Зокрема, джерелом випадковим послідовності можуть бути шуми пристроїв комп'ютера (наприклад, процесора), системний час, часові інтервали між натисканнями клавіш, руху миші і таке інше. Як

правило, послідовності, що вийшли в внаслідок таких процесів, потребують постобробки. До того ж їх швидкість отримання є досить низькою, особливо при генерації послідовностей досить великих обсягів.

Послідовність називається істинно випадковою (ІВП), якщо її не можна відтворити. Це означає, що якщо запустити генератор дійсно випадкових послідовностей двічі при тому самому вході, то на його виході вийдуть різні випадкові послідовності. Основна труднощі полягає в тому, щоб відрізнити випадкову послідовність від невипадкової. Однак на практиці далеко не завжди можна безпосередньо використовувати вихідні дані джерел істинно випадкових чисел. Тому зазвичай доводиться використовувати звані псевдовипадкові послідовності. Псевдовипадкова послідовність (ПВП) – це послідовність, що складається із псевдовипадкових двійкових чисел, що отримуються за допомогою заданого детермінованого алгоритму, але застосовуваних як випадкові. При цьому зазвичай алгоритми отримання ПВП використовують спеціальне випадкове початкове значення, або "зерно" (seed).

Для того, щоб ПВП могли використовуватися в якості випадкових послідовностей, вони повинні відповідати статистичними властивостям бути близькими до ІВП.

Вимоги до якісного генератора випадкових чисел [3]:

1) Непередбачуваність результатів роботи: при невідомому ключі/початковому стан генератора на основі відомої кінцевої частини ПВП неможливо визначити як її наступний елемент (пряма непередбачуваність, або непередбачуваність вправо), так і попередній (зворотній непередбачуваність, непередбачуваність вліво);

2) Невідмінність статистичних властивостей ПВП, що генеруються, від аналогічних властивостей істинно випадкової послідовності;

3) Великий період послідовності;

4) Можливість ефективної апаратної та програмної реалізації.

На практиці домогтися виконання всіх цих умов, як правило, не є можливим. Більше того, часто ці умови є взаємовиключними. Тому доводиться

шукати баланс між ними та насамперед прагнути до виконання того, що є найважливішим у контексті розв'язуваного завдання.

Оскільки метою даної роботи є підвищення стійкості криптографічних алгоритмів доцільним буде розробити генератор випадкових чисел результатом роботи якого буде істино випадкова послідовність. Використання апаратних генераторів не завжди є доцільним та можливим, тому найкращим варіантом буде дослідження можливості програмної реалізації такого генератора.

1.3 Аналіз існуючих програмних генератора випадкових чисел

Оскільки поставлена раніше задача для досягнення мети кваліфікаційної робота передбачує розробку програмного генератора випадкових чисел, доцільним буде розглянути існуючі ГПВЧ, щоб під час розробки уникнути недоліків вже реалізованих генераторів, та вибрати напрямок розробки.

За еталон генератора випадкових чисел (ГВЧ) прийнятий такий генератор, який породжує послідовність випадкових чисел з рівномірним законом розподілу в інтервалі (0; 1). За одне звернення даний генератор повертає одне випадкове число. Якщо спостерігати такий ГСЧ досить тривалий час, то виявиться, що, наприклад, в кожен з десяти інтервалів (0; 0.1), (0.1; 0.2), (0.2; 0.3), ..., (0.9; 1) потрапить практично однакову кількість випадкових чисел - тобто вони будуть розподілені рівномірно по всьому інтервалу (0; 1). Якщо зобразити на графіку k = 10 інтервалів і частоти Ni влучень в них, то вийде експериментальна крива щільності розподілу випадкових чисел (рис. 1.1).

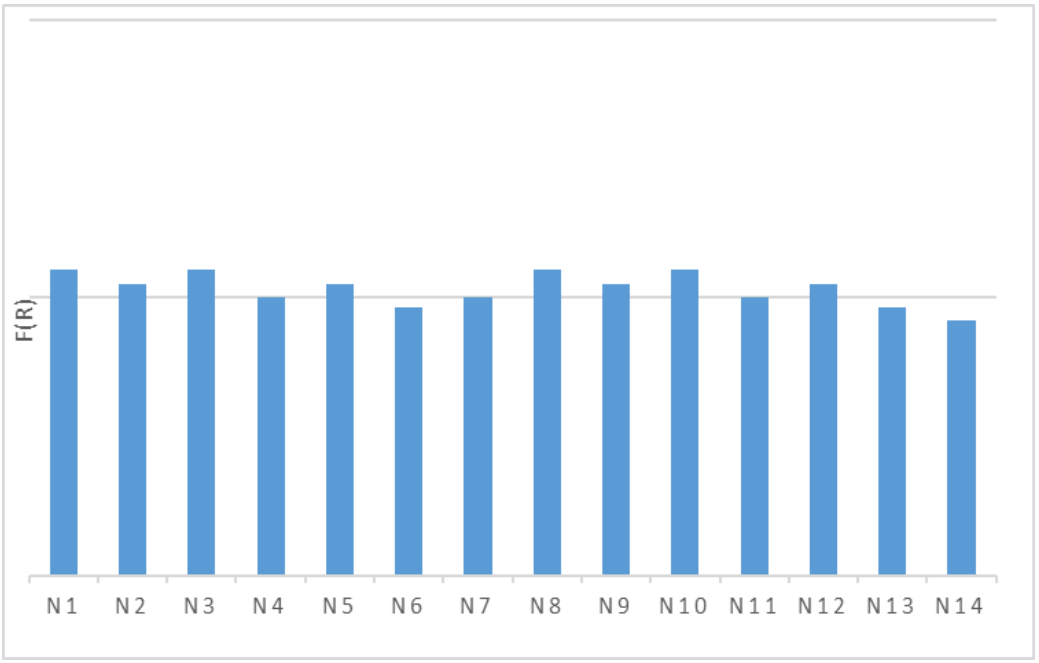


Рисунок 1.2 – Частотна діаграма випадання випадкових чисел, породжуваних реальним генератором

Зауважимо, що в ідеалі крива щільності розподілу випадкових чисел виглядала б так, як показано на рис. 1.2. Тобто в ідеальному випадку в кожен інтервал потрапляє однакове число точок: Ni = N / k, де N - загальне число точок, k - кількість інтервалів, i = 1, ..., k.

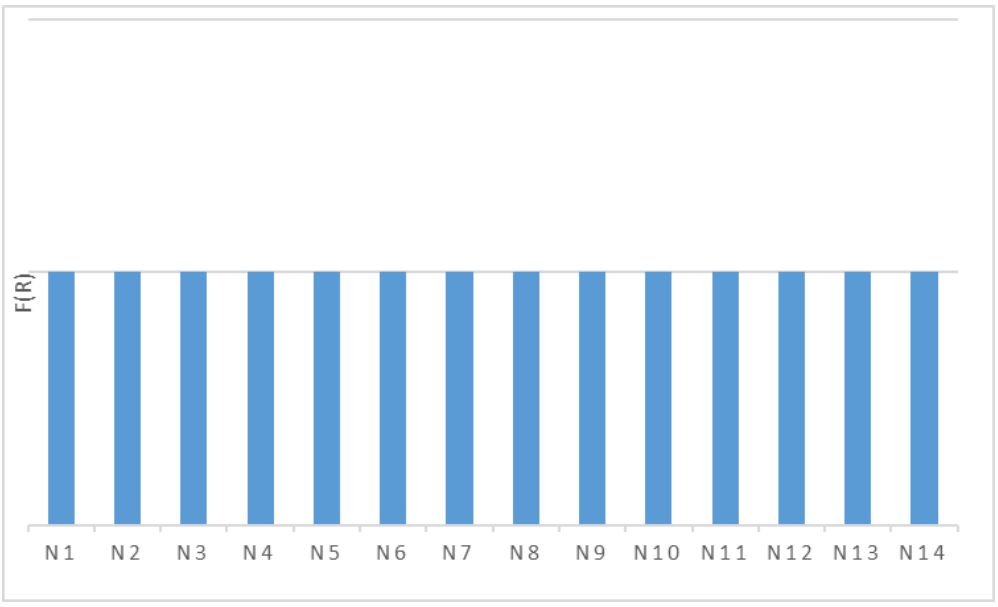


Рисунок 1.3 – Частотна діаграма випадання випадкових чисел, породжуваних ідеальним генератором

Слід пам'ятати, що генерація довільного випадкового числа складається з двох етапів:

1) генерація нормалізованого випадкового числа (тобто рівномірно розподіленого від 0 до 1);

2) перетворення нормалізованих випадкових чисел ri в випадкові числа xi, які розподілені по необхідному користувачеві (безпідставного) закону розподілу або в необхідному інтервалі.

На сьогоднішній день існує кілька тисяч різних генераторів псевдовипадкових чисел.

Розглянемо основні методи генерації псевдовипадкових послідовностей, які найбільше підходять для комп'ютерної криптографії:

1) Використання стандартних функцій мов програмування.

Функція Rand створює псевдовипадкове число у вказаному діапазоні (метод вказівки діапазону відрізняється різними мовами програмування).

Початкове ініціювання генератора випадкових чисел відбувається при системному виклику спеціальної функції. Длят C це функція srand, в C# початкове значення встановлюється через конструктор класу. У реальних криптосистемах ця функція не використовується, оскільки вона має низьку криптостійкість - через її відкритий код. [2][3].

2) Лінійний конгруентний генератор псевдовипадкових чисел

Лінійний конгруентний генератор (ЛКГ) є послідовністю чисел від 0 до m-1, яка задовольняє рекурентному виразу:

$$X_{k+1} = X_K a + b \; mod \; m \qquad (1.1)$$

де X_0 - початкове значення, a - множник, b - збільшення, m – модуль.

Такий генератор має період менший за m. Якщо a, b і m обрані правильно, генератор буде з максимальною довжиною і з періодом m (наприклад, gcd (m, b) = 1, де gcd – найбільший спільний дільник). [4]

Якщо b = 0, то ЛКГ має вигляд:

$$X_{k+1} = X_K a \; mod \; m \qquad (1.2)$$

В цьому випадку отримуємо найпростішу послідовність, яка може бути запропонована для генератора з рівномірним розподілом. При правильному виборі констант a може набувати значень 75 або 16807, а m - 231-1=2147483647.

Такий генератор матиме максимальний період повторення. Ці константи були представлені Парком та Міллером, тому ЛКГ виду:

$$X_{k+1} = 75X_k \, mod(2^{31}-1) \quad (1.3)$$

Носить назву генератора Парка-Миллера.

Основною перевагою ЛКГ є їхня швидкість через невелику кількість операцій на кожен байт і простоту реалізації. Однак такі генератори рідко використовуються в криптографії, тому що вони передбачувані. Вперше ЛКГ зламано Д.Бояр [2].

3) Нелінійні конгруентні генератори

У деяких випадках використовуються квадратичні та кубічні конгруентні генератори (КГ), які мають більш високу стійкість до злому. [5]

Квадратичний конгруентний генератор:

$$X_{k+1} = \left(aX_k^2 + bX_k \ + c\right) mod \ m \quad (1.4)$$

Кубічний конгруентний генератор:

$$X_{k+1} = (\ aX_k^3 + bX_k^2 + cX_k + d) mod \ m \quad (1.5)$$

Щоб збільшити період КГ, часто використовується їх об'єднання (суперпозиція) з допомогою нелінійного перетворення. У той же час їхня безпека не змінюється, але такі генератори мають найкращі властивості в деяких статистичних тестах [2].

4) Лінійні регістри зі зворотним зв'язком

Лінійний регістр зі зворотним зв'язком (ЛРЗЗ/LFSR) складається з двох частин: регістру зсуву та послідовності відгалуження (tap sequence).

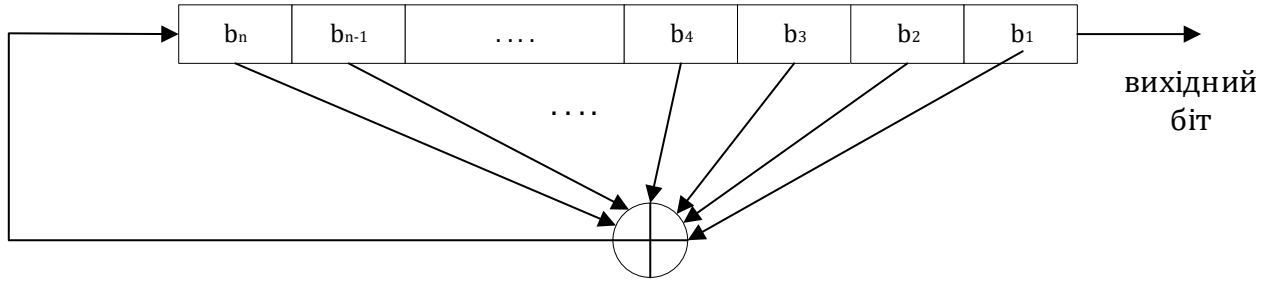


Рисунок 1.4 – Лінійний регістр зі зворотним зв'язком (ЛРЗЗ)

У цій схемі ЛРЗЗ є послідовністю біт. Як тільки знадобиться наступний біт, всі біти регістру зсуваються праворуч, а ЛРЗЗ видає найбільш значний біт. У

цьому випадку найменший значний біт визначається за допомогою обчислень XOR з інших бітів регістру зсуву відповідно до послідовності відгалуження.

Теоретично, n-бітний лінійний регістр зі зворотним зв'язком може генерувати псевдовипадкову послідовність довжиною 2n -1 біт перед зациклюванням. Для цього регістр зсуву повинен набути всіх 2n −1 станах. [5]

Тільки деякі послідовності, що відгалуження, проходять через всі 2n −1 внутрішні стани, довжина ЛРЗЗ з такими відгалуженнями буде максимальною.

Примітивні рівняння з трьома невідомими особливо зручні, тому що тільки 2 біти регістру повинні бути перетворені за допомогою операцій XOR, але вони більш вразливі. ЛРЗЗ – хороші генератори випадкових чисел, але вони мають малоприємні властивості. Бітові послідовності є лінійними, що унеможливлює їх використання для шифрування. Для регістрів зі зворотним зв'язком довжини n внутрішній стан може бути розпізнано вихідними n бітами генератора. Якщо схема зворотний зв'язок не відома, її визначення досить 2n вихідних біт. Випадкові числа, що генеруються з послідовних біт ЛРЗЗ, сильно кореловані, а іноді навіть не випадкові. Однак лінійні регістри часто використовуються як базовий алгоритм шифрування. [2]

5) Генератор Геффа

Цей генератор використовує три ЛРЗЗ, які об'єднані нелінійно. Два ЛРЗЗ є входами мультиплексора, а третій управляє виходом. Якщо a1, a2 та a3 є виходами трьох ЛРЗЗ, вихід генератора може бути описаний як:

$$b = (a_1 \wedge a_2) \oplus ((\neg a_1) \wedge a_3) \quad (1.6)$$

Якщо довжини ЛРЗЗ дорівнюють n_1,n_2 і n_3, то лінійна складність генератора дорівнює:

$$(n_1 + 1)n_2 + n_1 n_3 \quad (1.7)$$

Період такого генератора дорівнюватиме НСД (найменшому спільному дільнику) періодів трьох ЛРОС. Якщо ступеня трьох багаточленів зворотного зв'язку взаємно прості, то період дорівнюватиме добутку трьох періодів.

Цей генератор криптографічно слабкий і не може протистояти кореляційному розтині. У 75% часу вихід генератора дорівнює виходу ЛРЗЗ -2. Тому, якщо послідовність зворотного зв'язку відома, досить легко вгадати

початкове значення ЛРЗЗ -2 та сформувати вихідну послідовність цього регістру. Тоді можна розрахувати, скільки разів вихід ЛРЗЗ збігається з виходом генератора. Якщо вихідне значення неправильне, дві послідовності будуть узгоджені у 50% випадків, і якщо значення визначено правильно, то 75%. За таких кореляцій генератор може бути легко зламаний. Наприклад, якщо примітивні поліноми складалися лише з трьох членів, і довжина найбільшого ЛРЗЗ дорівнює n, то для того, щоб відновити ефективні стани всіх трьох ЛРЗЗ, необхідна частина вихідної послідовності довжиною 37n біт [2].

6) Адитивні генератори

Цей вид генераторів псевдовипадкових чисел (ще їх називають відкладеними генераторами Фібоначчі) є дуже ефективним, тому що на виході будуть випадкові слова, а не биті. Адитивні генератори не є безпечними, але вони використовуються як складові блоки.

Вихідним станом генератора є масив n-бітних слів: 8, 16, 32 і т. д.: X1 X2, X3, ..., Xm. Цей початковий стан є ключем, i-е слово:

$$X_i = (X_{i-a} + X_{i-b} + X_{i-c} + + X_{i-m}) \bmod 2^n \qquad (1.8)$$

При правильних коефіцієнтах a, b, c, . . . m період буде не менше 2n-1. Молодший біт формує максимальну довжину ЛРЗЗ.

Якщо (55,24,0) є примітивним багаточленом mod 2. Отже, довжина наступного генератора максимальна:

$$X_i = (X_{i-55} + X_{i-24}) \bmod 2^n \qquad (1.8)$$

Це працює, оскільки примітивний багаточлен має три коефіцієнти [2].

7) Fish

Даний генератор адитивний, що використовує принципи генератора, що проріджується. Він виводить потік 32-розрядних слів, які можуть використовуватися (з використанням операції XOR) з потоком відкритого тексту для отримання зашифрованого тексту або потоком зашифрованого тексту для отримання відкритого тексту. Назва «fish» - скорочення від проріджуваного генератора Фіббоначі.

Спочатку використовуються два генератори. Ключ – початковий стан.

$$A_i = (A_{i-55} + X_{i-24}) mod\ 2^{32} \quad (1.9)$$

$$B_i = (B_{i-52} + B_{i-19}) mod\ 2^{32} \quad (1.10)$$

Послідовності проріджуються парами залежно від найменш значущого біта B_i: якщо він дорівнює 1, то пара використовується, при рівні 0 – ігнорується. C_j – послідовність слів Ai, а D_j – використані слова B_i. Для генерації слів K_{2j} та $K_{2j}+_1$ ці слова використовуються парами –C_{2j}, C_{2j+1}, D_{2j}, D_{2j+1}.

$$E_{2j} = C_{2j} \oplus (D_{2j} \wedge D_{2j+1}) \quad (1.11)$$

Fish працює швидко але він не єбезпечний.

1.4 Висновки та постановка задачі

Обґрунтувавши залежність стійкості криптографічних алгоритмів від якості використовуваних у них ГВЧ, а також спираючись на досліджену раніше існуючи алгоритми генераторів випадкових чисел та алгоритми генераторів псевдовипадкових чисел, розглянувши їхні переваги та проаналізувавши недоліки, можна зробити висновок, що є необхідність досліджень та розробок спрямованих на створення швидкого та ефективного генератора випадкових чисел.

Отже, у даному розділі були розглянуті теоретичні відомості, що пояснюють значення та доцільність використання криптографічно стійких ГВЧ для вирішення задач інформаційної безпеки. У першій частині розділу наведені основні поняття про випадкові числа та їх послідовності. У другому підрозділі було дано поняття «багатокористувацького веб-ресурсу» та обґрунтовано необхідності використання в ньому якісних ГВЧ. Для отримання загального уявлення про будову ГВЧ, що допоможе в розробці алгоритму в третьому підрозділі були розглянуті та проаналізовані принципи роботи, а також основні переваги та недоліки, існуючих популярних модифікацій генераторів псевдовипадкових чисел.

Також у розділі описана постановка задачі та цілі роботи. Розглянутий теоретичний матеріал та проведений аналіз дають підґрунтя, що дозволяє спроектувати та розробити табличний генератор випадкових чисел на основі ентропії поведінки користувачів.

2 РОЗРОБКА АЛГОРИТМУ РОБОТИ ТАБЛИЧНОГО ГЕНЕРАТОРА ВИПАДКОВИХ ЧИСЕЛ НА ОСНОВІ ЕНТРОПІЇ ПОВЕДІНКИ КОРИСТУВАЧА

Враховуючи мету та цілі даної магістерської кваліфікаційної роботи, а саме підвищення захищеності веб-ресурсів стійкими криптографічними алгоритмами на основі генераторів випадкових чисел, що враховують ентропію поведінки користувача у багатокористувацькому середовищі доцільним буде розглянути та проаналізувати табличні ГВЧ як альтернативу фізичним та програмним генераторам випадкових послідовностей.

2.1 Аналіз можливості підвищення стійкості криптографічних алгоритмів за рахунок використання табличних генераторів випадкових чисел

Оскільки велика кількість криптографічних алгоритмів використовують в своїй роботі ГВЧ, від якості цих генераторів залежить стійкість таких алгоритмів. Розглянуті раніше програмні генератори псевдовипадкових чисел не можуть в повній мірі задовольнити потреби криптографії через наявні в них недоліки, фізичні генератори в свою чергу мають перевагу над ГПВЧ але їх використання потребує фізичної наявності такого генератора, що збільшує архітектурну складність системи та потребує додаткових фінансових витрат на їх розробку та підтримку.

Альтернативою може виступати табличний генератор випадкових чисел, оскільки він видає істино випадкову послідовність та не потребує фізичної наявності модуля генерації в системі.

Табличні ГВЧ як джерело випадкових чисел використовують спеціальним чином складені таблиці, що містять перевірені некорельовані, тобто ніяк не залежать один від одного, цифри. В таблиці 2.1 наведено невеликий фрагмент такої таблиці.

Таблиця 2.1 Випадкові цифри. Рівномірно розподілені від 0 до 1 випадкові числа

Випадкові числа								Рівномірно розподілені від 0 до 1 випадкові числа
9	2	9	2	0	4	2	6	0.929
9	5	7	3	4	9	0	3	0.204
5	9	1	6	6	5	7	6	0.269
…								…

Принцип роботи такого генератора можна умовно розділити на два етапи:

1) Заповнення таблиці істино випадковими числами;

2) Вибір з таблиці випадкових чисел.

Для роботи генератора потрібно мати завчасно зібрану таблицю некорельованих чисел. Для цієї цілі можуть використовуватись будь-які джерела ентропії. Розповсюдженою практикою є отримання набору чисел з апаратного генератора випадкових чисел, який в свою чергу використовує фізичні джерела ентропії для отримання випадкової послідовності чисел.

Наступним етапом буде безпосередня робота табличного ГВЧ:

Крок 1. З лічильника береться значення індексу останньої комірки таблиці яка використовувалась в попередньому циклі.

Крок 2. З таблиці вибирається число відповідного індексу i+1.

Крок 3. В лічильник записується новий індекс i+1.

Крок 4. Число передається на модуль постобробки.

Кроки 2-4 повторюються n разів, де n- кількість розрядів числа.

Крок 5. Конкатенація отриманих чисел.

Крок 6. Приведення отриманого числа до необхідного діапазону.

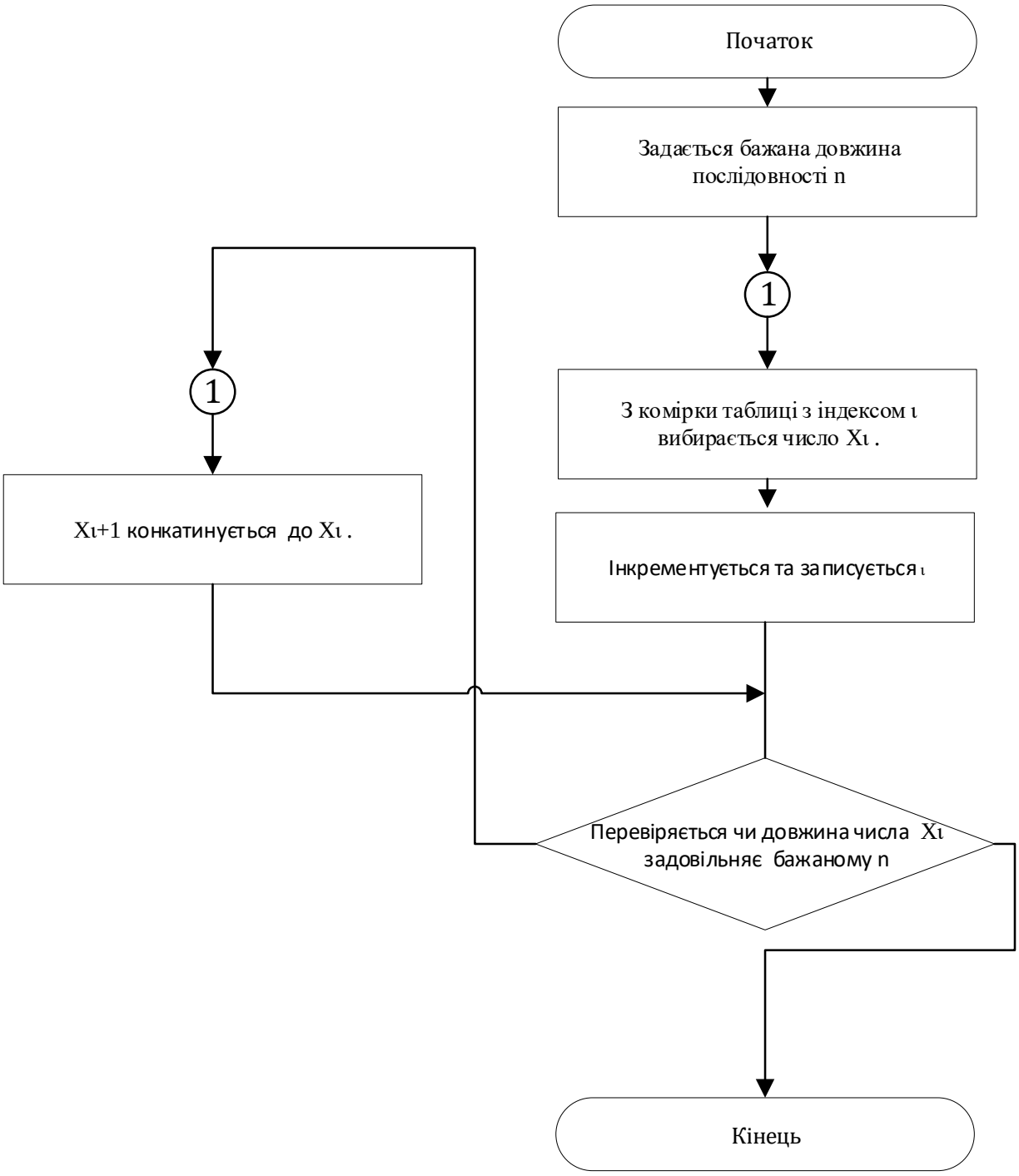


Рисунок 2.1 – алгоритм роботи табличного ГВЧ

Обходячи таблицю зліва направо зверху вниз, можна отримувати рівномірно розподілені від 0 до 1 випадкові числа з потрібним числом знаків після коми (в нашому прикладі ми використовуємо для кожного числа по три знаки).

Так як числа в таблиці не залежать один від одного, то числа з неї можна обирати різними способами, наприклад, зверху вниз, або справа наліво, або, скажімо, можна вибирати цифри, що знаходяться на парних позиціях.

Перевага даного методу в тому, що він дає дійсно випадкові числа, так як таблиця містить перевірені некорельовані цифри. Недоліки методу: для зберігання великої кількості цифр потрібно багато пам'яті; великі труднощі породження і перевірки такого роду таблиць, повтори при використанні таблиці вже не гарантують випадковості числової послідовності, а значить, і надійності результату.

Табличні генератори випадкових чисел можуть виступати альтернативою апаратним генераторам для використання їх в криптографічних алгоритмах за умови використання швидкодіючого джерела істино випадкових послідовностей для їх ініціалізації.

2.2 Розробка алгоритму надійного джерела ентропії на основі поведінки користувача веб-ресурсу

Якщо розглядати випадок використання ГВЧ як складову криптографічних алгоритмів в роботів веб-ресурсу, наприклад соціальної мережі, то поведінка користувачів даної системи може слугувати в якості джерела ентропії істино випадкових чисел для табличного генератора випадкових чисел.

Розглянемо отримання випадкових значень на прикладі популярної соціальної мережі Facebook.

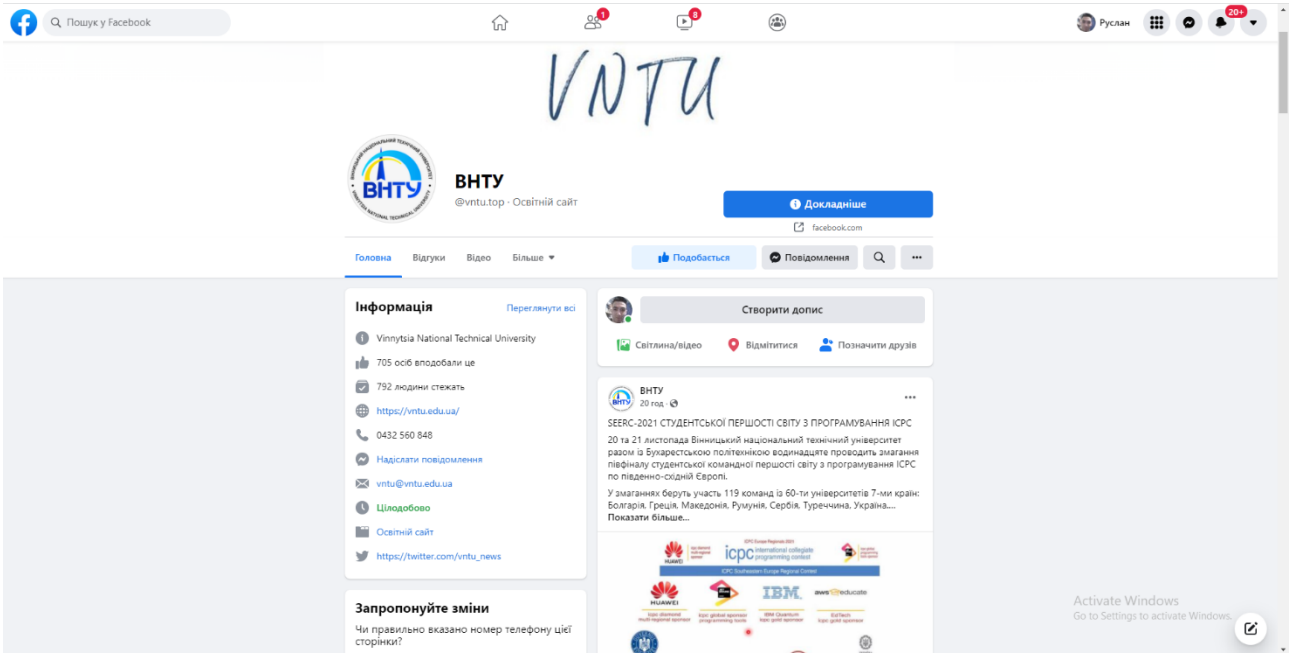


Рисунок 2.2 – вигляд інтерфейсу користувача соціальної мережі Facebook

Листаючи стрічку новин користувач підсвідомо здійснює випадкові дії, відслідковуючи які, ми можемо їх використовувати в якості ентропії для генератора випадкових чисел.

Для аналізу випадковості таких дій проведемо експеримент, в якості випадкового значення будемо використовувати значення координат курсору користувача.

Відкривши консоль розробника в веб-браузері Chrome підпишемось на подію «onmousemove», після кожного спрацювання події ми будемо отримувати об’єкт інтерфейсу MouseEvent який представляє події, які відбуваються внаслідок взаємодії користувача з вказівним пристроєм (наприклад, мишею).

Об’єкт MouseEvent має в собі властивості наведені в таблиці 2.1

Для отримання випадкових значень ми можемо брати одне або декілька значень властивостей комбінуючи їх.

Таблиця 2.1 Властивості об’єкту MouseEvent

Назва	Опис
MouseEvent.altKey	Повертає true, якщо клавіша alt була натиснута під час запуску події миші.
MouseEvent.button	Номер кнопки, яка була натиснута (якщо є) під час запуску події миші.
MouseEvent.buttons	Кнопки, які натискаються (якщо є) під час запуску події миші.
MouseEvent.clientX	Координата X вказівника миші в локальних координатах (вміст DOM).
MouseEvent.clientY	Координата Y вказівника миші в локальних координатах (вміст DOM).
MouseEvent.ctrlKey	Повертає істину, якщо клавіша керування була натиснутою під час запуску події миші.
MouseEvent.movementX	Координата X вказівника миші відносно позиції останньої події mousemove.

Продовження таблиці 2.1

MouseEvent.movementY	Координата Y вказівника миші відносно позиції останньої події переміщення миші.
MouseEvent.pageX	Координата X вказівника миші відносно всього документа.
MouseEvent.pageY	Координата Y вказівника миші відносно всього документа
MouseEvent.screenX	Координата X вказівника миші в глобальних (екранних) координатах.
MouseEvent.screenY	Координата Y вказівника миші в глобальних (екранних) координатах

Під час кожного спрацювання події onmousemove ми будемо брати значення властивостей MouseEvent.clientX та MouseEvent.clientY та записувати їх в глобальні зміні X та Y.

Також запустимо функцію setInterval в якій кожні 500 мс будемо брати значення з змінних X та Y та записувати їх в масив ArrayX та ArrayY.

В результаті отримає наступний програмний код.

```
var x=0;
var y=0;
var arrayX=[];
var arrayY=[];
        window.onmousemove = function(e) { x = e.clientX;y = e.clientY; }
setInterval(function() { arrayX.push(x);arrayY.push(y)},500)
```

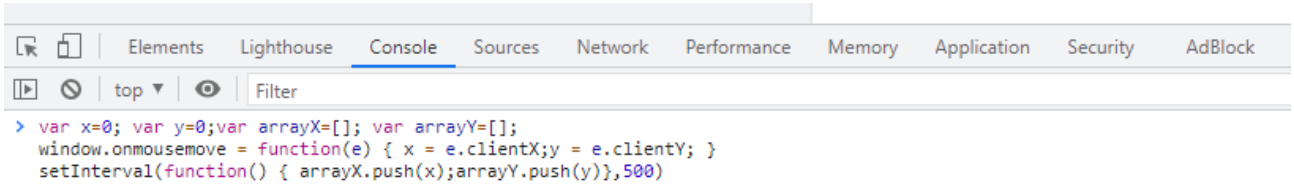


Рисунок 2.3 – вигляд консолі розробника веб-браузера Chrome

Для збору даних, двом користувачам було запропоновано переглянути сторінку новин, не оголошуючи їм суті експерименту. Після того як кожен користувач переглянув стрічку на основі зібраних даних було побудовано два графіки.

Оскільки користувачам потрібен був деякий час, щоб сісти за робоче місце та почати листати стрічку новин та в кінці був не обхідний час щоб зупинити збір даних, початок та кінець вибірки був відкинутий оскільки мишка була в нерухомому стані. Даний факт потрібно враховувати в розробці оскільки користувач може відійти від робочого місця або просто не використовувати комп'ютерну мишку, що може призвести до заповнення таблиці однаковими даними.

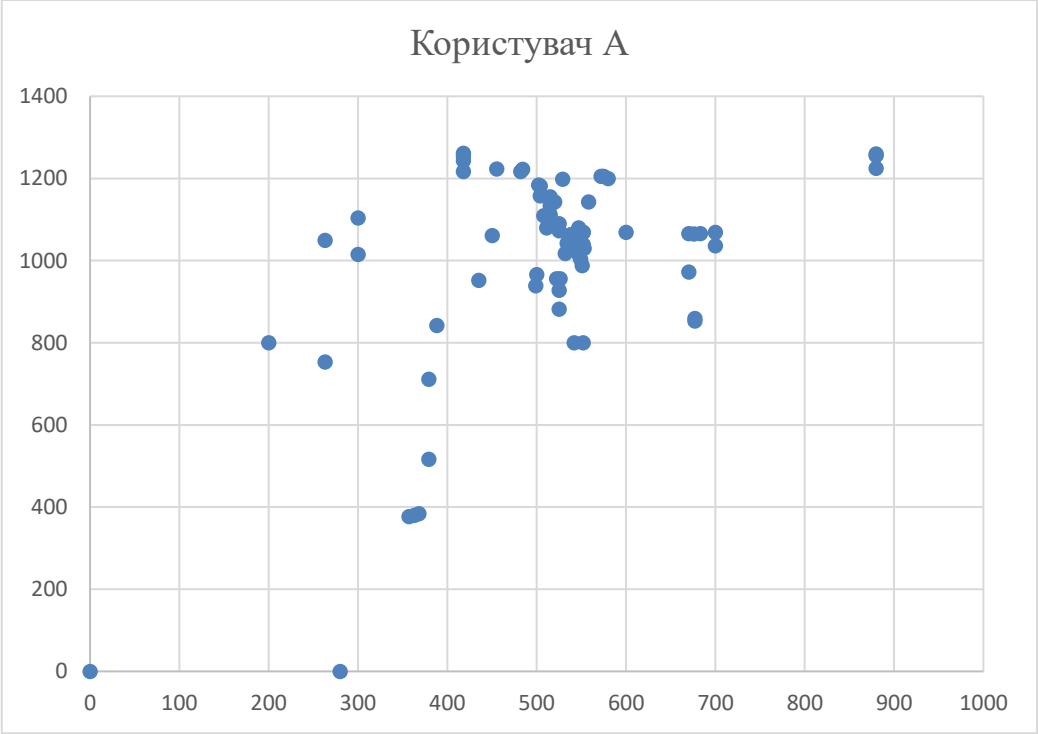


Рисунок 2.4 – Графік розподілу координат курсору користувача А.

З графіків видно, що між точками які є координатами вказівника миші в локальних координатах (вміст DOM) немає ніякої залежності.

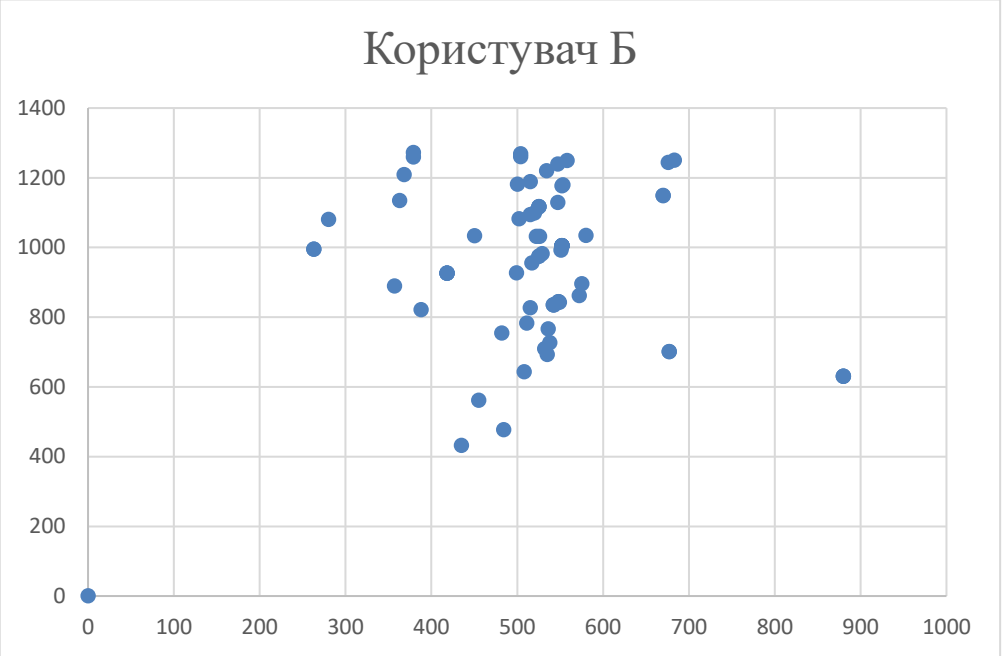


Рисунок 2.5 – Графік розподілу координат курсору користувача Б.

Результуюча послідовність бітів повинна мати розподіл, максимально близький до рівномірного. Для досягнення бажаного результату потрібен окремий важливий етап, званий постобробкою та реалізований шляхом застосування різноманітних спеціальних алгоритмів (рис. 2.5).

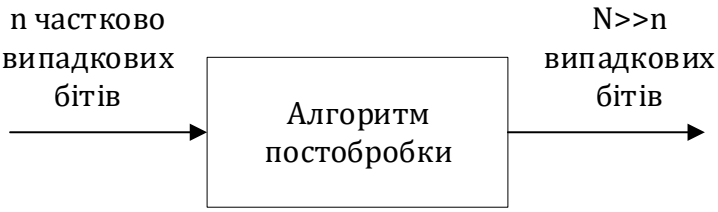


Рисунок 2.6 – Схема процесу постобробки ВП.

Сенс постобробки полягає у готовності пожертвувати певною часткою бітів, щоб отримати менший, але який володіє більш високим ступенем випадковості набір.

В якості методу постобробки використаємо один з варіантів спеціальних простих коректорів, а саме застосування операції XOR до біт послідовностей, отриманих від двох або більше паралельно працюючих генераторів.

Застосуємо XOR для послідовностей отриманих від користувача А та користувача Б.

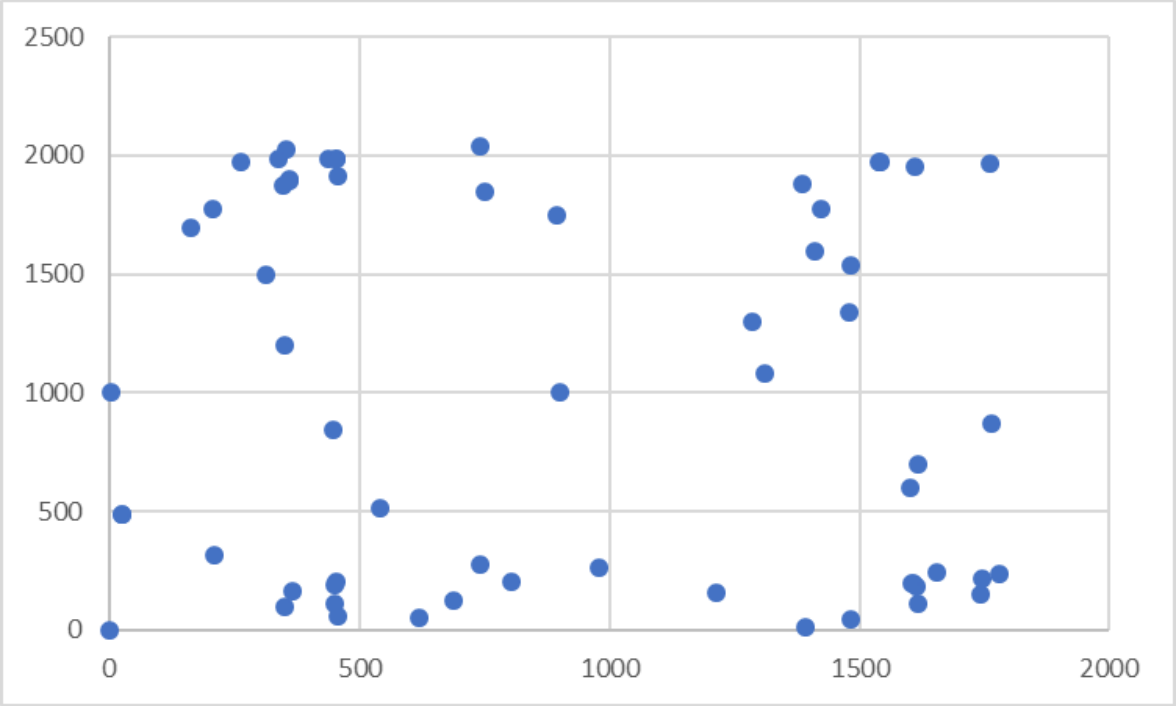


Рисунок 2.7 – Графік розподілу після постобробки

Розглянувши графік зображений на рисунку 2.7 можна зробити висновок про позитивний вплив функції постобробки та наближення розподілу до рівномірного.

Враховуючи кількість користувачів які одночасно активно використовують систему кількість ентропії яку можна отримати розробленим методом є нескінченою, також оскільки в алгоритмі не приймають участь жодні складні математичні операції ми можемо судити про швидкодію такого джерела ентропії.

Використовуючи поведінку користувачів як джерело ентропії для табличного генератора випадкових чисел можна позбутись основних його недоліків, а саме, відпаде необхідність тримати в пам’яті таблицю великих розмірів, оскільки таке джерело ентропії є швидкодіючим та фактично не скінченим. Таблицю можна буде заповнювати під час роботи системи, що дасть можливість використовувати менші об’єми пам’яті, скільки відпаде необхідність в великій таблиці.

2.3 Розробка алгоритму роботи табличного генератора випадкових чисел

Проаналізувавши отриману інформацію ми можемо об’єднати отримані окремі модулі ГВЧ, а саме алгоритм заповнення буфера табличного генератора випадкових чисел та алгоритм вибору з нього випадкового числа, для подальшої програмної реалізації підсистеми шифрування.

Розроблений алгоритм можна розділити на два етапи:

Першим буде заповнення буферу табличного ГВЧ.

Він складається з наступних кроків:

Крок 1. Вибирається деякі величини які характеризують стан системи джерелом зміни якого є користувач.

В розробленій реалізації в якості показників ентропії будуть використовуватись координати курсора користувача X та Y.

Крок 2. Вираховується $Z_n = X_n \oplus Y_n$ яке є результатом ентропії поведінки користувача.

Крок 3. Перевіряється чи Z_n не рівне Z_{n-1}.

Даний крок необхідний для недопущення заповнення буферу табличного ГВЧ однаковими бітами. Такий сценарій можливий коли користувач не здійснює жодних дій, в такому випадку користувач перестає бути джерелом ентропії.

Крок 4. Z_n надсилається з клієнтської частина на сервер.

Крок 5. На стороні сервера вираховується $Q = Z_{An} \oplus Z_{Bn}$, де Z_{An} - значення ентропії користувача A, а Z_{Bn} значення ентропії користувача B отримане аналогічним чином.

Крок 6. Q записується в комірку таблиці з індексом ι

Крок 7. Інкрементується та записується ι.

Крок 8. Перевіряється якщо ι більше розміру таблиці, ι присвоюється 0

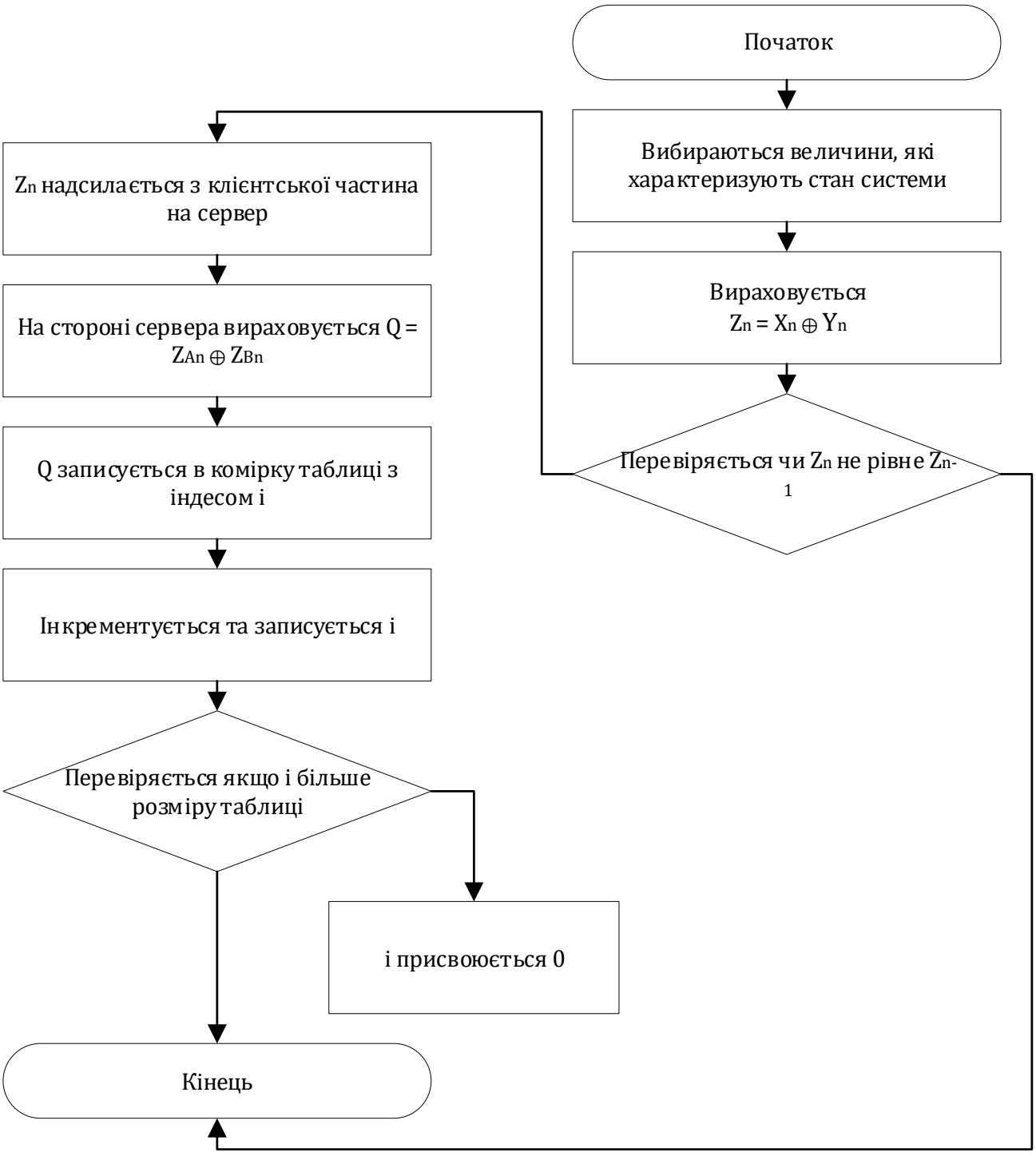


Рисунок 2.8 – Алгоритм заповнення буферу табличного ГВЧ

Наступним етапом буде вибір випадкового числа.

Даний етап складається з наступних кроків:

Крок 1. В модуль передається бажана довженна послідовності n.

Крок 2. З комірки таблиці з індексом ι вибирається число X_i.

Крок 3. Інкрементується та записується ι.

Крок 4. Перевіряється чи довжина числа X_i задовільняє бажаному n.

Крок 5. Якщо довжина X_i менша n, вибирається наступне число X_{i+1} (повторюються кроки 2,3) та конкатинується до X_i. Числа вибираються та конкатинуються до тих пір доки довжина числа X_i не задовольняє бажаному n.

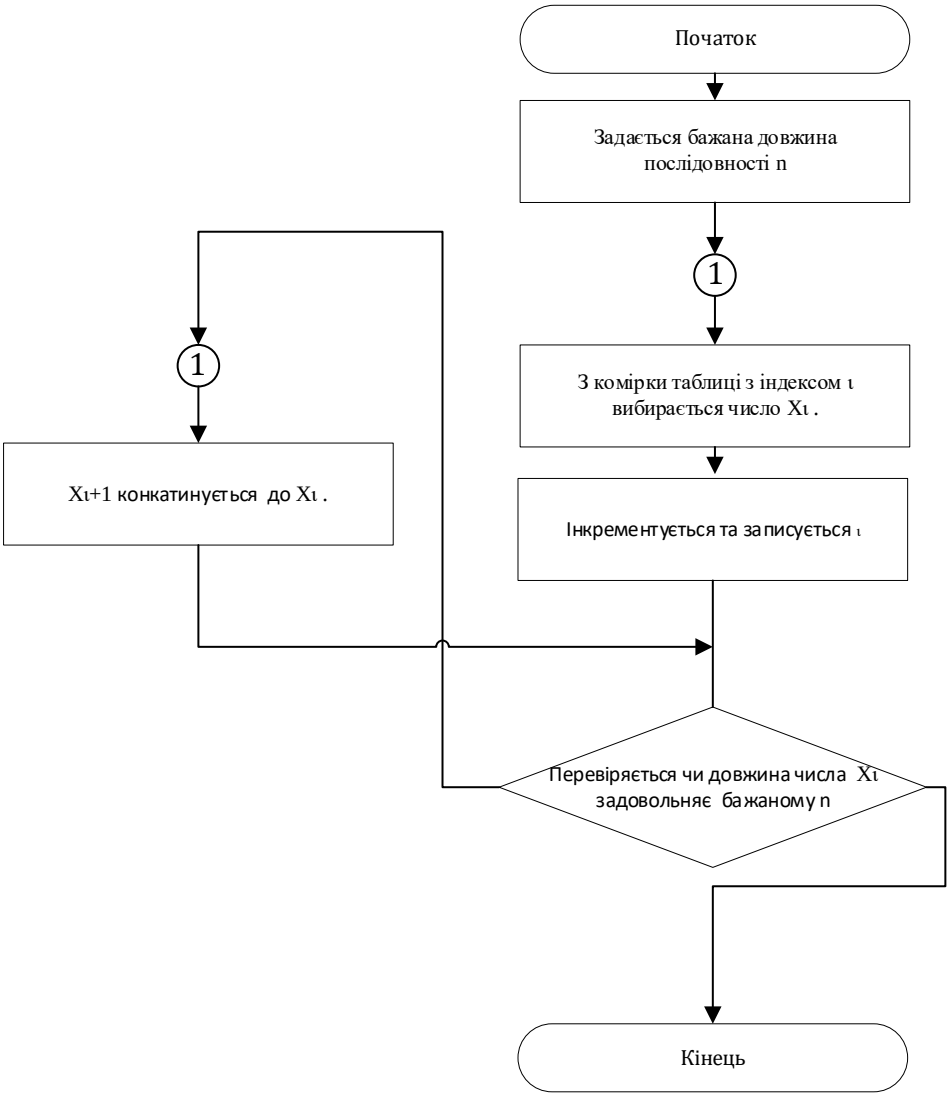


Рисунок 2.9 – Алгоритм вибору випадкового числа

3 РОЗРОБКА РОЗРОБЛЕНОГО ТАБЛИЧНОГО ГЕНЕРАТОРА ВИПАДКОВИХ ЧИСЕЛ

Даний розділ передбачає програмну розробку табличного генератора випадкових чисел. У розділі розглянуто обрану мову програмування, розроблені модулі ГВЧ, а також наведено результати статистичного тестування розробленого генератора випадкових послідовностей.

3.1 Обґрунтування вибору мови програмування

Оскільки архітектура розробленого табличного генератора випадкових чисел представляє собою клієнт-серверну для його програмної реалізації необхідно буде розробити два програмних модуля перший з них буде підключатись на стороні сервера а другий на стороні клієнта.

Для серверної частини була об’єктно-орієнтована мова програмування C# та інтегроване середовище програмування Visual Studio 2022.

C# — об’єктно-орієнтована мова програмування для платформи .NET Framework. Мова заснована на компонентній архітектурі і реалізує передові механізми забезпечення безпеки типів [18].

C# був створений спеціально для платформи .NET. У той же час на C # повністю написана і сама платформа .NET. C# розроблявся як мова програмування прикладного рівня для CLR, залежить, насамперед, від можливостей самої CLR. CLR надає C #, як і всім іншим .NET-орієнтованим мовам, багато можливостей, яких позбавлені «класичні» мови програмування (C++, Java, Pascal, Delphy та ін.).

Мова C# розроблялась «з нуля» і увібрала в себе багато корисних властивостей таких мов, як C++, Java, Visual Basic, а також Pascal, Delphy та ін. При цьому необхідність зворотної сумісності з попередніми версіями відсутня, що дозволило мові C # уникнути багатьох негативних сторін своїх попередників. Вона містить лише необхідні конструкції. Мова програмування C# є однією з найпростіших і найдоступніших. [18].

C# — це повнофункціональна об’єктно-орієнтована мова, що підтримує всі три «стовпи» об’єктно-орієнтованого програмування: інкапсуляцію, успадкування і поліморфізм. Всі змінні і методи, включаючи метод Main - точку входу додатку - інкапсулюються в визначення класів. Клас може успадковувати безпосередньо з одного родового класу, але може реалізовувати будь-яке число інтерфейсів. Для методів, які перевизначать віртуальні методи в батьківському класі, необхідно ключове слово override, щоб виключити випадкове повторне визначення. У мові C# структура схожа на полегшений клас: це тип, який розподіляється по стопках, реалізовує інтерфейси, але не підтримує спадкування. [18].

У C # використовуються звичайні привласнення для простих змінних; структурні або масові присвоювання не підтримуються. Оператори розгалуження теж досить традиційні (if, switch), але володіють двома особливостями. По-перше, умова в операторі if має виробляти саме булеве значення (тобто цілого значення, що виробляється при привласненні недостатньо), а по-друге, кожна гілка case всередині оператора switch повинна містити явну вказівку про подальший потік управління (тобто або break, або goto на якусь змінну, наприклад, на мітку іншої вітки) [18].. Що стосується циклів, то C# підтримує цілком традиційні цикли, такі як dowhile, while-do і цикли з ітерацією for, але крім цього, підтримує перебір масивів і колекцій за допомогою оператора foreach. C # підтримує структурну обробку виключень за допомогою конструкцій try, catch і finally. Винятки можна генерувати і явним чином за допомогою конструкції throw. [9]

Також дана мова програмування має вбудовані методи, такі як Regular Expressions та багато інших.

До числа принципово важливих рішень, які реалізовані корпорацією Microsoft у мові програмування C #, можна віднести наступні:

– компонентно-орієнтований підхід до програмування (який характерний і для ідеології Microsoft. NET в цілому);

– чітка типізація змінних;

– властивості як засіб інкапсуляції даних (характерно також в цілому для ООП);

– автоматична ініціалізація змінних;

– обробка подій;

– обробка виключень;

– делегати (delegate - розвиток покажчика на функцію в мовах C і C);

– індексатори (indexer - оператори індексу для звернення до елементів класу-контейнера);

– статична типізація;

– перевантажені оператори (розвиток ООП);

– оператор foreach (обробка всіх елементів класів-колекцій, аналог Visual Basic);

– підтримка компонентів;

– використання «збору сміття»;

– механізми boxing і unboxing для перетворення типів;

– атрибути (засіб оперування метаданими в COM-моделі);

– прямокутні масиви (набір елементів з доступом за номером індексу і однаковою кількістю стовпців і рядків) [18].

Для клієнтської частини генератора біла обрана мова програмування JavaScript

JavaScript – об'єктно-орієнтована мова програмування для написання сценаріїв. Найчастіше JavaScript використовується для написання сценаріїв роботи з web-сторінками, які відображаються web-браузером. Web-браузер інтерпретує код сценарію мови JavaScript, і на основі описаних у сценарії дій здійснює маніпуляції з розміткою web-сторінки. Таким чином, за допомогою JavaScript реалізується можливість програмування на стороні клієнта. Мова JavaScript надає можливість доступу до елементів розмітки веб-сторінки за допомогою об'єктів. При створенні сценаріїв JavaScript доводиться стикатися з труднощами, пов'язаними з тим, що різні web-браузери можуть по-різному інтерпретувати ці сценарії. Найсерйозніші труднощі виникають, якщо будь-який із браузерів не підтримує той чи інший об'єкт, метод чи властивість. Найбільш

практичним та сучасним способом вирішення цієї проблеми є використання вільної бібліотеки jQuery [7]. Ця бібліотека реалізована мовою JavaScript і розширює можливості цієї мови, нівелюючи різницю між браузерами

Також для розробки клієнтської частини буде використаний front-end фреймворк Angular.

Angular - бібліотека від компанії Google, призначена в основному розробки SPA (single page application) додатків. Бібліотека адаптує та розширює HTML за допомогою директив, а також синхронізує модель та подання за рахунок різного виду прив'язок даних.

Плюси Angular:

1) Angular поділяє DOM-маніпуляції та логіку програми. Цедопомагає зробити процес розробки та тестування простіше.

2) Angular добре працює з бібліотеками Karma та Jasmin, які призначені для тестування.

3) Angular дотримується MVC (Model View Controller) шаблону проектування та заохочує слабкий зв'язок між поданням, даними та логікою компонентів.

4) Використання типізованої мови Typescript запобігаєбезліч помилок у процесі розробки.

5) Angular можна використовувати для створення мобільних та десктоп додатків.

Отже, проаналізувавши всі переваги мов C# та JavaScript, можна зробити висновок, що вони найкраще підійдуть для реалізації поставленої задачі.

3.2 Програмна реалізація розробленого алгоритму генерації випадкових чисел

Розробивши алгоритм роботи та обравши технології для його реалізації можемо розпочати програмну реалізацію твбличного ГВЧ.

Першим етапом буде розробити серверної частину підсистеми. В якості платформи будемо використовувати .net фреймворк ASP.net core 3.1.

Першим кроком в класі Startup.cs зареєструємо маршрут та обробник для нього. На маршрут "/rng" будуть приходити POST запити з даними з клієнтської частини підсистеми.

```
app.Map("/rng", versionApp =>
      versionApp.UseMiddleware<RNGMiddleware>());
```

Клас RNGMiddleware описує обробник запитів, його програмний код виглядає наступним чином.

```
public class RNGMiddleware
  {
     readonly RequestDelegate _next;

     public RNGMiddleware(RequestDelegate next, ITabularRandomNumberGeneration
tabularRandomNumberGeneration)
     {
        _next = next;
        _tabularRandomNumberGeneration = tabularRandomNumberGeneration;
     }

     public ITabularRandomNumberGeneration _tabularRandomNumberGeneration { get; }

     public async Task Invoke(HttpContext context)
     {
        var requestReader = new StreamReader(context.Request.Body);
        var json = await requestReader.ReadToEndAsync();

        var Body = Newtonsoft.Json.JsonConvert.DeserializeObject<TraceModel>(json);
        await _tabularRandomNumberGeneration.Seed(Body.Seed);
        context.Response.StatusCode = 200;
        await context.Response.StartAsync();
     }
  }
```

В методі Invoke з тіла запиту зчитуються дані та після чого вони десеріалізуються в об’єкт класу TraceModel.

```
class TraceModel
  {
     public int Seed { get; set; }
```

```
}
```

Клас TraceModel описує одну властивість Seed, вона відображає етропію поведінки користувача.

В конструкторі класу RNGMiddleware ми можемо бачити ін’єкцію залежності сервісу ITabularRandomNumberGeneration.

```
public RNGMiddleware(RequestDelegate next, ITabularRandomNumberGeneration
tabularRandomNumberGeneration)
{
  _next = next;
  _tabularRandomNumberGeneration = tabularRandomNumberGeneration;
}
```

Даний сервіс є генератором випадкових послідовностей. Отримане значення Body.Seed, з клієнтської сторони, передається в метод Seed.

В методі Seed значення отримане з клієнтської сторони записується в таблицю слідуючи розробленому алгоритму.

```
public async Task Seed(int seed)
{
  if (_seed == null)
  {
    _seed = seed;
  }
  else
  {
    var digits=GetDigits((_seed ^ seed).Value);
    for (int i = 0; i < digits.Length; i++)
    {
      Buffer[_seedIndex + i] = digits[i];
    }

    _seedIndex = _seedIndex + digits.Length;
  }

}
```

Розглянемо розроблений клас TabularRandomNumberGeneration для кращого розуміння логіки роботи програми.

В класі описані наступні властивості

```
private int[] Buffer { get; set; }
private int? _seed { get; set; }
```

```
private int _seedIndex { get; set; }
private int _nextIndex { get; set; }
```

Buffer - безпосередньо таблиця в яку записуються та з якої вибираються випадкові числа.

_seed - зміна як тимчасово зберігає значення отримане з клієнтської сторони.

_seedIndex – індекс комірки таблиці з в яку було записано seed значення на минулої ітерації алгоритму.

_nextIndex – індекс комірки таблиці з на якій завершилась генерації послідовності на минулої ітерації алгоритму.

Для генерації випадкового числа з будь-якого місця веб-ресурсу, наприклад з модуля авторизації, розробнику необхідно буде викликати метод Next.

```
public int Next(int from, int to)
{
   if (to == 0 || to < from)
   {
      throw new ArgumentException();
   }
   int digitLength = (int)Math.Floor(Math.Log10(to) + 1);
   var randomNumber = GetRandom(digitLength);
   if (randomNumber > from && randomNumber < to)
   {
      return randomNumber;
   }
   else
   {
      return next(from, to);
   }

   return 0;

}
public async Task Seed(int seed)
```

В якості параметрів він приймай два значення from, to і використовує їх для вибору послідовності необхідного розміру.

Наступним кроком буде розробка клієнтської частини підсистеми.

Старт будь-якого Angular проекту розпочинається з компонента AppComponent.

В його конструкторі оголосимо ін’єкцію сервісів які нам знадобляться для реалізації, а саме:

```
constructor(private http: HttpClient, private AuthorizeService: AuthorizeService)
```

1) HttpClient – вбудований Angular сервіс для здійснення HTTP запитів;

2) AuthorizeService- сервіс який відповідає за авторизацію користувача.

Повний лістинг конструктора компонента AppComponent виглядає наступним чином

```
constructor(private http: HttpClient, private AuthorizeService: AuthorizeService) {
  var x;
  var y;
  this.subscription =
   fromEvent(document, 'mousemove')
     .subscribe((e: any) => {
       console.log(e)
       x = e.clientX; y = e.clientY;
     });


  setInterval(() => {
   if (AuthorizeService.isAuthenticated) {
     const entropy = x^y
     //  console.log( this.x )

     http.post("/rng", { Seed: entropy }).subscribe()

   }
  }, 500)


 }
```

В конструкторі оголошується дві змінні x та y для збереження координат курсору користувача.

Наступним кроком підписуємся на подію 'mousemove' та вказуємо її обробник.

В обробнику присвоюємо значення координат курсору e.clientX та e.clientY в зміні x,y.

Після чого запускаємо функцію setInterval яка кожні 500 мс буде перевіряти чо користувач був аутентифікований, якщо так то відправляти POST запит, на серверну частину під системи, із значенням ентропії отриманим шлях виконання операції XOR над змінними x та y.

3.3 Аналіз підвищення стійкості криптографічних алгоритмів за рахунок використання в них розробленого ГВЧ

Оскільки є пряма залежність стійкості криптографічних алгоритмів від якості та стійкості до криптоаналізу ГВЧ які в них використовуються, можна стверджувати, що мета даною магістерської кваліфікаційної роботи була досягнута у разі якщо, характеристик розробленого ГВЧ, а також його загальна стійкість до криптоаналізу буде перевершувати класичні та часто використовування ГВЧ.

Першим етапом, без проходження якого подальша оцінка ГВЧ не нестиме ніякого значення, є проходження статистичного тестування.

Національний інститут стандартів та технологій (NIST) винайшов 16 тестів випадковості.

Для їх походження скористаємся офіційним додатком національного інститут стандартів та технологій NIST Statistical Test Suite.

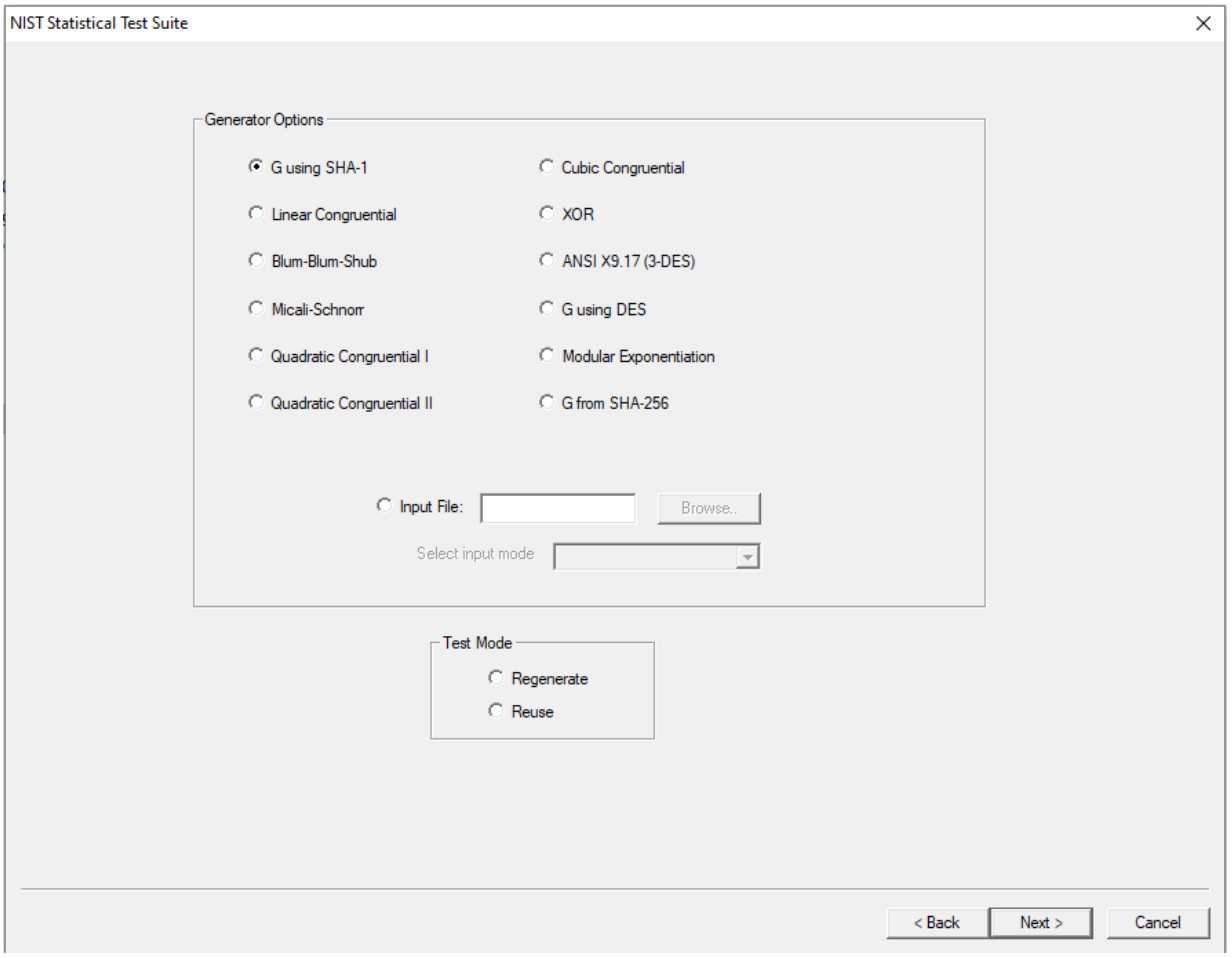


Рисунок 3.1 – головне вікно додатку NIST Statistical Test Suite.

Вибравши опцію Input File ми можемо вказати шлях до файлу в якому знаходиться заздалегідь згенерована розробленим генератором послідовність.

Наступним кроком потрібно обрати які саме тести потрібно пройти

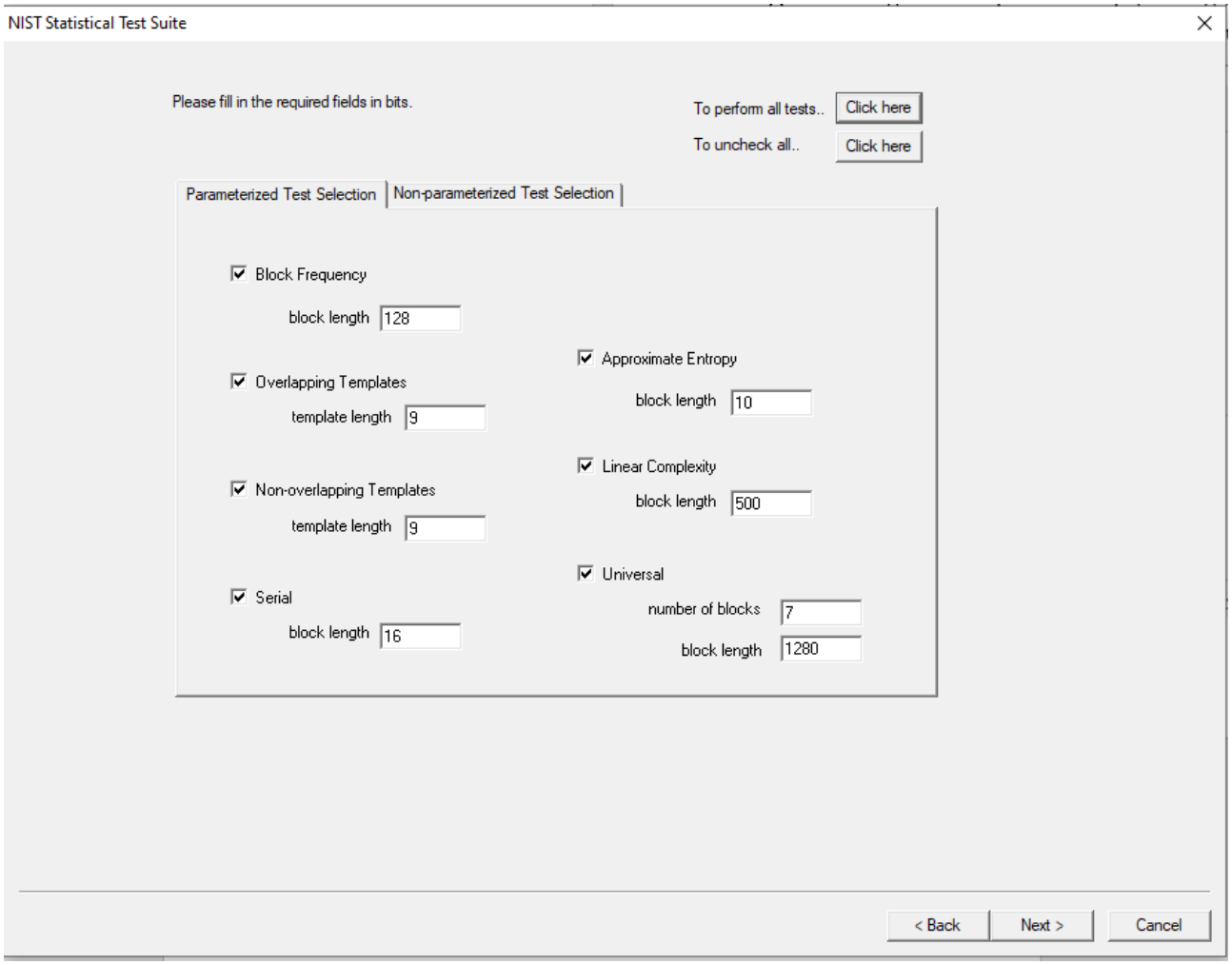


Рисунок 3.2 – Вікно вибору необхідних тестів.

Обравши всі можливі тести запускаємо тестування послідовності.

Результати будуть збережені в текстовому файлі в кореневій директорії програми.

Фрагмент файлу кінцевого звіту, формованого пакетом NIST STS, для кілька перших тестів наведено в на рисунку 3.3

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	PROPORTION	STATISTICAL TEST
135	149	122	96	117	151	164	120	103	123	0.00014	0.993	frequency
121	109	134	121	126	133	137	134	137	128	0.775277	0.9906	block-frequency
126	133	142	143	102	128	83	142	153	128	0.000292	0.9914	cumulative-sums
131	130	114	114	108	146	108	140	154	135	0.027549	0.9922	cumulative-sums
131	130	120	132	140	128	139	131	108	121	0.701879	0.9906	runs

Рисунок 3.3 – фрагмент результату проходження тестів

На рисунку 3.4 представлена діаграма, що характеризую попадання частки послідовностей, що пройшли кожен тест в довірчий інтервал [0,96; 1].

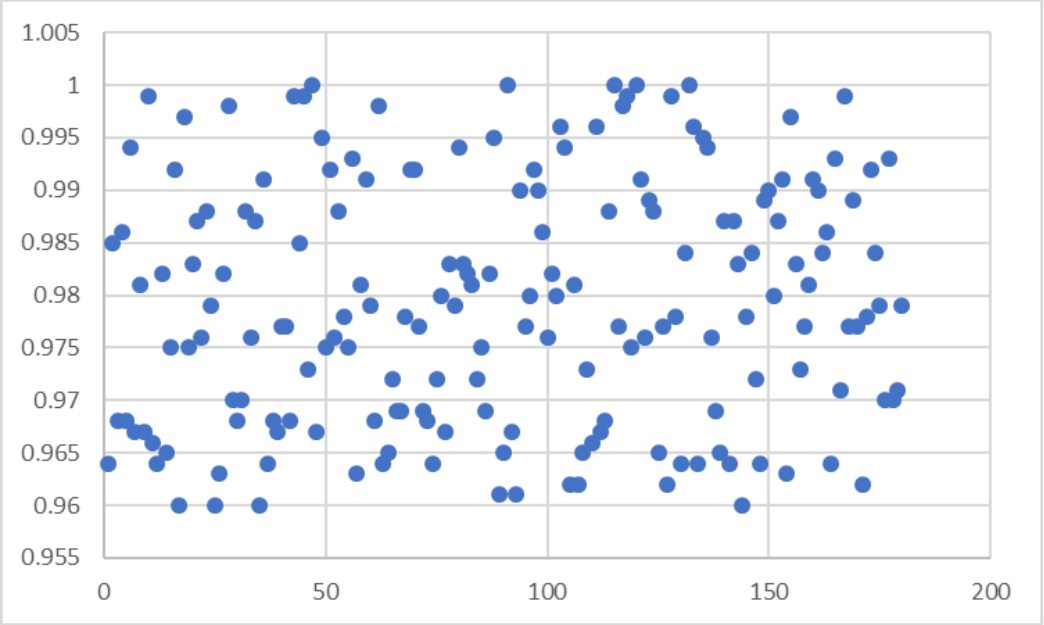


Рисунок 3.4 – Результати тестування послідовності згенерованої розробленим ГВЧ

З отриманих результатів можна зробити висновок, що послідовність повністю задовольняє критеріям випадковості.

Наступним кроком проведемо порівняльний аналіз розробленого ГВЧ з класичними реалізаціями.

Таблиця 3.1 Порівняльна характеристика типів ГВЧ

Характеристика	Апаратні ГВЧ	Програмні ГВЧ	Розроблений ГВЧ
Відсутність періоду	Так	Ні	Так
Непередбачуваність	Так	Умовна	Так
Незалежність значень	Так	Умовна	Так
Рівень крипостійкості	Високий	Умовний	Високий
Швидкість генерації	Низька	Висока	Висока
Відтворюваність	Ні	Так	Так
Простота генерації	Ні	Так	Так
Вартість генерації	Висока	Низька	Низька
Необхідність апаратної реалізації	Так	Ні	Ні

Проаналізувавши дані наведені в таблиці 3.1 можна зробити висновок, що розроблений ГВЧ комбінує переваги апаратних і програмних ГВЧ і також в ньому відсутні недоліки обох видів.

Оскільки не завжди є фінансова та фізична змога використовувати апаратні ГВЧ, а вони є більш криптографічно стійкими в порівняні з програмними, використання розробленого табличного ГВЧ дасть змогу отримувати істино випадкові та криптографічно якісні послідовності чисел без апаратної реалізації.

Цей факт дає підґрунтя вважати, що за рахунок використання використання розробленого ГВЧ можна підвищити стійкість криптографічних алгоритмів в умовах коли використання апаратних ГВЧ не є можливим або доцільним.

3.4 Висновок

Отже, у даному розділі представлена робота, що мала на меті розробку підсистему генерації випадкових чисел, з використанням табличного ГВЧ на основі ентропії поведінки користувача, для використання її в багатокористувацьких веб ресурсах. У першій частині розділу обґрунтовано обрану мову програмування – а саме інтерпретована об’єктно-орієнтована мова програмування C#, можливості та ресурси якої дозволяють у повній мірі здійснити програмну реалізацію підсистеми генерації випадкових чисел.

В другій частині розділу описано програмну реалізацію розробленого раніше табличного ГВЧ.

В третій частині розділу проаналізовано підвищення криптографічної стійкості алгоритмів за рахунок використання в них розробленого табличного генератора випадкових чисел на основі поведінки користувача в багатокористувацькому веб ресурсі .

Таким чином, даний розділ описує та пояснює програмну реалізацію розробленого табличного генератора випадкових чисел на основі ентропії поведінки користувача в багатокористувацькому веб-ресурсі, а також аргументую якого використання для досягнення мети роботи.

4 ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПІДВИЩЕННЯ СТІЙКОСТІ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ

4.1 Проведення наукового аудиту науково-дослідної роботи

Суб'єктами виконання науково-дослідних робіт і розробок є наукові організації, науково-дослідні центри при закладах вищої освіти, науково-дослідні, проектно-конструкторські організації, експериментальні підприємства, а також науково-виробничі об'єднання.

Метою фундаментальних і частково пошукових досліджень не є одержання продукту, виробу або послуги, що можуть стати товаром і оформитися у вигляді певного комерційного інвестиційного проекту. Однак на їхній основі здійснюється генерація ідей, які можуть трансформуватися в проекти науково-дослідних і дослідно-конструкторських робіт. Тому пошукові роботи можуть мати деяку комерційну цінність.

Наукова і науково-технічна результативність НДР не може бути оцінена з використанням методу дисконтування грошових потоків, за винятком випадків, коли дослідження мають вартісні характеристики результату НДР як наукової інформації, тому у деяких випадках результати дослідження можуть мати вартісні характеристики результату науково-дослідної роботи як наукової інформації, яку купує замовник. Тобто у такому випадку може виникнути фактична ефективність науково-дослідної роботи.

Для наукових і пошукових науково-дослідних робіт зазвичай здійснюють оцінювання наукового ефекту.

Основними ознаками наукового ефекту науково-дослідної роботи є новизна роботи, рівень її теоретичного опрацювання, перспективність, рівень розповсюдження результатів, можливість реалізації. Науковий ефект НДР можна охарактеризувати двома показниками: ступенем наукової новизни та рівнем теоретичного опрацювання.

Значення показників ступеня новизни і рівня теоретичного опрацювання

науково-дослідної роботи в балах наведено в табл. 4.1 та 4.2.

Таблиця 4.1 – Показники ступеня новизни науково-дослідної роботи

Ступінь новизни	Характеристика ступеня новизни	Значення показника ступеня новизни, бали
1	*2*	*3*
Принципово нова	Робота якісно нова за постановкою задачі і ґрунтується на застосуванні оригінальних методів дослідження. Результати дослідження відкривають новий напрям в цій галузі науки і техніки. Отримано принципово нові факти, закономірності; розроблено нову теорію. Створено принципово новий пристрій, спосіб, метод	60…100
Нова	Отримано нову інформацію, яка суттєво зменшує невизначеність наявних значень (по-новому або вперше пояснено відомі факти, закономірності, впроваджено нові поняття, розкрито структуру змісту). Проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів	40…60
Відносно нова	Робота має елементи новизни в постановці задачі і методах дослідження. Результати дослідження систематизують і узагальнюють наявну інформацію, визначають шляхи подальших досліджень; вперше знайдено зв’язок (або знайдено новий зв’язок) між явищами. В принципі, відомі положення поширено на велику кількість об’єктів, в результаті чого знайдено ефективне рішення. Розроблено більш прості способи для досягнення відомих результатів. Проведено часткову раціональну модифікацію (з ознаками новизни)	10…40
Традиційна	Робота виконана за традиційною методикою. Результати дослідження мають інформаційний характер. Підтверджено або поставлено під сумнів відомі факти та твердження, які потребують перевірки. Знайдено новий варіант рішення, який не дає суттєвих переваг порівняно з існуючим	2…10
Не нова	Отримано результат, який раніше зафіксований в інформаційному полі та не був відомий авторам	1…2

Таблиця 4.2 – Показники рівня теоретичного опрацювання науково-дослідної роботи

Характеристика рівня теоретичного опрацювання	Значення показника рівня теоретичного опрацювання, бали
1	2
Відкриття закону, розробка теорії	80…100

Продовження таблиці 4.2

1	2
Глибоке опрацювання проблеми: багатоаспектний аналіз зв'язків, взаємозалежності між фактами з наявністю пояснень, наукової систематизації з побудовою евристичної моделі або комплексного прогнозу	60…80
Розробка способу (алгоритму, програми), пристрою, отримання нової речовини	20…60
Елементарний аналіз зв'язків між фактами та наявною гіпотезою, класифікація, практичні рекомендації для окремого випадку тощо	6…20
Опис окремих елементарних фактів, викладення досвіду, результатів спостережень, вимірювань тощо	1…5

Показник, який характеризує науковий ефект, визначається за формулою:

$$\mathrm{E_{нау}} = 0{,}6 \cdot k_{\text{нов}} + 0{,}4 \cdot k_{\text{теор}},,,, \qquad (4.1)$$

де $k_{нов}$, $k_{теор}$ – показники ступенів новизни та рівня теоретичного опрацювання науково-дослідницької роботи, бали;

0,6 та 0,4 – питома вага (значимість) показників ступеня новизни та рівня теоретичного опрацювання науково-дослідної роботи

$$\mathrm{E_{нау}} = 0{,}6 \cdot 45 + 0{,}4 \cdot 60 = 51,,,$$

Визначення характеристики показника $E_{нау}$ проводиться на основі висновків експертів, виходячи з граничних значень, які наведено в табл. 2.3.

Таблиця 4.3 – Граничні значення показника наукового ефекту

Досягнутий рівень показника	Кількість балів
Високий	70…100
Середній	50…69
Достатній	15…49
Низький (помилкові дослідження)	1…14

Відповідно до таблиці 4.3 значення показника наукового ефекту має середній рівень і становить 51 бал, адже для розробки приладу було проведено суттєве вдосконалення, доповнення і уточнення раніше досягнутих результатів.

4.2 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності, тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, які є провідними викладачами випускової кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснімо із застосуванням п'ятибальної системи оцінювання за 12-ма критеріями, наведеними в табл. 4.4.

Таблиця 4.4 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
№	0	1	2	3	4
1	2	3	4	5	6
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 4.4

1	2	3	4	5	6

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної Динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна Конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно зведемо до таблиці 4.5.

Таблиця 4.5 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки

Критерії	Експерт		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	3	3	4
2. Ринкові переваги (наявність аналогів)	4	2	4
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	4	4	2
5. Ринкові переваги (експлуатаційні витрати)	4	3	3
6. Ринкові перспективи (розмір ринку)	3	3	2
7. Ринкові перспективи (конкуренція)	3	2	2
8. Практична здійсненність (наявність фахівців)	2	3	3
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	4	2	3
11. Практична здійсненність (термін реалізації)	2	1	4
12. Практична здійсненність (розробка документів)	3	3	4
Сума балів	$СБ_1=39$	$СБ_2=33$	$СБ_3=35$
Середньоарифметична сума балів $СБ_с$	$СБ_с = \frac{\sum_{i=1}^{3} СБ_1}{3}=35,66$		

За результатами розрахунків, наведених в таблиці 4.5, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 4.6.

Таблиця 4.6 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вищий середнього
21…30	Середній
11…20	Нижчий середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки становить 35,66 бали, що, згідно таблиці 4.6, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього. Дане програмне забезпечення надає підвищення стійкості криптографічних алгоритмів за рахунок використання в них генератора випадкових чисел на основі ентропії поведінки користувача в багатокористувацькому веб-ресурсі. Однією з практичних функцій на відмінну

від вже існуючих аналогів є використання табличного генератора випадкових чисел на основі ентропії поведінки користувача для підвищення стійкості криптографічних алгоритмів.

4.3 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної, дослідно-конструкторської, конструкторсько-технологічної роботи, створенням дослідного зразка і здійсненням виробничих випробувань, під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуються за такими статтями:

- витрати на оплату праці;
- відрахування на соціальні заходи;
- паливо та енергія для науково-виробничих цілей;
- витрати на службові відрядження;
- спецустаткування для наукових (експериментальних) робіт;
- програмне забезпечення для наукових (експериментальних) робіт;
- витрати на роботи, які виконують сторонні підприємства, установи і організації;
- інші витрати;
- накладні (загальновиробничі) витрати.

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на

оплату праці».

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників (Зо) розраховують відповідно до посадових окладів працівників, за формулою:

$$З_о = \sum_{i=1}^{k} \frac{М_{ni} \cdot t_i}{Т_р}, \quad (4.2)$$

де k – кількість посад дослідників, залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – кількість днів роботи конкретного дослідника, дн.;

T_p – середня кількість робочих днів в місяці, T_p=*21…23* дні.

Проведені розрахунки зведемо до таблиці 4.7.

Таблиця 4.7 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Кількість днів роботи	Витрати на заробітну плату, грн
Керівник проекту	11800	513	30	15 390
Науковий співробітник	9 900	430	30	12 900
Аспірант	7 500	326	30	9 780
Всього				38 070

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт розраховують за формулою:

$$З_р = \sum_{i=1} C_i \cdot t_i, \quad (4.3)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за вико-нану відповідну роботу, грн/год;

t_i – час роботи робітника на виконання певної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$С_і = \frac{М_м \cdot К_і \cdot К_с}{Т_р \cdot Т_{зм}}, \quad (4.4)$$

де M_M – розмір прожиткового мінімуму працездатної особи або мінімальної місячної заробітної плати (залежно від діючого законодавства), грн;

K_i – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середня кількість робочих днів в місяці, приблизно $T_p = 21…23$ *дні*;

$t_{зм}$ – тривалість зміни, год.

$$С_п = \frac{6500 \cdot 1{,}8 \cdot 2{,}2}{21 \cdot 8} = 153{,}21 \text{ грн}$$

Таблиця 4.8 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Інженер-програміст	168	7	2,2	153,21	25 740
Програміст	168	6	2	139,28	23 400
Системний аналітик	168	5	1,7	118,39	19 890
Тестувальник	168	4	1,5	104,46	17 550
Всього					86 580

Додаткова заробітна плата розраховується як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = З_о + З_р \cdot \frac{Н_{дод}}{100\%}, \quad (4.5)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати.

$$З_{дод} = (86580 + 38070) \cdot 0{,}1 = 12\,465 \text{ грн.}$$

До статті «Відрахування на соціальні заходи» належать відрахування внеску на загальнообов'язкове державне соціальне страхування та для здійснення заходів щодо соціального захисту населення (ЄСВ – єдиний соціальний внесок).

Нарахування на заробітну плату дослідників та робітників розраховується як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формуло:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%}, \quad (4.6)$$

де $Н_{зп}$ – норма нарахування на заробітну плату.

$$З_{дод} = (86580 + 38070 + 12\,465) \cdot 0{,}22 = 30165{,}3 \text{ грн.}$$

Спецустаткування для наукових (експериментальних) робіт

Вартість спецустаткування визначається за прейскурантом гуртових цін або за даними базових підприємств за відпускними і договірними ці- нами. До балансової вартості устаткування окрім прейскурантної вартості входять витрати на його транспортування і монтаж, тому ці витрати беруться додатково в розмірі 10…12% від вартості устаткування.

Балансову вартість спецустаткування розраховують за формулою:

$$В_{спец} = \sum_{i=1}^{k} Ц_і \cdot С_{пр.1} \cdot К_і \quad (4.7)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

Спр - кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.

$К_і$ - коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($К_і$ =1,10 ….1,12)

k = кількість найменувань устаткування

Таблиця 4.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Комп'ютер	4	25 000	100 000
Принтер	1	2 000	2 000
Всього			102 000

Витрати на придбання спецустаткування становитимуть:

$В_{спец}$= 102 000 ·1,1=112 200 грн.

До статті «Програмне забезпечення для наукових робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% відвартості програмного забезпечення.

Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{к} Ц_{іпрг} \cdot С_{прг} \cdot К_і \qquad (4.8)$$

де $Ц_{іпрг}$ – ціна придбання одиниці програмного засобу цього виду, грн;

$С_{прг}$ - кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного

засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів. Отримані результати зведемо до таблиці 4.10

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Операційна система Windows	4	4 000	16 000
Microsoft Visual Studio 2022 Professional	4	3 000	12 000
Всього			28 000

Витрати на придбання програмних засобів становитимуть :

$$В_{прог}=28000\cdot1,1=30\ 800 \text{ грн.}$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувалися під час (чи для) виконання даного етапу роботи розраховуються за формулою:

$$А = \frac{Ц}{Тв} \cdot \frac{tk}{12}, \quad (4.9)$$

де Ц – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.;

$Т_в$ – термін корисного використання, роки;

t_k – термін використання обладнання, приміщень тощо, місяці.

Розрахунки амортизаційних витрат для даного програмного забезпечення зведені в табл. 4.11.

Таблиця 4.11 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Кількість	Балансова вартість, грн.	Термін корисного використання, роки	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
Комп'ютер	4	25 000	3	2	5 555
Принтер	1	2 000	3	2	111
Операційна система Windows	4	4000	1	2	2666
Microsoft Visual Studio 2022 Professional	4	3000	1	2	2000
	Всього:				10 332

Витрати на силову електроенергію (B_e) розраховують за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (4.10)$$

де B – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії);

П– встановлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

$$В_е = 2{,}28 \cdot 0{,}84 \cdot 7 \cdot 0{,}7 = 9{,}38 \text{ грн.}$$

Проведені розрахунки занесемо до таблиці 4.12.

Таблиця 4.12 – Витрати на електроенергію

Найменування операції	Найменування обладнання	Встановлена потужність, кВт	Тривалість операції, год	Кількість днів	Сума, грн
Програмування	Комп'ютер	0,84	7	30	281,4
Програмування	Комп'ютер	0,84	7	30	281,4
Аналіз даних	Комп'ютер	0,84	7	30	281,4
Тестування	Комп'ютер	0,84	7	30	281,4
Друк, сканування	Принтер	0,70	2	30	67
			Всього		1 192,6

Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{зп}}{100\%}, \quad (4.11)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Інші витрати складатимуть:

$І_в$ = (86580+38070) · 0,5 = 124 650 грн.

Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + В_{спец} + В_{прг} + А_{обл} + В_е + І_в\,, \qquad (4.12)$$

$В_{заг}$= 86580+38070+12465+30165,3+112200+30800+10332+1192,6+321804,9= 446454,9 грн

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\mu}, \qquad (4.13)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи. Так, якщо науково-технічна розробка знаходиться настадії: науково-дослідних робіт, то μ =0,1; технічного проектування, то μ =0,2; розробки конструкторської документації, то μ =0,3; розробки технологій, то μ=0,4; розробки дослідного зразка, то μ =0,5; розробки промислового зразка, то μ=0,7; впровадження, то μ =0,9.

ЗВ= 446454,9/0,9= 496 061 грн.

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

У даному підрозділі кількісно спрогнозуємо, яку вигоду можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою

$$\Delta П_i = \sum_1^n (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\upsilon}{100}), \qquad (4.13)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році;

N - основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;ΔN – покращення

основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість.

Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333;

p – коефіцієнт, який враховує рентабельність продукту, p = 0,2;

υ – ставка податку на прибуток. У 2021 році — 18%.

Припустимо, що при прогнозованій ціні 10 000 грн. за послугу доступу до сервісу, в перший рік термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти його ціну на 1000 грн. Кількість наданих послуг також збільшиться: протягом першого року — на 500 шт., протягом другого року — на 300 шт., протягом третього року на 150 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1$ = (0*500+ (10000 + 1000)*500* 0,8333* 0,2) * (1 - 0,18) = 819 967,2 грн.

$\Delta П_2$ = (0*500+ (10000 + 1000)*(500+300)* 0,8333* 0,2) * (1 - 0,18) = 1 311 947,5 грн.

$\Delta П_3$ = (0*500+ (10000 + 1000)*(500+300+150)* 0,8333* 0,2) * (1 - 0,18) = 1 557 937,7 грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 3 689 852,4 грн.

Далі розраховують приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (4.15)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої МКР, грн;

Т – період часу, протягом якого виявляються результати впровадженої МКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,15;

t – період часу (в роках).

$$ПП = \frac{819\,967,2}{(1+0,15)^1} + \frac{1\,311\,947,5}{(1+0,15)^2} + \frac{1\,557\,937,7}{(1+0,15)^3} = 713014,9 + 992020,8 +$$

$+630106,3 = 2335142$ грн.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково- технічної розробки. Для цього можна використати формулу:

$$PV = k_{\text{інв}} \cdot ЗВ \qquad (4.16)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки. $k_{інв} = 2….5$

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 \cdot 496061 = 992\,122 \text{ грн}$$

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід(NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП – PV, \qquad (4.17)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.

$Е_{абс} = 2\,335\,142 – 992\,122 = 1\,343\,020$ грн.

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів роботи може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$ за формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.18)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн.;

PV – теперішня вартість інвестицій PV = ЗВ, грн.;

$Т_ж$ – життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{1343020}{528487,16}} - 1 = 1,57 - 1 = 0,57$$

Далі слід порівняти розраховану величину $E_в$ з мінімальною (бар'єрною) ставкою дисконтування τ_{min}, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування визначається за формулою:

$$\tau_{min} = d + f, \quad (4.19)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = (0,9...0,12);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f= (0,05...0,5), але може бути і значно більше.

$$\tau_{min} = 0,12 + 0,06 = 0,18.$$

Оскільки $E_в > \tau_{min}$, то інвестор буде зацікавлений у фінансуванні програмного забезпечення для підвищення стійкості криптографічних алгоритмів за рахунок використання в них генератора випадкових чисел на основі ентропії поведінки користувача в багатокористувацькому веб-ресурсі.

Термін окупності вкладених у реалізацію наукового проекту інвестицій $Т_{ок}$ можна розрахувати за формулою:

$$Т_{ок} = \frac{1}{E_в}, \quad (4.20)$$

Для програмного забезпечення підвищення стійкості криптографічних алгоритмів термін окупності складе:

$$T_{ок} = 1/0,57= 1,75 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

4.5 Висновки до розділу

У даному розділі було проаналізовано економічну доцільність розробки програмного забезпечення для підвищення стійкості криптографічних алгоритмів за рахунок використання в них генератора випадкових чисел на основі ентропії поведінки користувача в багатокористувацькому веб-ресурсі. Опитування експертів показало, що запропонована розробка має високий рівень комерційного потенціалу. Вкладання коштів на виконання та впровадження результатів МКР є доцільним. Дана розробка має високий рівень щорічної ефективності та термін окупності складає 1,75р.

ВИСНОВОК

В даній магістерської кваліфікаційної роботі було проведено роботу по підвищенню стійкості криптографічних алгоритмів.

У першому розділі було наведено пряму залежність стійкості криптографічних алгоритмів від якості ГВЧ які в них використовуються.

Проаналізовано основні особливості та проблеми, пов’язані з розробкою генераторів випадкових послідовностей.

Отримані вході аналізу дані дали підґрунтя для пошуку можливості отримувати криптографічно якісних та істино випадкових послідовностей чисел.

У другому розділі для вирішення поставленої задачі було розроблено алгоритм роботи табличного ГВЧ, а також проаналізовано математичний апарат, на якому він базується алгоритм.

Розроблено алгоритм отримання ентропії поведінки користувача для використання її в якості Seed даних для табличного ГВЧ.

Також розроблено повний алгоритм роботи підсистеми генерації випадкових чисел.

Було проведено обґрунтування вибору програмного середовища, що є важливим для продовження подальших досліджень. Наведено опис розробки модулів, що відповідають за реалізацію підсистеми. Проведено аналіз підвищення стійкості криптографічних алгоритмів за рахунок використання в розробленої підсистеми генерації випадкових послідовностей.

Отримані результати свідчать про випадкову природу послідовності отриманої з розробленого ГВЧ. Оскільки, розроблений ГВЧ є генератором істино випадкових послідовностей, також не має класичних недоліків псевдовипадкових ГВЧ, використання його в криптографічних алгоритмах підвищує їх криптографічну стійкість, оскільки розроблений генератор не можливо скомпрометувати. Розроблена підсистема ГВЧ дає змогу отримувати істино випадкові послідовності не накладаючи а систему обмежень притаманні фізичним генераторам випадкових чисел.

ПЕРЕЛІК ВИКОРИСТАНИХ ПОСИЛАНЬ

1) W. Schindler: Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators. Version 2.0 (02.12.1999), mathematical-technical reference of [1] (English translation); www.bsi.bund.de/zertifiz/zert/interpr/ais20e.pdf

ANSI X3.92, “American National Standard for Data Encryption Algorithm (DEA),” American National Standards Institute, 1981.

Алфёров, А.П. Основы криптографии / А.П. Алфёров, А.Ю. Зубов, А.С. Кузьмин, А.В. Черёмушкин. – Изд.-во: Гелиос-АРВ, 2002. – с. 323–326.

Авдошин С.М., Савельева А.А. Криптографические методы защиты информационных систем // Известия АИН им. А.М. Прохорова. Бизнес-информатика. – 2006. – Т. 17. – С. 91–99.

AIS 20: Functionality Classes and Evaluation Methodology for Deterministic Random Number Generators. Version 1 (02.12.1999) (mandatory if a German IT security certificate is applied for; English translation). www.bsi.bund.de/zertifiz/zert/interpr/ais20e.pdf

Чмора, А.Л. Современная прикладная криптография / А.Л. Чмора. – М.: Гелиос АРВ, 2001. – 256 с.

Diffie, W. New Directions in Cryptography/ W. Diffie, M.E. Hellman // IEEE Transactions on Information Theory. – 1976. – v. 22. – №6. – p.644–654

Фергюсон, Н. Практическая криптография / Н. Фергюсон, Б. Шнайер. – М.:Издательский дом «Вильямс», 2005. – с. 235-236

W. Schindler: Efficient Online Tests for True Random Number Generators. In: C¸ .K. Ko¸c, D. Naccache, C. Paar (eds.): Cryptographic Hardware and Embedded Systems — CHES 2001. Springer, Lecture Notes in Computer Science, Vol. 2162, Berlin (2001), 103–117.

NIST: Security Requirements for Cryptographic Modules. FIPS PUB 140-2 (25.05.2001) and Change Notice 1 (10.10.2001). Василенко, О.Н. Теоретико-числовые алгоритмы в криптографии/ О.Н. Василенко. – М.: МЦН-МО, 2003. – с.108-129

Fairfield R.C., Mortenson R.L. and Koulthart K.B. An LSI Random Number Generator, Advances in Cryptology: Proceedings of CRYPTO 84, Springer Verlag.– 1985. – P. 203 – 230

Gardner, M. Mathematical Games: A new kind of cipher that would take millions of years to break/ M. Gardner// Scientific American. – 1977. – v.237–№2. – p. 120-124

Коблиц, Н. Курс теории чисел и криптографии, Н. Коблиц. – М.:науное изд-во ТВП, 2001. – с. 188-217

Авдошин, С.М. Дискретная математика. Модулярная алгебра, криптография, кодирование / С.М. Авдошин, А.А. Небебин. – ДМК-Пресс, 2017. – с.352.

Gabriel C., Wittmann C., Sych D. Dong R., Mauerer W., Andersen U. L., Marquard C. and Leuchs G. A Generator for Unique Quantum Random Numbers Based on Vacuum States // Nature Photonics.– 2010. – № 4. – P. 711 – 715.

Dalkiran, I. Artificial neural network based chaotic generator for cryptology / I. Dalkiran, K. Danis // Turk J Elec Eng and Comp Sci. – 2010. – v.18. – №2. – p. 240-255.

Davis D., Ihaka R., Fenstermacher P. Cryptographic Randomnes from Air Turbulence in Disk Drives. – In: Desmedit Y. G. (ed). Advances in Cryptology – CRYPTO 94. Lecture Notes in Computer Science. Springer-Verlag. – 1994. – Vol. 839. – P. 114–120

Джозеф А. C# 9.0 in a Nutshell: The Definitive Reference / Джозеф А.-2021.

Ковалев А.В. Реализация генераторов случайных чисел. – М.: Научная сессия МИФИ, 2007. – Том 12. – С. 176–177.

M. Bakiri, C. Guyeux, J.-F. Couchot, and A. K. Oudjida, “Survey on hardware implementation of random number generators on FPGA: theory and experimental analyses,” *Computer Science Review*, vol. 27, pp. 135–153, 2018.

Mikhail J. Atallah and Keith B. Frikken. Securely outsourcing linear algebra computations. In Proceedings of the ACM Symposium on Information, Computer and Communications Security (ASIACCS), pages 48–59, 2010.

Mikhail J. Atallah, Konstantinos N. Pantazopoulos, John R. Rice, and Eugene H. Spafford. Secure outsourcing of scientific computations. Advances in Computers, 54:215–272, 2001.

F. Özkaynak, "Cryptographically secure random number generator with chaotic additional input," *Nonlinear Dynamics*, vol. 78, no. 3, pp. 2015–2020, 2014.

Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In Proceedings of the Public Key Cryptography Conference (PKC), pages 85–99, 2003.

David Benjamin and Mikhail J. Atallah. Private and cheating-free outsourcing of algebraic computations. In Proceedings of the Annual Conference on Privacy, Security and Trust (PST), pages 240–245, 2008.

Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. Electronic Colloquium on Computational Complexity (ECCC), 18:109, 2011.

T. Tuncer, "Implementation of duplicate TRNG on FPGA by using two different randomness source," *Elektronika ir Elektrotechnika*, vol. 21, no. 4, pp. 35–39, 2015.

Xiaofeng Chen, Xinyi Huang, Jin Li, Jianfeng Ma, Wenjing Lou, and Duncan S. Wong. New algorithms for secure outsourcing of large-scale systems of linear equations. IEEE Transactions on Information Forensics and Security, 10(1):69–78, 2015.

M. Dichtl, Bad and Good Ways of Post-processing Biased Physical Random Numbers, vol. 4593 of Lecture Notes in Computer Science, Springer, Berlin, Germany, 2007.

E. Avaroglu, T. Tuncer, A. B. Özer, and M. Türk, "A new method for hybrid pseudo random number generator," *Journal of Microelectronics, Electronic Components and Materials*, vol. 4, no. 4, pp. 303–311, 2014.

W. Xingyuan, Q. Xue, and T. Lin, “A novel true random number generator based on mouse movement and a one-dimensional chaotic map,” *Mathematical Problems in Engineering*, vol. 2012, Article ID 931802, 9 pages, 2012.

Ivan Damg˚ard and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In Proceedings of the Advances in Cryptology (CRYPTO), pages 378–394, 2005.

Wenliang Du and Mikhail J. Atallah. Privacy-preserving cooperative scientific computations. In Proceedings of the IEEE Computer Security Foundations Workshop (CSFW), pages 273–294, 2001.

Wenliang Du, Yunghsiang S. Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In Proceedings of the International Conference on Data Mining (SDM), pages 222–233, 2004.

Dario Fiore and Rosario Gennaro. Publicly verifiable delegation of large polynomials and matrix computations, with applications. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), pages 501–512, 2012.

Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE Transactions on Information Theory, 31(4):469–472, 1985.

Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Proceedings of the Advances in Cryptology (CRYPTO), pages 465–482, 2010.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In Proceedings of the Annual ACM Symposium on Theory of Computing (STOC), pages 169–178, 2009.

Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In Proceedings of the Advances in Cryptology (CRYPTO), pages 116–137, 2010.

Susan Hohenberger and Anna Lysyanskaya. How to securely outsource cryptographic computations. In Proceedings of the Theory of Cryptography Conference (TCC), pages 264–282, 2005.

Seny Kamara, Payman Mohassel, and Ben Riva. Salus: a system for server-aided secure function evaluation. In Proceedings of the ACM Conference on Computer and Communications Security (CCS), pages 797–808, 2012.

Alan F. Karr, Xiaodong Lin, Ashish P. Sanil, and Jerome P. Reiter. Privacy-preserving analysis of vertically partitioned data using secure matrix products. Journal of Official Statistics, 25(1):125–138, 2009.

Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In Proceedings of the Advances in Cryptology (CRYPTO), pages 335–354, 2004.

Eike Kiltz, Payman Mohassel, Enav Weinreb, and Matthew K. Franklin. Secure linear algebra using linearly recurrent sequences. In Proceedings of the Theory of Cryptography Conference (TCC), pages 291–310, 2007.

Kaoru Kurosawa. Multi-recipient public-key encryption with shortened ciphertext. In Proceedings of the Public Key Cryptography Conference (PKC), pages 48–63, 2002.

T. Tuncer, "Implementation of duplicate TRNG on FPGA by using two different randomness source," *Elektronika ir Elektrotechnika*, vol. 21, no. 4, pp. 35–39, 2015.

Xinyu Lei, Xiaofeng Liao, Tingwen Huang, Huaqing Li, and Chunqiang Hu. Outsourcing large matrix inversion computation to a public cloud. IEEE Transactions on Cloud Computing, 1(1):78–87, 2013.

Chae Hoon Lim and Pil Joong Lee. Security and performance of server-aided RSA computation protocols. In Proceedings of the Advances in Cryptology (CRYPTO), pages 70–83, 1995.

Y. Hu, X. Liao, K.-W. Wong, and Q. Zhou, "A true random number generator based on mouse movement and chaotic cryptography," *Chaos, Solitons & Fractals*, vol. 40, no. 5, pp. 2286–2293, 2009.

ДОДАТКИ

Додаток А. Технічне завдання

Вінницький національний технічний університет

Факультет менеджменту та інформаційної безпеки

Кафедра менеджменту та безпеки інформаційних систем

ЗАТВЕРДЖУЮ

Голова секції “Управління інформаційною
безпекою” кафедри МБІС
д.т.н., професор
_____Ю.Є.Яремчук
“24” вересня 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

до магістерської кваліфікаційної роботи на тему:
«Підвищення захищеності веб-ресурсів стійкими криптографічними алгоритмами на основі генераторів випадкових чисел, що враховують ентропію поведінки користувача у багатокористувацькому середовищі»
08-42.МКР.007.00.000.ТЗ

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент, завідувач кафедри МБІС

Карпінець В.В

Вінниця – 2021 р.

1. Найменування та область застосування

Програмний модуль для генерації випадкових чисел, реалізуючий табличний генератор випадкових чисел на основі ентропії поведінки користувачів у багатокористувацькому веб-ресурсі.

2.Підстава для розробки

Розробка виконується на основі наказу ректора ВНТУ №277 від 24.09.2021 р.

3. Мета та призначення розробки

3.1 Мета розробки: програмно реалізувати табличний генератор випадкових чисел на основі ентропії поведінки користувачів у багатокористувацькому веб-ресурсі.

3.2 Призначення: розроблений програмний модуль здійснює генерацію випадкових чисел для використання їх в криптографічних алгоритмах.

4. Джерела розробки

4.1. W. Schindler: Efficient Online Tests for True Random Number Generators. In: C¸ .K. Ko¸c, D. Naccache, C. Paar (eds.): Cryptographic Hardware and Embedded Systems — CHES 2001. Springer, Lecture Notes in Computer Science, Vol. 2162, Berlin (2001).

4.2. T. Tuncer, “Implementation of duplicate TRNG on FPGA by using two different randomness source,” Elektronika ir Elektrotechnika, vol. 21, no. 4, pp. 35–39, 2015.

4.3. W. Xingyuan, Q. Xue, and T. Lin, “A novel true random number generator based on mouse movement and a one-dimensional chaotic map,” *Mathematical Problems in Engineering*, vol. 2012, Article ID 931802, 9 pages, 2012.

4.4. F. Özkaynak, “Cryptographically secure random number generator with chaotic additional input,” Nonlinear Dynamics, vol. 78, no. 3, pp. 2015–2020, 2014.

4.5 Mihir Bellare, Alexandra Boldyreva, and Jessica Staddon. Randomness re-use in multi-recipient encryption schemeas. In Proceedings of the Public Key Cryptography Conference (PKC), pages 85–99, 2003.

5. Вимоги до програми

5.1 Всі програмні модулі мають виконувати одну функцію і у разі збоїв не впливати на роботу інших модулів.

5.2 Вимоги до надійності:

5.2.1 Програмний засіб повинен працювати без помилок, у випадку виникнення критичних ситуацій необхідно передбачити виведення відповідних повідомлень;

5.2.2 Програмний засіб повинен виконувати свої функції.

5.3 Вимоги до складу і параметрів технічних засобів:

– оперативна пам’ять – не менше 512 Mb;

– середовище функціонування – операційна система сімейство Windows;

– вимоги до техніки безпеки при роботі з програмою повинні відповідати існуючим вимогам та стандартам з техніки безпеки при користуванні комп’ютерною технікою.

5.4 Програма має мати читабельний вигляд, містити коментарі.

5.5 Програма має працювати без помилок, а у разі їх виникнення інформувати користувача.

6. Вимоги до програмної документації

6.1 Код програми повинен відповідати єдиному стилю.

7. Вимоги до технічного захисту інформації

7.1 Необхідно забезпечити захист розроблюваного програмного засобу від несанкціонованого використання.

7.2 Необхідно забезпечити збереження отриманих результатів.

8. Техніко-економічні показники

8.1 Цінність результатів використання даного проекту повинна перевищувати витрати на його реалізацію.

8.2 Має бути реалізований таким чином, щоб підходити для використання широкого загалу.

9. Стадії та етапи розробки

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Початок	Закінчення
1	Аналіз предметної області обраної теми	27.09.2021	10.10.2021
2	Апробація отриманих результатів	11.10.2021	15.10.2021
3	Розробка алгоритму роботи	16.10.2021	25.10.2021
4	Написання магістерської роботи на основі розробленої теми	26.10.2021	26.11.2021
5	Розробка економічної частини	27.11.2021	16.12.2021
6	Передзахист магістерської кваліфікаційної роботи	25.11.2021	26.11.2021
7	Виправлення, уточнення, корегування магістерської кваліфікаційної роботи	27.11.2021	16.12.2021
8	Захист магістерської кваліфікаційної роботи	20.12.2021	20.12.2021

10. Порядок контролю та прийому

10.1 До приймання магістерської кваліфікаційної роботи надається:

– ПЗ до магістерської кваліфікаційної роботи;

– програмний додаток;

– презентація;

– відзив керівника роботи;

– відзив опонента

Технічне завдання до виконання прийняв _____ Пархоменко Р.М.

Додаток Б. Лістинг програми

```
public class TabularRandomNumberGeneration : ITabularRandomNumberGeneration
 {
    private int[] Buffer { get; set; }
    public TabularRandomNumberGeneration()
    {
       Buffer = new int[100000];
       _seedIndex = 0;

    }
    private int? _seed { get; set; }
    private int _seedIndex { get; set; }
    private int _nextIndex { get; set; }

    public int next(int from, int to)
    {
       if (to == 0 || to < from)
       {
          throw new ArgumentException();
       }
       int digitLength = (int)Math.Floor(Math.Log10(to) + 1);
       var randomNumber = GetRandom(digitLength);
       if (randomNumber > from && randomNumber < to)
       {
          return randomNumber;
       }
       else
       {
          return next(from, to);
       }

       return 0;

    }
    public async Task Seed(int seed)
    {
       if (_seed == null)
       {
          _seed = seed;
       }
       else
       {
          _seed = seed;
          var digits=GetDigits((_seed ^ seed).Value);
```

```
            for (int i = 0; i < digits.Length; i++)
            {
                Buffer[_seedIndex + i] = digits[i];
            }


            _seedIndex = _seedIndex + digits.Length;
        }

    }

    private int[] GetDigits(int val)
    {
        var length = (int)Math.Floor(Math.Log10(val) + 1);
        int[] stack = new int[length];



        int number = val;
        for (int i = 0; i < length; i++)
        {

            var digit = number % 10;
            stack[i] = digit;

            number /= 10;
        }

        return stack;
    }

    private int GetRandom(int length)
    {
        int[] result = new int[length];
        Array.Copy(Buffer, _nextIndex, result, 0, length);
        _nextIndex = _nextIndex + length + 1;
        return int.Parse(string.Join("", result)); ;
    }

}
```

```
public class RNGMiddleware
  {
    readonly RequestDelegate _next;


    public RNGMiddleware(RequestDelegate next, ITabularRandomNumberGeneration
tabularRandomNumberGeneration)
    {
      _next = next;
      _tabularRandomNumberGeneration = tabularRandomNumberGeneration;
    }

    public ITabularRandomNumberGeneration _tabularRandomNumberGeneration { get;
}

    public async Task Invoke(HttpContext context)
    {
      var requestReader = new StreamReader(context.Request.Body);
      var json = await requestReader.ReadToEndAsync();

      var Body = Newtonsoft.Json.JsonConvert.DeserializeObject<TraceModel>(json);
      await _tabularRandomNumberGeneration.Seed(Body.Seed);


      context.Response.StatusCode = 200;
      await context.Response.StartAsync();


    }
  }


  class TraceModel
  {
    public int Seed { get; set; }
  }

public interface ITabularRandomNumberGeneration
  {
    Task Seed(int seed);
  }
```

```
export class AppComponent {


  private _x: number;
  public get x(): number {
    return this._x;
  }
  public set x(v: number) {
    this._x = v;
  }

  private _y: number;
  public get y(): number {
    return this._y;
  }
  public set y(v: number) {
    this._y = v;
  }
  subscription: Subscription;

  constructor(private http: HttpClient, private AuthorizeService: AuthorizeService) {
    var xx;
    var yy;
    this.subscription =
      fromEvent(document, 'mousemove')
        .subscribe((e: any) => {
          console.log(e)
          xx = e.clientX; yy = e.clientY;
        });

    //  window.onmousemove = function (e) { this.x = e.clientX; this.y = e.clientY; }
    setInterval(() => {
      if (AuthorizeService.isAuthenticated) {
        const entropy = xx
        //  console.log( this.x )

        http.post("/rng", { Seed: entropy }).subscribe()

      }
    }, 500)


  }
  title = 'app';
}
```

Додаток В. Ілюстративний матеріал

Підвищення захищеності веб-ресурсів стійкими криптографічними алгоритмами на основі генераторів випадкових чисел, що враховують ентропію поведінки користувача у багатокористувацькому середовищі

ВИКОНАВ: ПАРХОМЕНКО Р.М.

КЕРІВНИК: КАРПІНЕЦЬ В.В.

Актуальність теми

Оскільки зловмисні криптоаналітики постій шукають вразливості систем захисту, а також враховуючи той факт, що комп'ютеризація сучасного суспільства досягла рівня коли майже вся інформація обробляється, зберігається та передається за допомогою комп'ютерних технологій, удосконалення алгоритмів захисту є вкрай важливою метою. Якщо розглядати криптографічну стійкість алгоритм як стійкість всіх модулів які в них використовуються то вході їх удосконалення необхідно звертати увагу на всі, навіть найменш значущі на перший погляд, складові цих алгоритмів. Такою складовою можуть виступати генератори випадкових послідовностей. Не завжди є можливість використовувати апаратні генератори які можуть задовольняти потреби під час їх використання в криптографії, оскільки вони потребують розроби на впровадження апаратних модулів у систему, що не завжди є фінансово та практично доцільним.
Цей факт змушує нас шукати нові алгоритми для генерації випадкових чисел для захисту та підвищувати стійкість криптографічних алгоритмів.

Постановка задачі

- Мета роботи – Метою роботи підвищення стійкості криптографічних алгоритмів в багатокористувацьких веб-ресурсах.
- Об'єкт дослідження – підвищення криптографічної стійкості криптографічних алгоритмів.
- Предмет дослідження – залежність стійкості криптографічних алгоритмів від криптографічної якості генераторів випадкових чисел які в них використовуються.

3

Використання ГВЧ в криптографічних алгоритмах

- Генератори сесій (PHPSESSID);
- Генерація тексту для капчі;
- Шифрування;
- Генерація солі;
- Генератор тимчасових паролів;
- Генерація ключів;
- Та інше.

4

Різновиди ГВЧ

ГПВЧ	ГІВП	
Програмні	Апаратні	Табличні

5

Порівняльний аналіз існуючих ГВЧ

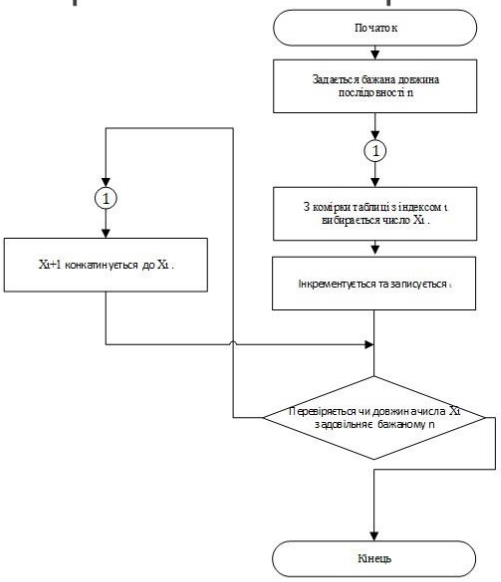
Характеристика	Апаратні ГВЧ	Програмні ГВЧ
Відсутність періоду	Так	Ні
Непередбачуваність	Так	Умовна
Незалежність значень	Так	Умовна
Рівень крипостійкості	Високий	Умовний
Швидкість генерації	Низька	Висока
Відтворюваність	Ні	Так
Простота генерації	Ні	Так
Вартість генерації	Висока	Низька
Необхідність апаратної реалізації	Так	Ні

6

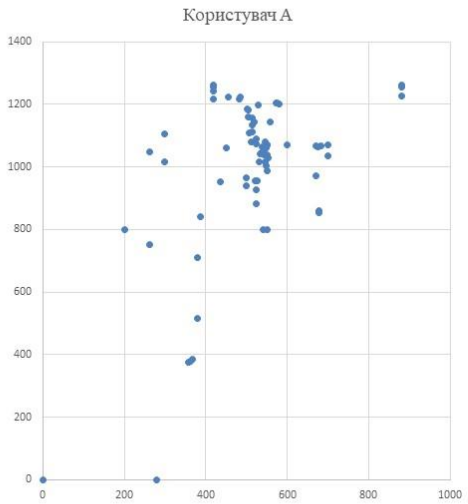
Табличний

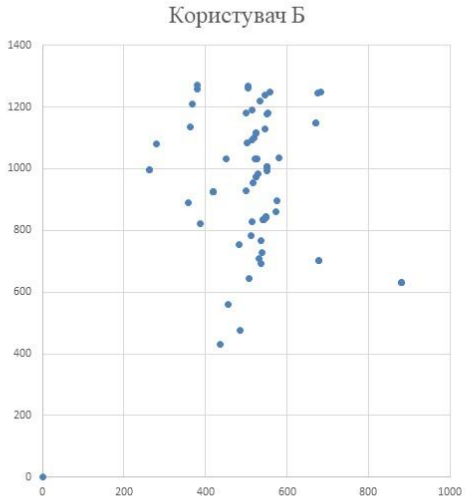
Випадкові числа								Рівномірно розподілені від 0 до 1 випадкові числа
9	2	9	2	0	4	2	6	0.929
9	5	7	3	4	9	0	3	0.204
5	9	1	6	6	5	7	6	0.269
...								...

Блок-схема роботи алгоритму табличниго ГВЧ



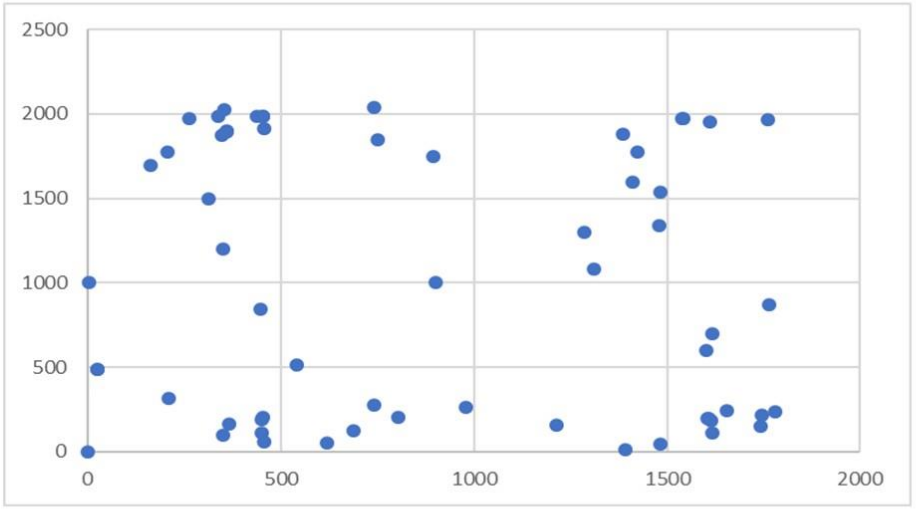
Ентропія поведінки користувача





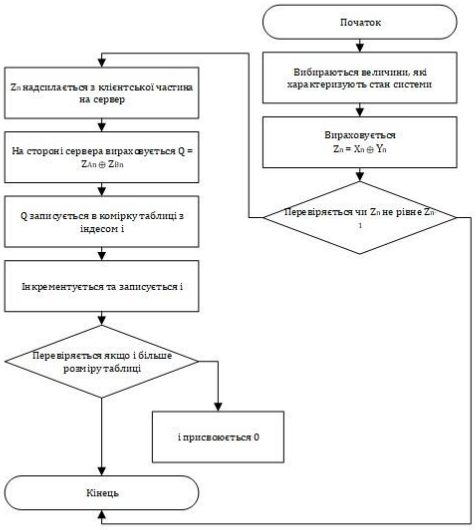
9

Результати постобробки XOR



10

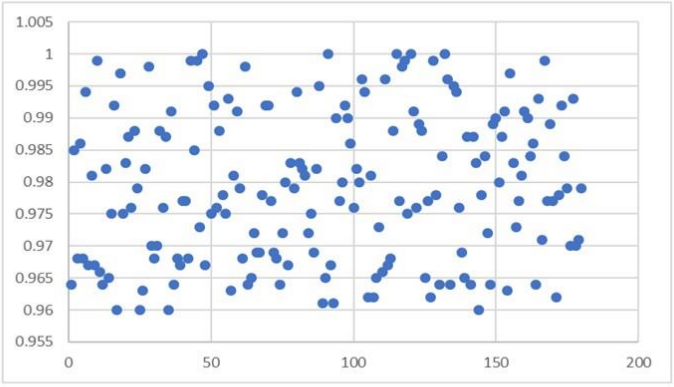
Блок-схема роботи алгоритму табличниго ГВЧ



Вибір технологій

- Мова програмування – C#
  реалізація серверної частини додатку
- Платформа – AngularJS
  реалізація клієнтської частини додатку

Результати тестування послідовності згенерованої розробленим ГВЧ



13

Порівняльна характеристика типів ГВЧ

Характеристика	Апаратні ГВЧ	Програмні ГВЧ	Розроблений ГВЧ
Відсутність періоду	Так	Ні	Так
Непередбачуваність	Так	Умовна	Так
Незалежність значень	Так	Умовна	Так
Рівень крипостійкості	Високий	Умовний	Високий
Швидкість генерації	Низька	Висока	Висока
Відтворюваність	Ні	Так	Так
Простота генерації	Ні	Так	Так
Вартість генерації	Висока	Низька	Низька
Необхідність апаратної реалізації	Так	Ні	Ні

14

Висновки

Отримані результати свідчать про випадкову природу послідовності отриманої з розробленого ГВЧ. Оскільки, розроблений ГВЧ є генератором істино випадкових послідовностей, також не має класичних недоліків псевдовипадкових ГВЧ, використання його в криптографічних алгоритмах підвищує їх криптографічну стійкість, оскільки розроблений генератор не можливо скомпрометувати. Розроблена підсистема ГВЧ дає змогу отримувати істино випадкові послідовності не накладаючи а систему обмежень притаманні фізичним генераторам випадкових чисел.

Дякую за увагу!

Додаток Г. ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: Підвищення захищеності веб-ресурсів стійкими криптографічними алгоритмами на основі генераторів випадкових чисел, що враховують ентропію поведінки користувача у багатокористувацькому середовищі

Тип роботи: Магістерська кваліфікаційна робота

Підрозділ: Факультет МІБ, кафедра менеджменту та безпеки інформаційних систем,

гр. УБ-20м

Науковий керівник Карпінець В.В., зав.каф. МБІС, доцент, к.т.н.

Показники звіту подібності

Plagiat.pl (StrikePlagiarism)		Unicheck	
КП1			
КП2		Оригінальність	88 %
Тривога/Білі знаки	/	Схожість	12 %

Аналіз звіту подібності (відмінити подібне)

- ☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
- ☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.
- ☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений(-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Пархоменко Р.М.

(підпис) (прізвище, ініціали)

Опис прийнятого рішення

Ступінь оригінальності роботи відповідає вимогам, що висуваються до МКР

Особа, відповідальна за перевірку_____ Коваль Н.П.

(підпис) (прізвище, ініціали)

Експерт _____ _____

(за потреби) (підпис) (прізвище, ініціали)