МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кваліфікаційна наукова
праця на правах рукопису

ГРИЦАК АНАТОЛІЙ ВАСИЛЬОВИЧ

УДК 004.056.5:004.7

ДИСЕРТАЦІЯ

МЕТОДИ ПОБУДОВИ ЕФЕКТИВНИХ КРИПТОГРАФІЧНИХ ФУНКЦІЙ ГЕШУВАННЯ

05.13.21 – «Системи захисту інформації»

Подається на здобуття наукового ступеня доктора технічних наук

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело _____ А.В. Грицак

Науковий керівник:
Яремчук Юрій Євгенович, доктор технічних наук, професор, професор кафедри менеджменту та безпеки інформаційних систем, Вінницький національний технічний університет.

Київ – 2020

АНОТАЦІЯ

Грицак А.В. Методи побудови ефективних криптографічних функцій гешування. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.21 – Системи захисту інформації. – Національний авіаційний університет, Київ, 2020.

Дисертаційна робота присвячена розв'язанню актуальної наукової задачі розробки та дослідження нових ефективних геш-функцій, які при достатньо високій швидкодії забезпечуватимуть необхідний рівень стійкості.

Проведено аналіз сучасних методів і алгоритмів побудови та реалізації ефективних криптографічних функцій гешування, що дозволило виявити їх недоліки і формалізувати завдання наукового дослідження. Серед виявлених недоліків основними є уразливість відомих геш-функцій до криптоаналітичних атак, низька швидкість шифрування і високі вимоги до обчислювальних засобів.

На сьогодні існує багато криптографічних алгоритмів, серед яких зустрічаються достатньо вдалі та широко використовувані, що розроблені не тільки спецслужбами, а й приватними особами. Сучасна криптографія застосовується для розв'язання таких задач:

1) забезпечення конфіденційності даних;

2) перевірка справжності відправника (аутенти-фікація);

3) не заперечення авторства;

4) забезпечення цілісності даних.

Остання задача полягає у тому, що отримувач може перевірити несанкціоновану модифікацію в тексті, а зловмисник не може видати змінений текст за справжній. Одним із найбільш ефективних способів розв'язання зазначеної задачі є використання методів гешування, тобто перетворення вхідних даних довільної довжини у вихідні дані (бітовий рядок) фіксованої довжини (процес перетворення називається геш-функцією,

а вихідні дані геш-кодом, або дайджестом). Криптографічні функції гешування дають змогу перевірити відповідність вхідних даних дайджесту, проте не дозволяють відновити вхідні дані за наявним дайджестом – саме ця властивість дозволяє забезпечити цілісність даних. Крім зазначеної, ефективна функція гешування має забезпечувати так властивості, як висока швидкість обчислення та стійкість до колізій першого і другого роду. Серед сучасних геш-функцій варто відзначити Keccak, SHA-2, BCA, MD-5, Haval, N-hash, RIPE-MD, Kupyna та інші.

Розроблено метод побудови функцій гешування, який за рахунок доповнення вхідного повідомлення розміром цього повідомлення та ПВП salt (розраховується на основі вхідного повідомлення), використання у функції стиснення нової послідовності операцій (на основі 6-ти нелінійних функцій, операцій підстановки, додавання за модулем 2 і 2^n, циклічних і лінійних зсувів), дозволив будувати криптостійкі функції гешування. На основі цього методу було розроблено, реалізовано програмно і дослі-джено три нові стійкі геш-функції, які дозволяють підвищити швидкість у 1.15-1.36 разів та можуть застосовуватись у системах, для яких критичним є параметр стійкості.

Розроблено метод побудови функцій гешування, який за рахунок доповнення вхідного повідомлення ПВП salt (розраховується на основі вхідного повідомлення та його розміру), використання у функції стиснення додаткового вектору внутрішнього стану та нової послідовності операцій (на основі 4-ьох не лінійних функцій, операцій підстановки, перестановки, додавання за модулем 2 і 2^n та циклічного зсуву), дозволив будувати швидкісні функції гешування. На основі цього методу було розроблено, реалізовано програмно і досліджено три нові швидкісні геш-функції, які дозволяють підвищити швидкість у 1.16-1.53 разів і можуть застосовуватись у системах, для яких критичним є параметр швидкості.

Удосконалено метод побудови генераторів ПВП, який за рахунок обробки вектора внутрішнього стану та ключового вектору операціями підстановки, циклічного зсуву, складання за модулем 2 і 2^n та 4-ма нелінійними функціями, дозволив будувати ефективні генератори ПВП. На основі цього методу розроблено і реалізовано програмно три генератори ПВП, які будуть корисними як для функцій гешування, так і для інших криптографічних застосувань (генерування ключів, потокові шифри тощо). Крім того, розроблені генератори ПВП Viriy є більш швидкими за аналоги (зокрема, у 1.02-1.22 разів в порівнянні з алгоритмом Snow).

Удосконалено метод криптографічного захисту інформації, що за рахунок фіксування інформації про ідентифікатор користувача, ідентифікатор сесії, час відправлення, довжину повідомлення та його порядковий номер, а також використання нової процедури формування сеансового ключа для шифрування, дає можливість забезпечити конфіденційність і цілісність даних в ІКС.

Розроблено спеціалізоване програмне забезпечення у вигляді консольних додатків на мові програмування C++ (середовище розробки Microsoft Visual Studio 2013 (Release Version)) та методику, що дозволило провести експерименти і верифікувати запропоновані методи. Результати дисертаційної роботи використовуються у навчальному процесі Вінницького національного технічного університету, науковому процесі Національного авіаційного університету та ННВК “Інформаційно-комунікаційні системи”, що підтверджено відповідними актами впровадження.

Ключові слова: захист інформації, гешування, цілісність, конфіденційність, блокчейн, геш-функція, генератор ПВП.

ABSTRACT

Hrytsak A. Methods for effective cryptographic hash functions construction. –Qualifying scientific work as a manuscript.

Thesis for a Candidate of Technical Science degree in specialty 05.13.21 – Information security systems. – National Aviation University, Kyiv, 2020.

The dissertation is devoted to solving the actual scientific problem of developing and researching new effective hash functions that will provide the necessary level of the security with a sufficiently high speed.

The analysis of modern methods and algorithms for the construction and implementation of effective cryptographic hashing functions was carried out, which made it possible to identify their shortcomings and formalize the tasks of scientific research. Among the shortcomings identified are the vulnerabilities of known hash functions to cryptoanalytic attacks, low encryption speed and high requirements for computing facilities. It has allowed formalizing problem and defining task of research study.

Today, there are many cryptographic algorithms, among which there are quite successful and widely used, developed not only by intelligence services but also by individuals. Modern cryptography is used to solve the following problems:

1) ensuring the confidentiality of data;

2) authentication of the sender (authentication);

3) no denial of authorship;

4) ensuring data integrity.

The last task is that the recipient can check the unauthorized modification in the text, and the attacker can not pass the modified text as genuine. One of the most effective ways to solve this problem is to use hashing methods, ie converting input data of arbitrary length into output data (bit string) of fixed length (the conversion process is called a hash function, and the output data is called a hash code or digest). Cryptographic hashing functions make it possible to check the

correspondence of the input data of the digest, but do not allow to restore the input data on the existing digest - this property allows to ensure the integrity of the data.

In addition to this, the effective hashing function must provide properties such as high computational speed and resistance to collisions of the first and second kind. Among the modern hash functions are Keccak, SHA-2, BCA, MD-5, Haval, N-hash, RIPE-MD, Kupyna and others.

A method of constructing hashing functions was developed, which made it possible to increase the speed of cryptographic data processing. Based on this method, three new stable hash functions have been developed, programmatically and investigated, which can increase the speed in 1.15-1.36 times and provide resistance to cryptanalytic attacks and can be used in systems for which the stability parameter is critical (this method was titled “secure method”).

A method of constructing hashing functions was developed, which made it possible to provide resistance to cryptanalytic attacks. Based on this method, three new high-speed hash functions have been developed, programmatically and investigated, which increase the speed in 1.16-1.53 times and can be applied to systems for which the speed parameter is critical (in the work this method was titled “high-speed method”).

The method of pseudorandom number generators construction has been improved, which allowed forming a statistically stable range for cryptographic applications. Based on this method, three pseudorandom number generators have been developed and implemented software that will be useful for both hashing functions and other cryptographic applications (key generation, stream ciphers, etc.). In addition, the developed Viriy pseudorandom number generators are faster than their counterparts;

The method of cryptographic protection of information has been improved, which by means of fixing information on user ID, session ID, time of sending, length of message and its serial number, as well as use of the new procedure of formation of session key and encryption, made it possible to ensure confidentiality

and integrity of data in the modern information and communication systems and technologies.

Last chapter of the dissertation contains research study devoted to collision characteristics of proposed hash functions using so-called “baby versions” of hashing functions based on the existed experimental technique (relevant in cryptography).

Specialized software was developed in the form of console applications in C ++ programming language (Microsoft Visual Studio 2013 (Release Version)) and a technique that allowed us to conduct experiments and verify the proposed methods. the results of the dissertation are used in the educational process of Vinnytsa National Technical University (to increase the efficiency of training of specialists in the specialty 125 “Cybersecurity”) as well as in scientific process of National Aviation University and Educational & Research Complex “Information and Communication Systems”. It was confirmed by the acts of implementation.

Keywords: information security, hashing, integrity, privacy, blockchain, hash function, pseudo random number generator.

Список основних публікацій здобувача

1. S. Gnatyuk, A. Hrytsak, V. Kinzeryavyy, N. Seilova et al, “Modern Method and Software Tool for Guaranteed Data Deletion an Advanced Big Data Systems”, Advances in Intelligent Systems and Computing, Vol. 902, pp. 581-590, 2019, ISSN 2194-5357 (Scopus).

2. А. Грицак, І. Березовий, І. Гринь, В. Кінзерявий, “Програмна система захисту засобів зберігання криптовалют”, Вісник Інженерної академії України, №1, с. 128-139, 2018.

3. Н. Остапенко, В. Кінзерявий, А. Грицак, К. Кириченко, “Удосконалена функція гешування MD4”, Безпека інформації, Том 24, №2, 2018.

4. Ю. Яремчук, А. Грицак, Д. Присяжний, “Сучасні підходи до побудови і реалізації ефективних криптографічних функцій гешування”, Вісник Інженерної академії України, №4, с. 221-228, 2017.

5. A. Hrytsak, V. Kinzeryavyy, D. Prysiazhnyi, Yu. Burmak and Ye. Samoylik, “High-Speed and Secure Hash Function for Blockchain Security Mechanisms”, Scientific and Practical Cyber Security Journal (SPCSJ), Vol. 4, Issue 1, pp. 65-70, 2020.

6. А. Грицак, В. Катаєв, В. Леонтьєв, Н. Ляховченко, “Проблеми активного захисту інформації від витоку через віброакустичні канали”, Реєстрація, зберігання і обробка даних, Том 18, №3, с.54-59, 2016.

7. А.В. Грицак, “Застосування алгоритмів гешування в технології блокчейн”, тези доповідей XVII міжнар. наук.-практ. конф. молодих учених і студентів “Політ. Сучасні проблеми науки”, 5-7 квітня 2017 р., К., с. 77, 2017.

8. К.С. Кириченко, В.М. Кінзерявий, А.В. Грицак, М.Б. Александер, “Перспективна криптографічна функція гешування”, Матеріали міжнар. наук.-практ. конф. “Актуальні питання забезпечення кібернетичної безпеки та захисту інформації”, 21-24 лютого 2018 р., Верхній Студений, с. 68-69, 2018.

9. А.В. Грицак, “Дослідження методу побудови функції гешування на основі алгоритму MD4”, тези доповідей XVI міжнар. наук.-практ. конф. молодих учених і студентів “Політ. Сучасні проблеми науки”, 04-05 квітня 2016 р., К., с. 104, 2016.

10. А.В. Грицак, “Экспериментальное исследование криптостойкости разработанных функций хеширования”, материалы XXIII международной научно-технической конференции “Современные средства связи”, 17-18 октября 2019 р., Минск, с. 56-59, 2019.

11. А.В. Грицак, “Дослідження удосконаленого метода забезпечення конфіденційності та цілісності даних в інформаційно-телекомунікаційних системах”, тези доповідей Міжнар. наук.-практ. конф. молодих учених і студентів “Політ. Сучасні проблеми науки”, 01-03 квітня 2020 р., К., с. 159, 2020.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ГПВП – генератор псевдовипадкових послідовностей;

ЕЦП – електронний цифровий підпис;

ІБ – інформаційна безпека;

ІКС – інформаційно-комунікаційна система;

ІКТ – інформаційно-комунікаційні технології;

КБ – кібербезпека;

КЦД – конфіденційність, цілісність, доступність;

МППК – міжнародна науково-практична конференція;

МНТК – міжнародна науково-технічна конференція;

НДР – науково-дослідна робота;

НСД – несанкціонований доступ;

ПВП – псевдовипадкова послідовність;

ПЗ – програмне забезпечення;

ISO – Міжнародна організація зі стандартизації;

ITIL – бібліотека інфраструктури інформаційних технологій;

MD – система Меркла-Демґарда;

NIST – Національний інститут стандартизації і технологій.

ВСТУП

Актуальність. Серед спектру методів захисту інформації особливе місце займають криптографічні методи. На відміну від інших, ці методи спираються лише на властивості самої інформації і не використовують властивості її матеріальних носіїв, особливості вузлів її оброблення, передавання і зберігання. Широке використання і постійне збільшення об'єму інформаційних потоків викликає постійне зростання інтересу до криптографії. Останнім часом збільшується роль програмних криптографічних засобів захисту інформації, які не потребують великих фінансових витрат порівняно з апаратними криптосистемами. Сучасні методи шифрування гарантують надійний захист, але завжди є імовірність знаходження нових методів криптоаналізу, які дозволять послабити стійкість криптоалгоритмів.

На сьогодні існує багато криптографічних алгоритмів, серед яких зустрічаються достатньо вдалі та широко використовувані, що розроблені не тільки спецслужбами, а й приватними особами. Сучасна криптографія застосовується для розв’язання таких задач: 1) забезпечення конфіденційності даних; 2) перевірка справжності відправника (аутенти-фікація); 3) не заперечення авторства; 4) забезпечення цілісності даних. Остання задача полягає у тому, що отримувач може перевірити несанкціоновану модифікацію в тексті, а зловмисник не може видати змінений текст за справжній. Одним із найбільш ефективних способів розв’язання зазначеної задачі є використання методів гешування, тобто перетворення вхідних даних довільної довжини у вихідні дані (бітовий рядок) фіксованої довжини (процес перетворення називається геш-функцією, а вихідні дані геш-кодом, або дайджестом). Криптографічні функції гешування дають змогу перевірити відповідність вхідних даних дайджесту, проте не дозволяють відновити вхідні дані за наявним дайджестом – саме ця властивість дозволяє забезпечити цілісність даних. Крім зазначеної, ефективна функція гешування має забезпечувати так властивості, як висока швидкість обчислення та стійкість до колізій першого і другого роду. Серед сучасних геш-

функцій варто відзначити Keccak, SHA-2, BCA, MD-5, Haval, N-hash, RIPE-MD, Kupyna та інші.

Значний внесок у розвиток теорії й практики побудови ефективних функцій гешування внесли такі вітчизняні та закордонні вчені: Д. Бернштейн, Г. Бертоні, А. Бірюков, І. Горбенко, Й. Даймен, Ю. Женг, О. Король, Т. Ланге, А. Олексійчук, Р. Олійников, Й. Пєпжик, Б. Преніл, Р. Райвест, Дж. Себері, Б. Шнайєр та ін.

Переважна більшість сучасних наукових досліджень, пов'язаних із розробкою криптографічних функцій гешування, є орієнтованими або на забезпечення високого рівня криптостійкості (такі геш-функції потребують підвищення швидкодії), або ж на забезпечення високої швидкодії (такі геш-функції потребують підвищення стійкості до криптоаналітичних атак). З огляду на це, та незважаючи на велику номенклатуру існуючих методів і рішень, розробка та дослідження нових ефективних геш-функцій, які при достатньо високій швидкодії забезпечуватимуть необхідний рівень стійкості є *актуальною науково-технічною задачею,* що має теоретичне і практичне значення.

Зв'язок роботи з науковими програмами, планами, темами. Тематика дисертаційної роботи та одержані результати безпосередньо пов'язані з "Основними науковими напрямами та найважливішими проблемами фундаментальних досліджень у галузі природничих, технічних і гуманітарних наук НАН України на 2014-2018 роки" в частині п.1.2.8.1. "Розробка методів та інформаційних технологій розв'язання задач комп'ютерної криптографії та стеганографії", зі Стратегією кібербезпеки України від 15 березня 2016 року № 96/2016 у контексті п. 4.1 "Розвиток та вдосконалення системи технічного і криптографічного захисту інформації" і Рамковою програмою ЄС з досліджень та інновацій "Горизонт 2020". Результати роботи відображені у звітах держбюджетних НДР Національного авіаційного університету "Методи забезпечення конфіденційності державних інформаційних ресурсів в інформаційно-комунікаційних системах" (№ 61/09.01.08) "Квантово-

криптографічні методи захисту критичної інформаційної інфраструктури держави” (0117U006770), у яких здобувач брав участь у якості виконавця.

Мета і задачі дослідження. Метою дисертаційної роботи є забезпечення цілісності даних в інформаційно-комунікаційних системах за рахунок розробки методів побудови і засобів реалізації ефективних криптографічних функцій гешування.

Для досягнення поставленої мети **необхідно розв’язати такі основні задачі:**

– проаналізувати сучасні методи і алгоритми побудови та реалізації ефективних криптографічних функцій гешування для виявлення їх недоліків і формалізації завдання наукового дослідження;

– розробити та удосконалити методи побудови функцій гешування для ефективного застосування у системах, для яких критичними є параметри швидкості і криптостійкості;

– удосконалити метод побудови генераторів псевдовипадкових послідовностей (ПВП) для формування статистично стійкої гами;

– удосконалити метод криптографічного захисту інформації для забезпечення конфіденційності і цілісності даних в інформаційно-комунікаційних системах;

– розробити спеціалізоване програмне забезпечення та методику для проведення експериментів і верифікації запропонованих методів.

Об’єктом дослідження є процес забезпечення цілісності та конфіденційності даних.

Предметом дослідження є методи, способи та алгоритми побудови ефективних криптографічних функцій гешування.

Методи дослідження. Проведені дослідження базуються на сучасних методах теорії криптографії (дослідження швидкості і стійкості геш-функцій), скінченних полів та елементів теорії чисел (побудова функцій гешування і криптоалгоритмів), об’єктно-орієнтованого програмування та математичної

статистики (розробка програмних засобів, проведення експериментів і обробка їх результатів, аналіз колізійних властивостей геш-функцій).

Наукова новизна одержаних результатів полягає у такому:

– *вперше розроблено* метод побудови функцій гешування, який базується на структурі Меркла-Демґарда та за рахунок доповнення вхідного повідомлення розміром цього повідомлення та псевдовипадковою послідовністю salt (розраховується на основі вхідного повідомлення), використання у функції стиснення нової послідовності операцій (на основі 6-ти не лінійних функцій, операцій підстановки, додавання за модулем 2 і 2^n, циклічних і лінійних зсувів), дозволив будувати криптостійкі функції гешування;

– *вперше розроблено* метод побудови функцій гешування, який базується на структурі Меркла-Демґарда та за рахунок доповнення вхідного повідомлення псевдовипадковою послідовністю salt (розраховується на основі вхідного повідомлення та його розміру), використання у функції стиснення додаткового вектору внутрішнього стану та нової послідовності операцій (на основі 4-х не лінійних функцій, операцій підстановки, перестановки, додавання за модулем 2 і 2^n та циклічного зсуву), дозволив будувати швидкісні функції гешування;

– *удосконалено* метод побудови генераторів псевдовипадкових послідовностей, який за рахунок обробки вектора внутрішнього стану та ключового вектору операціями підстановки, циклічного зсуву, складання за модулем 2 і 2^n та 4-ма нелінійними функціями, дозволив будувати ефективні генератори псевдовипадкових послідовностей;

– *удосконалено* метод криптографічного захисту інформації, який за рахунок фіксування інформації про ідентифікатор користувача, ідентифікатор сесії, час відправлення, довжину повідомлення та його порядковий номер, а також використання нової процедури формування сеансового ключа для

шифрування, дозволяє забезпечити конфіденційність і цілісність даних в інформаційно-комунікаційних системах.

Практичне значення одержаних результатів. Отримані в дисертаційній роботі результати можуть бути використані для підвищення ефективності забезпечення цілісності даних в ІКС та інших завданнях криптографічного захисту даних.

Зокрема, практична цінність роботи полягає у такому:

– розроблено і реалізовано програмно шість нових функцій гешування (стійкі – Oberih-1, Oberih-2, Oberih-3, швидкісні – Barvinok-1, Barvinok-2, Barvinok-3), які дозволяють забезпечити стійкість, підвищити швидкість у 1.15-1.36 разів (Oberih) або у 1.16-1.53 разів (Barvinok) і можуть бути використані для забезпечення цілісності даних в ІКС, блокчейн системах, електронній пошті, системах миттєвого обміну повідомленнями (месенджерах) та інших сучасних застосунках;

– розроблено і реалізовано програмно три генератори ПВП (Viriy-1, Viriy-2, Viriy-3), які є швидшими у 1.02-1.22 разів в порівнянні з аналогами, що можуть бути використанні для криптографічних застосувань (генерування ключів, потокові шифри тощо) для підвищення їх ефективності;

– подано заявку на отримання патенту України на корисну модель “Спосіб побудови стійких функцій гешування” від 27.05.2020 року;

– результати дисертації використовуються у навчальному процесі Вінницького національного технічного університету, науковому процесі Національного авіаційного університету та ННВК “Інформаційно-комунікаційні системи”.

Особистий внесок здобувача. Основні положення і результати дисертаційної роботи, що виносяться до захисту, отримані автором самостійно. У роботах, написаних у співавторстві, автору належать: [1,6] – розробка і реалізація стійких генераторів ПВП, [2] – реалізація і дослідження програмної системи для блокчейн; [3,5,7,8] – розробка і дослідження

швидкісних та криптостійких функцій гешування; [4] – огляд методів і засобів забезпечення цілісності даних в сучасних ІКС; [9-11] – експериментальне дослідження розроблених методів забезпечення конфіденційності і цілісності даних в ІКС. З робіт, що опубліковані в співавторстві, у дисертаційній роботі використовуються виключно результати, отримані особисто здобувачем.

Апробація результатів дисертації. Основні положення дисертаційної роботи доповідалися та обговорювалися міжнародних наукових конференціях, серед яких: МНПК молодих учених і студентів “Політ. Сучасні проблеми науки” (м. Київ, 2016-2020 роки), МНТК “Проблеми експлуатації та захисту інформаційно-комунікаційних систем” (Київ, 2017 рік), МНПК “Актуальні питання забезпечення кібернетичної безпеки та захисту інформації” (Верхній Студений, 2018 рік), МНТК “Сучасні засоби зв’язку” (Мінськ, 2019 рік).

Публікації. Основні положення дисертації опубліковано в 11 наукових працях, у тому числі – 6 наукових статей (1 – у міжнародному рецензованому періодичному виданні [1], що входить до бази даних Scopus, 5 – у вітчизняних [2-4,6] і закордонних [5] фахових наукових журналах), а також 5 матеріалів і тез доповідей на конференціях [7-11].

Структура роботи та її обсяг. Дисертація складається із анотації, вступу, чотирьох розділів, загальних висновків, додатків, списку використаних джерел і має 123 сторінки основного тексту, 31 рисунок, 17 таблиць, 5 сторінок додатків. Список використаних джерел містить 125 найменувань і займає 14 сторінок. Загальний обсяг роботи 142 сторінки.

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ПОБУДОВИ ЕФЕКТИВНИХ ФУНКЦІЙ ГЕШУВАННЯ

1.1. Основні визначення та поняття

Захист інформації (англ. Data protection) — сукупність методів і засобів, що забезпечують цілісність, конфіденційність і доступність інформації за умов впливу на неї загроз природного або штучного характеру, реалізація яких може призвести до завдання шкоди власникам і користувачам інформації [4].

Цінність інформації була усвідомлена дуже давно, внаслідок цього виникло завдання захисту від надмірно цікавих людей. Мистецтво тайнопису зародилося ще в доантичні часи і проіснувало аж до зовсім недавнього часу. Лише декілька десятиліть тому все змінилося корінним чином – інформація придбала самостійну комерційну цінність і стала широко поширеною, майже звичайним товаром [29]. Її передають, зберігають, продають і купують, а значить – крадуть і підроблюють. З цього випливає потреба її захищати. Сучасне суспільство все більшою мірою стає інформаційно-обумовленим, успіх будь-якого виду діяльності все сильніше залежить від володіння певними відомостями і від відсутності їх у конкурентів [49].

Серед всього спектру методів захисту даних від небажаного доступу особливе місце займають криптографічні методи.

Криптографія – наука про захист інформації від прочитання її сторонніми. Захист досягається шифруванням, тобто перетворенням, які роблять захищені вхідні дані складними для розкриття за вхідними даними без знання спеціальної ключової інформації – ключа [32]. Під ключем розуміється легко змінна частина криптосистеми, що зберігається в таємниці і визначає, які шифрувальні перетворення з можливих виконуються в даному випадку.

Криптографія довгий час була засекречена, так як застосовувалася, в основному, для захисту державних і військових секретів [23]. В даний час методи і засоби криптографії використовуються для забезпечення

інформаційної безпеки не тільки держави, а й приватних осіб, і організацій. Справа тут зовсім не обов'язково в секретах. Занадто багато різних відомостей поширюється по всьому світу в цифровому вигляді. І над цими відомостями висять загрози нелегітимного ознайомлення, накопичення, підміни, фальсифікації. Найбільш надійні методи захисту від таких загроз дає саме криптографія [2].

Одним із головних розділів криптографії є симетричне шифрування.

Симетрична шифрування – це сучасна назва процедури шифрування і розшифрування, які реалізуються на обох кінцях лінії зв'язку за допомогою однакових або майже однакових шифрувальних пристроїв (шифраторів). Симетрична шифрування є найбільш поширеним в сучасному світі, хоча зусилля відомих математиків в останні роки по ряду причин майже повністю були спрямовані на дослідження проблем, пов'язаних з асиметричним шифруванням і відкритим розподілом ключів [37].

При симетричному шифруванні нападаюча сторона (противник) не може прочитати зашифроване повідомлення через те, що він не знає алгоритму шифрування, який в значній мірі визначається секретним ключем.

Асиметричне шифрування принципово відрізняється від симетричного тим, що його алгоритм шифрування, який представляє собою відображення деякої множини в себе, загальновідомий. Стійкість цього шифрування ґрунтується на тому, що противник (нелегітимний користувач) не в змозі доступними йому засобами обчислити зворотне відображення [48]. Разом з тим обчислення зворотного відображення для легітимного користувача обчислювальною є через те, що він знає деякий секрет, який був використаний при побудові прямого відображення.

Асиметричне шифрування принципово відрізняється від симетричного ще також тим, що для нього не потрібен абсолютно надійний канал для розсилки секретних ключів [23]. Це властивість в деяких випадках дає асиметричним системам шифрування значні практичні переваги перед традиційним симетричним. Разом з тим необхідно відзначити, що, по-перше,

складність обчислення значень «одностороннього відображення» і його зворотного, тобто складність асиметричного шифрування і розшифрування, зазвичай значно вище, ніж складність цих процедур при традиційних симетричних автоматних методах шифрування, і, по-друге, в даний час невідомі практично реалізовані системи асиметричного шифрування, для яких досить переконливо доведена неможливість їх розколювання кваліфікованим незаконним користувачем [32].

Для захисту інформації використовуються різні типи криптографічних алгоритмів [2]. Криптоалгоритми діляться на три категорії (рис. 1.1):

- безключові алгоритми, які не використовують будь-яких ключів в процесі криптографічних перетворень;
- одноключові алгоритми, які використовують у своїх обчисленнях якийсь секретний ключ;
- двохключові алгоритми, в яких на різних етапах обчислень застосовуються два види ключів: приватні та публічні.

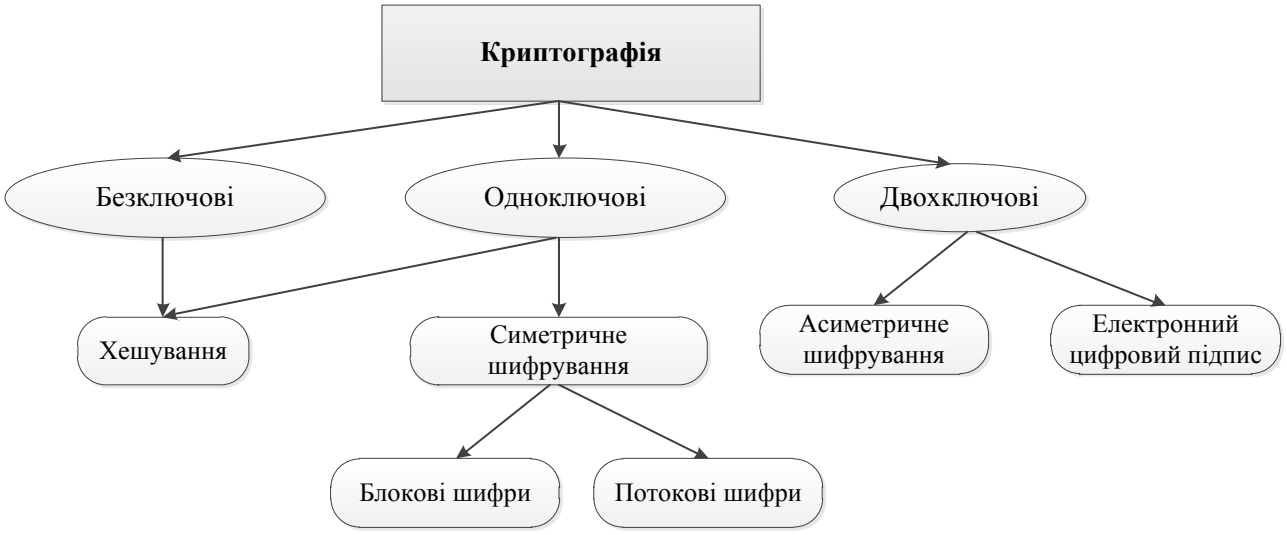

Рис. 1.1. Класифікація криптографічних алгоритмів

Розглянемо коротко основні типи криптоалгоритмів.

1. *Функції гешування (MD4[15], Snefru[18], SHA-2[1])*. Виконують стискання даних змінної довжини в послідовність фіксованого розміру – фактично це контрольне підсумовування даних, яке може виконуватися як за участю якогось ключа, так і без нього [43]. Такі функції мають досить широке застосування в області захисту комп'ютерної інформації [36], наприклад:

1.1. Для підтвердження цілісності будь-яких даних в тих випадках, коли використання електронного підпису або є надлишковим;

1.2. В самих схемах електронного підпису – підписується зазвичай геш-код даних, а не всі дані цілком;

1.3. В різних схемах аутентифікації користувачів.

2. *Генератори випадкових чисел* [48]. Випадкові числа необхідні, в основному, для генерації секретних ключів шифрування, які, в ідеалі, повинні бути абсолютно випадковими. Потрібні вони і для обчислення електронного цифрового підпису, і для роботи багатьох алгоритмів аутентифікації.

3. *Алгоритми симетричного шифрування* (проста перестановка, одинична перестановка, подвійна перестановка, магічний квадрат) – алгоритми шифрування, в яких для зашифровування і розшифрування використовується один і той же ключ, або ключ розшифрування легко обчислюється з ключа зашифровування і навпаки [30].

3.1.Блочне шифрування [2] – в цьому випадку інформація розбивається на блоки фіксованої довжини (наприклад, 64 або 128 бітів), після чого ці блоки по черзі шифруються. Причому в різних алгоритмах шифрування або навіть в різних режимах роботи одного і того ж алгоритму блоки можуть шифруватися незалежно один від одного або «зі зчепленням» – коли результат зашифровування поточного блоку даних залежить від значення попереднього блоку або від результату зашифруванні попереднього блоку;

3.2.Потокове шифрування – необхідно, перш за все, в тих випадках, коли інформацію неможливо розбити на блоки – скажімо, якийсь потік даних, кожен символ яких повинен бути зашифрований і відправити куди-небудь, не чекаючи інших даних, достатніх для формування блоку [47]. Тому алгоритми потокового

шифрування шифрують дані побітно або посимвольно. Хоча, деякі класифікації не поділяють блочне і потокове шифрування, вважаючи, що потокове шифрування – це шифрування блоків одиничної довжини [2].

4. *Генератори псевдовипадкових чисел (Xorshift, Вихор Мерсена, лінійний конгруентний метод)* [38]. Не завжди можливо отримання читається ну ніяк випадкових чисел – для цього необхідна наявність якісних апаратних генераторів. Однак на основі алгоритмів симетричного шифрування можна побудувати дуже якісний генератор псевдосліввипадкових чисел.

5. *Алгоритми аутентифікації (Фейге-Фиата-Шамира, Kerberos, RSA).* Дозволяють перевірити, що користувач (або віддалений комп'ютер) дійсно є тим, за кого себе видає. Найпростіша схема аутентифікації – парольний – не вимагає наявності будь-яких криптографічних ключів, але доведено є слабкою. А за допомогою секретного ключа можна побудувати помітно сильніші схеми аутентифікації. Тобто, рівність чисел означає, що користувач має необхідним секретним ключем, тобто йому вдалося довести свою легітимність.

6. *Алгоритми електронного підпису (RSA, Ель-Гамаля)* [43]. Використовують секретний ключ для обчислювання електронного цифрового підпису даних, а який вираховується з нього відкритий – для її перевірки.

7. *Алгоритми з двома ключами (RSA, Ель-Гамаля)* [31]. Простим обчисленням відкритого ключа з секретного і практичною неможливістю зворотного обчислення. Однак призначення ключів є зовсім іншим [30]:

7.1. Секретний ключ використовується для обчислення електронного підпису;

7.2. Відкритий ключ необхідний для її перевірки.

При дотриманні безпечного зберігання секретного ключа, ніхто, крім його власника, не в змозі обчислити вірний електронний підпис будь-якого електронного документа [4].

Надійність або якість шифрування визначається обсягом роботи криптоаналітика, необхідної для розкриття системи. Шифросистема може служити об'єктом нападу противника, що займає той чи інший інтелектуальним

і обчислювальним потенціалом [45]. Можливості потенціального противника визначають вимоги, що пред'являються до надійності шифрування.

Вихідна інформація і цілі криптоаналітика можуть бути різними. Безсумнівно, основна мета противника є в отриманні конфіденційної інформації. Метою нападу може служити також застосований секретний ключ, за допомогою якого криптоаналітик може розкривати інші криптограми. Шифросистема може бути несприйнятливою до одним загрозам і бути вразливою по відношенню до інших [49]. Спроби противника з добування зашифрованою інформації зазвичай називають криптоатаками.

У криптографії з секретним ключем зазвичай розглядають наступні криптоатаки [47]:

1) атака на основі підібраного шифротексту — криптографічна атака, за якої криптоаналітик збирає інформацію про шифр через підбір зашифрованого тексту і отримання відповідного відкритого тексту при невідомому ключі. Як правило, криптоаналітик може скористуватися пристроєм розшифрування один або декілька разів [27]. Використовуючи отримані дані, він може спробувати відновити секретний ключ для розшифрування. Існують шифри для яких подібні атаки можуть виявитися успішними. до них зараховують;

2) атака на основі підібраного шифротексту може бути адаптивною і не адаптивною;

3) атака на основі неадаптивно підібраного шифротексту;

4) при неадаптивній атаці криптоаналітик не використовує висліди попередніх розшифрувань, тобто шифротексти підбираються заздалегідь;

5) атака на основі адаптивно підібраного шифротексту;

6) в протилежному випадку криптоаналітик адаптивно підбирає шифротекст, що залежить від слідів попередніх розшифрувань (cca2);

7) атака на основі відомого відкритого тексту: криптоаналітик має парами (x, y) відкритих і таким, що відповідає їм шифрованих текстів. Потрібно визначити ключ хоча б однієї з пар. [2] В окремому випадку, коли потрібно визначити ключ до або, переконавшись у своїй нездатності зробити це,

визначити відкритий текст ще однієї криптограми, зашифрованої на тому ж ключі;

8) атака на основі обраного відкритого тексту відрізняється від попередньої лише тим, що криптоаналітик має можливість вибору відкритих текстів. Мета атаки – та ж, що і попередньої. Подібна атака можлива, наприклад, у разі, коли криптоаналітик має доступ до шифратору сторони, яка передає, або в системах впізнання "свій-чужий";

9) атака на основі обраного шифртекста відрізняється від другої атаки лише тим, що криптоаналитик має можливість вибору шифртекст. Мета атаки – та ж, що і в другому випадку. Подібна атака можлива, наприклад, у разі, коли криптоаналитик має доступ до шифратору приймаючої сторони;

10) атаки на основі вибраних текстів вважаються найбільш небезпечними. Іноді до вказаних атакам додають і інші. Шифр, що витримує всі можливі атаки, можна визнати стійким або надійним.

При оцінці ефективності будь-якої криптоатаки зазвичай користуються загальноприйнятим в криптографії правилом Кергофса (голландського криптографа XIX ст.). Це правило викладено в книзі "Військова криптографія", у ній сформульовано шість таких вимог до систем шифрування [32]:

1) система повинна не піддаватись криптоаналізу, якщо не теоретично, то практично;

2) компрометація системи не повинна завдавати незручностей її користувачам;

3) секретний ключ повинен легко запам'ятовується без будь-яких записів;

4) криптограма повинна бути представлена в такій формі, щоб її можна було передати по телеграфі;

5) апаратура шифрування повинна бути портативною і такою, щоб її могла обслуговувати одна людина;

6) система повинна бути простою. вона не повинна вимагати ні запам'ятовування довгого переліку правил, ні великого розумового напруження.

Друге з цих правил і стало називатися правилом Керкгофса. Суть його полягає в тому, що при проведенні криптоаналізу можна вважати відомою систему шифрування. Стійкість (або надійність) шифрування повинна визначатися лише секретністю ключа шифрування [46]. Визнання усіма цього принципу в криптографії пов'язано з тим, що рано чи пізно ті чи інші відомості про використовувану шифросистему стають відомими. Проте шифри, які використовуються спеціальними службами, всіляко охороняються. Це обумовлено необхідністю додаткового запасі міцності, оскільки до цього часу створення шифрів з доказовою стійкістю є дуже складною проблемою [43].

Криптостійкістю називається характеристика шифру, що його стійкість до розшифрування без знання ключа (тобто криптоатака). Показник криптостійкості - головний параметр будь криптосистеми [30]. Як показник криптостійкості можна вибрати:

- кількість всіх можливих ключів або ймовірність підбору ключа за заданий час із заданими ресурсами;
- кількість операцій або час (з заданими ресурсами), необхідне для злому шифру із заданою вірогідністю;
- вартість обчислення ключової інформації або вихідного тексту.

Всі ці показники повинні враховувати також рівень можливої криптоатаки. Однак слід розуміти, що ефективність захисту інформації криптографічними методами залежить не тільки від криптостійкості шифру, але і від безлічі інших чинників, включаючи питання реалізації криптосистем у вигляді пристроїв або програмного модуля.

1.2. Аналіз програмних модулів захисту

Програмний модуль — функціонально закінчений фрагмент програми, оформлений у вигляді окремого файлу з сирцевим кодом або його іменованої частин, призначений для використання в інших програмах.

Модуль характеризують:

- один вхід і вихід. На вході програмний модуль отримує певний набір початкових даних;
- функціональна завершеність. Модуль виконує набір визначених операцій для реалізації визначених операцій для реалізації кожної окремої функції, достатніх для завершення початої обробки даних;
- логічна незалежність. Результат роботи даного фрагменту програми не залежить від роботи інших модулів;
- слабкі інформаційні зв'язки з іншими програмними модулями. Обмін інформацією між окремими модулями повинен бути мінімальним;
- розмір і складність програмного елемента в розумних рамках.

Таким чином, модулі містять описання початкових даних, операції обробки даних і структури взаємозв'язку з іншими модулями. Програмний модуль є самостійним програмним продуктом. Тобто кожен програмний модуль розроблюється, компілюється і налагоджується окремо від інших модулів програми. Більш того, кожний розроблений ПМ може включатись у склад різних програмних систем при умові виконання вимог, які пред'являються до його використання в документації до цього модуля. Таким чином, ПМ може розглядатись і як засіб спрощення складних програм, і як засіб накопичення і багатократного використання програмних знань. Розмір модуля вимірюється числом операторів які містяться в ньому. Модуль не повинен бути надто малим чи надто великим. Великі модулі, як правило, складні для розуміння і незручні для внесення змін, вони можуть суттєво збільшити сумарний час повторних трансляцій програми при відладці. Маленькі модулі ускладнюють загальну структурну схему програми і можуть не оправдовувати накладних розходів, пов'язаних з їх оформленням. Зазвичай рекомендуються програмні модулі розміром від декількох десятків до декількох сотень операторів.

Надійність модуля – це міра його внутрішніх зв’язків. Чим вища надійність модуля, тим більше зв’язків приховано від зовнішньої по відношенню до нього частини програми і, відповідно, тим простіша сама програма. Найнижчим ступенем надійності володіє модуль, для перевірки на співпадіння. Такий ПМ оформлюються послідовність операторів які повторюються в декількох місцях програми. Якщо виникне необхідність зміни цієї послідовності в одному з контекстів, прийдеться змінювати сам модуль, що може зробити його використання в інших контекстах помилковим. Такий клас програмних модулів не рекомендується для використання.

Було здійснено аналіз програмних модулів, які використовуються:

1) Месенджери – це ПМ для обміну та шифрування текстових, голосових, відео повідомлень, чи файлів, які передаються через мобільний додаток або веб-сервіс для обміну миттєвими повідомленнями.

2) Модулі захисту для цифрового підпису. Цифровий підпис - це електронні дані, які додаються підписантом до інших електронних даних, або логічно пов'язуються з ними, і використовуються ним в якості підпису.

3) Blockchain системи - побудовані за певними правилами безперервні послідовні ланцюжки блоків (зв'язний список), що містять інформацію. Найчастіше копії ланцюжків блоків зберігаються на безлічі різних комп'ютерів незалежно один від одного. Кожен блок містить часову мітку та посилання на попередній блок геш дерева.

Розглянемо кожен з них більш детальніше.

1. <u>Месенджери</u>

MTProto 1.0 [51] – ПМ, який використовується для шифрування повідомлень при передачі клієнтами Telegram. Протокол підрозділяється на три фактично незалежні компоненти:

- компонент високого рівня (мова запиту API): визначає спосіб, за допомогою якого API запити та відповіді перетворюються на двійкові повідомлення;

- криптографічний (авторизаційний) рівень: визначає спосіб шифрування повідомлень перед передачею через транспортний протокол;
- транспортний компонент: визначає метод для клієнта та сервера для передачі повідомлень за допомогою іншого існуючого мережевого протоколу (наприклад, http, https, tcp, udp).

Signal Protocol [52] – використовується для шифрування миттєвих повідомлень Facebook Messenger. Функція доступна в розділі Secret Conversations (Секретна листування). У цих чатах діє повне або наскрізне шифрування (End-to-end encryption), при якому прочитати повідомлення можуть тільки користувачі, які беруть участь в листуванні, а провайдери, хакери, урядові відомства та інші сторонні особи не можуть отримати доступ до ключів, необхідних для розшифровки повідомлень. Зашифрувати можна персональні чати з текстом і стікерами, але відео або анімовані картинки вони не підтримують. Протокол поєднує в собі алгоритм подвійного ритча, товари та трійковий рухінг Діффі-Хеллмана (3-DH) і використовує Curve25519, AES-256 та HMAC-SHA256 як примітиви [53].

Протокол сигналу також підтримує кінцеві коди зашифрованих групових чатів. Протокол групового чату являє собою комбінацію подвійного подвійного храпового та багатоадресного шифрування. На додаток до властивостей, наданих протоколом "один на один", протокол групового чату забезпечує консистентність динаміків, стійкість до нестабільності, зменшену стійкість до повідомлень, рівновагу в обчислювальному пристрої, рівність довіри, передачу повідомлень підгруп, а також скорочення і розширення членства. Для автентифікації користувачі можуть вручну порівняти відбитки відкритих кнопок через зовнішній канал. Це дає змогу користувачам перевіряти ідентифікацію один одного та уникнути атаки між людьми. Реалізація також може вибрати використання довіри на механізм першого використання, щоб повідомляти користувачів, якщо змінюється ключ кореспондента [52].

TLS Skype [54] – використовується для миттєвих повідомлень використовується TLS (безпека на рівні транспорту) для шифрування повідомлень між клієнтом Skype та службою чату в нашій облас або AES, коли вони надсилаються безпосередньо між двома клієнтами Skype. Більшість повідомлень надсилаються двома способами, однак у майбутньому вони будуть надсилатися лише через нашу хмару, щоб забезпечити оптимальну роботу користувачів.

Голосові повідомлення зашифруються, коли вони доставляються вам. Однак після того, як Ви прослухали голосові повідомлення, він передається з наших серверів на місцевий комп'ютер, де він зберігається як незашифрований файл [54].

Skype використовує AES, також відомий як Rijndael, який використовується урядом США для захисту конфіденційної інформації, і Skype протягом деякого часу завжди використовувала 256-бітне шифрування. Користувацькі ключі сертифіковані сервером Skype за реєстрацією за допомогою 1536 або 2048-бітних сертифікатів RSA.

2. <u>Для електронно-цифрового підпису</u>

2.1. Digital Signature eSign Genie TLS Skype [55] – це проста у використанні програма, яка є дійсно найпростішим програмним забезпеченням eSign. Безпека є головним пріоритетом при надсиланні та отримання контрактів, угод та інших необхідних документів підпису. eSign Genie дозволяє завжди залишатися безпечним. До основних переваг підпису можна віднести:

- шифрування документів 256-розрядне шифрування забезпечує безпеку всіх документів у системі;
- відстежування документу за IP-адресою та талонами часу;
- сертифіковане завершення. Потрібне підтвердження наявності засвідченого документа завершення, що містить IP-адресу та штамп часу;

- контроль видимості, можна обмежувати або дозволяти. Підписувач може переглядати кожну частину документа і мати повний контроль над процесом подання.

Автентифікація електронної пошти: перевірте підписувача електронною поштою.

eSign Genie пропонує повний спектр API-інтерфейсів для автоматизації процесу надсилання та надсилання документів. API можна розробити для [55] :

- режим єдиної компанії: єдиний обліковий запис/компанія, щоб ініціювати та завершити автоматизацію автоматизації з веб-сайту програми;
- модель партнера або мульти-орендарна модель: де партнер з розробки може розвивати спільну інтеграцію для всіх своїх компаній-користувачів. Кожна компанія також підпишеться на послуги eSign Genie. Процес реєстрації для партнерських облікових записів може бути ініційований через партнерську заявку через API.

API процеси можна використовувати як:

- Створюйте шаблони електронної пошти в eSign Genie із PDF-файлу, завантаженого через API.
- Надіслати інформацію та створювати документи з шаблону.
- Створіть PDF-документ з вашої програми з тегами заповнюваних полів.

2.2. SignEasy [56] – безпечне рішення для бізнесу та професіоналів для підписання та отримання документів із смартфону, планшету та комп’ютера. ПМ дозволяє підписувати документ у послідовній або паралельний послідовності, контролювати хід документу, перевіряти хто підписав документ і хто ще має. SignEasy дозволяє надсилати підписувачам нагадування. Дозволяється імпорт та експорт документа з електронної пошти та інших програм. Відповідно до закону ESIGN, eIDAS і визнаний у всьому світі. Додатковий захист з кодом доступу та автентифікацією відбитками пальців.

Документи надсилаються та зберігаються за допомогою SSL-шифрування. Перевірка аудиту для юридичних даних, включаючи адресу електронної пошти підписувача, IP-адресу пристрою, відбиток на документі та позначку часу.

2.3. Adobe EchoSign [57] прискорює процес підписання документів за допомогою миттєвих факсів або електронних підписів. Adobe Sign перейде на новий сертифікат SSL 2 травня 2018 року. Не змінюється відкритий ключ, що лежить в основі криптографічних протоколів або схеми. Змінюється лише сертифікат Adobe Sign (вказаний нижче). Public Key, Root CA, і Intermediate CA залишаться колишніми. Зробіть прокруткою і краєм просту, щоб збирати та надсилати юридичні електронні підписи - на будь-якому пристрої. Завдяки потужності Adobe Sensei ви навіть можете легко створювати заповнювані, підписувані форми. Adobe Sign є кращим рішенням електронного підпису Microsoft. Це глибоко інтегроване з Office 365, що спрощує запит та відстеження підписів у ваших улюблених інструментах. Adobe Sign захищає дані та документи, які допоможуть вам виконати найсуворіші стандарти безпеки [58]. Він також відповідає найвибагливішим галузевим і державним нормам, таким як FedRAMP Tailored, і найширшим спектром юридичних вимог, включаючи GDPR Європейського Союзу.

2.4. Signable [59] забезпечує найкращий спосіб захоплення електронних підписів на документі між двома (або більше) сторонами.

Підписані також допоможуть підприємствам відповідати юридич-ним вимогам електронних процедур підписання в законодавчому законодавстві у всьому світі.

Ми записуємо кожну дію процесу підписання документів з аудиторськими маршрутами, до яких ви можете отримати доступ до свого облікового запису.

Підписані умови повинні перевищувати юридичні вимоги, які повинен відповідати документ, який буде прийнято на законних підставах. До них відносяться:

- Будучи однозначно пов'язаним з підписуючою стороною і будучи здатним ідентифікувати підписувача.
- Використовуючи дані електронного підпису, будь-яка зміна даних може бути виявлена та позначена.

Відстеження ваших онлайн-документів - це вітер. Ви можете миттєво дізнатись, які клієнти мають підписані незавершені документи, які документи прийнято або відхилено, а їхня дата оброблена.

Коли кожен підписав документ, ви отримаєте повну копію всіх підписів із сертифікатом, що покаже вас:

- Кожна партія підписів.
- Дати, IP-адреси та відбитки пальців.
- Повний журнал перевірки.

Цей документ надійно зберігається у вашому обліковому записі для зручного доступу, як раз у вас, у будь-який час.

Щоб усі ваші документи були захищені та юридично обов'язковими, ми забезпечуємо:

- Відповідність eIDAS та перевищення правил eSign у Великобританії.
- Детальний журнал перевірки зберігається для кожної дії підпису.
- Ідентифікація користувачів, яким дозволено надсилати документи.
- Цілісність документів перевіряється, завжди.
- Унікальні солоні відбитки пальців для кожного документа.
- Безпечне зберігання документів та даних.
- Ідентифікатор підпису підтверджено електронною поштою, мітками та географічним відстеженням.
- SSL 256-розрядне шифрування AES / RSA.
- Доступ до документів доступний лише тим і тими, кого ви дозволили.

3. Для Blockchain

3.1. Bitcoin [60] – електронна валюта, концепт якої був озвучений 2008 року Сатосі Накамото, і представлений ним 2009 року, базується на самоопублікованому документі Сатосі Накамото [17]. Повна капіталізація ринку біткоїнів на 5 грудня 2017 року, коли курс сягав 12 000 $, становить 200 млрдUSD. У грудні 2017 року став шостою за капіталізацією валютою світу, обійшовши рубль, фунт і південнокорейську вону. 7 грудня курс досяг свого історичного чергового максимуму в 17,7 тис. дол., наступний ріст до 20 тис доларів відбувся 17 грудня, потім курс впав до 16 тис. У 2018 році курс продовжив падати. Періодично підіймаючись і падаючи на 10-20%, станом на 5 квітня 2018 року коштує 6800 дол. 17 липня курс біткоїна підскочив на 10%, що стало максимальним денним приростом із 12 квітня. У результаті він подолав психологічно важливу відмітку в $7000. Крім того, ціна криптовалюти піднялася вище за середнє значення за останні 50 днів, що також розглядається ринком як важливий позитивний індикатор. У кінці 2018 року валюта продовжила падіння. Найбільше падіння було зафіксовано 15 грудня, коли курс впав до $3194, проте через 2 доби ціна зросла до $3276. Обсяги торгів теж знижувалися і біткоїна, й інших криптовалют.

Bitcoin не має централізованого управління та емітентів. Транзакції із цифровим підписом між двома вузлами передаються до всіх вузлів мережі peer-to-peer, а самі дані про переміщення коштів зберігаються у розподіленій базі даних. Для запобігання можливості трати чужих грошей або використання своїх коштів двічі використовуються криптографічні методи.

Bitcoin покладається на криптографічні принципи, щоб створити унікальні, невідтворювані і ділені маркери валюти. Користувачі зберігають криптографічні ключі до своїх власних грошей локально на персональному комп'ютері, і проводять транзакції безпосередньо один з одним через пірингову мережу, перевіряючи за допомогою мережі достовірність грошових переказів. Фізично кожна монета в системі має свій унікальний ключ.

При здійсненні транзакції користувач додає до монети відкритий ключ адресата і підписує її своїм особистим закритим ключем. Щоб запобігти подвійному списанню однієї монети, всі транзакції транслюються іншим учасникам, а повний список транзакцій в анонімному вигляді зберігається в розподіленій мережі. При кожній новій транзакції ключі перевіряються за списком попередніх транзакцій. Інакше кажучи, Bitcoin заснований на записі переміщень грошових коштів з використанням асиметричного шифрування.

Для запобігання багаторазової витрати однієї і тієї ж суми мережа реалізує щось подібне розподіленому серверу часу, використовуючи ідею ланцюжка гешів, кожен з яких обчислюється на базі попереднього. Для зменшення розміру розподіленої мережевої БД використовується деревоподібне гешування.

Наразі кількість монет в обігу системи становить понад 16.7 млн. Фактично, сьогодні Bitcoin — це хмарна мережа розподілених обчислень. Дохід в Bitcoin монетизується за рахунок валюти, цінність якої забезпечує електрична енергія і робота процесора. Фактично, номінал однієї монети дорівнює певній кількості процесорного часу. На практиці вартість її визначається співвідношенням біржових пропозицій та попиту, що з певною затримкою впливає на необхідні комп'ютерні ресурси для генерації монети.

Для збереження конфіденційності може бути застосований «біткоїн-міксер», котрий в одній транзакції змішує на вході біткоїни різних користувачів і робить одночасно багато платежів. Це ускладнює співставлення, хто куди платив.

Для скорочення замість слова «Bitcoin» часто пишуть абревіатуру латинськими буквами BTC. Такий запис схожий на коди валют, однак подібний код міжнародним стандартом ISO 4217 поки не присвоєно. 7 жовтня 2014 року Bitcoin Foundation опублікувала плани домогтися стандартизації коду для біткоїнів. Запис BTC суперечить прийнятій в стандарті системі — іменувати «глобальні товари» починаючи з X (наприклад, золото має код XAU). Як кандидата розглядають варіант XBT.

При вказівці BTC або XBT мається на увазі розрахункова одиниця, а не мережа, набір алгоритмів або будь-яка інша річ, що належить до них. Знак біткоїнів затверджений для включення в планований стандарт юнікоду версії 9.0, йому присвоєно номер U + 20BF. Іноді застосовують символ ฿ — знак тайського бата, але він сумісний не із усіма кодуваннями і шрифтами. URI-схема «Bitcoin:» офіційно включена в специфікації WHATWG для HTML5. Біткоїн також планується додати в список валют в Microsoft Excel 2016.

3.2. Ethereum – це децентралізована платформа з інтелектуальними контрактами: додатки, які працюють точно так же, як запрограмовано без будь-якої можливості простоїв, цензури, шахрайства або втручання третіх сторін. Ці програми працюють на настроюється блок-ланцюжку, надзвичайно потужної загальної глобальної інфраструктури, яка може переміщати вартість і представляти власність на власність. Це дозволяє розробникам створювати ринки, зберігати реєстри боргів або обіцянок, переказувати кошти відповідно до інструкцій, даними в минулому (наприклад, волею або ф'ючерсним контрактом) і багатьма іншими речами, які ще не були винайдені, все без середньої людини або ризик контрагента.

Можливості:

- Ефіріума Гаманець є шлюзом для децентралізованих додатків на blockchain Ефіріума. Він дозволяє зберігати і захищати ефір і інші кріптоактівние активи, створені на Ethereum, а також писати, розгортати і використовувати смарт-контракти.
- Створити комерційний цифровий токен, який можна використовувати як валюту, уявлення активу, віртуальну частку, доказ членства або що-небудь взагалі. Ці жетони використовують стандартний API-інтерфейс для монет, тому ваш контракт буде автоматично сумісний з будь-яким гаманцем, іншим контрактом або обміном, також використовуючи цей стандарт. Загальна кількість токенов в зверненні може бути встановлено на просту

фіксовану суму або змінюватися в залежності від будь-якого запрограмованого набору правил.

- Використовуючи платформу, ви можете створити контракт, який буде містити гроші вкладника, поки не буде досягнута якась конкретна дата або мета. Залежно від результату кошти будуть або передані власникам проектів, або будуть повернуті назад учасникам. Все це можливо, не вимагаючи централізованого арбітра, посередництва чи кому-небудь довіряти. Ви навіть можете використовувати маркер, який ви створили раніше, щоб відстежувати розподіл винагород.

Коли ви розробили свою ідею і забезпечені фонди, що далі? Ви повинні найняти менеджерів, знайти надійного фінансового директора для обробки рахунків, проведення рад директорів і створення купу документов.Ілі ви можете просто залишити все це в контракті Ethereum. Він буде збирати пропозиції від ваших прихильників і представляти їх через повністю прозорий процес голосованія. Одно з багатьох переваг того, що робот управляє вашою організацією, полягає в тому, що він несприйнятливий до будь-якого зовнішнього впливу, оскільки він гарантує виконання тільки того, на що він був запрограмований. І оскільки мережа Ethereum децентралізована, ви зможете надавати послуги зі 100% гарантією безвідмовної роботи.

3.3. Ripple [62] – єдине корпоративне рішення blockchain в світі для глобальних платежів. Розробники Ripple зрозуміли, що, незважаючи на численні досягнення в області технологій, інфраструктура платежів, яку ми продовжуємо використовувати сьогодні, фактично була побудована до того, як Інтернет злетів або навіть розвинувся. З Ripple вони націлені на створення нової платіжної інфраструктури, яка підвищує надійність і швидкість при одночасному зниженні вартості.

Серед різних проблем, пов'язаних з поточною інфраструктурою платежів, Ripple фокусується на вирішенні кількох великих. Ripple виникає у відповідь на вимоги до безпрецедентного, економічному і надійному досвіду.

RippleNet є мережею Ripple, що об'єднує постачальників платежів, банків, корпорацій і цифрових обмінів активами. Він забезпечує єдиний досвід без тертя, який дозволяє відправляти і отримувати гроші по всьому світу. RippleNet забезпечує можливість підключення через різні платіжні мережі з миттєвим розрахунком на вимогу. Завжди є впевненість, оскільки ви можете відстежувати кошти в режимі реального часу. На довершення всього, він має низькі експлуатаційні витрати і низьку вартість ліквідності.

Ripple краще, ніж інші системи blockchain, які ви знайдете, тому що вони побудовані на найбільш передовою технологією blockchain. Це дозволяє масштабованість, а також безпеку при взаємодії з різними мережами. Ripple навіть дає користувачам додатковий доступ до найшвидшим, а також до найбільш масштабованим цифровим активів, які використовуються для платежів в світі, XRP [62].

3.4. Monero [63] пропонує безліч послуг для своїх користувачів. По-перше, ви можете використовувати його для проведення грошових транзакцій в Інтернеті. Це включає покупку товарів або послуг. Крім того, ви можете використовувати його для торгівлі іншими цифровими валютами на біржових ринках, таких як Poloniex, BitSquare і ShapeShift. Нарешті, ви можете використовувати його, і його алгоритм був спеціально розроблений, щоб підтримувати життєздатність розробки ЦП.

Як було сказано раніше, основна перевага цієї криптовалюта полегшує безпрецедентну анонімність користувачів. Він включає в себе ряд вельми складних і складних криптографічних методів, які дозволяють йому забезпечити винятковий рівень конфіденційності [63].

Повторне використання адрес є однією з найпоширеніших проблем конфіденційності, з якими часто стикаються багато цифрові валюти, в тому числі Bitcoin. Всі адреси призначення закриті в блокової ланцюжку, щоб ідентифікувати їх тільки відправника і одержувача. Будь-який аналіз, виконаний на його Blockchain, ніколи не буде розкривати точну адресу призначення, який ви використовуєте для отримання коштів. Швидше за все,

буде видно тільки криптографічний геш адреси призначення, який відрізняється від всіх транзакцій.

3.5. Litecoin [64] – це криптовалюта на основі Blockchain, яка працює аналогічно Ethereum, Bitcoin. Це однорангова інтернет-валюта, яка дозволяє миттєво отримувати майже нульові платежі будь-якій людині в світі. Це децентралізована платіжна мережа, якої не керують жодні центральні органи. Мережа, як і інші блок-ланцюга, забезпечується математикою. Особи можуть контролювати свої власні фінанси, не покладаючись на третіх осіб (банки або інші фінансові установи). Litecoin пропонує всі наступні функції:

Широкодоступні ресурси: Ви можете знайти загальну інформацію, а також список послуг і обмінів, які підтримують Litecoin. Загальна інформація може бути знайдена в Litecoin Wiki, в той час як самі останні мережеві статистики можна знайти в Litecoin Block Explorer Charts. Тим часом, вихідний код для Litecoin Core відкритий і доступний для всіх через GitHub.

Litecoin - це програмне забезпечення з відкритим вихідним кодом. Програмний проект був випущений під ліцензією MIT/X11, що означає, що користувачі мають право запускати, змінювати і копіювати програмне забезпечення і поширювати за своїм вибором змінені версії програмного забезпечення. Litecoin має прозорий процес звільнення, який полегшує незалежну перевірку довічних файлів і їх відповідного вихідного коду.

Blockchain: Блок-ланцюжок Litecoin може обробляти більш високий обсяг транзакції, ніж біткойн. Це пов'язано з тим, що блоковий блок Litecoin має більш часту генерацію блоків. Мережа підтримує більше транзакцій без необхідності зміни програмного забезпечення в майбутньому. В результаті торговці користуються більш швидким часом підтвердження, все ще маючи можливість чекати додаткових підтверджень при продажу більших сум [64].

Шифрування гаманця: Як і всі хороші криптовалюта, свої Litecoin можна зашифрувати. Ви можете захистити свій гаманець, переглянути транзакції і перевірити баланс свого облікового запису, використовуючи власний гаманець

проекту Litecoin. Однак, перш ніж витрачати Litecoin, вам потрібно буде ввести свій пароль.

Майнінг: Майнінг - важлива частина будь-якої ланцюжка. З блокуванням Litecoin Майнер в даний час нагороджуються (станом на червень 2017 року) 25 новими Litecoins за блок. Ця сума зменшується вдвічі приблизно раз в 4 роки (або кожні 840 000 блоків). Очікується, що мережа Litecoin виросте до 84 млн. Litecoin, що в 4 рази більше, ніж у одиниць біткойнов.

3.6. Dash [65] - криптовалюта, націлена на анонімність. Dash захищає ваші особисті дані, роблячи транзакції анонімними за допомогою мережевої технології, розробленої командою Dash, відомої як DarkSend. DarkSend був створений під враженням від проекту CoinJoin, який повинен був анонімізувати Bitcoin-транзакції.

Після ребрендингу Dash став позиціонуватися як анонімна цифрова валюта з відкритим кодом для широкого загалу. Основна увага приділяється питанням безпеки та швидкості проведення транзакцій.

Dash використовує алгоритм гешування, що складається з 11 етапів, а саме blake, bmw, groestl, jh, keccak, skein, luffa, cubehash, shavite, simd і echo, що робить цю монету особливо захищеною.

Є й інші особливості, які визначають принципи роботи криптовалюта. Ось деякі з них:

- Максимальна емісія обмежена;
- Обсяг виробленої криптовалюта буде зменшуватися на 7% щороку, поки не наблизиться до 0 в 2150;
- Дворівнева система Майнер і майстер-вузлів забезпечує безпеку в мережі;
- Згода майстер-вузлів необхідно для підтвердження Майнінг блоку Майнер;
- Нагорода за Майнінг ділиться між майстер-вузлами і Майнер, при цьому майстер-вузли отримують близько 45%;

- Новий блок з'являється приблизно кожні 2 з половиною хвилини;
- Встановлена нагорода за блок - 5 Dash.

Таблиця 1.1

Аналіз програмних модулів захисту інформації

№	Програмні модулі	вання	ЕЦП	Блокчейн системи	Месенджери, де використовується	Криптоалгоритм	Шв. роботи	Зручний інтерфейс	Кросплатформе-ність
1.	MTProto 1.0	+	–	–	Telegtam	SHA-256, AES-256	+	+	+/-
2.	Signal Protocol	+	–	–	Fasebook Messanger, WhatsApp	Curve25519, AES-256, HMAC, SHA256	+/-	+	+
3.	TLS Skype	+	–	–	Skype	AES-256, RSA	+/-	-	+/-
4.	eSign Genie	–	+	–	–	AES-256	+	+/-	+
5.	SignEasy	–	+	–	–	AES-256	+	+	+/-
6.	Adobe EchoSign	–	+	–	–	AES-256	+	+/-	+/-
7.	Signable	–	+	–	–	AES-RSA	+/-	+	+/-
8	Bitcoin	–	+	+	–	SHA-256, ECDSA	+/-	+/-	+
9.	Entherem	–	+	+	–	Ethash, AES 256.	+/-	+	+
10	Ripple	–	+	+	–	RPCA, ECDSA, ripemd160	+	+/-	+
11.	Monero	–	+	+	–	CryptoNightV7, EdDSA	+/-	+	+/-
12.	Litecoin	–	+	+	–	Scrypt, ECDSA	+/-	+/-	+/-
13.	Dash	–	+	+	–	X11, DGW	+/-	+	+/-

У Dash є набір механізмів, які добре визначені для користувачів, щоб відправляти пропозиції в свою мережу для поліпшення послуг. Це корисно для кінцевих користувачів, оскільки вони можуть запросити то, що вони вважають найбільш важливим. У мережі Dash використовується розширене шифрування і дворівнева структура для забезпечення повної безпеки своїх користувачів.

Алгоритми криптографічного захисту інформації використовуються практично у всіх сферах життєдіяльності людини. В результаті проведеного аналізу зроблено висновок, що існуючі програмні модулі, які використовуються, хоч і забезпечують захист даних від несанкціонованого доступу, але мають певні недоліки. У програмних модулях Signal Protocol, SignEasy, Entherem, Bitcoin дуже високий рівень стійкості, але вони потребують покращення, щоб збільшити швидкодію. У MTProto 1.0, Adobe EchoSign, протилежна ситуація, вони забезпечують швидку роботу, проте рівень стійкості при цьому неможливо назвати високим. Можна зробити висновок, що хоча існує багато алгоритмів криптографічного захисту інформації з використанням функцій гешування, розробка нових програмних модулів, які при достатньо високій швидкодії забезпечуватимуть високий рівень стійкості.

1.3. Аналіз існуючих функцій гешування

Функція гешування – це функції, що здійснюють стискання даних змінної довжини в послідовність фіксованого розміру (геш-значення). Отримані геш-значення використовується для перевірки відсутності модифікації даних [43].

Криптографічна функція гешування – це функція гешування, яка є алгоритмом, що приймає довільний блок даних і повертає рядок встановленого розміру, (криптографічне) геш-значення, таке що (випадкові або навмисні) зміни даних (з дуже високою ймовірністю) змінять геш-значення [34].

Дані до кодування часто звуть «повідомлення», а геш-значення іноді називають дайджест повідомлення (англ. message digest) або просто дайджест.

Ідеальна криптографічна функція гешування має чотири основні або значимі властивості [29]:

- легкість обчислення геш-значення для будь-якого повідомлення;
- неможливо утворити повідомлення для заданого геш-значення;
- неможливо змінити повідомлення без зміни геша;
- неможливо знайти два різних повідомлення з тим самим гешем.

Нижче наведені відомі функції гешування.

MD4 (Message Digest 4) [15] – функція гешування, розроблена професором Массачусетського університету Рональдом Рівестом в 1990 році, і вперше описана в RFC 1186 [3]. Для довільного вхідного повідомлення функція генерує 128-розрядне геш-значення, зване дайджестом повідомлення [35].

Рівень безпеки, закладений у MD4, був розрахований на створення досить стійких гібридних систем електронного цифрового підпису, заснованих на MD4 і криптосистемі з відкритим ключем. Уразливості в MD4 були продемонстро-вані у статті Берта ден Бура та Антона Босселарса в 1991 році. Перша колізія була знайдена Гансом Доббертіном в 1996 році.

MD5 (Message Digest 5) [16] – 128-бітний алгоритм гешування, розроблю-ний професором Рональдом Л. Рівестом в 1991 році. Призначений для створен-ня «відбитків» або «дайджестів» повідомлень довільної довжини [22]. Прийшов на зміну MD4, що був недосконалим [13]. Описаний в RFC 1321 [16].

Її використовують для різних схем електронного цифрового підпису, а також необоротного шифрування паролів [43]. У випадку необоротного шифрування паролів основною вимогою до функції гешування стає неможливість визначення вихідного ключа за значенням його геш-коду. В даний час не існує алгоритмів, які дозволили б відновити вихідний або еквівалентний ключ по його геш-коду MD5 або виділити безліч можливих ключів. Таким чином, пошук ключа зводиться до прямого перебору і середня кількість необхідних спроб оцінюється як 280, що робить цей метод практично не придатним [17].

Через можливість існування колізій, перебір може завершитися до знаходження вихідного ключа підбором еквівалентного значення, також методи пошуку колізій дозволяють використовувати їх для підробки електронного підпису [35].

SHA-1 (Secure Hash Algorithm 1) [8] — алгоритм криптографічного гешування. Описано в RFC 3174 [8]. Для вхідного повідомлення довільної довжини (максимум 2^{64} біт) алгоритм генерує 160-бітове геш-значення, відоме також дайджестом повідомлення.

Вважається, що SHA-1 не гарантує достатнього захисту проти атак. В 2005 дослідниками були відкриті методи атаки на SHA-1, які поставили під сумнів тривалість використання цього алгоритму [46]. Вже з 2010 року низка організацій та компаній стали рекомендувати використання SHA-2 замість нього. Microsoft, Google, Apple та Mozilla оголосили, що їхні веб-браузери припинять приймати SSL сертифікати з SHA-1 починаючи з 2017 року [13].

23 лютого 2017 року була доведена практична досяжність обчислення колізій для функції SHA-1 без потреби звертатись до повного перебору.

SHA-2 (Secure Hash Algorithm Version 2) [1] — збірна назва односторонніх функцій гешування SHA-224, SHA-256, SHA-384 і SHA-512. Функції гешування призначені для створення дайджестів повідомлень довільної бітової довжини. Застосовуються в різних додатках або компонентах, пов'язаних із захистом інформації. Функції гешування сімейства SHA-2 побудовані на основі структури Меркла-Демґарда.

SHA-224, SHA-256, SHA-384 і SHA-512 допускаються законом США до використання в деяких урядових програмах, включаючи використання в рамках інших криптографічних алгоритмів та протоколів, для захисту інформації, яка не має грифа секретності. Стандарт також допускає використання SHA-2 приватними та комерційними організаціями.

Як показали дослідження, алгоритми SHA-2 працюють удвічі-втричі повільніше від інших популярних геш-алгоритмів MD5, SHA-1, Tiger та RIPEMD-160.

SHA-3 (Keccak) [66] – алгоритм гешування змінної розрядності, розроблений групою авторів на чолі з Йоаном Дайменом, співавтором Rijndael, автором шифрів MMB, SHARK, Noekeon, SQUARE і BaseKing. 2 жовтня 2012 року Keccak став переможцем конкурсу криптографічних алгоритмів, що проводяться Національним інститутом стандартів і технологій США. 5 серпня 2015 року алгоритм затверджений і опублікований в якості стандарту FIPS202. У програмній реалізації автори заявляють про 12,5 циклах на байт при виконанні на ПК з процесором Intel Core 2. Однак в апаратних реалізаціях Keccak виявився набагато швидше, ніж всі інші фіналісти.

ГОСТ Р 34.11-2012 [6] «Інформаційна технологія. Криптографічний захист інформації. Функція гешування » – чинний российскій криптографічний стандарт, що визначає алгоритм і процедуру обчислення функції гешування. Розроблено Центром захисту інформації та спеціального зв'язку ФСБ Росії за участю ВАТ «ІнфоТеКС» і введений в дію 1 січня 2013 року.

Розмір гешу – 256 або 512 біт; розмір блоку вхідних даних – 512 біт.

Стандарт визначає алгоритм і процедуру обчислення функції гешування для послідовності символів. Цей стандарт розроблений і введений в якості заміни застарілого стандарту ГОСТ Р 34.11-94. Назва функції гешування - «Стрибог», на честь слов'янського божества, - часто використовується замість офіційної назви стандарту.

RIPEMD [12] – це функція гешування [4] була розроблена в 1992 р в рамках європейського проекту RIPE (RACE Integrity Primitives Evaluation) як альтернатива популярній на той час функції гешування MD4 [22].

Фактично, функція стиснення RIPEMD являє собою дві працюючі паралельно функції стиснення MD4 (ліва і права гілки RIPEMD), що відрізняються один від одного аддитивними константами. Уже в 1997 р Х. Доббертін [47] знайшов колізії для урізаною до двох раундів версії RIPEMD, а в 2001 р К. Дебарт і Г. Гілберт [13] показали, що окремо і ліва і права гілки RIPEMD не стійкі до колізій. Для повної версії RIPEMD колізії були побудовані лише в 2004 р і пред'явлені в знаменитій замітці К. Вонг і ін. [39], трохи пізніше

вони опублікували деталі свого алгоритму пошуку колізій і привели оцінку середньої трудомісткості, яка є найкращою на сьогоднішний день.

До поточного моменту розроблені посилені варіанти функції гешування RIPEMD, наприклад RIPEMD-160 [7, 8], які рекомендуються до використання в багатьох міжнародних і національних стандартах, зокрема ISO/IEC 10118-3: 2004. Функції гешування сімейства RIPEMD набули широкого поширення і на практиці, наприклад RIPEMD-160 використовується для генерації ключа шифрування на основі пароля в популярному програмному комплексі створення шифрованих дисків TrueCrypt [9].

N-Hash [41] – криптографічна функція гешування на основі циклічної функції FEAL. Була розроблена в 1990 році телекомунікаційною компанією Nippon Teleg-raph and Telephone. Спочатку, функція N-Hash була призначена для того, щоб вирішити проблему підміни інформації на шляху між двома користувачами телефонного зв'язку і прискорити пошук даних.

Протягом деякого часу алгоритм N-Hash використовувався фірмою Nippon Telegraph and Telephone згідно з цілями даної функції, але через деякий час був розроблений метод днів народження, який з легкістю зламував цей алгоритм. У зв'язку зі зломом відмовилися не тільки від N-Hash, а й майже від всіх функцій, заснованих на блокових шифрах, так як для всіх них характерна одна і та ж проблема: вони легко вразливі методом днів народження.

Snefru [18] – криптографічна функція гешування, запропонована Ральфом Меркле. (Сама назва Snefru, продовжуючи традиції блокових шифрів Khufu і Khafre, також розроблених Ральфом Меркле, являє собою ім'я єгипетського фараона). Функція Snefru перетворює повідомлення довільної довжини в геш довжини m. (зазвичай $m = 128$ або $m = 256$). У березні 1990 року була призначена нагорода в 1000 $ першому, хто зможе зламати двопрохідний варіант Snefru, знайшовши два повідомлення з однаковим геш-кодом (тобто показати, що Snefru не є стійкою до колізій 2-го роду). Аналогічна нагорода була оголошена пізніше за злом чотирьохпрохідного варіанту Snefru.

Використовуючи засоби диференційного аналізу, Елі Біхам і Аді Шамір показали, що двопрохідні функція Snefru 128 - розрядним гешем не є стійкою до колізій 1-го роду і 2-го роду.

Купина [13] — ітеративна криптографічна функція гешування. Прийнята як національний стандарт України ДСТУ 7564:2014 «Інформаційні технології. Криптографічний захист інформації. Функція гешування» [13].

Функція стиснення Купини складається з двох фіксованих 2n-бітних перестановок T⊕ і T+, структура яких запозичена у шифра Калина. Зокрема, використовуються чотири таких самих S-блоків. Результат роботи функції гешування може мати довжину від 8 до 512 біт.

Автори цієї функції гешування запевняють, що диференціальні атаки і rebound-атаки неефективні вже після 4 ітерацій функцій перестановок [6]. У результаті незалежного криптоаналізу вдалося провести атаку тільки на перші 5 раундів; складність знаходження колізії для скороченої до 5 раундів функції Купина-256 складає 2^{120}. Геш-функція Купина реалізована у бібліотеці з відкритим вихідним кодом cppcrypto.

Результати проведеного аналізу відображені в таблиці 1.2.

Таблиця 1.2

Аналіз функцій гешування

№ п/п	Засоби реалізації функцій гешування	Макс розмір повідомлення	Довжина геш-значення (біт)	Шв. шифрування (Мбіт/с)	Розмір блоку	К-сть раундів	Розмір слова	Стійкість до методів криптоаналізу
1.	MD4	$<2^{64}$	128	2,36	512	48	32	-
2.	MD5	$<2^{64}$	128	1,74	512	64	32	-
3.	SHA-1	$<2^{64}$	160	0,75	512	80	32	-/+
4.	SHA-2/256	$<2^{64}$	256	1,85	512	64	32	+
5.	SHA-2/512	$<2^{128}$	1024	1,76	1024	80	64	+

6.	SHA-3 (Keccak)	-	512	0,12	1600	24	64	+
7.	ГОСТ Р 34.11-2012	$<2^{64}$	256	0,11	512	12	32	+
8.	RIPEMD(160)	$<2^{64}$	128	3,60	512	8	32	-
9.	N-геш (12 етапів)	$<2^{64}$	128	1,66	512	8	128	-
10.	Snerfu	$<2^{64}$	128	1,70	512	64	128	-
11.	Купина–512	$<2^{96}$	512	3,25	1024	10	64	+

Функції гешування займають важливе місце у сучасних системах криптографічного захисту інформації. В результаті проведеного аналізу зроблено висновок, що криптостійкі функції гешування SHA-2/256, SHA-2/512, ГОСТ Р 34.11-2012, Купина–512 потребують підвищення швидкодії. Варто відмітити, що функціям гешування з високою швидкодією MD4, MD5, RIPEMD(160), Snerfu більше властиво піддаватися криптоаналізу. Можна зробити висновок, що хоча існує багато криптографічних функцій гешування є актуальною розробка нових функцій, які при достатньо високій швидкодії забезпечуватимуть достатній рівень стійкості.

1.4. Висновки до розділу 1

При написанні першого розділу було розглянуто основні визначення криптографії, класифікацію криптографічних алгоритмів та програмних модулів для захисту інформації. Також розглянуті функції гешування, як основні алгоритми для контролю цілісності інформації, що використовуються програмними модулями.

Програмні модулі криптографічного захисту інформації з використанням функцій гешування мають важливе значення для сучасних систем криптографічного захисту інформації. Внаслідок використання та передачі великих обсягів конфіденційної інформації, можливість перевіряти відсутність несанкціонованих модифікацій є особливо актуальною задачею. Сформовано

основні вимоги для функцій гешування, що можуть використовуватися у криптосистемах, з метою пошуку недоліків та можливих шляхів їх усунення.

Проведено аналіз існуючих програмних модулів криптографічного захисту інформації з використанням функцій гешування, а також сучасних функцій гешування. На основі проведеного аналізу отримано їх порівняльну характеристику. Помічена загальна тенденція втрати швидкості шифрування у криптостійких, на сьогоднішній день, функцій гешування.

РОЗДІЛ 2
МЕТОДИ ПОБУДОВИ ФУНКЦІЙ ГЕШУВАННЯ

Розділ присвячено розробці двох методів побудови функцій гешування – перший метод орієнтований на застосування у системах, для яких критичним є параметр стійкість, а другий – у системах, для яких критичним є параметр швидкість.

2.1. Стійкий метод побудови функцій гешування

Веб-браузери постійно розширюють свої функціональні можливості та надають ко- ристувачам можливість збереження своїх конфіденційних даних, документів, пошти та ін. У зв’язку з цим, забезпечення захищеного доступу до Веб-ресурсів та обмін інформацією між ними займає одне з пріоритетних напрямів в процесі забезпечен- ня захисту інформації та потребує постійного вдосконалення. Одним з найпоширеніших методів захисту є використання криптографічних сертифікатів – цифрових сертифікатів, які забезпечують конфіденційний обмін даними між клієнтом та сервером шляхом шифрування та аутентифікації цифрового сертифікату. Цифровий сертифікат являє собою відкритий ключ користувача, завірений ЕЦП сертифікаційного центру. Однак цифровий сертифікат це не лише відкритий ключ з інформацією, а так званий підпис сервера чи веб-ресурсу, який реалізується використовуючи геш-функції. За останні роки тенденція зростання кількості кібератак збільшується в геометричній прогресії. Так, при збільшення кількості атак, а отже і виявленні нових уразливостей, спостерігається ряд проблем з реалізацією і застосуванням цифрових сертифікатів. Відомі атаки, як DROWN (дозволяє розшифрувати шифротекст без знання закритого ключа), FREAK (дозволяє проникнути у встановлене зашифроване з'єднання та аналізувати трафік), LOGJAM (дозволяє читання та модифікацію даних, що передаються по захищеному каналу зв’язку), завдали великих збитків багатьом власникам веб-ресурсів, в тому числі таким гігантам як Google, Mozilla, Yahoo та ін. та поставили під питання надійність цифрових сер- тифікатів. Тому підвищення надійності цифрових

сертифікатів, як найпоширеніших методів захисту обміну даними через канали зв'язку, є актуальним та потребує вдосконалення.

Сьогодні є багато систем, для яких критичним є параметр стійкості – це цифрові сертифікати, захищені протоколи і системи, що містять інформацію з обмеженим доступом тощо.

Прототипом для першого методу побудови функцій гешування (стійкого) було обрано функцію гешування SHA-2 [36]. Дана функція гешування базується на структурі Меркла-Демґарда [109]. Порівняно з прототипом було змінено:

1. Початкове повідомлення доповнюється розміром цього повідомлення та псевдовипадковою послідовністю salt (розраховується на основі самого повідомлення за допомогою функції F_{Gen}). Слід зауважити, що для кожної нової функції гешування розробленої за допомогою даного методу можна задавати свою унікальну функцію F_{Gen} .
2. Введено параметри l, L, при фіксації яких формується нова структура нової функції гешування (змінюється розрядність операцій).
3. Кількість раундів у функції стиснення F_g визначається за допомогою фіксації параметра R.
4. У функцію стиснення F_g для підвищення не лінійності введено операцію підстановки $S(x)$. Слід зауважити, що для кожної нової функції гешування розробленої за допомогою даного методу можна задати свою унікальну операцію підстановки $S(x)$.
5. У функції стиснення F_g запропоновано свої етапи розбиття блоків на слова та ініціалізації змінних (на основі використання операцій підстановок $S(x)$).
6. У функції стиснення F_g в етапі безпосереднього стиснення запропоновано свій порядок операцій (на основі 6-ти не лінійних функцій, операцій підстановки, додавання за модулем 2 і 2^n,

циклічних і лінійних зсуві), введено дві нових нелінійні функції $JQ(x,y)$, $SH(x,y)$, введено використання операцій підстановок $S(x)$, змінено дві нелінійні функції $Sigma_0(x,y)$, $Sigma_1(x)$.

При зміні/фіксації параметрів l, L, R визначенні операцій F_{Gen} та $S(x)$ можна будувати різноманітні функції гешування.

Опис стійкого методу побудови функцій гешування

Нехай M – вхідне повідомлення, $M \in V_N$, $V_N \in \{0,1\}^N$, $N \in Z_+$, $N < 2^{128}$, H – дайджест повідомлення M, $H \in V_L$, $L = 256 \cdot l$, $l \in Z_+$. Тоді, обчислення H з M виконується у два етапи (рис. 2.1):

Вхідні дані:

M – вхідне повідомлення, $M \in V_N$,

$V_N \in \{0,1\}^N$, $N \in Z_+$, $N < 2^{128}$.

Вихідні дані:

H – дайджест повідомлення M, $H \in V_L$, $L = 256 \cdot l$, $l \in Z_+$.

Етап 1. Етап попередньої обробки:

$M_{rez} = (M, D, salt)$, $M_{rez} \in V_{NN}$,

$NN = N + N_D + N_{salt} = 2 \cdot L \cdot t$, $t \in Z_+$,

$D \in V_{N_D}$, $N_D = 128$,

$salt = F_{Gen}(M)$, $salt \in V_{N_{salt}}$, $N_{salt} = 4L - ((N + N_D) \bmod 2L)$.

Етап 2. Визначення дайджесту повідомлення:

$(m_1, m_2 \ldots m_t) = M_{rez}$, $m_i \in V_{2 \cdot L}$, $i = \overline{1,t}$, $t = NN / 2L$.

$h_i = F_g(h_{i-1}, m_i, i)$, $i = \overline{1,t}$, $h_0 = IV$, $IV \in V_L$, $h_i \in V_L$, $i = \overline{1,t}$,

$H = H(IV, M_{rez}) = h_t$, $H \in V_L$.

Рис. 2.1 – Схема реалізації першого методу побудови функцій гешування

Етап 1. Етап попередньої обробки. На даному етапі вхідне повідомлення M доповнюється додатковою інформацією, таким чином, щоб результуюча довжина повідомлення була кратна $2 \cdot L$:

$$M_{rez} = (M, D, salt),$$

де M_{rez} – результуюче повідомлення, з якого буде обраховуватись H, $M_{rez} \in V_{NN}$, $NN = N + N_D + N_{salt} = 2 \cdot L \cdot t$, $t \in Z_+$, D – довжина повідомлення M, $D \in V_{N_D}$, $N_D = 128$, $salt$ – псевдовипадкова послідовність, що формується на основі M, $salt = F_{Gen}(M)$, $salt \in V_{N_{salt}}$, $N_{salt} = 4L - ((N + N_D) \bmod 2L)$, F_{Gen} – деяка функція генерування псевдовипадкової послідовності на основі M.

Етап 2. Визначення дайджесту повідомлення. Спочатку повідомлення M_{rez}, $M_{rez} \in V_{NN}$, розбивається на t $2 \cdot L$ – бітних блоків:

$$M_{rez} = (m_1, m_2 .... m_t),$$

де $m_i \in V_{2 \cdot L}$, $i = \overline{1,t}$, $t = NN / 2L$.

Далі послідовно обробляється функцію стиснення F_g (див. рис. 2.2) кожен i-й блок повідомлення M_{rez}, проміжний дайджест $(i-1)$-го блоку та індекс i:

$$h_i = F_g(h_{i-1}, m_i, i),\ i = \overline{1,t},$$

де $h_0 = IV$, IV – вектор ініціалізації, $IV \in V_L$, h_i – проміжні значення дайджесту, $h_i \in V_L$, $i = \overline{1,t}$, F_g - функція стиснення.

Результат обробки останнього блоку t і буде дайджестом повідомлення M:

$$H = H(IV, M_{rez}) = h_t,$$

де H – дайджест повідомлення M, $H \in V_L$.

Функція стиснення F_g i-го блоку повідомлення M_{rez} виконується в три етапи: 1) розбиття блоків на слова; 2) ініціалізація змінних; 3) безпосереднє стиснення.

Вхідні дані:

m_i , $m_i \in V_{2 \cdot L}$, $i = \overline{1,t}$,

h_{i-1} , $h_{i-1} \in V_L$, $i = \overline{1,t}$.

Вихідні дані:

h_i , $h_i \in V_L$, $i = \overline{1,t}$.

Етап 1. Розбиття блоків на слова:

$$\left(W_0^i,...,W_{15}^i\right) = m_i,$$

$$W_u = W_{u-16} \oplus Delta_0\left(W_{u-15}\right) \oplus S\left(W_{u-7}\right) \oplus Delta_1\left(W_{u-2}\right) \oplus C_{i \bmod c},$$

$$u = \overline{16,R},\ W_u^i \in V_{L/8},\ u = \overline{16,R}.$$

Етап 2. Ініціалізація змінних:

$$T_z = S\left(h_{i-1}^z\right),\ z = \overline{1,8},\ h_{i-1} = \left(h_{i-1}^1,...,h_{i-1}^8\right),\ h_{i-1}^z \in V_{L/8},\ z = \overline{1,8}.$$

Етап 3. Безпосереднє стиснення:

1. У j-му раунді виконуватися наступні дії, $j = \overline{0,R}$:

$$F_{g_1} = T_8 \oplus Sigma_1\left(T_5\right) \oplus Ch\left(T_5, T_6, T_7\right) + W_j + K_j,$$

$$F_{g_2} = Sigma_0\left(T_1, i\right) \oplus Maj\left(T_1, T_2, T_3\right),$$

$$F_{g3} = JQ\left(T_3, T_6\right) \oplus Maj\left(T_2, T_3\right),$$

$$F_{g_4} = SH\left(T_8, T_7\right) \oplus Sigma_0\left(T_8, T_1\right),$$

$$T_8 = T_7 \oplus F_{g_4},\ T_7 = T_6,\ T_6 = T_5,\ T_5 = T_4 \oplus S\left(F_{g_1}\right),$$

$$T_4 = T_3,\ T_3 = T_2 + F_{g_3},\ T_2 = T_1;\ T_1 = F_{g_1} + S\left(F_{g_2}\right).$$

2. $T_Z = T_Z \oplus h_{i-1}^Z$, $h_{i-1} = \left(h_{i-1}^1,...,h_{i-1}^8\right)$, $h_{i-1}^z \in V_{L/8}$, $z = \overline{1,8}$.

3. $h_i = F_g\left(h_{i-1}, m_i, i\right) = \left(T_1,...,T_z\right)$.

Рис. 2.2 – Функція стиснення першого методу побудови функцій гешування

Етап 1 функції стиснення F_g. Кожен m_i блок повідомлення M_{rez}, $m_i \in V_{2 \cdot L}$, $i = \overline{1,t}$, розкладається на 16 слів:

$$m_i = \left(W_0^i,...,W_{15}^i\right),$$

де $W_j^i \in V_{L/8}$, $j = \overline{0,15}$.

На основі слів W_j^i, $j = \overline{0,15}$, розраховуються слова W_u^i, $W_u^i \in V_{L/8}$, $u = \overline{16,R}$:

$$W_u = W_{u-16} \oplus Delta_0(W_{u-15}) \oplus S(W_{u-7}) \oplus Delta_1(W_{u-2}) \oplus C_{i \bmod c},\ u = \overline{16,R}$$

де $Delta_0(W_u) = Rotr(W_u,1) \oplus Rotr(W_u,8) \oplus SHR(W_u,7)$,

$Delta_1(W_u) = Rotr(W_u,19) \oplus Rotr(W_u,23) \oplus SHR(W_u,6)$,

$Rotr(x,l)$ – правий побітовий циклічний зсув аргументу x на l – біт;

$SHR(x,l)$ – лівий зсув аргументу x на l – біт,

$S(x)$ – деяка операція підстановки,

C_v - наперед визначені константи, $C_v \in V_{L/8}$, $v = \overline{0,c-1}$, $c \in Z_+$,

R – кількість раундів стиснення, $R \in Z_+$.

Етап 2 функції стиснення F_g. Виконується ініціалізація векторів внутрішнього стану T, $T = (T_1,...,T_8)$, $T_z \in V_{L/8}$, $z = \overline{1,8}$:

$$T_z = S\left(h_{i-1}^z\right),\ z = \overline{1,8},$$

де $h_{i-1} = \left(h_{i-1}^1,...,h_{i-1}^8\right)$, h_{i-1} – значення дайджесту $(i-1)$-го блоку, що подається на вхід функції F_g, $h_{i-1}^z \in V_{L/8}$, $z = \overline{1,8}$, $S(x)$ – деяка операція підстановки.

Етап 3 функції стиснення F_g. На даному етапі відбувається безпосереднє стиснення блоку даних $m_i \in V_{2 \cdot L}$, $i = \overline{1,t}$, $t = NN / 2L$, при цьому у кожному j-му раунді ($j = \overline{0,R}$, $R \in Z_+$) буде змінюватись значення векторів внутрішнього стану $T = (T_1,...,T_8)$, $T_z \in V_{L/8}$, $z = \overline{1,8}$, за допомогою їх перемішування із векторами W_j та константами K_j.

У кожному j-му раунді послідовно будуть виконуватися наступні дії, приведені нижче, $j = \overline{0,R}$:

$$F_{g_1} = T_8 \oplus Sigma_1(T_5) \oplus Ch(T_5, T_6, T_7) + W_j + K_j,$$

$$F_{g_2} = Sigma_0(T_1, i) \oplus Maj(T_1, T_2, T_3),$$

$$F_{g3} = JQ(T_3, T_6) \oplus Maj(T_2, T_3),$$

$$F_{g_4} = SH(T_8, T_7) \oplus Sigma_0(T_8, T_1),$$

$$T_8 = T_7 \oplus F_{g_4},$$

$$T_7 = T_6,$$

$$T_6 = T_5,$$

$$T_5 = T_4 \oplus S(F_{g_1}),$$

$$T_4 = T_3,$$

$$T_3 = T_2 + F_{g_3},$$

$$T_2 = T_1;$$

$$T_1 = F_{g_1} + S(F_{g_2}),$$

де T_z – вектори внутрішнього стану, $T_z \in V_{L/8}$, $z = \overline{1,8}$, W_j – слова, на які розбивається кожен m_i блок, K_j – наперед визначені константи, $K_j \in V_{L/8}$,

$Sigma_0(x, y) = Rotr(x, 12 + y) \oplus Rotr(x, 3) \oplus Rotr(x, 25 + y) \oplus y,$

$Sigma_1(x) = Rotr(x, 14) \oplus Rotr(x, 18) \oplus Rotr(x, 3),$

$Ch(x, y, z) = (x + y) \oplus (\bar{x} + z),$

$Maj(x, y, z) = (x + y) \oplus (x + z) \oplus (y + z),$

$JQ(x, y) = (\bar{x} + y) \oplus Rotr(x, 13) \oplus SHR(\bar{y}, 2),$

$SH(x, y) = SHR(x, 3) \oplus Rotr(y, 8) \oplus Rotr(\bar{x}, y),$

$S(x)$ – деяка операція підстановки.

Після виконання останнього R-го раунду значення векторів внутрішнього стану $T = (T_1, ..., T_8)$, $T_z \in V_{L/8}$, $z = \overline{1,8}$, остаточно змінюються наступним чином:

$$T_z = T_z \oplus h_{i-1}^z,$$

де $h_{i-1}-$ попереднє значення дайджесту, що подається на вхід функції F_g $h_{i-1} = \left(h_{i-1}^1, ..., h_{i-1}^8\right)$, $h_{i-1}^z \in V_{L/8}$, $z = \overline{1,8}$.

Виходом функції стиснення F_g буде вектор h_i, $h_i \in V_L$ складений із $T_z \in V_{L/8}$, $z = \overline{1,8}$:

$$h_i = F_g\left(h_{i-1}, m_i, i\right) = \left(T_1, ..., T_z\right).$$

Отже, у даному підрозділі запропоновано новий метод побудови функцій гешування, який базується на структурі Меркла-Демґарда та за рахунок доповнення вхідного повідомлення розміром цього повідомлення та псевдовипадковою послідовністю salt (розраховується на основі вхідного повідомлення), використання у функції стиснення нової послідовності операцій (на основі 6-ти не лінійних функцій, операцій підстановки, додавання за модулем 2 і 2^n, циклічних і лінійних зсувів), дозволить будувати криптостійкі функції гешування (при фіксації параметрів l, L, R, визначенні операцій F_{Gen} та $S(x)$).

2.2. Швидкісний метод побудови функцій гешування

У сучасних ІКС крім параметру стійкості критичним є швидкість у таких часткових випадках наприклад:

- он-лайнові застосунки
- трансакції
- браузери
- протоколи 4G/5G

Прототипом для другого методу побудови функцій гешування (швидкісного) було обрано функцію гешування MD4 [114]. Дана функція

гешування базується на структурі Меркла-Демґарда [116]. Порівняно з прототипом було змінено:

1. Початкове повідомлення доповнюється псевдовипадковою послідовністю salt (розраховується на основі самого повідомлення та його довжини за допомогою функції F_{Gen}). Слід зауважити, що для кожної нової функції гешування розробленої за допомогою даного методу можна задавати свою унікальну функцію F_{Gen} .
2. Введено параметри l, p, L, при фіксації яких формується нова структура нової функції гешування (змінюється розрядність операцій).
3. Кількість раундів у функції стиснення F_g визначається за допомогою фіксації параметра o.
4. У функцію стиснення F_g для підвищення не лінійності та розсіювання даних введено операції підстановки $S(x)$ та операції перестановки $P(x)$. Слід зауважити, що для кожної нової функції гешування розробленої за допомогою даного методу можна задати свою унікальну операцію підстановки $S(x)$ та перестановки $P(x)$.
5. При генерації дайджесту повідомлення окрім проміжних векторів дайджесту h_i використовуються додаткові вектори v_i.
6. У функції стиснення F_g запропоновано свій етап ініціалізації змінних (на основі використання операцій підстановок $S(x)$ та векторів h_i і v_i).
7. У функції стиснення F_g в етапі безпосереднього стиснення запропоновано свій порядок операцій (на основі 4-х не лінійних функцій, операцій підстановки, перестановки, додавання за модулем 2 і 2^n та циклічного зсуву), використовуються 8 векторів внутрішнього стану T_z, введено 4 нові нелінійні функції F_1, F_2, F_3, F_4, введено

використання операцій підстановок $S(x)$ та операцій перестановки $P(x)$.

При зміні/фіксації параметрів l, p, L, o визначенні операцій F_{Gen} та $S(x)$ і $P(x)$ можна будувати різноманітні функції гешування.

Опис швидкісного методу побудови функцій гешування

Нехай M – вхідне повідомлення, $M \in V_N$, $V_N \in \{0,1\}^N$, $N \in Z_+$, $N < 2^{128}$, H – дайджест повідомлення M, $H \in V_L$, $L = 32 \cdot p \cdot l$, $p \in Z_+$, $l \in Z_+$. Тоді, обчислення H з M виконується у два етапи (рис. 2.3):

Вхідні дані:

M – вхідне повідомлення, $M \in V_N$,

$V_N \in \{0,1\}^N$, $N \in Z_+$, $N < 2^{128}$.

Вихідні дані:

H – дайджест повідомлення M, $H \in V_L$, $L = 32 \cdot p \cdot l$, $p \in Z_+$, $l \in Z_+$.

Етап 1. Етап попередньої обробки:

$M_{rez} = (M, salt)$, $M_{rez} \in V_{NN}$,

$NN = N + N_{salt} = 10 \cdot L \cdot t$, $t \in Z_+$,

$salt = F_{Gen}(M, D)$, $salt \in V_{N_{salt}}$,

$N_{salt} = 20L - (N \bmod 10L)$, $D \in V_{N_D}$, $N_D = 128$.

Етап 2. Визначення дайджесту повідомлення:

$(m_1, m_2 \ldots m_t) = M_{rez}$, $m_i \in V_{10 \cdot L}$, $i = \overline{1,t}$, $t = NN / 10L$,

$(h_i, v_i) = F_g(h_{i-1}, v_{i-1}, m_i, i)$, $i = \overline{1,t}$,

$h_0 = IV$, $IV \in V_L$, $v_0 = 0$, $h_i \in V_L$, $i = \overline{1,t}$, $v_i \in V_L$, $i = \overline{1,t}$,

$H = H(IV, 0, M_{rez}) = h_t$, $H \in V_L$.

Рис. 2.3 – Схема реалізації другого методу побудови функцій гешування

Етап 1. Етап попередньої обробки. На даному етапі вхідне повідомлення M доповнюється додатковою інформацією, таким чином, щоб результуюча довжина повідомлення була кратна $10 \cdot L$:

$$M_{rez} = (M, salt),$$

де M_{rez} – результуюче повідомлення, з якого буде обраховуватись H, $M_{rez} \in V_{NN}$, $NN = N + N_{salt} = 10 \cdot L \cdot t$, $t \in Z_+$, $salt$ – псевдовипадкова послідовність, що формується на основі M і довжини D повідомлення M, $salt = F_{Gen}(M, D)$, $salt \in V_{N_{salt}}$, $N_{salt} = 20L - (N \bmod 10L)$, D – довжина повідомлення M, $D \in V_{N_D}$, $N_D = 128$, F_{Gen} – деяка функція генерування псевдовипадкової послідовності на основі M і D.

Етап 2. Визначення дайджесту повідомлення. Спочатку повідомлення M_{rez}, $M_{rez} \in V_{NN}$, розбивається на t $10 \cdot L$ – бітних блоків:

$$M_{rez} = (m_1, m_2 .... m_t),$$

де $m_i \in V_{10 \cdot L}$, $i = \overline{1,t}$, $t = NN / 10L$.

Далі послідовно обробляється функцію стиснення F_g (див. рис. 2.4) кожен i-й блок повідомлення M_{rez}, проміжний дайджест $(i-1)$-го блоку, допоміжний вектор v_{i-1} та індекс i:

$$(h_i, v_i) = F_g(h_{i-1}, v_{i-1}, m_i, i),\ i = \overline{1,t},$$

де $h_0 = IV$, IV – вектор ініціалізації, $IV \in V_L$, $v_0 = 0$, h_i – проміжні значення дайджесту, $h_i \in V_L$, $i = \overline{1,t}$, v_i – значення допоміжного вектору, $v_i \in V_L$, $i = \overline{1,t}$, F_g – функція стиснення.

Результат обробки останнього блоку t і буде дайджестом повідомлення M:

$$H = H(IV, 0, M_{rez}) = h_t,$$

де H - дайджест повідомлення M, $H \in V_L$.

Функція стиснення F_g i-го блоку повідомлення M_{rez} виконується в три етапи: 1) розбиття блоків на слова; 2) ініціалізація змінних; 3) безпосереднє стиснення.

Етап 1 функції стиснення F_g. Кожен m_i блок повідомлення M_{rez}, $m_i \in V_{10 \cdot L}$, $i = \overline{1,t}$, розкладається на 40 слів:

$$m_i = \left(W_1^i, ..., W_{40}^i\right),$$

де $W_j^i \in V_{L/4}$, $j = \overline{1,40}$.

Вхідні дані:

m_i, $m_i \in V_{2 \cdot L}$, $i = \overline{1,t}$,

h_{i-1}, $h_{i-1} \in V_L$, $i = \overline{1,t}$,

v_{i-1}, $v_{i-1} \in V_L$, $i = \overline{1,t}$.

Вихідні дані:

h_i, $h_i \in V_L$, $i = \overline{1,t}$,

v_i, $v_i \in V_L$, $i = \overline{1,t}$.

Етап 1. Розбиття блоків на слова:

$\left(W_1^i, ..., W_{40}^i\right) = m_i$, $W_j^i \in V_{L/4}$, $j = \overline{1,40}$.

Етап 2. Ініціалізація змінних:

$T_j = h_{i-1}^j$, $j = \overline{1,4}$, $h_{i-1} = \left(h_{i-1}^1, ..., h_{i-1}^4\right)$, $h_{i-1}^j \in V_{L/4}$,

$T_y = S\left(v_{i-1}^{y-4}\right)$, $y = \overline{5,8}$, $v_{i-1} = \left(v_{i-1}^1, ..., v_{i-1}^4\right)$, $v_{i-1}^y \in V_{L/4}$.

Етап 3. Безпосереднє стиснення:

1. У j-му раунді виконуватися наступні дії, $j = \overline{0,R} = \overline{0,5 \cdot o}$

$T_1 = F_1\left(T_1, T_2, T_3, T_4, T_5, W_{(8j+0) \bmod 40}, i, j\right)$, $T_2 = F_2\left(T_2, T_3, T_4, T_5, T_6, W_{(8j+1) \bmod 40}, i, j\right)$,

$T_3 = F_3\left(T_3, T_4, T_5, T_6, T_7, W_{(8j+2) \bmod 40}, i, j\right)$, $T_4 = F_4\left(T_4, T_5, T_6, T_7, T_8, W_{(8j+3) \bmod 40}, i, j\right)$,

$T_5 = F_1\left(T_5, T_6, T_7, T_8, T_1, W_{(8j+4) \bmod 40}, i, j\right)$, $T_6 = F_2\left(T_6, T_7, T_8, T_1, T_2, W_{(8j+5) \bmod 40}, i, j\right)$,

$T_7 = F_3\left(T_7, T_8, T_1, T_2, T_3, W_{(8j+6) \bmod 40}, i, j\right)$ $T_8 = F_4\left(T_8, T_1, T_2, T_3, T_4, W_{(8j+7) \bmod 40}, i, j\right)$,

$T_z \in V_{L/4}$, $z = \overline{1,8}$, $W_e \in V_{L/4}$.

2. $\left(h_i, v_i\right) = F_g\left(h_{i-1}, v_{i-1}, m_i, i\right) = \left(\left(T_1, ..., T_4\right), \left(T_5, ..., T_8\right)\right)$.

Рис. 2.4 – Функція стиснення другого методу побудови функцій гешування

Етап 2 функції стиснення F_g. Виконується ініціалізація векторів внутрішнього стану T, $T=(T_1,...,T_8)$, $T_z \in V_{L/4}$, $z=\overline{1,8}$:

$$T_j = h_{i-1}^j,\ j=\overline{1,4},$$
$$T_y = S\left(v_{i-1}^{y-4}\right),\ y=\overline{5,8},$$

де $h_{i-1}=\left(h_{i-1}^1,...,h_{i-1}^4\right)$, h_{i-1} – значення дайджесту $(i-1)$-го блоку, що подається на вхід функції F_g, $h_{i-1}^j \in V_{L/4}$, $j=\overline{1,4}$, $v_{i-1}=\left(v_{i-1}^1,...,v_{i-1}^4\right)$, v_{i-1} – значення допоміжного вектору, що подається на вхід функції F_g, $v_{i-1}^y \in V_{L/4}$, $y=\overline{1,4}$, $S(x)$ – деяка операція підстановки.

Етап 3 функції стиснення F_g. На даному етапі відбувається безпосереднє стиснення блоку даних $m_i \in V_{10\cdot L}$, $i=\overline{1,t}$, $t=NN/10L$, при цьому у кожному j-му раунді ($j=\overline{0,R}$, $R=5\cdot o$, $o \in Z_+$) буде змінюватись значення векторів внутрішнього стану $T=(T_1,...,T_8)$, $T_z \in V_{L/4}$, $z=\overline{1,8}$, за допомогою їх перемішування із векторами W_j.

У кожному j-му раунді послідовно будуть виконуватися наступні дії, приведені нижче, $j=\overline{0,R}=\overline{0,5\cdot o}$:

$$T_1 = F_1\left(T_1,T_2,T_3,T_4,T_5,W_{(8j+0)\bmod 40},i,\ j\right),$$
$$T_2 = F_2\left(T_2,T_3,T_4,T_5,T_6,W_{(8j+1)\bmod 40},i,\ j\right),$$
$$T_3 = F_3\left(T_3,T_4,T_5,T_6,T_7,W_{(8j+2)\bmod 40},i,\ j\right),$$
$$T_4 = F_4\left(T_4,T_5,T_6,T_7,T_8,W_{(8j+3)\bmod 40},i,\ j\right),$$

$$T_5 = F_1\left(T_5, T_6, T_7, T_8, T_1, W_{(8j+4) \bmod 40}, i,\ j\right),$$

$$T_6 = F_2\left(T_6, T_7, T_8, T_1, T_2, W_{(8j+5) \bmod 40}, i,\ j\right),$$

$$T_7 = F_3\left(T_7, T_8, T_1, T_2, T_3, W_{(8j+6) \bmod 40}, i,\ j\right),$$

$$T_8 = F_4\left(T_8, T_1, T_2, T_3, T_4, W_{(8j+7) \bmod 40}, i,\ j\right),$$

де T_z – вектори внутрішнього стану, $T_z \in V_{L/4}$, $z = \overline{1,8}$, W_e – слова, на які розбивається кожен m_i блок, $W_e \in V_{L/4}$,

$$F_1\left(A, B, C, D, E, W, i,\ j\right) = \left(\left(A + \overline{E}\right) \oplus P\left(B\right) \oplus S\left(\overline{C} \oplus W\right) \oplus i\right) <<< \left(D + j\right),$$

$$F_2\left(A, B, C, D, E, W, i,\ j\right) = \left(\left(A + C\right) \oplus P\left(B \oplus S(D + j + \overline{E})\right) \oplus W \oplus i\right) <<< C,$$

$$F_3\left(A, B, C, D, E, W, i,\ j\right) = \left(A \oplus S\left(S\left(B \oplus W \oplus D\right) + j + C + \overline{W}\right) \oplus i\right) <<< E,$$

$$F_1\left(A, B, C, D, E, W, i,\ j\right) = \left(A \oplus S\left(P\left(C + D + \overline{W}\right) \oplus E\right) \oplus i\right) <<< \left(B + j\right),$$

$S\left(x\right)$ − деяка операція підстановки,

$P\left(x\right)$ − деяка операція перестановки.

Виходом функції стиснення F_g буде вектори h_i і v_i, $h_i \in V_L$ складений із $T_j \in V_{L/4}$, $j = \overline{1,4}$, $v_i \in V_L$ складений із $T_y \in V_{L/4}$, $y = \overline{5,8}$:

$$\left(h_i, v_i\right) = F_g\left(h_{i-1},\ v_{i-1}, m_i, i\right) = \left(\left(T_1, ..., T_4\right), \left(T_5, ..., T_8\right)\right).$$

Отже, у даному підрозділі запропоновано новий метод побудови функцій гешування, який базується на структурі Меркла-Демґарда та за рахунок доповнення вхідного повідомлення псевдовипадковою послідовністю salt (розраховується на основі вхідного повідомлення та його розміру), використання у функції стиснення додаткового вектору внутрішнього стану та нової послідовності операцій (на основі 4-х не лінійних функцій, операцій

підстановки, перестановки, додавання за модулем 2 і 2^n та циклічного зсуву), дозволить будувати швидкісні функції гешування (при фіксації параметрів l, p, L, o, визначенні операцій F_{Gen}, $S(x)$ та $P(x)$).

2.3. Висновки до розділу 2

1. Запропоновано перший метод побудови функцій гешування, який базується на структурі Меркла-Демґарда та за рахунок доповнення вхідного повідомлення розміром цього повідомлення та псевдовипадковою послідовністю salt, використання у функції стиснення нової послідовності операцій, дозволив будувати криптостійкі функції гешування. Для побудови нової функції гешування потрібно зафіксувати параметри l, L (для фіксації розрядності операцій), R (для фіксації кількості раундів), визначити операцій F_{Gen} (для формування псевдовипадкової послідовності salt на основі самого повідомлення) та $S(\text{x})$ (для фіксації операції підстановки).

2. Запропоновано другий метод побудови функцій гешування, який базується на структурі Меркла-Демґарда та за рахунок доповнення вхідного повідомлення псевдовипадковою послідовністю salt, використання у функції стиснення додаткового вектору внутрішнього стану та нової послідовності операцій, дозволив будувати швидкісні функції гешування. Для побудови нової функції гешування потрібно зафіксувати параметри l, p, L (для фіксації розрядності операцій), o(для фіксації кількості раундів), визначенні операцій F_{Gen} (для формування псевдовипадкової послідовності salt на основі самого повідомлення та його довжини), $S(x)$ (для фіксації операції підстановки) та $P(x)$ (для фіксації операції перестановки).

РОЗДІЛ 3

МЕТОД ПОБУДОВИ ГЕНЕРАТОРІВ ПСЕВДОВИПАДКОВИХ ПОСЛІДОВНОСТЕЙ ТА МЕТОД КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

У розділі наведено розробку методу побудови генераторів псевдовипадкових послідовностей та методу криптографічного захисту інформації.

3.1. Метод побудови генераторів псевдовипадкових послідовностей

Для генерування дійсно випадкової послідовності за допомогою комп'ютера необхідно використовувати його апаратні засоби. Ці засоби можуть фіксувати наступні явища;

- шум від напівпровідникових приладів;
- біти оцифрованого звуку з мікрофону;
- інтервали між перериванням зовнішніх або внутрішніх пристроїв;
- інтервали між натисканням клавіш;
- температура повітря на апаратних складових.

Також існують генератори випадкових чисел у вигляді плат або зовнішніх пристроїв. Такі генератори виористовуються в сучасних криптосистемах для військових. Вони підключаються до комп’ютерів за допомогою портів вводу-виводу. Основні джерела для них слугують:

- білий Гаусівський шум;
- запис радіоефіру;
- виміри теплових флуктуацій.

Незважаючи на те, що для проектування генераторів псевдовипадкових наборів необхідне комплексе тестування на випадковість, вони в багатьох випадках знаходять використання в комп'ютерних програмах прикладного

характеру. Також можуть бути реалізовані для будь-яких типів комп'ютерних систем.

Прототипом для методу побудови генераторів псевдовипадкових послідовностей було обрано генератор Trivium [111]. Порівняно з прототипом було змінено:

1. Введено параметри n, t, e, k, при фіксації яких формується нова структура генератора псевдовипадкових послідовностей (змінюється розрядність операцій). Всі операції виконуються не над бітами, а над векторами певного розміру (байтами).
2. Для підвищення показників не лінійності введено використання операції підстановки $S(x)$. Слід зауважити, що для кожного нового генератора псевдовипадкових послідовностей можна задати свою унікальну операцію підстановки $S(x)$.
3. Для генерації псевдовипадкових послідовностей використовується вектор внутрішнього стану генератора E_i, ключовий вектор для генерації послідовності K та індекс поточної ітерація генерування i.
4. У функції генерування F_{gen} змінено етап ініціалізація змінних, введено операцію динамічного циклічного зсуву та операцію підстановки.
5. У функції генерування F_{gen} запропоновано використання незалежних функцій F_A, F_B, F_C і F_D, що залежать від значень вектору внутрішнього попереднього етапу генерації, ключового вектора K та індексу поточної ітерація генерування i. Виходом функцій F_A, F_B, F_C і F_D будуть дані необхідного розміру (розмір даних входу і виходу будуть різними). Слід зауважити, що для кожного нового генератора псевдовипадкових послідовностей можна задати свої унікальні функції F_A, F_B, F_C і F_D. По суті дані функції це є окремі байт орієнтовані генератори послідовностей (не обов’язково криптостійкі), які можуть

працювати паралельно, тому для кращої оптимізації швидкість генерації послідовностей в цих функціях має бути приблизно однаковою.

6. Змінений фінальний крок формування послідовності m, змінена послідовність операція, введено використання операції підстановки.

При зміні/фіксації параметрів n, t, e, k визначенні функцій F_A, F_B, F_C і F_D та $S(x)$ можна будувати різноманітні генератори псевдовипадкових послідовностей.

Опис методу побудови генераторів псевдовипадкових послідовностей

Нехай $n, t \in Z_+$, тоді для генерації псевдовипадкової послідовності M, $M \in V_N$, $V_N \in \{0,1\}^N$, довжиною $N = n \cdot t$ біт, потрібно сформувати t послідовностей довжиною n біт кожна:

$$M = (m_1, m_2, ..., m_{t-1}, m_t),\ m_i \in V_n,\ i = \overline{1,t}.$$

Процес генерації кожного m_i, $m_i \in V_n$, $i = \overline{1,t}$, відбувається наступним чином (див. рис. 3.1):

$$m_i, E_i = F_{gen}(E_{i-1}, K, i),\ i = \overline{1,t},$$

де E_i – вектор внутрішнього стану генератора після генерації i-го m_i, $E_i \in V_e$, $e \in Z_+$, $E_0 = IV$, IV – вектор ініціалізації, $IV \in V_e$, K – ключовий вектор для генерації послідовності, $K \in V_k$, $k \in Z_+$, F_{gen} - функція генерації послідовності m_i.

Вхідні дані:

$n, t, e, k \in Z_+$,

$S(x) \to V_n$, $F_A \to V_{n+a}$, $F_B \to V_{n+b}$, $F_C \to V_{n+c}$ і $F_D \to V_{n+d}$,

$K \in V_k$, $k \in Z_+$.

Вихідні дані:

$M = (m_1, m_2, ..., m_{t-1}, m_t)$, $m_i \in V_n$, $i = \overline{1,t}$,

$M \in V_N$, $V_N \in \{0,1\}^N$, $N = n \cdot t$.

Генерація псевдовипадкової послідовності:

$m_i, E_i = F_{gen}(E_{i-1}, K, i)$, $m_i \in V_n$, $i = \overline{1,t}$,

$E_i \in V_e$, $e \in Z_+$, $E_0 = IV$, $IV \in V_e$, $K \in V_k$, $k \in Z_+$.

Рис. 3.1 – Схема реалізації методу генерації псевдовипадкових послідовностей

Функція $F_{gen}(E, K, i)$ виконується в два етапи:

1) ініціалізація змінних;

2) формування послідовності.

Етап 1 функції $F_{gen}(E, K, i)$. На початку виконується обробка вектора внутрішнього стану E, $E \in V_e$:

$$E = S(E) <<< i,$$

де $x <<< y$ – операція правого побітового циклічного зсуву аргументу x на y – біт, $S(x)$ – деяка операція підстановки.

Далі вектор внутрішнього стану генератора E і ключовий вектор K розкладаються на 4 частини:

$$E = (E_a, E_b, E_c, E_d),\ E \in V_e,\ e = a + b + c + d,$$

$$E_a \in V_a,\ E_b \in V_b,\ E_c \in V_c,\ E_d \in V_d,\ a, b, c, d \in Z_+,$$

$$K = (K_a, K_b, K_c, K_d),\ K \in V_k,\ k = a' + b' + c' + d',$$

$$K_a \in V_{a'},\ K_b \in V_{b'},\ K_c \in V_{c'},\ K_d \in V_{d'},\ a', b', c', d' \in Z_+.$$

Вектори E_a, E_b, E_c, E_d і K_a, K_b, K_c, K_d будуть використовуватись в наступному етапі функції $F_{gen}(E, K, i)$.

Вхідні дані:

$E_{i-1} \in V_e$, $K \in V_k$, $i \in Z_+$

Вихідні дані:

$m \in V_n$

Етап 1. Ініціалізація змінних:

$E = S(E) <<< i$, $E \in V_e$

$E = (E_a, E_b, E_c, E_d)$, $E \in V_e$, $e = a + b + c + d$,

$E_a \in V_a$, $E_b \in V_b$, $E_c \in V_c$, $E_d \in V_d$, $a, b, c, d \in Z_+$,

$K = (K_a, K_b, K_c, K_d)$, $K \in V_k$, $k = a' + b' + c' + d'$,

$K_a \in V_{a'}$, $K_b \in V_{b'}$, $K_c \in V_{c'}$, $K_d \in V_{d'}$, $a', b', c', d' \in Z_+$.

Етап 2. Формування послідовностей:

Крок 1. Сформувати вектори A, B, C, D, E_a, E_b, E_c і E_d:

$A, E_a = F_A(E_a, K_a, i)$, $A \in V_n$, $E_a \in V_a$,

$B, E_b = F_B(E_b, K_b, i)$, $B \in V_n$, $E_b \in V_b$,

$C, E_c = F_C(E_c, K_c, i)$, $C \in V_n$, $E_c \in V_c$,

$D, E_d = F_D(E_d, K_d, i)$, $D \in V_n$, $E_d \in V_d$.

Крок 2. Розрахувати E, $E \in V_e$:

$E = (E_b, E_d, E_a, E_c)$.

Крок 3. Сформувати послідовності m, $m \in V_n$:

$AB = A <<< B$, $AB \in V_n$,

$CD = C <<< D$, $CD \in V_n$,

$BC = B + CD$, $BC \in V_n$,

$AD = AB \oplus D$, $AD \in V_n$,

$m = \overline{AD} \oplus S(BC)$, $m \in V_n$.

Рис. 3.2 – Функція генерації послідовностей

Етап 2 функції $F_{gen}(E, K, i)$. На даному етапі виконується формування послідовності m, $m \in V_n$. Для цього використовуються чотири додаткових функції $F_A(E_a, K_a, i)$, $F_B(E_b, K_b, i)$, $F_C(E_c, K_c, i)$ і $F_D(E_d, K_d, i)$, функції F_A, F_B, F_C і F_D – деякі функції, що на вхід приймають значення певного вектора внутрішнього стану і ключового вектора, а на вихід передається послідовність

довжини n біт (ці функції можуть бути побудовані на основі нелінійних регістрів зсуву, блокових і потокових шифрів, геш-функцій тощо).

Тоді, процес генерації послідовності m, $m \in V_n$ та нового значення вектора внутрішнього стану E, $E \in V_e$ в функції $F_{gen}(E, K, i)$ буде таким:

Крок 1. Сформувати додаткові вектори A, B, C і D, та отримати нові значення векторів E_a, E_b, E_c і E_d:

$$A, E_a = F_A(E_a, K_a, i),\ A \in V_n,\ E_a \in V_a,$$
$$B, E_b = F_B(E_b, K_b, i),\ B \in V_n,\ E_b \in V_b,$$
$$C, E_c = F_C(E_c, K_c, i),\ C \in V_n,\ E_c \in V_c,$$
$$D, E_d = F_D(E_d, K_d, i),\ D \in V_n,\ E_d \in V_d.$$

Крок 2. Розрахувати нове значення вектору внутрішнього стану E, $E \in V_e$:

$$E = (E_b, E_d, E_a, E_c).$$

Крок 3. Сформувати послідовності m, $m \in V_n$:

$$AB = A <<< B,\ AB \in V_n,$$
$$CD = C <<< D,\ CD \in V_n,$$
$$BC = B + CD,\ BC \in V_n,$$
$$AD = AB \oplus D,\ AD \in V_n,$$
$$m = \overline{AD} \oplus S(BC),\ m \in V_n,$$

де $\oplus$ і $+$ відповідають відповідно операціям додавання за модулем 2 і 2^n, $S(x)$ – операція підстановки.

Виходом функції $F_{gen}(E, K, i)$ будуть вектори m, $m \in V_n$ і E, $E \in V_e$:

$$(m, E) = F_{gen}(E, K, i).$$

Отже, у даному підрозділі запропоновано новий метод побудови генераторів псевдовипадкових послідовностей, який за рахунок обробки вектора внутрішнього стану та ключового вектору операціями підстановки, циклічного зсуву, складання за модулем 2 і 2^n та 4-ма нелінійними функціями, дозволить будувати ефективні генератори псевдовипадкових послідовностей (при фіксації параметрів n, t, e, k визначенні функцій F_A, F_B, F_C і F_D та $S(x)$.

3.2. Метод криптографічного захисту інформації

Вимоги до рівня захисту інформації почали зростати зі збільшенням кількост атак від зловмисників не тільки на великі технологічні компанії, але і на рядових користувачів. Після викриття Сноуденом фактів прослуховування спецслужбами пересічних громадян США все, хто користуються мобільними мессенджерами, відразу стали приділяти увагу особистій безпеці даних і захисту інформації. Саме тому на ринку з'явився ряд месенджерів (див. аналіз у 1 розділі роботи), які використовують повне або часткове шифрування повідомлень, файлів, фотографій або відео, які ви пересилаєте іншим людям.

Крім власне шифрування, також з'явилася опція самознищення повідомлень і цілих чатів і навіть блокування можливостей для скріншотів. Месенджери Wickr, Wiper і ряд аналогів пропонують саме такі суперможливості.

Прототипом для методу побудови генераторів псевдовипадкових послідовностей було обрано MTProto Mobile Protocol v.1.0 [45]. Порівняно з прототипом було змінено:

1. Змінені вхідні та вихідні дані методу. На вході приймаються і обробляються наступні дані: повідомлення M, інформацію про

ідентифікатор користувача та ідентифікатор сесії S, інформацію про час відправлення і довжину повідомлення ID та порядковий номер повідомлення PD. На виході тільки отримуємо $mHash$ – геш значення DB ($DB = (S, ID, M)$) та $EncP$ – зашифроване повідомлення P.

2. Замість використання геш функції SHA-1 введено використання певної криптостійкої геш функції F_{hash}. Слід зауважити, що у якості F_{hash} може бути використана функція гешування, що побудована на основі одного із методів що наведені у 2 розділі даної роботи.
3. Замість використання блокового шифру AES введено використання функції F_{enc}. Слід зауважити, що у якості F_{enc} може бути використаний певний криптостійкий алгоритм шифрування, побудований на основі блокових, потокових шифрів чи геш функцій тощо (зокрема при виростанні генератора псевдовипадкових послідовностей на основі описаного у 3 розділі методу або функцій гешування на основі методів описаних у 2 розділі).
4. У якості $authKey$, ввдено використання заздалегідь узгодженого секретного ключа користувачів, наприклад за допомогою протоколів асиметричної криптографії.

Опис методу криптографічного захисту інформації

Нехай маємо повідомлення M, $M \in V_m$, $V_m \in \{0,1\}^m$, яке потрібно зашифрувати для передавання.

Тоді для передавання підготовлюється наступне повідомлення (містить окрім повідомлення M, інформацію про ідентифікатор користувача та ідентифікатор сесії (S, $S \in V_s$, $s \in Z_+$), інформацію про час відправлення і довжину повідомлення (ID, $ID \in V_{id}$, $id \in Z_+$) та порядковий номер повідомлення (PD, $PD \in V_{pd}$, $pd \in Z_+$)):

$$P = (S, ID, M, PD),$$

де $P \in V_p$, $p = m + s + id + pd$.

Розглянемо поетапно схему роботи *алгоритму шифрування* (рис. 3.3).

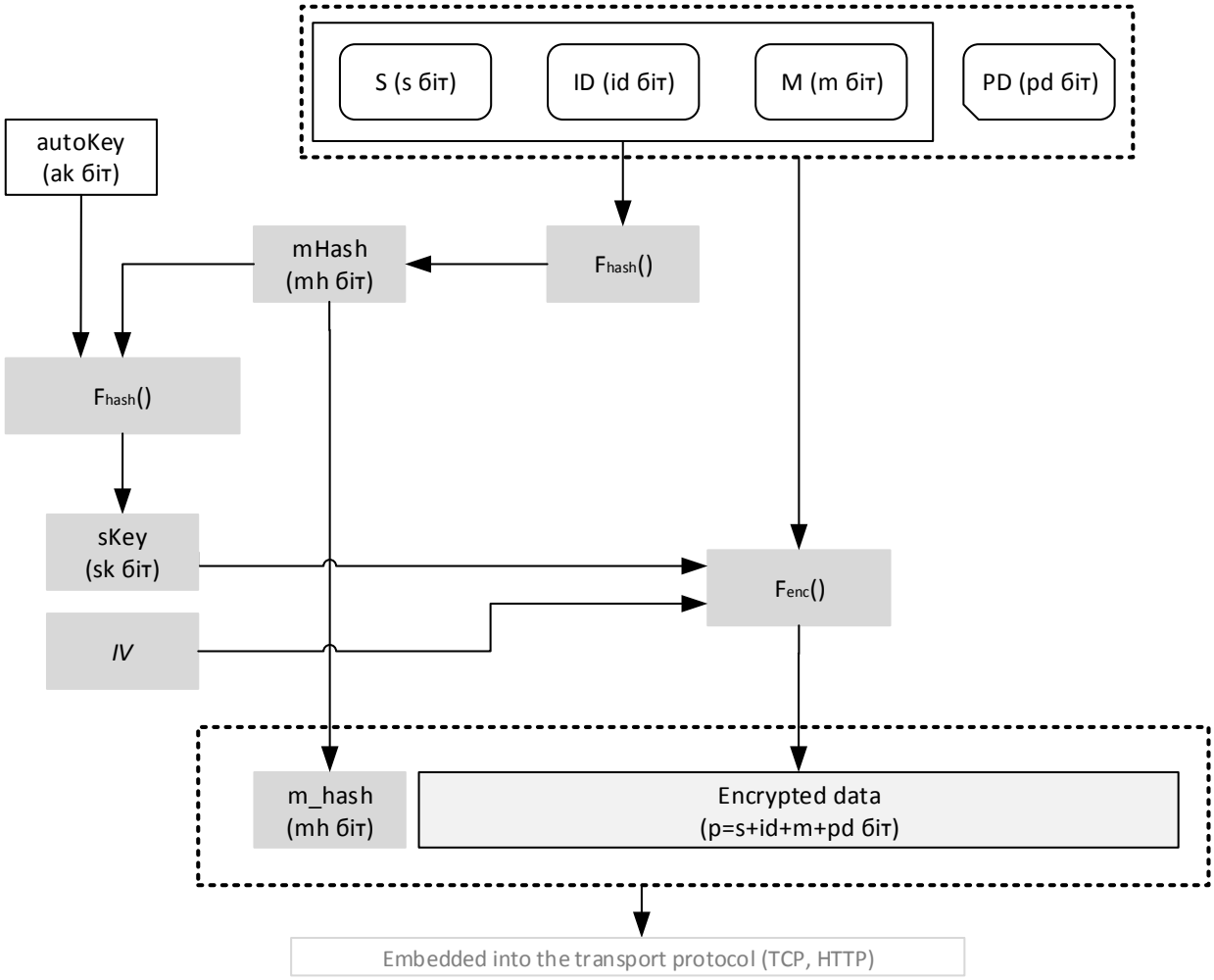

Рис. 3.3. Схема роботи розробленого методу криптографічного захисту інформації

Етап 1. Формування блоку даних *DB* для обрахунку геш значення:

$$DB = (S, ID, M),$$

де $DB \in V_{db}$, $db = m + s + id$, S – ідентифікатор користувача та ідентифікатор сесії, $S \in V_s$, $s \in Z_+$, ID – інформація про час відправлення і довжину повідомлення, $ID \in V_{id}$, $id \in Z_+$, M – саме повідомлення, $M \in V_m$.

Етап 2. Формування геш значення повідомлення DB:

$$mHash = F_{hash}(DB),$$

де $mHash$ – геш значення DB, $DB \in V_{db}$, $mHash \in V_{mh}$, $mh \in Z_+$, $F_{hash}(x)$ – деяка функція гешування.

Етап 3. Формування ключа сеансу $sKey$:

$$sKey = F_{hash}(authKey, mHash),$$

де $sKey$ – сеансовий ключ, $sKey \in V_{sk}$, $sk \in Z_+$, $authKey$ – секретний ключ автентифікації (користувачам потрібно заздалегідь узгодити даний ключ, наприклад за допомогою протоколів асиметричної криптографії), $authKey \in V_{ak}$, $ak \in Z_+$, $F_{hash}(x)$ – деяка функція гешування.

Етап 4. Шифрування за допомого криптографічного алгоритму:

$$EncP = F_{enc}(P, sKey, IV),$$

де $EncP$ – зашифроване повідомлення P, $EncP \in V_p$, P – повідомлення із додатковою інформацією, $P \in V_p$, $sKey$ – сеансовий ключ, $sKey \in V_{sk}$, IV – вектор ініціалізації, $IV \in V_{iv}$, $iv \in Z_+$, $F_{enc}(P, sKey, IV)$ – деяка функція шифрування (може бути побудована на основі блокових і потокових шифрів, геш функцій тощо).

Етап 5. Формування кінцевого повідомлення:

$$EncMes = (mHash, EncP),$$

де $EncMes$ – кінцеве зашифроване повідомлення, $EncMes \in V_{p+mh}$, $mHash$ – геш значення DB, $DB \in V_{db}$, $EncP$ – зашифроване повідомлення P, $EncP \in V_p$

.

Розглянемо поетапно схему роботи *алгоритму розшифрування*.

Етап 1. Розкладаємо отримане зашифроване повідомлення на частини:

$$(mHash, EncP) = EncMes,$$

де $EncMes$ – отримане зашифроване повідомлення, $EncMes \in V_{p+mh}$, $mHash$ – геш значення DB, $DB \in V_{db}$, $EncP$ – зашифроване повідомлення P, $EncP \in V_p$.

Етап 2. Формування ключа сеансу $sKey$:

$$sKey = F_{hash}(authKey, mHash),$$

де $mHash$ – геш значення DB, $DB \in V_{db}$, $mHash \in V_{mh}$, $mh \in Z_+$, $sKey$ – сеансовий ключ, $sKey \in V_{sk}$, $sk \in Z_+$, $authKey$ – секретний ключ автентифікації (користувачам потрібно заздалегідь узгодити даний ключ, наприклад за допомогою протоколів асиметричної криптографії), $authKey \in V_{ak}$, $ak \in Z_+$.

Етап 3. Розшифрування за допомого криптографічного алгоритму:

$$P = F_{dec}(EncP, sKey, IV),$$

де $EncP$ – зашифроване повідомлення P, $EncP \in V_p$, P – повідомлення із додатковою інформацією, $P \in V_p$, $sKey$ – сеансовий ключ, $sKey \in V_{sk}$, IV – вектор ініціалізації, $IV \in V_{iv}$, $iv \in Z_+$, $F_{dec}(EncP, sKey, IV)$ – функція розшифрування обернена до $F_{enc}(P, sKey, IV)$.

Етап 4. Розкладання блоку даних P на частини:

$$(S, ID, M, PD) = P,$$

де $P \in V_p$, $p = m + s + id + pd$, S – ідентифікатор користувача та ідентифікатор сесії, $S \in V_s$, $s \in Z_+$, ID – інформація про час відправлення і довжину повідомлення, $ID \in V_{id}$, $id \in Z_+$, M – саме повідомлення, $M \in V_m$, порядковий номер повідомлення PD, $PD \in V_{pd}$, $pd \in Z_+$).

Етап 5. Формування перевірочного блоку даних DB' для обрахунку геш значення:

$$DB' = (S, ID, M),$$

де $DB' \in V_{db}$, $db = m + s + id$, S – ідентифікатор користувача та ідентифікатор сесії, $S \in V_s$, $s \in Z_+$, ID – інформація про час відправлення і довжину повідомлення, $ID \in V_{id}$, $id \in Z_+$, M – саме повідомлення, $M \in V_m$.

Етап 6. Формування перевірочного геш значення повідомлення DB':

$$mHash' = F_{hash}(DB'),$$

де $mHash'$ – перевірочне геш значення DB', $DB' \in V_{db}$, $mHash \in V_{mh}$, $mh \in Z_+$.

Етап 7. Звіряємо $mHash'$ та $mHash$:

$$mHash' === mHash$$

Якщо $mHash'$ і $mHash$ рівні значить повідомлення не було змінено зловмисником.

Етап 8. Перевіряємо системні дані: отриманий ідентифікатор користувача та ідентифікатор сесії, інформацію про час відправлення і довжину повідомлення, порядковий номер повідомлення. Якщо дані істині та коректні, то повідомлення надіслав легітимний користувач, і можемо прочитати повідомлення M.

Отже, у даному підрозділі запропоновано метод криптографічного захисту інформації, який за рахунок фіксування інформації про ідентифікатор користувача, ідентифікатор сесії, час відправлення, довжину повідомлення та його порядковий номер, а також використання нової процедури формування сеансового ключа для шифрування, дозволяє забезпечити конфіденційність і цілісність даних в інформаційно-комунікаційних системах. Для використання на практиці даного методу потрібно визначитись/зафіксувати функцієї гешування F_{hash} та шифрування F_{enc}.

3.3. Висновки до розділу 3

1. Запропоновано метод побудови генераторів псевдовипадкових послідовностей, який за рахунок обробки вектора внутрішнього стану та ключового вектору операціями підстановки, циклічного зсуву, складання за модулем 2 і 2^n та 4-ма нелінійними функціями, дозволить будувати ефективні генератори псевдовипадкових послідовностей. Для побудови нових генераторів псевдовипадкових послідовностей потрібно зафіксувати параметри n, t, e, k (для фіксації розрядності операцій), визначити функції/генератори F_A, F_B, F_C і F_D (для формування допоміжної гами) та $S(x)$ (для фіксації операції підстановки).

2. Запропоновано метод криптографічного захисту інформації, який за рахунок фіксування інформації про ідентифікатор користувача, ідентифікатор сесії, час відправлення, довжину повідомлення та його порядковий номер, а також використання нової процедури формування сеансового ключа для шифрування, дозволяє забезпечити конфіденційність і цілісність даних в інформаційно-комунікаційних системах. Основним для його використання є вибір криптостійких методів шифрування F_{enc} та гешування F_{hash} та синхронізація секретний ключа *authKey*. В якості F_{enc} та F_{hash} можуть бути використані алгоритми, що побудовані на основі методів описаних у 2 і 3 розділі даної роботи.

РОЗДІЛ 4
ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ РІШЕНЬ

4.1. Методика проведення експериментального дослідження

Метою експериментального дослідження є вивчення якостей оцінюваних об’єктів, перевірка правильності формування та достовірності гіпотез, глибоке вивчення досліджуваної наукової тематики.

Правильний вибір методики експерименту займає особливе значення при його проведенні. Методика експерименту – визначена послідовність процесів, у результаті якої досягається мета дослідження. Цінність експерименту визначає правильність розробки методики його дослідження.

Перший крок у проведенні експериментального дослідження займає складання плану – програми дослідження:

1. Визначення мети та задач експериментів.
2. Вибір вхідних та вихідних параметрів.
3. Визначення послідовності дій.
4. Визначення засобів для використання.
5. Аналіз результатів.

Другий крок, здійснюється після затвердження методики, – це визначення об’єму експериментальних досліджень та необхідних програмних засобів. Третім кроком є безпосереднє проведення експерименту, а заключним кроком – обробка експериментальних даних, систематизація усіх числових даних, перевірка зведення до єдиної системи одиниць, побудова графіків, таблиць, діаграм.

Розроблено методики проведення 3 експериментів.

Експеримент 1. Дослідження статистичних характеристик.

1. Мета та задачі експерименту: дослідити статистичні характеристики запропонованих функцій гешування і генераторів псевдовипадкових послідовностей за методиками NIST STS та DIEHARD.

2. Вибір вхідних та вихідних параметрів:

Вхідні параметри. Розроблені алгоритми Oberih-1, Oberih-2, Oberih-3, Barvinok-1, Barvinok-2, Barvinok-3, Viriy-1, Viriy-2, Viriy-3, генератор BBS, функції гешування SHA-256, SHA-512 та потокові шифри Snow і Trivium. По 10 файлів із псевдовипадковими послідовностями (розміром 100 Мбіт) згенерованими досліджуваними алгоритмами.

Вихідні параметри. Результати проходження за методиками NIST STS та DIEHARD.

3. Послідовність дій:

3.1. За допомогою кожного алгоритму потрібно згенерувати по 10 файлів із псевдовипадковими послідовностями розміром 100 Мбіт.

3.2. Перевірити файли консольними додатками NIST STS та DIEHARD.

3.3. Навести результати досліджень і вивести усереднені результати.

4. Використовувані засоби: Microsoft Visual Studio 2013 - для створення програмних засобів, що дозволяють провести дослідження. Консольні версії додатків NIST STS та DIEHARD – для проведення дослідження.

5. Провести аналіз отриманих результатів.

Експеримент 2. Дослідження швидкісних характеристик.

1. Мета та задачі експерименту: дослідити швидкісні характеристики запропонованих функцій гешування і генераторів псевдовипадкових послідовностей.

2. Вибір вхідних та вихідних параметрів:

Вхідні параметри. Розроблені алгоритми Oberih-1, Oberih-2, Oberih-3, Barvinok-1, Barvinok-2, Barvinok-3, Viriy-1, Viriy-2, Viriy-3, функція гешування SHA-512 та потоковий шифр Snow. Оптимізовані програмні реалізації досліджуваних алгоритмів. Файли розміром 1 МБ, 10 МБ, 100 МБ.

Вихідні параметри. Швидкісні характеристики досліджуваних алгоритмів.

3. Послідовність дій:

3.1. Розробити та оптимізувати програмні реалізації досліджуваних алгоритмів.

3.2. Зашифрувати/визначити дайджест файлів розміром 1 МБ, 10 МБ, 100 МБ при цьому робити заміри часу роботи для визначення швидкодії (для кожного файлу зробити 10 замірів часу їх шифрування).

3.3. Навести результати досліджень і вивести усереднені результати.

4. Використовувані засоби: Microsoft Visual Studio 2013 - для створення програмних засобів, що дозволяють провести дослідження.

5. Провести аналіз отриманих результатів.

Експеримент 3. Дослідження колізійних властивостей.

1. Мета та задачі експерименту: дослідити колізійні властивості запропонованих функцій гешування, згідно методики, що наведена в [84].

2. Вибір вхідних та вихідних параметрів:

Вхідні параметри. Розроблені зменшені версії алгоритмів Oberih-1, Oberih-2, Oberih-3, Barvinok-1, Barvinok-2, Barvinok-3 та зменшена функція гешування SHA-512. Програмний додаток для дослідження колізійних властивостей згідно [35].

Вихідні параметри. Колізійні характеристики досліджуваних алгоритмів.

3. Послідовність дій:

3.1. Розробити зменшені версії досліджуваних алгоритмів.

3.2. Розробити програмний додаток для дослідження колізійних властивостей.

3.3. Провести експериментальні дослідження і навести результати досліджень.

4. Використовувані засоби: Microsoft Visual Studio 2013 - для створення програмного засобу, що дозволить провести дослідження.

5. Провести аналіз отриманих результатів.

Відповідно до описаних методик проведено експериментальні дослідження, опис та обробка яких наведені в 4.3.

4.2. Розробка криптографічних алгоритмів для проведення досліджень

Побудова таблиць підстановок

При розробці криптографічних алгоритмів використовувались одна схема побудови таблиць підстановок (аналогічна [68]). Для генерації таблиці замін на множині V_a ($V_a \in \{0,1\}^a$), необхідно обрахувати для кожного X, $X \in V_a$ при фіксованих C, V, M ($C \in V_a$, $V \in V_a$, M – квадратна не вироджена матриця над полем $GF(2)$ розміром $a \times a$) усі значення таблиці підстановок [69]:

$$S(X) = M \cdot (C / X)^{-1} \oplus V \qquad (4.1)$$

Параметри C, V, M окремо підбиралися для кожної таблиці замін.

У якості поліному, що не приводиться, обрано поліном: $m(x) = x^8 + x^7 + x^5 + x^4 + x + 1$. Також зафіксовано поліном $c(x)$: $c(x) = 3x^7 + 7x^6 + x^5 + 3x^4 + 7x^3 + 4x^2 + 1Dx + 1$.

Функції гешування Oberih

На базі першого методу побудови функцій гешування було розроблено три функції гешування: Oberih-1, Oberih-2 та Oberih-3.

Oberih-1 був спроектований з такими параметрами: $l = 1$, $L = 256 \cdot 1 = 256$, $H \in V_{256}$, у якості операції $S(x)$ – використовується операція виду: $S(x) = \left(s_0(x_1), s_0(x_0)\right)$, де $x_j \in V_{16}$, $j = \overline{0,1}$, s_0 – підстановка на множині V_{16}, $R = 48$.

Підстановка s_0 побудована на основі (4.1) з параметрами, що наведені у табл. 4.1.

Таблиця 4.1

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Oberih-1

M	C	V
{ 903, 3206, 640C, C818, 9031, 2063, 40C6, 818C, 319, 632, C64, 18C8,	6D71	19CF

Oberih-2 був спроектований з такими параметрами $l = 2$, $L = 256 \cdot 2 = 512$, $H \in V_{512}$, у якості операції $S(x)-$ використовується операція виду: $S(x) = \left(s_1\left(x_3\right), ..., s_0\left(x_0\right)\right)$, де $x_j \in V_{16}$, $j = \overline{0,3}$, s_b- підстановка на множині V_{16}, $b = \overline{0,1}$ (почергово використовуються 2 різні таблиці замін), $R = 48$.

Підстановки s_0 і s_1 побудована на основі (4.1) з параметрами, що наведені у табл. 4.2 і 4.3 відповідно.

Таблиця 4.2

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Oberih-2

M	C	V
{ 1906, 320C, 6418, C830, 9061, 20C3, 4186, 830C, 619, C32, 1864, 30C8,	6692	43CC

Таблиця 4.3

Параметри C, V та M для побудови таблиці замін s_1 алгоритму Oberih-2

M	C	V
{ 298E, 531C, A638, 4C71, 98E2, 31C5, 638A, C714, 8E29, 1C53, 38A6, 714C,	27F7	519A

Oberih-3 був спроектований з такими параметрами $l = 4$, $L = 256 \cdot 4 = 1024$, $H \in V_{1024}$, у якості операції $S(x)-$ використовується операція виду: $S(x) = \left(s_3\left(x_7\right), ..., s_0\left(x_0\right)\right)$, де $x_j \in V_{16}$, $j = \overline{0,7}$, s_b- підстановка на множині V_{16}, $b = \overline{0,3}$ (почергово використовуються 4 різні таблиці замін), $R = 48$.

Підстановки s_0, s_1, s_2, s_3 побудована на основі (4.1) з параметрами, що наведені у табл. 4.4, 4.5, 4.6 і 4.7 відповідно.

Таблиця 4.4

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Oberih-3

M	*C*	*V*
{ 2995, 532A, A654, 4CA9, 9952, 32A5, 654A, CA94, 9529, 2A53, 54A6, A94C,	63F5	2593

Таблиця 4.5

Параметри C, V та M для побудови таблиці замін s_1 алгоритму Oberih-3

M	*C*	*V*
{ 32B7, 656E, CADC, 95B9, 2B73, 56E6, ADCC, 5B99, B732, 6E65, DCCA, B995,	7EA6	1386

Таблиця 4.6

Параметри C, V та M для побудови таблиці замін s_2 алгоритму Oberih-3

M	*C*	*V*
{ 32BD, 657A, CAF4, 95E9, 2BD3, 57A6, AF4C, 5E99, BD32, 7A65, F4CA, E995,	A37	677F

Таблиця 4.7

Параметри C, V та M для побудови таблиці замін S_3 алгоритму Oberih-3

M	C	V
{ 3643, 6C86, D90C, B219, 6433, C866, 90CD, 219B, 4336, 866C, CD9, 19B2,	4606	2B9F

Функції гешування Barvinok

На базі другого методу побудови функцій гешування було розроблено три функцій гешування: Barvinok-1, Barvinok-2 та Barvinok-3.

Barvinok-1 був спроектований з такими параметрами $p = 8$, $l = 1$, $L = 32 \cdot p \cdot l = 256$, $H \in V_{256}$, у якості операції $S(x)$ – використовується операція виду: $S(x) = (S_0(x_3), ..., S_0(x_0))$, де $x_j \in V_8$, $j = \overline{0,3}$, S_0 – підстановка на множині V_8, $o = 8$. Перестановка $P(x) = x <<< 16$ - перестановка за допомогою динамічного циклічного зсуву.

Підстановка S_0 побудована на основі (4.1) з параметрами, що наведені у табл. 4.8.

Таблиця 4.8

Параметри C, V та M для побудови таблиці замін S_0 алгоритму Barvinok-1

M	C	V
{ 29, 52, A4, 49, 92, 25, 4a, 94 }	07	D8

Barvinok-2 був спроектований з такими параметрами $p = 8$, $l = 2$, $L = 32 \cdot p \cdot l = 512$, $H \in V_{512}$, у якості операції $S(x)$ – використовується операція виду: $S(x) = (S_1(x_7), ..., S_0(x_0))$, де $x_j \in V_8$, $j = \overline{0,7}$, S_b – підстановка на множині V_8, $b = \overline{0,1}$ (почергово використовуються 2 різні таблиці замін), $o = 8$.

Перестановка $P(x) = x <<< 16$ - перестановка за допомогою динамічного циклічного зсуву.

Підстановки S_0 і S_1 побудована на основі (4.1) з параметрами, що наведені у табл. 4.9 і 4.10 відповідно.

Таблиця 4.9

Параметри C, V та M для побудови таблиці замін S_0 алгоритму Barvinok-2

M	C	V
{ 70, E0, C1, 83, 7, E, 1C, 38 }	A2	44

Таблиця 4.10

Параметри C, V та M для побудови таблиці замін S_1 алгоритму Barvinok-2

M	C	V
{ 3E, 7C, F8, F1, E3, C7, 8F, 1F }	72	63

Barvinok-3 був спроектований з такими параметрами $p = 8$, $l = 4$, $L = 32 \cdot p \cdot l = 1024$, $H \in V_{1024}$, у якості операції $S(x)$ – використовується операція виду: $S(x) = (s_3(x_{15}), ..., s_0(x_0))$, де $x_j \in V_8$, $j = \overline{0,15}$, s_b – підстановка на множині V_8, $b = \overline{0,3}$ (почергово використовуються 4 різні таблиці замін), $o = 8$. Перестановка $P(x) = x <<< 16$ - перестановка за допомогою динамічного циклічного зсуву.

Підстановки S_0, S_1, S_2, S_3 побудована на основі (4.1) з параметрами, що наведені у табл. 4.11, 4.12, 4.13 і 4.14 відповідно

Таблиця 4.11

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Barvinok-3

M	C	V
{ E3, C7, 8F, 1F, 3E, 7C, F8, F1 }	43	9B

Таблиця 4.12

Параметри C, V та M для побудови таблиці замін s_1 алгоритму Barvinok-3

M	C	V
{ E5, CB, 97, 2F, 5E, BC, 79, F2 }	A0	8C

Таблиця 4.13

Параметри C, V та M для побудови таблиці замін s_2 алгоритму Barvinok-3

M	C	V
{ AB, 57, AE, 5D, BA, 75, EA, D5 }	7B	C6

Таблиця 4.14

Параметри C, V та M для побудови таблиці замін s_3 алгоритму Barvinok-3

M	C	V
{ 91, 23, 46, 8C, 19, 32, 64, C8 }	ED	B0

Генератори псевдовипадкових послідовностей Viriy

На базі методу побудови генераторів псевдовипадкових послідовностей було розроблено три генератори: Viriy-1, Viriy-2 та Viriy-3.

Viriy-1 був спроектований з такими параметрами $n=128$, $a=128$, $b=100$, $c=111$, $d=173$, $e=a+b+c+d=512$, $a'=128$, $b'=128$, $c'=128$, $d'=128$, $k=a'+b'+c'+d'=512$. F_A, F_B, F_C і F_D – функції, що побудовані на основі нелінійних регістрів зсуву. У якості операції $S(x)$ – використовується операція виду: $S(x)=\left(s_0\left(x_{31}\right),...,s_0\left(x_0\right)\right)$, де $x_j \in V_{16}$, $j=\overline{0,31}$, s_0 – підстановка на множині V_{16}.

Підстановка s_0 побудована на основі (4.1) з параметрами, що наведені у табл. 4.15.

Таблиця 4.15

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Viriy-1

M	C	V
{ 903, 3206, 640C, C818, 9031, 2063, 40C6, 818C, 319, 632, C64, 18C8,	6D71	19CF

Viriy-2 був спроектований з такими параметрами $n=256$, $a=138$, $b=120$, $c=116$, $d=138$, $e=a+b+c+d=512$, $a'=128$, $b'=128$, $c'=128$, $d'=128$, $k=a'+b'+c'+d'=512$. F_A, F_B, F_C і F_D – функції, що побудовані на основі нелінійних регістрів зсуву. У якості операції $S(x)$ – використовується операція виду: $S(x)=\left(s_7\left(x_{63}\right),...,s_0\left(x_0\right)\right)$, де $x_j \in V_8$, $j=\overline{0,63}$, s_b – підстановка на множині V_8, $b=\overline{0,7}$ (почергово використовуються 8 різні таблиці замін).

Підстановки s_i, $i=\overline{0,7}$ побудована на основі (4.1) з параметрами, що наведені у табл. 4.16-4.23 відповідно.

Таблиця 4.16

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Viriy-2

M	C	V
{ 29, 52, A4, 49, 92, 25, 4a, 94 }	07	D8

Таблиця 4.17

Параметри C, V та M для побудови таблиці замін s_1 алгоритму Viriy-2

M	C	V
{ 70, E0, C1, 83, 7, E, 1C, 38 }	A2	44

Таблиця 4.18

Параметри C, V та M для побудови таблиці замін s_2 алгоритму Viriy-2

M	C	V
{ 3E, 7C, F8, F1, E3, C7, 8F, 1F }	72	63

Таблиця 4.19

Параметри C, V та M для побудови таблиці замін s_3 алгоритму Viriy-2

M	C	V
{ E3, C7, 8F, 1F, 3E, 7C, F8, F1 }	43	9B

Таблиця 4.20

Параметри C, V та M для побудови таблиці замін s_4 алгоритму Viriy-2

M	C	V
{ E5, CB, 97, 2F, 5E, BC, 79, F2 }	A0	8C

Таблиця 4.21

Параметри C, V та M для побудови таблиці замін s_5 алгоритму Viriy-2

M	C	V
{ AB, 57, AE, 5D, BA, 75, EA, D5 }	7B	C6

Таблиця 4.22

Параметри C, V та M для побудови таблиці замін s_6 алгоритму Viriy-2

M	C	V
{ 91, 23, 46, 8C, 19, 32, 64, C8 }	ED	B0

Таблиця 4.23

Параметри C, V та M для побудови таблиці замін s_7 алгоритму Viriy-2

M	C	V
{ F8, F1, E3, C7, 8F, 1F, 3E, 7C}	18	75

Viriy-3 був спроектований з такими параметрами $n=128$, $a=128$, $b=128$, $c=128$, $d=128$, $e=a+b+c+d=512$, $a'=128$, $b'=128$, $c'=128$, $d'=128$, $k=a'+b'+c'+d'=512$. F_A, F_B, F_C і F_D – функції, що побудовані на основі AES-128. У якості операції $S(x)$ – використовується операція виду: $S(x)=\left(s_0\left(x_{63}\right),...,s_0\left(x_0\right)\right)$, де $x_j \in V_8$, $j=\overline{0,63}$, s_0 – підстановка на множині V_8.

Підстановка s_0 побудована на основі (4.1) з параметрами, що наведені у табл. 4.24.

Таблиця 4.24

Параметри C, V та M для побудови таблиці замін s_0 алгоритму Viriy-3

M	C	V
{ 20, 40, 80, 1, 2, 4, 8, 10}	59	6B

4.3. Проведення експериментальних досліджень

Дослідження статистичних характеристик (NIST STS)

Статистичні характеристики розроблених функцій гешування і генераторів псевдовипадкових послідовностей досліджувались за методикою NIST STS. Результати порівнювались із результатами генератора псевдовипадкових послідовностей BBS, функціями гешування SHA-256, SHA-512, потоковими шифрами Snow і Trivium. Зауважимо, що для цього дослідження, на основі розроблених хеш-функцій і функцій SHA-256 та SHA-512, були побудовані генератори псевдовипадкових послідовностей для створення файлів необхідної довжини для статистичних тестів NIST STS.

Для здійснення тестувань були обрані такі параметри:

1. Довжина послідовності, що тестується $n = 10^6$ біт;
2. Кількість послідовностей, що тестується $m = 100$;
3. Рівень значущості $\alpha = 0{,}01$.
4. Кількість тестів $q = 188$

Таким чином, обсяг вибірки, що тестується, склав $N = 10^6 \times 100 = 10^8$ біт, кількість тестів (q) для різних довжин $q = 188$, таким чином, статистичний портрет генератора містить 18800 значень імовірності P.

В ідеальному випадку при $m = 100$ і $\alpha = 0{,}01$ у ході тестування може бути відкинута тільки одна послідовність зі ста, тобто коефіцієнт проходження кожного тесту має складати 99%. Але це занадто жорстке правило. Тому застосовується правило на основі довірчого інтервалу. Нижня межа дорівнює 0,96015.

Для кожного алгоритму генерувалось 10 файлів із псевдовипадковими послідовностями розміром 100 Мбіт, які і досліджувались за методикою NIST STS.

У табл. 4.25 наведено усереднені результати експериментальних досліджень, а на рис. 4.1-4.14 наведено один статистичний портрет для кожного

із досліджуваних алгоритмів (наведено результати дослідження 1-го із 10-ти файлів).

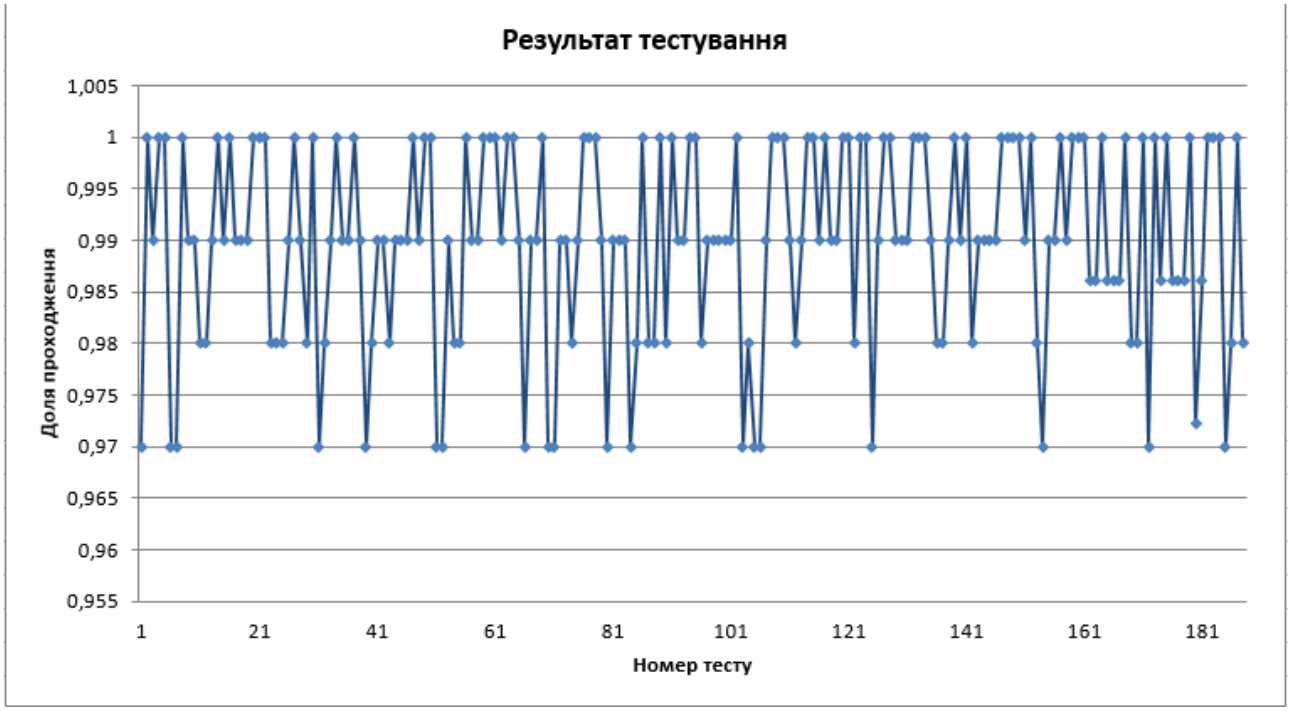

Рис.4.1. Статистичний портрет алгоритму BBS

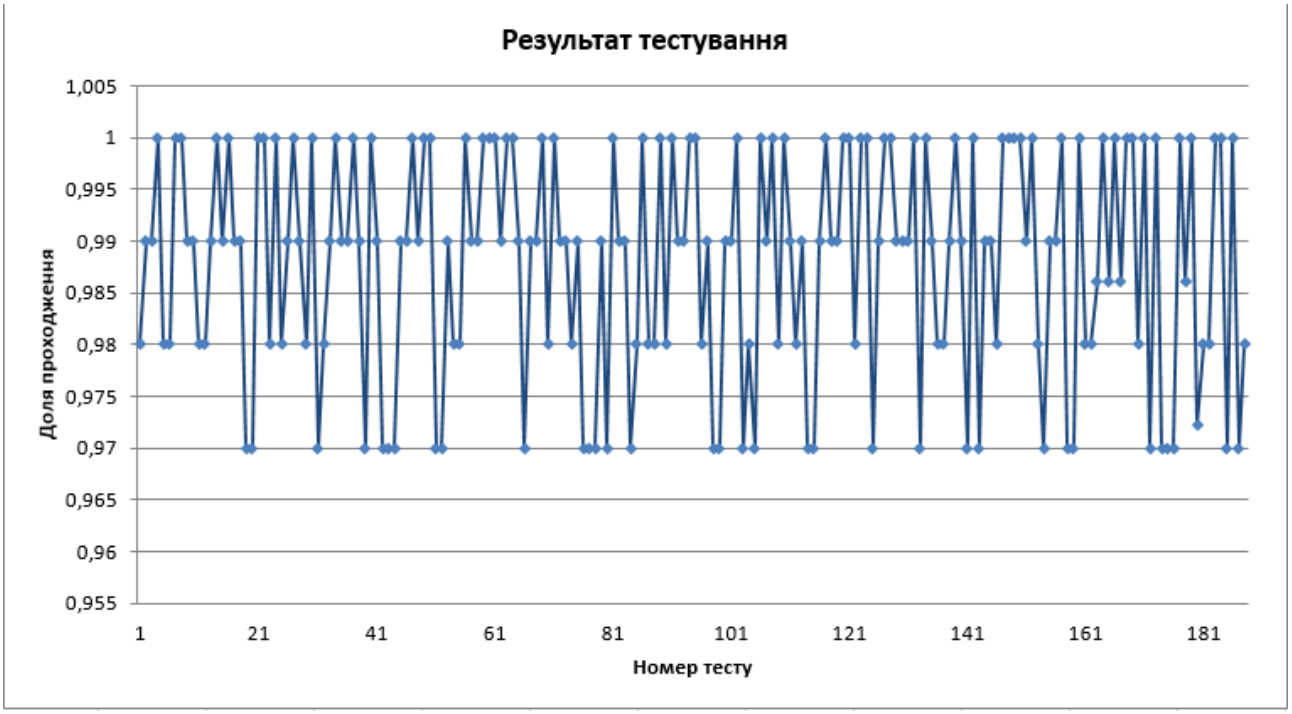

Рис.4.2. Статистичний портрет алгоритму SHA-256

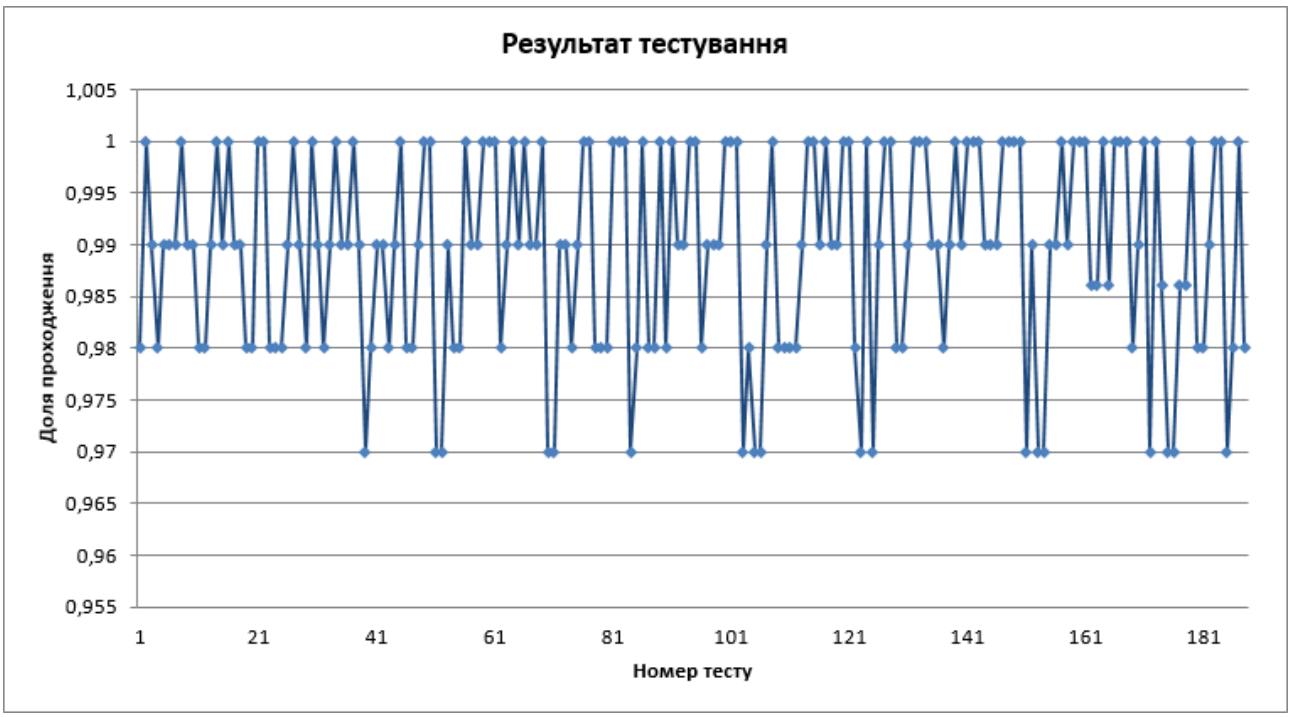

Рис.4.3. Статистичний портрет алгоритму SHA-512

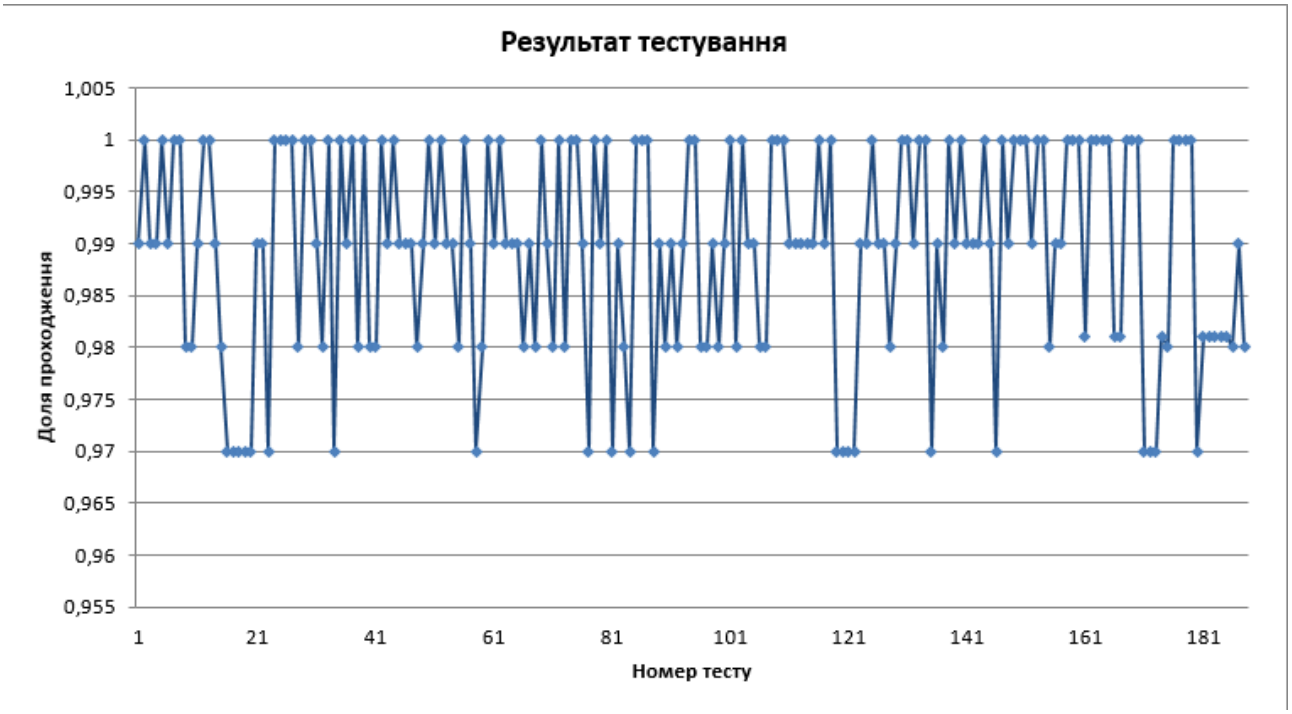

Рис.4.4. Статистичний портрет алгоритму Snow

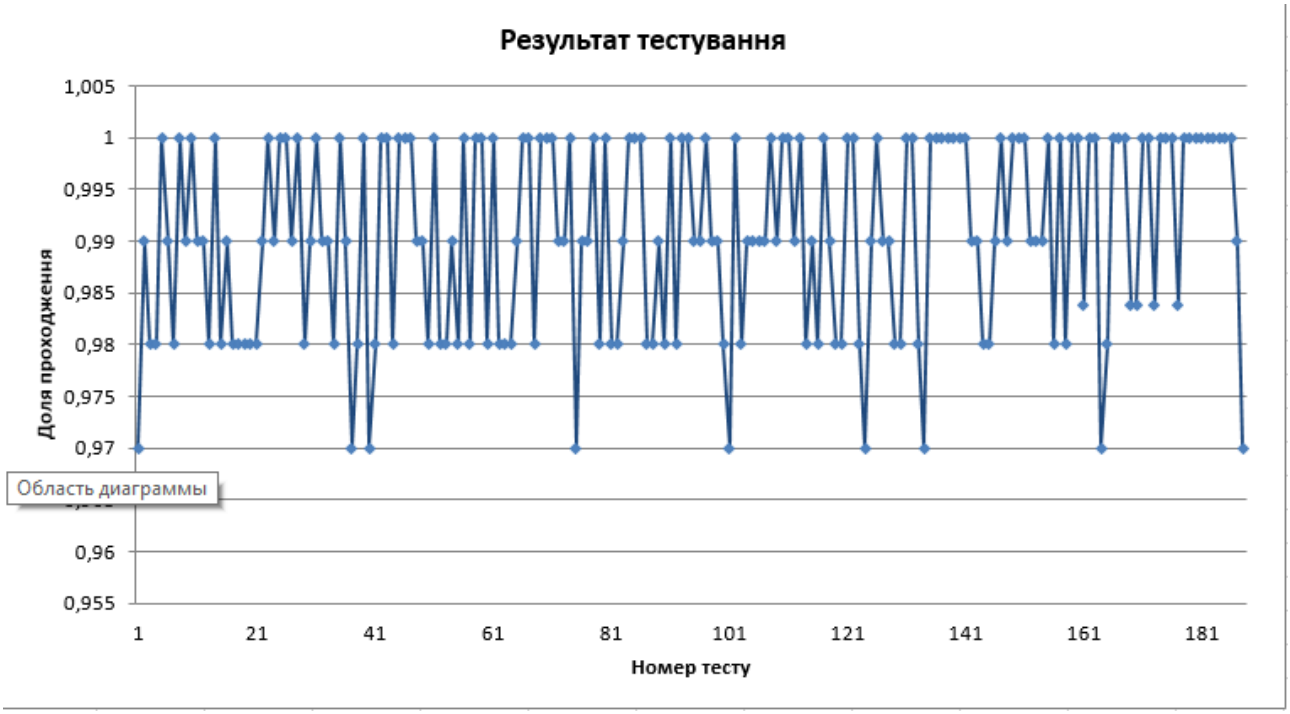

Рис.4.5. Статистичний портрет алгоритму Trivium

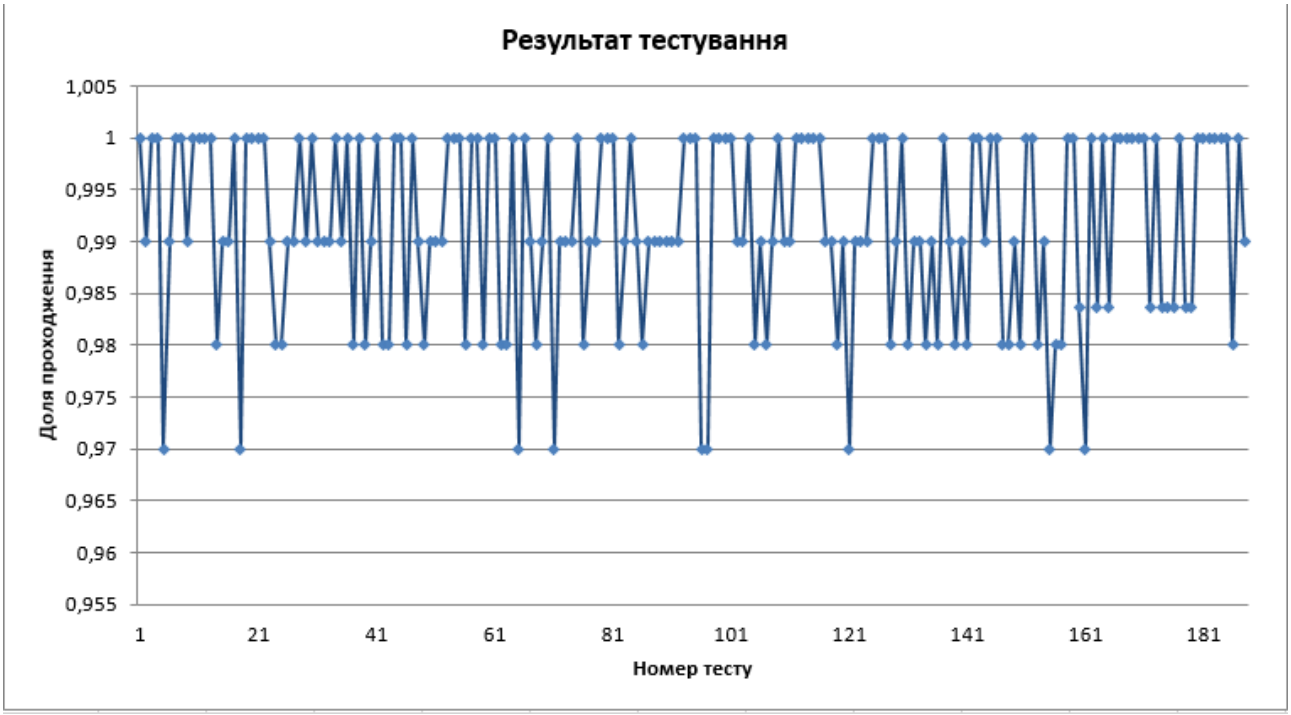

Рис.4.6. Статистичний портрет алгоритму Oberih-1

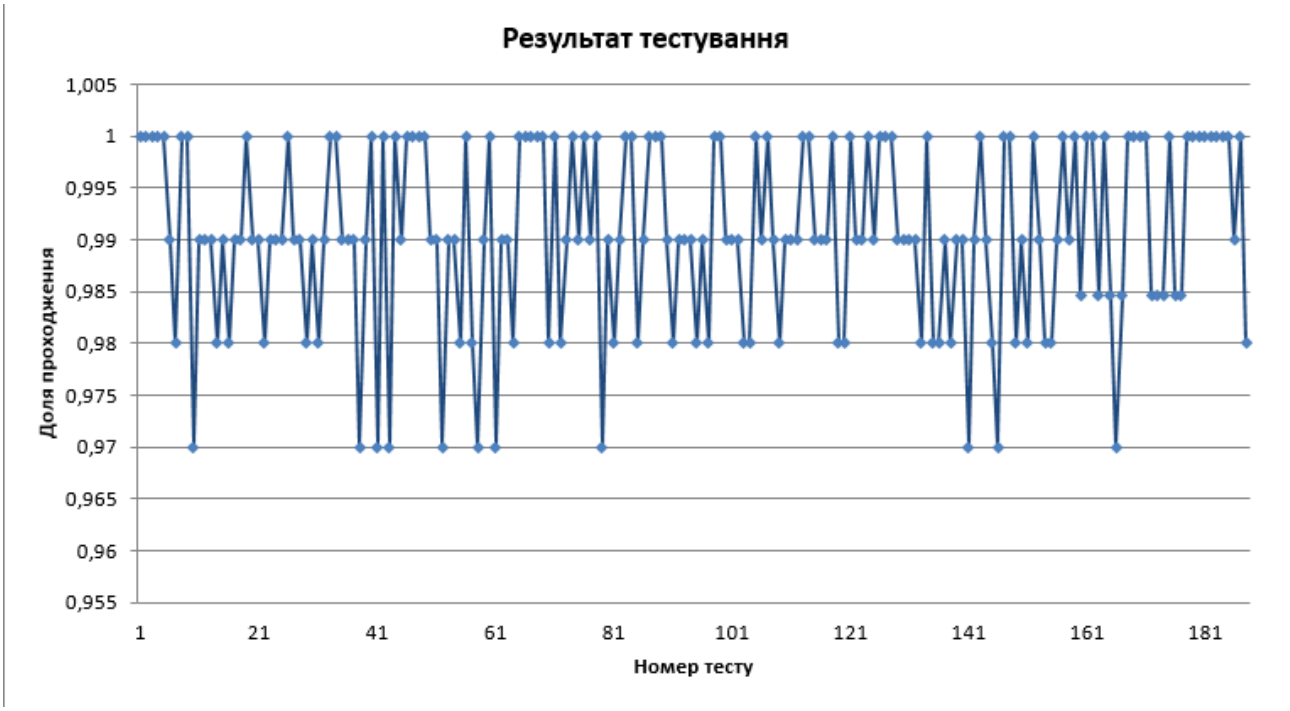

Рис.4.7. Статистичний портрет алгоритму Oberih-2

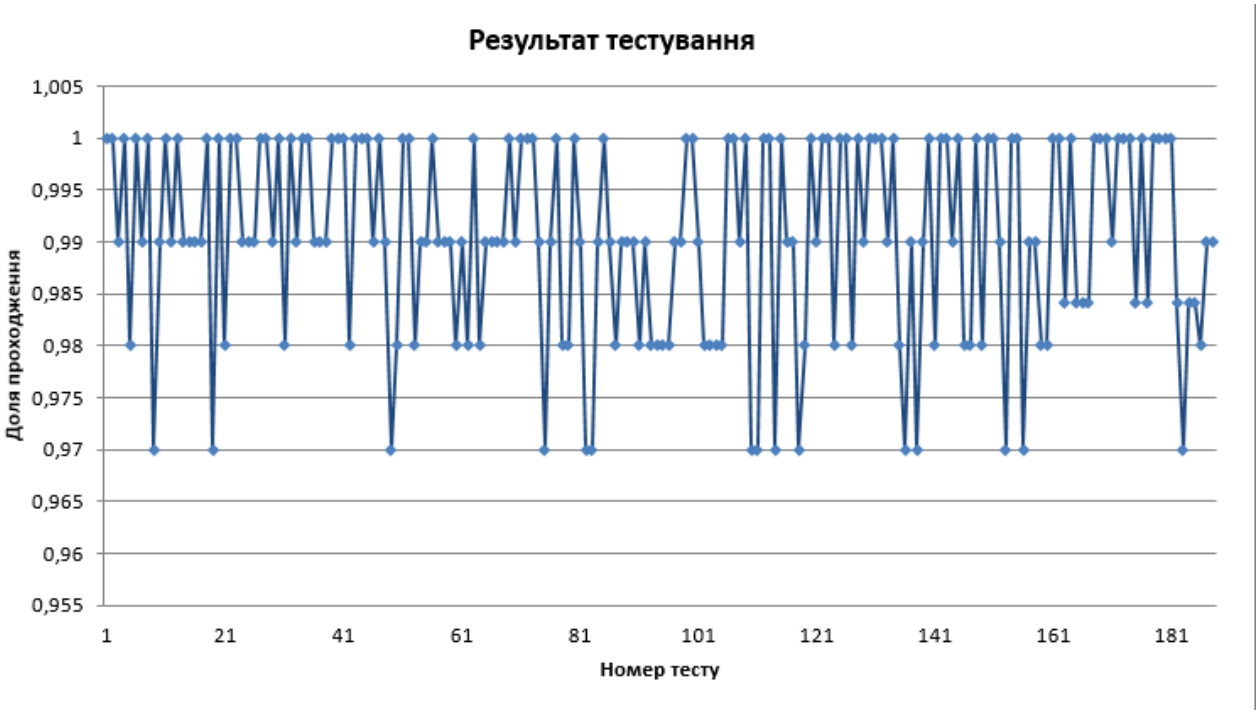

Рис.4.8. Статистичний портрет алгоритму Oberih-3

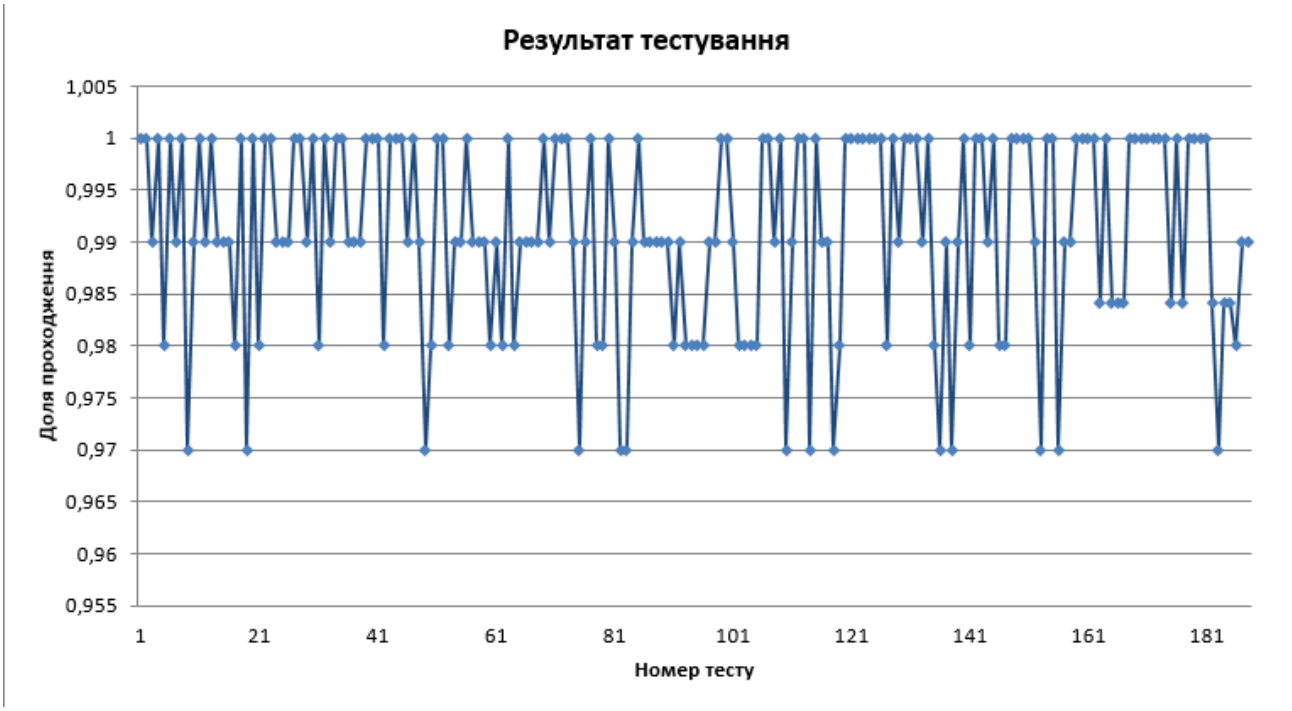

Рис.4.9. Статистичний портрет алгоритму Barvinok-1

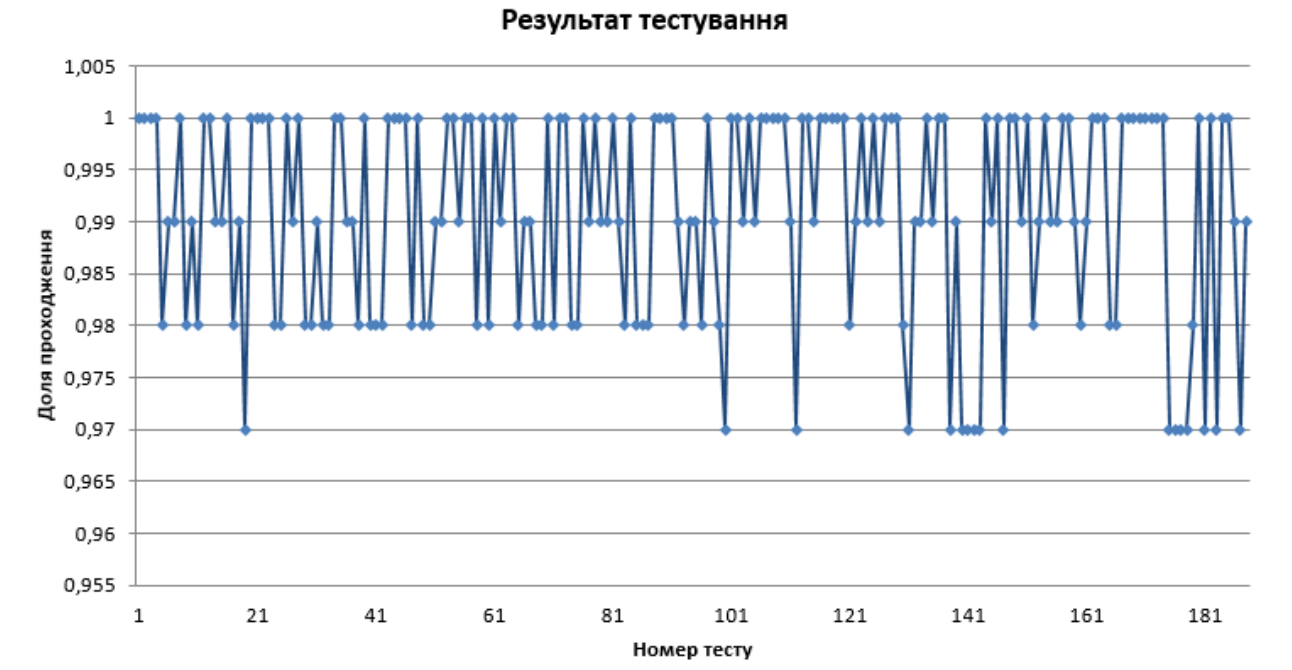

Рис.4.10. Статистичний портрет алгоритму Barvinok-2

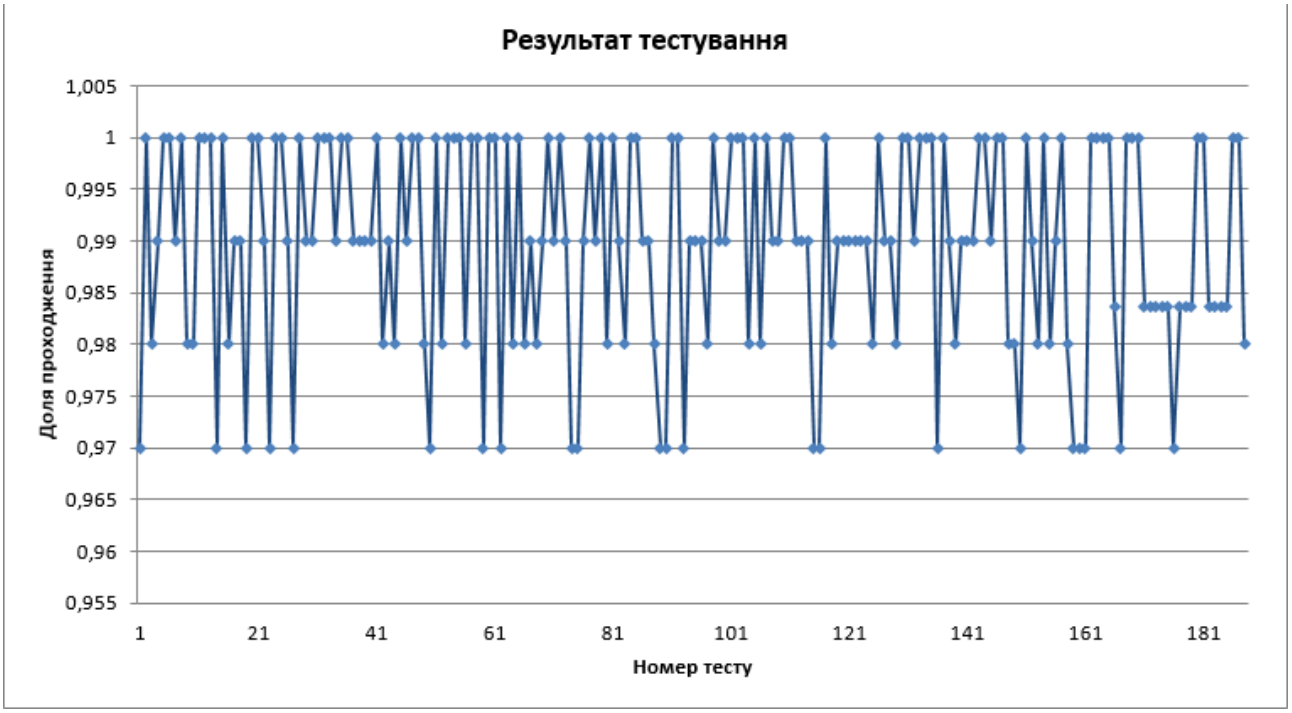

Рис.4.11. Статистичний портрет алгоритму Barvinok-3

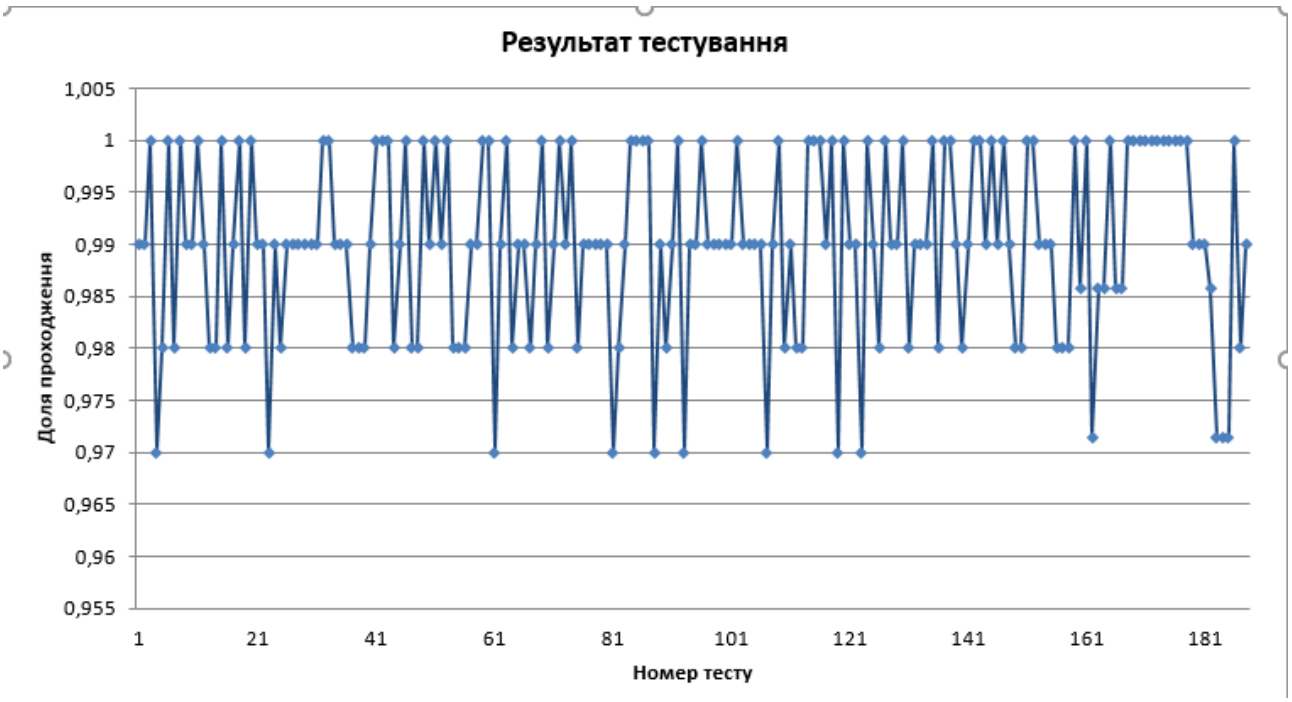

Рис.4.12. Статистичний портрет алгоритму Viriy-1

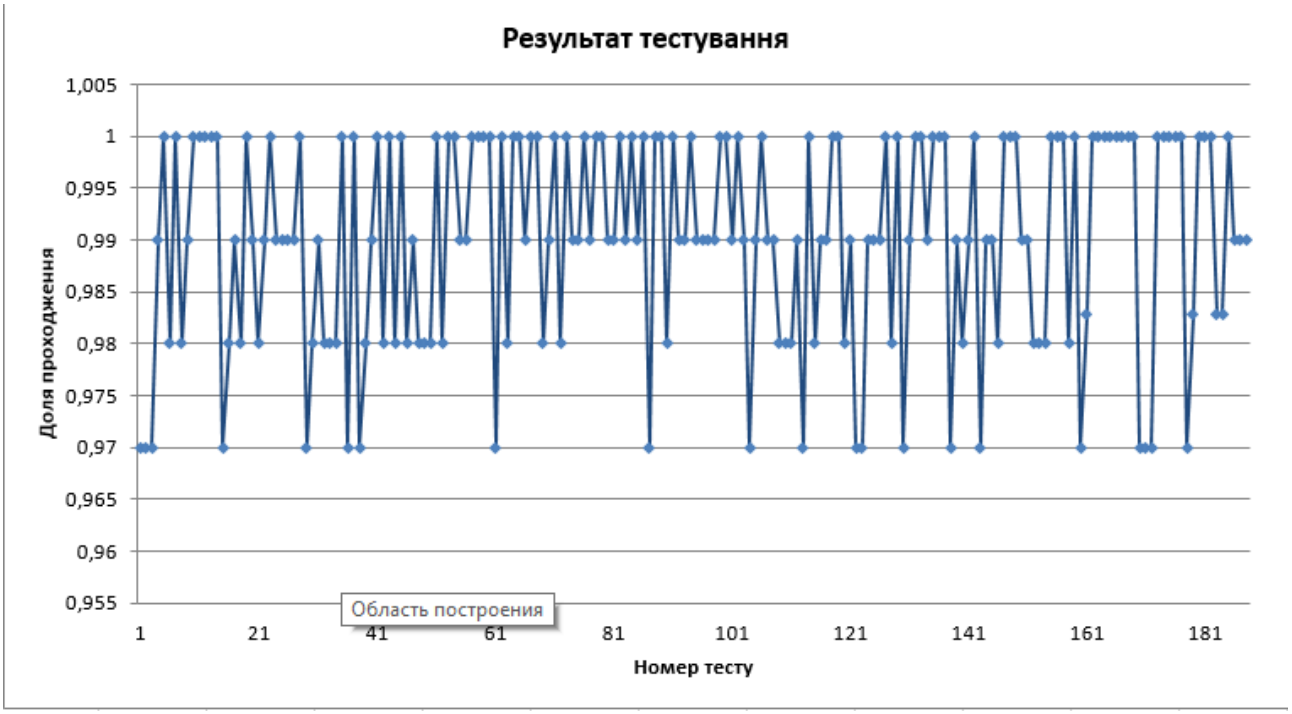

Рис.4.13. Статистичний портрет алгоритму Viriy-2

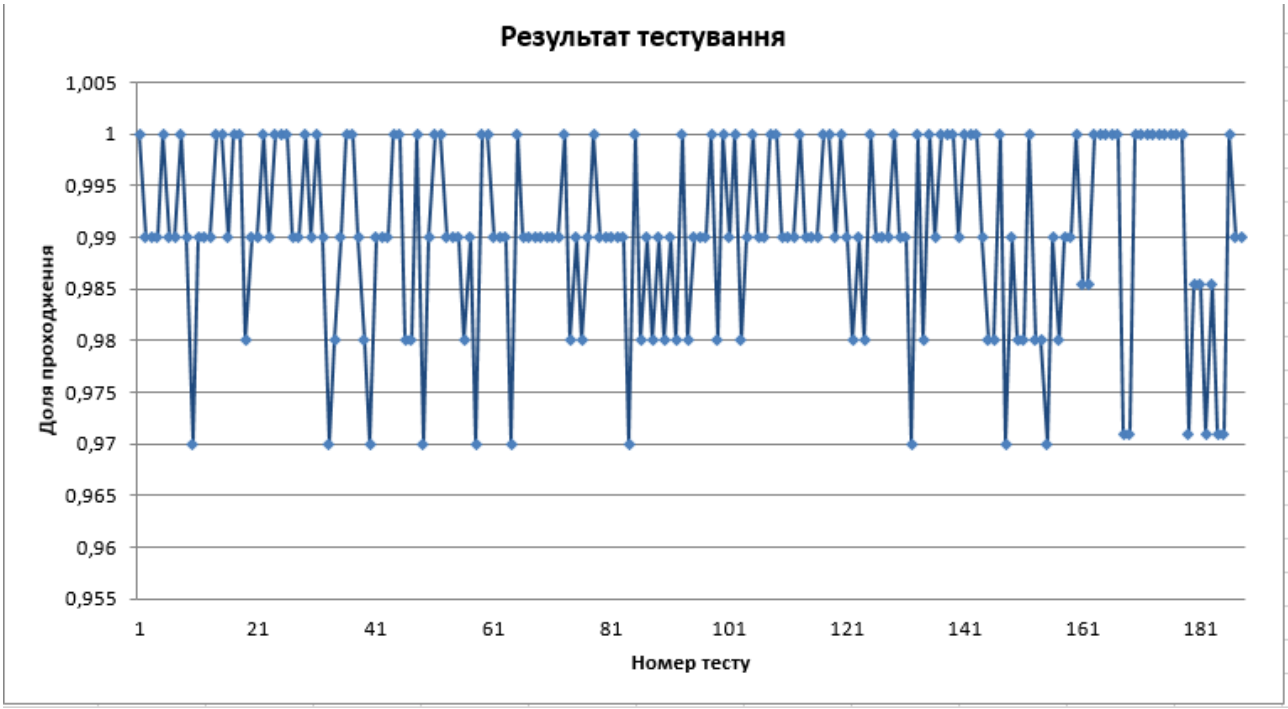

Рис.4.14. Статистичний портрет алгоритму Viriy-3

Таблиця 4.25

Результати дослідження за методикою NIST STS

Генератор ПВП	Кількість тестів, у яких тестування пройшло	
	99% послід.	96% послід.
BBS	133.4 (70.96%)	188 (100%)
SHA-256	132.2 (70.32%)	187.9 (99.94%)
SHA-512	134.3 (71.44%)	188 (100%)
Snow	134.8 (71.70%)	188 (100%)
Trivium	130.1 (69.20%)	187.6 (99.78%)
Oberih-1	133.0 (70.74%)	188 (100%)
Oberih-2	134.6 (71.59%)	188 (100%)
Oberih-3	136.1 (72,39%)	188 (100%)
Barvinok-1	132.3 (70.37%)	187.9 (99.94%)
Barvinok-2	131.7 (70.05%)	188 (100%)
Barvinok-3	135.5 (72,07%)	188 (100%)
Viriy-1	134.1 (71,32%)	188 (100%)
Viriy-2	136.4 (72,55%)	187.8 (99.89%)
Viriy-3	137.3 (73,03%)	188 (100%)

Як видно з результатів, розроблені алгоритми пройшли комплексний контроль за методикою NIST STS, та показали не гірші, а в деяких випадках і кращі результати ніж відомі алгоритми.

Дослідження статистичних характеристик (DIEHARD)

Результати дослідження алгоритмів SHA-256, SHA-512, Snow, Trivium, Oberih-1, Oberih-2, Oberih-3, Barvinok-1, Barvinok-2, Barvinok-3, Viriy-1, Viriy-2 та Viriy-3 за методикою DIEHARD наведені на рис. 4.15.

Test name	ntup	P - value	Assessment
Birthdays	0	0.56138918	Passed
OPERM5	0	0.35408526	Passed
32x32 Binary Rank	0	0.60999066	Passed
6x8 Binary Rank	0	0.9916336	Passed
Bitstream	0	0.69571052	Passed
OPSO	0	0.00110249	Weak
OQSO	0	0.07521348	Passed
DNA	0	0	Failed
Count the 1s (stream)	0	0.79998899	Passed
Count the 1s (byte)	0	0.91337086	Passed
Parking Lot	0	0.45602493	Passed
Minimum Distance (2d Circle)	2	0.03270875	Passed
Minimum Distance (3d Sphere)	3	0.64429711	Passed
Squeeze	0	0.06475503	Passed
Sums	0	0.12714483	Passed
Runs	0	0.60978969	Passed
Runs	0	0.43995034	Passed
Craps	0	0.9779192	Passed
Craps	0	0.9501001	Passed

SHA-256 (a)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

SHA-512 (b)

Test name	ntup	P - value	Assessment
Birthdays	0	0.81512846	Passed
OPERM5	0	0.62228798	Passed
32x32 Binary Rank	0	0.30837818	Passed
6x8 Binary Rank	0	0.46302877	Passed
Bitstream	0	0.9987632	Weak
OPSO	0	0.37088973	Passed
OQSO	0	0.23529634	Passed
DNA	0	0.55171308	Passed
Count the 1s (stream)	0	0.72866517	Passed
Count the 1s (byte)	0	0.99815572	Weak
Parking Lot	0	0.84067258	Passed
Minimum Distance (2d Circle)	2	0.81537116	Passed
Minimum Distance (3d Sphere)	3	0.72015725	Passed
Squeeze	0	0.99930286	Weak
Sums	0	0.00853179	Passed
Runs	0	0.96202615	Passed
Runs	0	0.72277123	Passed
Craps	0	0.08256381	Passed
Craps	0	0.15457383	Passed

Snow (c)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

Trivium (d)

Test name	ntup	P - value	Assessment
Birthdays	0	0.81512846	Passed
OPERM5	0	0.62228798	Passed
32x32 Binary Rank	0	0.30837818	Passed
6x8 Binary Rank	0	0.46302877	Passed
Bitstream	0	0.9987632	Weak
OPSO	0	0.37088973	Passed
OQSO	0	0.23529634	Passed
DNA	0	0.55171308	Passed
Count the 1s (stream)	0	0.72866517	Passed
Count the 1s (byte)	0	0.99815572	Weak
Parking Lot	0	0.84067258	Passed
Minimum Distance (2d Circle)	2	0.81537116	Passed
Minimum Distance (3d Sphere)	3	0.72015725	Passed
Squeeze	0	0.99930286	Weak
Sums	0	0.00853179	Passed
Runs	0	0.96202615	Passed
Runs	0	0.72277123	Passed
Craps	0	0.08256381	Passed
Craps	0	0.15457383	Passed

Oberih-1 (e)

Test name	ntup	P - value	Assessment
Birthdays	0	0.28466726	Passed
OPERM5	0	0.19558726	Passed
32x32 Binary Rank	0	0.09379646	Passed
6x8 Binary Rank	0	0.59616303	Passed
Bitstream	0	0.34938907	Passed
OPSO	0	0.21334675	Passed
OQSO	0	0.57791349	Passed
DNA	0	0	Failed
Count the 1s (stream)	0	0.69604802	Passed
Count the 1s (byte)	0	0.75772348	Passed
Parking Lot	0	0.50331801	Passed
Minimum Distance (2d Circle)	2	0.6131528	Passed
Minimum Distance (3d Sphere)	3	0.03060612	Passed
Squeeze	0	0.63218305	Passed
Sums	0	0.21471818	Passed
Runs	0	0.99739326	Weak
Runs	0	0.24672802	Passed
Craps	0	0.94959445	Passed
Craps	0	0.99104684	Passed

Oberih-2 (f)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

Oberih-3 (g)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

Barvinok-1 (h)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

Barvinok-2 (i)

Test name	ntup	P - value	Assessment
Birthdays	0	0.56138918	Passed
OPERM5	0	0.35408526	Passed
32x32 Binary Rank	0	0.60999066	Passed
6x8 Binary Rank	0	0.9916336	Passed
Bitstream	0	0.69571052	Passed
OPSO	0	0.00110249	Weak
OQSO	0	0.07521348	Passed
DNA	0	0	Failed
Count the 1s (stream)	0	0.79998899	Passed
Count the 1s (byte)	0	0.91337086	Passed
Parking Lot	0	0.45602493	Passed
Minimum Distance (2d Circle)	2	0.03270875	Passed
Minimum Distance (3d Sphere)	3	0.64429711	Passed
Squeeze	0	0.06475503	Passed
Sums	0	0.12714483	Passed
Runs	0	0.60978969	Passed
Runs	0	0.43995034	Passed
Craps	0	0.9779192	Passed
Craps	0	0.9501001	Passed

Barvinok-3 (j)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

Viriy-1 (k)

Test name	ntup	P - value	Assessment
Birthdays	0	0.46632136	Passed
OPERM5	0	0.67512015	Passed
32x32 Binary Rank	0	0.69527276	Passed
6x8 Binary Rank	0	0.10123586	Passed
Bitstream	0	0.98654155	Passed
OPSO	0	0.79734863	Passed
OQSO	0	0.9987444	Weak
DNA	0	0.89704567	Passed
Count the 1s (stream)	0	0.63873474	Passed
Count the 1s (byte)	0	0.94012168	Passed
Parking Lot	0	0.24635381	Passed
Minimum Distance (2d Circle)	2	0.89675937	Passed
Minimum Distance (3d Sphere)	3	0.37439327	Passed
Squeeze	0	0.78310853	Passed
Sums	0	0.81065417	Passed
Runs	0	0.99245722	Passed
Runs	0	0.45528189	Passed
Craps	0	0.75941456	Passed
Craps	0	0.95427721	Passed

Viriy-2 (l)

Test name	ntup	P - value	Assessment
Birthdays	0	0.81512846	Passed
OPERM5	0	0.62228798	Passed
32x32 Binary Rank	0	0.30837818	Passed
6x8 Binary Rank	0	0.46302877	Passed
Bitstream	0	0.9987632	Weak
OPSO	0	0.37088973	Passed
OQSO	0	0.23529634	Passed
DNA	0	0.55171308	Passed
Count the 1s (stream)	0	0.72866517	Passed
Count the 1s (byte)	0	0.99815572	Weak
Parking Lot	0	0.84067258	Passed
Minimum Distance (2d Circle)	2	0.81537116	Passed
Minimum Distance (3d Sphere)	3	0.72015725	Passed
Squeeze	0	0.99930286	Weak
Sums	0	0.00853179	Passed
Runs	0	0.96202615	Passed
Runs	0	0.72277123	Passed
Craps	0	0.08256381	Passed
Craps	0	0.15457383	Passed

Viriy-3 (m)

Рис.4.15. Результати тестування за методикою DIEHARD. генераторів:
SHA-256 (a), SHA-512 (b), Snow (c), Trivium (d),
Oberih-1 (e), Oberih-2 (f), Oberih-3 (g),
Barvinok-1 (h), Barvinok-2 (i), Barvinok-3 (j),
Viriy-1 (k), Viriy-2 (l) та Viriy-3 (m)

Дослідження швидкісних характеристик

Для дослідження розроблених функцій гешування Oberih-1, Oberih-2, Oberih-3, Barvinok-1, Barvinok-2, Barvinok-3 – дані алгоритми були програмно реалізовані на мові програмування C++. Результати порівнювались із функцією гешування SHA-512. Для дослідження були випадковим чином обрані кілька файлів різного розміру (файли розміром 1 МБ, 10 МБ, 100 МБ), для кожного з яких визначався його дайджест досліджуваним алгоритмом, при цьому замірявся час обробки. Кожен файл оброблявся 10 разів кожним алгоритмом.

Дослідження проводилися в однакових умовах на Intel Core i3-3220 3.3GHz.

Усереднені результати досліджень наведено у табл. 4.26.

Таблиця 4.26

Результати дослідження швидкісних характеристик функцій гешування

Функцій гешування	Файл 1, 1 МБ		Файл 2, 10 МБ		Файл 3, 100 МБ	
	t ,с	*v* ,МБ/с	*t* ,с	*v* ,МБ/с	*t* ,с	*v* ,МБ/с
SHA-512	0.015	66.67	0.145	68.96	1.38	72,46
Oberih-1	0.012	83.33	0.114	87.72	1.07	93.45
Oberih-2	0.011	90.91	0.109	91.74	1.01	99.01
Oberih-3	0.013	76.92	0.121	82.64	1.13	88.50
Barvinok-1	0.010	100.00	0.096	104.16	0.90	111.11
Barvinok-2	0.011	90.91	0.105	95.23	0.96	104.17
Barvinok-3	0.013	76.92	0.117	85.47	1.12	89.29

Згідно з результатами дослідження швидкість обробки розробленими функціями гешування кращі ніж у функції SHA-512, так алголитми Oberih швидші за SHA-512 у 1.15-1.36 рази, а алгоритми Barvinok – у 1.16-1.53 рази.

Для дослідження розроблених генераторів псевдовипадкових послідовностей Viriy-1, Viriy-2, Viriy-3 – дані алгоритми були програмно реалізовані на мові програмування C++. Результати порівнювались із генератором псевдовипадкових послідовностей Snow. Для дослідження були випадковим чином обрані кілька файлів різного розміру (файли розміром 1 МБ, 10 МБ, 100 МБ), кожен з яких зашифровувався досліджуваними алгоритмами, при цьому замірявся час обробки. Кожен файл оброблявся 10 разів кожним алгоритмом.

Дослідження проводилися в однакових умовах на Intel Core i3-3220 3.3GHz.

Усереднені результати досліджень наведено у табл. 4.27.

Таблиця 4.27

Результати дослідження швидкісних характеристик генераторів ПВП

Функцій гешування	Файл 1, 1 МБ		Файл 2, 10 МБ		Файл 3, 100 МБ	
	t ,с	v ,МБ/с	t ,с	v ,МБ/с	t ,с	v ,МБ/с
Snow	0.011	90.91	0.107	93.46	1.01	99.01
Viriy-1	0.009	111.11	0.091	109.89	0.88	113.64
Viriy-2	0.010	100.00	0.098	102.04	0.92	108.70
Viriy-3	0.014	71.43	0.112	89.29	0.99	101.01

Згідно з результатами дослідження швидкість алгоритму Viriy краща за Snow у 1.02-1.22 рази (за виключенням двох результатів алгоритму Viriy-3).

Дослідження колізійних властивостей

Для дослідження колізійних властивостей розроблених функцій гешування спочатку були розроблені зменшені версії функцій гешування, а далі згідно методики проводилось оцінювання даних алгоритмів.

Для цього уводяться статистичні показники, що характеризують колізійні властивості функцій гешування, що й дозволяють, використовуючи методи теорії імовірності й математичної статистики, одержувати оцінки із заданими довірчим інтервалом і необхідною точністю.

Перший показник $n_1\left(x_1, x_2\right)$ характеризує кількість правил гешування, при яких для заданих $x_1, x_2 \in A$, $x_1 \neq x_2$ виконується рівність (4.2), тобто кількість ключів, при яких існує колізія для двох вхідних послідовностей x_1 і x_2.

$$n_1\left(x_1, x_2\right) = \left|\left\{h \in H : h\left(x_1\right) = h\left(x_2\right)\right\}\right|, x_1, x_2 \in A, x_1 \neq x_2 \qquad (4.2)$$

Другий показник $n_2\left(x_1, y_1\right)$ характеризує кількість правил гешування, при яких для заданих $x_1 \in A$, $y_1 \in B$ виконується рівність (4.3), тобто кількість

ключів, при яких для вхідної послідовності x_1 значення геш-коду y_1 не змінюється.

$$n_2\left(x_1, y_1\right) = \left|\left\{h \in H : h\left(x_1\right) = y\right\}\right|, x_1 \in A, y_1 \in B \qquad (4.3)$$

Третій показник $n_3\left(x_1, x_2, y_1, y_2\right)$ характеризує кількість правил гешування, при яких для заданих $x_1, x_2 \in A$, $x_1 \neq x_2$, $y_1, y_2 \in B$ виконується рівність (4.4), тобто кількість ключів, при яких для двох вхідних послідовностей x_1 і x_2 відповідні їм значення геш-кодів y_1 і y_2 не змінюються.

$$n_3\left(x_1, x_2, y_1, y_2\right) = \left|\left\{h \in H : h\left(x_1\right) = y_1, h\left(x_2\right) = y_2\right\}\right|, x_1, x_2 \in A, x_1 \neq x_2, y_1, y_2 \in B \qquad (4.4)$$

У табл. 4.28 наведено результати даного дослідження.

Таблиця 4.28

Результати дослідження колізійних властивостей функцій гешування (на міні версіях)

	SHA-512	Ob.-1	Ob.-2	Ob.-3	Bar.-1	Bar.-2	Bar.-3
$\tilde{m}(n_1)$	4.01	3.11	2.72	2.99	3.51	4.07	3.81
$\tilde{D}(n_1)$	0.17	0.23	0.32	0.30	0.28	0.27	0.25
$P_\partial = P\left(\left\|\tilde{m}(n_1) - m(n_1)\right\| < 0.1\right)$	0.96	0.96	0.96	0.97	0.95	0.96	0.94
$\tilde{m}(n_2)$	4.41	4.02	3.89	4.11	4.38	4.29	4.35
$\tilde{D}(n_2)$	0.39	0.42	0.40	0.39	0.37	0.45	0.44
$P_\partial = P\left(\left\|\tilde{m}(n_2) - m(n_2)\right\| < 0.1\right)$	0.88	0.93	0.95	0.96	0.97	0.95	0.93
$\tilde{m}(n_3)$	5.42	4.87	4.75	4.99	5.31	5.39	5.48
$\tilde{D}(n_3)$	0.23	0.31	0.30	0.25	0.24	0.31	0.39
$P_\partial = P\left(\left\|\tilde{m}(n_3) - m(n_3)\right\| < 0.1\right)$	0.95	0.93	0.98	0.92	0.93	0.95	0.96

$m(n_1)$, $m(n_2)$ та $m(n_3)$ – математичні сподівання максимумів кількості правил гешування при яких виконуються рівності (4.2), (4.3) і (4.4) відповідно.

Дисперсії $D(n_1)$, $D(n_2)$ і $D(n_3)$, що характеризують розсіювання значень кількості правил гешування, при яких виконуються рівності (4.2), (4.3) і (4.4), щодо їх математичних сподівань $m(n_1)$, $m(n_2)$ і $m(n_3)$ відповідно.

Під час експериментальних дослідженнях колізійних властивостей гешування оцінювалось середнє арифметичне спостережуваних значень максимумів $\tilde{m}(n_i)$, дисперсію $\tilde{D}(n_i)$, $i=\overline{1,3}$ і довірчою вірогідністю отриманих середньостатистичних оцінок P_{∂}.

На основі отриманих результатів можна стверджувати, що запропоновані функції гешування мають кращі колізійні властивості ніж у функції гешування SHA-512.

4.4. Висновки до розділу 4

1. У даному розділі розроблена методика проведення експериментальних досліджень, описані їх результати та підтвердилась ефективність розроблених рішень.
2. На основі першого (стійкого) та другого (швидкісного) методу побудови функцій гешуванн, розроблено по 3 нові функції гешування, що були програмно реалізовані і досліджені. Були перевірені їхні колізійні властивості, перевірена ефективність перемішування інформації (формування псевдовипадкової гами) за методиками NIST STS і DIEHARD, досліджена їх швидкість і порівнянні із відомими алгоритмами. В результаті порівнянь, визначено, що розроблені геш функції не поступаються відомим, а по деяким показникам показують кращу результати.

3. На основі методу побудови генераторів псевдовипадкових послідовностей сформовано 3 генератора, що були програмно реалізовані і досліджені. Була перевірена ефективність формування псевдовипадкової гами за методиками NIST STS і DIEHARD, досліджена їх швидкість і порівнянні із відомими алгоритмами. В результаті порівнянь, визначено, що розроблені геш функції не поступаються відомим а по деяким показникам показують кращу результати.

ВИСНОВКИ

Результатом виконаної роботи є розв’язання актуальної і важливої науково-технічної задачі розробки та дослідження нових ефективних геш-функцій, які при достатньо високій швидкодії забезпечуватимуть необхідний рівень стійкості.

У процесі виконання дисертаційної роботи отримані такі вагомі результати:

1. Проведено аналіз сучасних методів і алгоритмів побудови та реалізації ефективних криптографічних функцій гешування, що дозволило виявити їх недоліки і формалізувати завдання наукового дослідження. Серед виявлених недоліків основними є уразливість відомих геш-функцій до криптоаналітичних атак, низька швидкість шифрування і високі вимоги до обчислювальних засобів.

2. Розроблено метод побудови функцій гешування, який за рахунок доповнення вхідного повідомлення розміром цього повідомлення та ПВП salt (розраховується на основі вхідного повідомлення), використання у функції стиснення нової послідовності операцій (на основі 6-ти нелінійних функцій, операцій підстановки, додавання за модулем 2 і 2^n, циклічних і лінійних зсувів), дозволив будувати криптостійкі функції гешування. На основі цього методу було розроблено, реалізовано програмно і досліджено три нові стійкі геш-функції, які дозволяють підвищити швидкість у 1.15-1.36 разів та можуть застосовуватись у системах, для яких критичним є параметр стійкості.

3. Розроблено метод побудови функцій гешування, який за рахунок доповнення вхідного повідомлення ПВП salt (розраховується на основі вхідного повідомлення та його розміру), використання у функції

стиснення додаткового вектору внутрішнього стану та нової послідовності операцій (на основі 4-х не лінійних функцій, операцій підстановки, перестановки, додавання за модулем 2 і 2^n та циклічного зсуву), дозволив будувати швидкісні функції гешування. На основі цього методу було розроблено, реалізовано програмно і досліджено три нові швидкісні геш-функції, які дозволяють підвищити швидкість у 1.16-1.53 разів і можуть застосовуватись у системах, для яких критичним є параметр швидкості.

4. Удосконалено метод побудови генераторів ПВП, який за рахунок обробки вектора внутрішнього стану та ключового вектору операціями підстановки, циклічного зсуву, складання за модулем 2 і 2^n та 4-ма нелінійними функціями, дозволив будувати ефективні генератори ПВП. На основі цього методу розроблено і реалізовано програмно три генератори ПВП, які будуть корисними як для функцій гешування, так і для інших криптографічних застосувань (генерування ключів, потокові шифри тощо). Крім того, розроблені генератори ПВП Viriy є більш швидкими за аналоги (зокрема, у 1.02-1.22 разів в порівнянні з алгоритмом Snow).

5. Удосконалено метод криптографічного захисту інформації, що за рахунок фіксування інформації про ідентифікатор користувача, ідентифікатор сесії, час відправлення, довжину повідомлення та його порядковий номер, а також використання нової процедури формування сеансового ключа для шифрування, дає можливість забезпечити конфіденційність і цілісність даних в ІКС.

6. Розроблено спеціалізоване програмне забезпечення у вигляді консольних додатків на мові програмування C++ (середовище розробки Microsoft Visual Studio 2013 (Release Version)) та методику, що дозволило провести експерименти і верифікувати запропоновані методи. Результати дисертаційної роботи використовуються у навчальному процесі

Вінницького національного технічного університету, науковому процесі Національного авіаційного університету та ННВК “Інформаційно-комунікаційні системи”, що підтверджено відповідними актами впровадження.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. S. Gnatyuk, V. Kinzeryavyy, T. Sapozhnik, I. Sopilko, N. Seilova, A. Hrytsak, «Modern Method and Software Tool for Guaranteed Data Deletion an Advanced Big Data Systems», Advances in Intelligent Systems and Computing, Vol. 902, pp. 581-590, 2019 (Scopus)

2. А.В. Грицак, «Удосконалений метод побудови генераторів псевдовипадкових послідовностей для стійкого гешування», Вісник Інженерної академії України, №3, с. 116-122, 2019.

3. Н.В. Остапенко, В.М. Кінзерявий, А.В. Грицак, К.С. Кириченко, «Удосконалена функція гешування MD4», Безпека інформації, Том 24, №2, 2018.

4. А.В. Грицак, І.Р. Березовий, І.В. Гринь, В.М. Кінзерявий, «Програмна система захисту засобів зберігання криптовалют», Вісник Інженерної академії України, №1, с. 128-139, 2018.

5. Ю.Є. Яремчук, А.В. Грицак, Д.В. Присяжний, «Сучасні підходи до побудови і реалізації ефективних криптографічних функцій гешування» Вісник Інженерної академії України, №4, с. 128-139, 2018.

6. А.В. Грицак, В.С. Катаєв, В.О. Леонтьєв, Н.В. Ляховченко, «Проблеми активного захисту інформації від витоку через віброакустичні канали», Реєстрація, зберігання і обробка даних, Том 18, №3, с.54-59, 2016.

7. А.В. Грицак, «Застосування алгоритмів гешування в технології блокчейн», тези доповідей XVII міжнар. наук.-практ. конф. молодих учених і студентів «Політ. Сучасні проблеми науки», 5-7 квітня 2017 р., К., с. 77, 2017.

8. К.С. Кириченко, В.М. Кінзерявий, А.В. Грицак, М.Б. Александер, «Перспективна криптографічна функція гешування», Матеріали міжнар. наук.-практ. конф. «Актуальні питання забезпечення кібернетичної безпеки та захисту інформації», 21-24 лютого 2018 р., Верхній Студений, с. 68-69, 2018.

9. А.В. Грицак, «Дослідження методу побудови функції гешування на основі алгоритму MD4», тези доповідей XVI міжнар. наук.-практ. конф. молодих учених і студентів «Політ. Сучасні проблеми науки», 4-5 квітня 2016 р., К., с. 104, 2016.

10. А.В. Грицак, «Экспериментальное исследование криптостойкости разработанных функций хеширования», материалы XXIII международной научно-технической конференции «Современные средства связи», 17-18 октября 2019 р., Минск, с. 56-59, 2019.

11. Bider D., SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol. [Electronic recourse] // URL: https://tools.ietf.org/html/rfc6668

12. Biham Eli , Differential Cryptanalysis of Feal and N-Hash. [Electronic recourse] / URL: https://link.springer.com/chapter/10.1007%2F3-540-46416-6_1

13. Boer B., An Attack on the Last Two Rounds of MD4 / Bert den Boer and A. Bosselaers // — C.: Lecture Notes in Computer Science, 1991, P. 194–203.

14. Daum Magnus, Cryptanalysis of Hash Functions of the MD4-Family / M. Daum. — B.: Electronic edition, 2005. — P. 23-34.

15. Debaert C. The RIPEMDL and RIPEMDR Improved Variants of MD4 Are Not Collision Free/ Debaert C., Gilbert H. // L.:LNCS, 2002. — P. 52–65.

16. Dobraunig Christoph, Analysis of the Kupyna-256 Hash Function. [Electronic recourse] /C. Dobraunig, M. Eichlseder, F. Mendel // URL: https://eprint.iacr.org/2015/956.pdf

17. Dolmatov V., GOST R 34.11-2012: Hash Function. [Electronic recourse] / URL: https://tools.ietf.org/html/rfc6986

18. Eastlake D., US Secure Hash Algorithm 1 (SHA1). [Electronic recourse] / / URL: https://tools.ietf.org/html/rfc3174

19. Hoch J. J. Breaking the ICE – Finding Multicollisions in Iterative Concatenated and Expanded Hash Functions [Electronic recourse] / J. J. Hoch, A. Shamir. – 2006. – URL: http://www.wisdom.weizmann.ac.il/ yaakovh/papers/hashpaper_submission.pdf

20. Joux A. Multicollisions in Iterated Hash Functions. Application to Cascaded Constructions / Antoine Joux // Lecture Notes in Computer Science. – 2004. – № 3152. – С. 306-316.

21. Kelsey J. Second preimages on n-bit Hash Function for Less than 2n Work [Electronic recourse] / J. Kelsey, B. Schneier // Cryptology ePrint Archive. – 2004. – 15 с. – URL: http://eprint.iacr.org/2004/304.pdf

22. Keromytis A., The Use of HMAC-RIPEMD-160-96 within ESP and AH. [Electronic recourse] / URL: https://www.ietf.org/rfc/rfc2857.txt

23. Leurent Gaetan, MD4 is Not One-Way / G. Leurent. — P.: Département d'Informatique, 2001. — pp. 3-10.

24. Oliynykov R., A New Standard of Ukraine: The Kupyna Hash Function. [Electronic recourse]/ R Oliynykov, I. Gorbenko, O. Kazymyrov, V. Ruzhentsev, O. Kuznetsov, Y. Gorbenko, A. Boiko, O. Dyrda, V. Dolgov, A. Pushkaryov // URL: https://eprint.iacr.org/2015/956.pdf

25. Rivest R., The MD4 Message-Digest Algorithm. [Electronic recourse] / URL: // https://tools.ietf.org/html/rfc1320

26. Rivest R., The MD5 Message-Digest Algorithm. [Electronic recourse]/ URL: https://www.ietf.org/rfc/rfc1321.txt

27. Shannon C.E. A Mathematical Theory of Communication. / C.E. Shannon. // Bell System Technical Journal. — 1948. — №27, pp. 379—423, 623—656.

28. Shirey R., Internet Security Glossary, Version 2. [Electronic recourse] / URL: https://www.ietf.org/rfc/rfc4949.txt

29. Stach P. MD4 Collision Generator. [Electronic recourse] URL: http://packetstormsecurity.org/files/41550/md4coll.c.html

30. Stevens Marc, Attacks on Hash Functions and Applications / M. Stevens. — N.: Ipskamp Drukkers, 2012. — C. 65-81.

31. Wang X., Cryptanalysis of the Hash Functions MD4 and RIPEMD. [Electronic recourse] URL: / http://www.infosec.sdu.edu.cn/uploadfile/papers/Cryptanalysis_of_Hash_Functions.pdf

32. Xiaoyun W., Collisions for Hash Functions MD4, MD5, HAVAL-128 and RIPEMD / X. Wang, D. Feng, X. Lai, and H. Yu // — Cryptology ePrint Archive, 2004. — P. 26-35

33. Адигеев М.Г. Введение в криптографию. Основные понятия, задачи и методы криптографии. /М.Г. Адигеев. – Ростов-на-Дону: РГУ, 2002. — 35 с.

34. Алферов А.П. Основы криптографии: Учебное пособие / А.П. Алферов, А.Ю. Зубов. — М. : Гелиос АРВ, 2002. — 480 с.

35. Шеннон К. Теория связи в секретних системах / Клод Шеннон; [пер. с англ. В.Ф. Писаренко] // Работы по теории информации и кибернетике. — Москва, 1963. — С. 333—369.

36. Бабенко Л.К. Современные алгоритмы блочного шифрования и методы их анализа / Бабенко Л.К., Ищукова Е.А. — М.: Гелиос АРВ — 2006.

37. Баричев С.Г. Криптография без секретов: Учебное пособие / С.Г. Баричев. — М. : Наука, 1999. – 44 с.

38. Баричев С.Г., Основы Современной Криптографии / С.Г. Баричев, Р.Е. Серов // — М.: Горячая Линия – Телеком, 2011. — С. 113-127.

39. Баричев С.Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. — М. : Горячая линия − Телеком, 2002. — с. 175.

40. Бондаренко В.В. Введение в криптографию : Учебное пособие / В.В. Бондаренко. — Ставрополь : СГУ, 2000. — с. 234.

41. Брассар Жиль, Современная криптология / Ж. Брассар. — М.: ПОЛИМЕД, 1999. — С.83-96.

42. Карпунин Г., Оценки сложности поиска коллизий для хэш-функции RIPEMD. [Електронний ресурс] // Г. Карпунин, Е. Ермолаева// URL: http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=pdma&paperid=22&option_lang=rus

43. A. Hrytsak, V. Kinzeryavyy, D. Prysiazhnyi, Yu. Burmak and Ye. Samoylik, “High-Speed and Secure Hash Function for Blockchain Security Mechanisms”, Scientific and Practical Cyber Security Journal (SPCSJ), Vol. 4, Issue 1, pp. 65-70, 2020.

44. Куприянов А.И. Основы защиты информации / А.И. Куприянов, А.В. Сахаров, В.А. Шевцов. — М. : Academia, 2006. — 256 с.

45. Майзаков К.Д., Алгоритмическая оптимизация вычисления прообразов необратимых хэш-функций MD5 И MD4. [Електронний ресурс] / К.Д. Майзаков, Д.А. Эдель, В.А. Новосядлый // URL: http://docplayer.ru/39360691-K-d-mayzakov-d-a-edel-v-a-novosiadliy-algorithmic-optimization-of-inverse-image-computation-for-md5-and-md4-digest-algorithms.html

46. Матов О. Я. Хеш-функції та цілісність інформаційних об’єктів / О. Я. Матов, В. С. Василенко // Реєстрація, зберігання і обробка даних. - 2014. - Т. 16, № 4. - С. 12-17.

47. Мухачева В.А. Методы практической криптографии / В.А. Мухачев, В.А. Хорошко. — К.: ООО «Полиграф Консалтинг», 2005. — 215 с.

48. Панасенко С.П. Алгоритмы шифрования : специальный справочник / С. П. Панасенко. — СПб. : БХВ-Петербург, 2009. — 576 с.

49. Петрашенко А. В., Аналіз алгоритмів пошуку дублікатів текстових документів. [Електронний ресурс] / Петрашенко А. В., Буряк М. М. // — URL:/http://pmk.fpm.kpi.ua/arhive_2010/47-ANALIZ%ALGORYTMIV.pdf

50. Рябко Б.Я. Основы современной криптографии для специалистов в информационных технологіях / Б.Я. Рябко, А.Н. Фионов. — М.: Научный мир, 2004. — с. 173.

51. Сидельников В.М. Криптография и теория кодирования / В.М. Сидельников. — Э.: Электронное издание, 2002. — С. 107-109.

52. Харин Ю. С., Математические и компьютерные основы криптологии. Учебное пособие / Харин Ю. С., Берник В. И., Матвеев Г. В., Агиевич С. В. — М.: Новое издание, — 2003. – с. 210.

53. Хеш функції и цифрові підписи. [Електронний ресурс] URL: // http://izi.vlsu.ru/teach/books/913/theory.html

54. Чекатков А.А. Методы и средства защиты информации: Учебное пособие / А. А. Чекатков, В. А. Хорошко. — М.: Юниор, 2003. — 504 с.

55. Шеннон К. Теория связи в секретних системах / Клод Шеннон; [пер. с англ. В.Ф. Писаренко] // Работы по теории информации и кибернетике. — Москва, 1963. — С. 333—369.

56. Шнайер Б., Практическая криптография / Б. Шнайер, Н. Фергюсон // — М: Диалектика, 2005. — С. 105-113.

57. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. — М. : Триумф, 2002. — с. 716.

58. Щербаков А., Прикладная криптография / А. Щербаков, А. Домашев// — М.: Русская Редакция, 2003. — С. 297-303.

59. Ямпольський Ю.С. Лекції до дисципліни «Захист інформації» / Ю.С.Ямпольський. — О.: ОНПУ, 2002. — С. 20-25.

60. Ященко В.В. Введение в криптографію / В.В. Ященко. — П.: Питер, 2001. — с. 288.

61. MTProto Mobile Protocol v.1.0 [Electronic recourse] / URL: https://core.telegram.org/mtproto_v1

62. Signal Protocol [Electronic recourse] / URL: https://signal.org/docs/

63. Матричный алгоритм Диффи–Хэллмана [Electronic recourse] / URL: www.hups.mil.gov.ua/periodic-app/article/7440/soi_2010_3_31.pdf

64. TLS 1.0/1.1 Deprecation [Electronic recourse] / URL: https://techcommunity.microsoft.com/t5/Skype-for-Business-Blog/Preparing-for-TLS-1-0-1-1-Deprecation-O365-Skype-for-Business/ba-p/222247

65. eSignatures from eSign Genie [Electronic recourse] / URL: https://www.esigngenie.com/electronic-signatures/

66. SignEasy [Electronic recourse] / URL: https://www.getsigneasy.com/

67. Adobe EchoSign [Electronic recourse] / URL: https://acrobat.adobe.com/us/en/sign.html

68. Adobe EchoSign [Electronic recourse] / URL: https://appexchange.salesforce.com/listingDetail?listingId=a0N300000016ZmCEAU

69. Signable [Electronic recourse]/ URL: https://www.signable.co.uk/

70. Bitcoin [Electronic recourse] / URL: https://bitcoin.org/ru/

71. Ethereum [Electronic recourse] / URL: https://www.ethereum.org/

72. Ripple [Electronic recourse] / URL: https://ripple.com/

73. Monero [Electronic recourse] / URL: https://ww.getmonero.org/

74. Litecoin [Electronic recourse] / URL: https://litecoin.org/ru/

75. Dash [Electronic recourse] / URL: https://www.dash.org/ru/

76. SHA-3 [Electronic recourse] / URL: https://ru.bitcoinwiki.org/wiki/SHA-3.

77. Уязвимости MD4 [Electronic recourse] / URL: https://www.securitylab.ru/blog/personal/shaurojen/22829.php

78. Програми та програмне забезпечення [Electronic recourse] / URL: http://khpi-iip.mipk.kharkiv.edu/library/sp/sp_book/sp01.html

79. Dolmatov V., GOST R 34.11-2012: Hash Function. [Electronic recourse] / URL: https://tools.ietf.org/html/rfc6986

80. Вентцель Е. С. Теория вероятностей / Е. С. Вентцель. – М. : Государственное издательство физико-математической литературы, 1958. – 564 с.

81. Вишневский В. М. Теоретические основы проектирования компьютерных сетей / В. М. Вишневский. – М. : Техносфера, 2003. – 512 с.

82. Галкин В. А. Телекоммуникации и сети / В. А. Галкин, Ю. А. Григорьев. – М. : МГТУ имени Н. Э. Баумана, 2003. – 608 с.

83. Гмурман В. Е. Теория вероятностей и математическая статистика / В. Е. Гмурман. – М. : Высшая школа, 2003. – 479 с.

84. Долгов В. И. Подход к криптоанализу современных шифров / В. И. Долгов, И. В. Лисицкая, Р. В. Олейников // Современные информационные системы. Проблемы и тенденции развития, материалы второй международной конференции, Харьков – Туапсе, Украина, 2 – 5 октября. – 2007. – С. 435 – 436.

85. ДСТУ 2481-94 Системи оброблення інформації. Інтелектуальні інформаційні технології. Терміни та визначення. – Х. : ДСТУ, 1994. – 33 с.

86. Дуглас Э. Камер. Сети TCP/IP. Принципы, протоколы и структура. Т. 1 / Камер Э. Дуглас. – 4-е изд. – М. : ИД «Вильямс», 2003. – 848 с.

87. Евсеев С. П. Механизмы обеспечения аутентичности банковских данных во внутриплатежных системах комерческого банка. / С. П. Евсеев, В. Е. Чевардин, С. А. Радковский. – Х. : ХНЕУ, 2008. – Вип. 6. – С. 40–44.

88. Городецкий А. Я. Информационные системы. Вероятностные модели и статистические решения / А. Я. Городецкий – СПб. : Изд-во СПбГПУ, 2003. – 326 с.

89. Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях. / М. А. Иванов – М. : Кудиц-Образ, 2001. – 368 с.

90. Ирвин Дж. Передача данных в сетях и инженерный подход / Дж. Ирвин, Д. Харль. – СПб. : Питер, 2002. – 405 с.

91. Исследование дифференциальных свойств блочно-симметричных шифров. / Л. С. Сорока, А. А. Кузнецов, И. В. Московченко и др. // Системи обробки інформації. – Х. : ХУПС, 2010. – Вип. 6(87). – С. 286–294.

92. Исследование дифференциальных свойств мини-шифров Baby-ADE и Baby-AES / В. И. Долгов, А. А. Кузнецов, Р. В. Сергиенко и др. // Прикладная радиоэлектроника. – Х. : ХНУРЭ, 2009. – Т. 8, № 3. – С. 252–257.

93. Король О. Г. Исследование коллизионных свойств кодов аутентификации сообщений UMAC / О. Г. Король, А. А. Кузнецов, С. П. Евсеев // Прикладная радиоэлектроника. – Х. : Изд-во ХНУР, 2012. – Т. 11, № 2. – С. 171–183.

94. Король О. Г. Исследование методов обеспечения аутентичности и целостности данных на основе односторонних хэш-функций / О. Г. Король, С. П. Евсеев // Захист інформації : науково-технічний журнал. Спецвипуск (40). – 2008. – С. 50–55.

95. Халимов Г. З. Каскадное универсальное хеширование с использованием АГК-кодов / Г. З. Халимов, А. Ю. Иохов // Восточно-

европейский журнал передовых технологий. – Х., 2005. – Вып. 2/2(14). – С. 155–159.

96. Korchenko O., Vasiliu Y., Gnatyuk S. Modern quantum technologies of information security against cyber-terrorist attacks, Aviation, Vol. 14, №3, pp. 58-69, 2010.

97. S. Gnatyuk, M. Kovtun, V. Kovtun, A. Okhrimenko, Development of a search method of birationally equivalent binary Edwards curves for binary Weierstrass curves from DSTU 4145-2002, Proceedings of 2nd Intern. Scientific-Practical Conf. on the Problems of Infocommunications. Science and Technology (PIC S&T 2015), Kharkiv, Ukraine, October 13-15, 2015, pp. 5-8.

98. S. Gnatyuk, T. Zhmurko, P. Falat, Efficiency Increasing Method for Quantum Secure Direct Communication Protocols, Proceedings of the 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS'2015), Warsaw, Poland, September 24-26, Vol. 1, 2015, pp. 468-472.

99. S. Gnatyuk, A. Okhrimenko, M. Kovtun, T. Gancarczyk, V. Karpinskyi, Method of Algorithm Building for Modular Reducing by Irreducible Polynomial, Proceedings of the 16th International Conference on Control, Automation and Systems, Oct. 16-19, Gyeongju, Korea, 2016, pp. 1476-1479.

100. Hu Z., Gnatyuk S., Kovtun M., Seilova N. Method of searching birationally equivalent Edwards curves over binary fields, Advances in Intelligent Systems and Computing, Vol. 754, pp. 309-319, 2019.

101. S. Gnatyuk, V. Kinzeryavyy, M. Iavich, D. Prysiazhnyi, Kh. Yubuzova, High-Performance Reliable Block Encryption Algorithms Secured against Linear and Differential Cryptanalytic Attacks, CEUR Workshop Proceedings, Vol. 2104, pp. 657-668, 2018.

102. M. Iavich, S. Gnatyuk, E. Jintcharadze, Yu. Polishchuk, R. Odarchenko, Hybrid Encryption Model of AES and ElGamal Cryptosystems for Flight Control

Systems, Proceedings of the 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control, October 16-18, 2018, Kyiv, Ukraine, pp. 229-233.

103. Tynymbayev S., Gnatyuk S.A., Aitkhozhayeva Y.Z., Berdibayev R.S., Namazbayev T.A. Modular reduction based on the divider by blocking negative remainders, News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, №2 (434), pp. 238-248, 2019.

104. Kalimoldayev M., Tynymbayev S., Gnatyuk S., Ibraimov M., Magzom M. The device for multiplying polynomials modulo an irreducible polynomial, News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, №2 (434), pp. 199-205, 2019.

105. Zh. Hu, S. Gnatyuk, T. Okhrimenko (Zhmurko), V. Kinzeryavyy, M. Iavich, Kh. Yubuzova, High-Speed Privacy Amplification Method for Deterministic Quantum Cryptography Protocols Using Pairs of Entangled Qutrits, CEUR Workshop Proceedings, Vol. 2393, pp. 810-821, 2019.

106. Kalimoldayev M., Tynymbayev S., Gnatyuk S., Khokhlov S., Magzom M., Kozhagulov Y. Matrix multiplier of polynomials modulo analysis starting with the lower order digits of the multiplier, News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, №4 (436), pp. 181-187, 2019.

107. Iavich M., Gagnidze A., Iashvili G., Gnatyuk S., Vialkova V. Lattice based Merkle, CEUR Workshop Proceedings, Vol. 2470, pp. 13-16, 2019.

108. M. Iavich, S. Gnatyuk, E. Jintcharadze, Y. Polishchuk, A. Fesenko and A. Abisheva, Comparison and Hybrid Implementation of Blowfish, Twofish and RSA Cryptosystems, Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), Lviv, Ukraine, 2019, pp. 970-974.

109. S. Tynymbayev, E. Aitkhozhayeva, R. Berdibayev, S. Gnatyuk, T. Okhrimenko and T. Namazbayev, Development of Modular Reduction Based on the Divider by Blocking Negative Remainders for Critical Cryptographic Applications, Proceedings of 2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON), Lviv, Ukraine, 2019, pp. 809-812.

110. Qoussini A.E., Daradkeh Y.I., Al Tabib S.M., Gnatyuk S., Okhrimenko T., Kinzeryavyy V. Improved model of quantum deterministic protocol implementation in channel with noise, Proceedings of the 2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2019), 2019, pp. 572-578.

111. Gnatyuk S., Kinzeryavyy V., Kyrychenko K., Yubuzova Kh., Aleksander M., Odarchenko R. Secure Hash Function Constructing for Future Communication Systems and Networks, Advances in Intelligent Systems and Computing, Vol. 902, pp. 561-569, 2020.

112. Gnatyuk S., Akhmetov B., Kozlovskyi V., Kinzeryavyy V., Aleksander M., Prysiazhnyi D. New Secure Block Cipher for Critical Applications: Design, Implementation, Speed and Security Analysis, Advances in Intelligent Systems and Computing, Vol. 1126, pp. 93-104, 2020.

113. Hu Z., Gnatyuk S., Okhrimenko T., Tynymbayev S., Iavich M. High-speed and secure PRNG for cryptographic applications, International Journal of Computer Network and Information Security, Issue 12 (3), pp. 1-10, 2020.

114. Gnatyuk S., Okhrimenko T., Azarenko O., Fesenko A., Berdibayev R. Experimental Study of Secure PRNG for Q-trits Quantum Cryptography Protocols, Proceedings of the 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT 2020), Kyiv, Ukraine, May 14, 2020, pp. 183-188.

115. Iavich M., Gnatyuk S., Arakelian A., Iashvili G., Polishchuk Y., Prysiazhnyy D., Improved Post-quantum Merkle Algorithm Based on Threads,

Advances in Intelligent Systems and Computing, Vol. 1247 AISC, pp. 454-464, 2021.

116. Kalimoldayev M., Tynymbayev S., Gnatyuk S., Ibraimov M., Magzom M. The device for multiplying polynomials modulo with analysis of two least significant bits of the multiplier per step, News of the National Academy of Sciences of the Republic of Kazakhstan, Series of Geology and Technical Sciences, №3 (441), pp. 102-109, 2020.

117. B. Akhmetov, S. Gnatyuk, V. Kinzeryavyy, Kh.Yubuzova, Studies on practical cryptographic security analysis for block ciphers with random substitutions, International Journal of Computing, Vol. 19, Issue 2, pp. 298-308, 2020.

118. Tynymbayev S., Berdibayev R., Omar T., Gnatyuk S., Namazbayev T., Adilbekkyzy S. Devices for modular multiplication of numbers with analysis of two least significant bits of the multiplier, CEUR Workshop Proceedings, Vol. 2654, pp. 762-772, 2020.

119. Bierbrauer J. Authentification via algebraic-geometric codes [Electronic resource] / J. Bierbrauer. – Access mode : http://www.math.mtu.edu/~jbierbra/ potpap.ps.

120. Bierbrauer J. On families of hash function via geometric codes and concatenation / J. Bierbrauer, T. Johansson, G. Kabatianskii // Advances in Cryptology – CRYPTO 93. Lecture Notes in Computer Science. – 1994 – № 773. – P. 331–342.

121. Daemen J. AES Proposal: Rijndael, AES Algorithm Submission [Electronic resource] / J. Daemen, V. Rijmen. – 1999, 3 September. – Access mode : http://www.docstoc.com/docs/14641406/AES-Implementation-and-Performance-Evaluation-on-8-bit-Microcontrollers.

122. Fast hashing on the Pentium // Advances in Cryptology ‘CRYPTO '96 (1996) [Electronic resource] / A. Bosselaers, R. Govaerts, J. Vandewalle ; vol. 1109

of Lecture Notes in Computer Science, Springer-Verlag. – P. 298–312. – Access mode : http://www.esat.kuleuven.ac.be/ bosselae/fast.html.

123. Ferguson Niels A Cryptographic Evaluation of IPsec Counterpane Internet Security [Electronic resource] / Niels Ferguson, Bruce Schneier – Access mode : ttp://ftp.csci.csusb.edu/ykarant/courses/f2007/csci530/papers/counterpane-ipsec.pdf.

124. Final report of European project number IST-1999-12324, named New European Schemes for Signatures, Integrity and Encryption. – April, 19. – 2004 – Version 0.15 (beta), Springer-Verlag.

125. Halevi S. CRYPTO '99, LNCS. / S. Halevi H., Krawczyk, T. Krovetz, P. Rogaway. – Springer-Verlag, 1999. – vol. 1666. – P. 216-233