Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(назва факультету)
Кафедра кібербезпеки
(повна назва кафедри)

ПОЯСНЮВАЛЬНА ЗАПИСКА
до дипломного проекту (роботи)

магістр
(освітній рівень)

на тему: *«Дослідження методів безпечного гешування для забезпечення цілісності та автентичності інформації у автоматизованих банківських системах»*

Виконав: студент (ка) *VI* курсу, групи _____

Спеціальності:
125 «Кібербезпека»
(шифр і назва напряму підготовки, спеціальності)

Очеретний В.О.
підпис (прізвище та ініціали)

Керівник _____ *Карпінський М.П.*
підпис (прізвище та ініціали)

Нормоконтроль _____ *Лобур Т.Б.*
підпис (прізвище та ініціали)

Рецензент _____
підпис (прізвище та ініціали)

м. Тернопіль – 20*20*

АНОТАЦІЯ

Дослідження методів безпечного гешування для забезпечення цілісності та автентичності інформації у автоматизованих банківських системах // Дипломна робота ОР «Магістр» // Очеретний Владислав Олегович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра кібербезпеки, група СБм-61 // Тернопіль, 2020 // 76 стор., 62 рис., 11 табл., 3 додатки, 40 джерел.

Ключові слова: ЦІЛІСНІСТЬ ІНФОРМАЦІЇ, АВТЕНТИФІКАЦІЯ ПОВІДОМЛЕНЯ, ЕЛЕКТРОННИЙ ЦИФРОВИЙ ПІДПИС, ГЕШ-ФУНКЦІЯ, НАЦІОНАЛЬНА ПЛАТІЖНА СИСТЕМА.

Дана магістерська кваліфікаційна робота присвячена дослідженню механізмів забезпечення автентичності за допомогою геш-функцій на основі MAC та MDC-кодів. Об'єктом дослідження є процес забезпечення автентичності та цілісності банківської інформації.

Предметом дослідження є методи ключового гешування забезпечення автентичності та цілісності банківської інформації.

Метою роботи є моделювання ключового гешування забезпечення автентичності та цілісності банківської інформації.

При розробленні програмного продукту було використано програмне середовище NetBeans IDE 6.9.1 та мова програмування – Java.

Методами розробки обрано:

При аналізі побудови MAC- (MDC-) кодів використані методи теорії захисту інформації. При оцінці технічних характеристик MAC- (MDC-) кодів використані методи теорії ймовірності та математичної статистики.

У результаті роботи проведений аналіз механізмів забезпечення автентичності на основі MAC та MDC-кодів, обґрунтовані рекомендації щодо їх використання в організаціях банківського сектора.

ANNOTATION

Research of secure hashing methods to ensure the integrity and authenticity of information in automated banking systems // Thesis OR “Master” // Ocheretnyi Vladyslav Olehovych // Ternopol National Technical University named after Ivan Pulyuy, Faculty of Computer Information Systems and Software Engineering, Department of Cybersecurity -61 // Ternopol, 2020 // 76 pages, 62 figures, 11 tables, 3 appendices, 40 sources.

Keywords: INFORMATION INTEGRITY, AUTHENTICATION OF ELECTRONIC DIGITAL SIGNATURE MESSAGES, HASH FUNCTIONS, NATIONAL PAYMENT SYSTEM.

This master's thesis is devoted to the study of mechanisms for ensuring authenticity using hash functions based on MAC and MDC codes. The object of the research is the process of ensuring the authenticity and integrity of banking information. The subject of the research is the methods of key hashing to ensure the authenticity and integrity of banking information.

The aim of the work is to simulate key hashing to ensure the authenticity and integrity of banking information. During the development of the software product, the NetBeans IDE 6.9.1 software environment and the programming language Java used.

The development methods selected:

When analyzing the construction of MAC- (MDC-) codes, the methods of information security theory used. When evaluating the technical characteristics of MAC- (MDC-) codes, the methods of probability theory and mathematical statistics used. As a result of the work, an analysis of mechanisms for ensuring authenticity based on MAC and MDC codes was carried out, recommendations for their use in organizations of the banking sector were substantiated.

ЗМІСТ

ВСТУП

Розвиток сучасних банківських систем, стрімке зростання обчислювальних можливостей ІТ-технологій дозволяє суттєво поширити спектр електронних послуг цифрового банкінгу. Це з одного боку дозволяє використовувати клієнтам організацій банківського сектору запропонованих можливостей в умовах розвитку кіберпростору, з іншого – дозволяє зловмисникам і хакерським угрупуванням на основі комплексування сучасних загроз з методами соціальної інженерії отримувати несанкціонований доступ к банківської інформації (БіН).

Для забезпечення основних послуг безпеки: конфіденційності, цілісності, доступності і автентичності в авторизованих банківських системах (АБС) використовуються сертифіковані Національним банком України сучасні стандартні механізми. Серед яких для забезпечення автентичності та цілісності БіН можуть використовуватись сучасні алгоритми геш-функцій на основі формування MAC- та MDC-кодів. Відомо що перші формуються на основі блоково-симетричних алгоритмів шифрування в режимах CBC та OFB., модулярної арифметики, а також за допомогою спеціалізованих геш-функцій. При цьому забезпечується формування криптопримітиву фіксованої довжини. Другі (MDC-коди (Manipulation Detection Code)) забезпечують формування криптопримітиву, але для його криптостійкості необхідно використання додаткових механізмів шифрування. Основні алгоритми гешування, які є переможцями європейського конкурсу NESSIE закріплені в міжнародних стандартах ISO/IEC 10118– (1–3).

Метою роботи є дослідження властивостей алгоритмів гешування другого туру конкурсу SHA-3.

Для досягнення мети необхідно вирішити часткові задачі:

проаналізувати відомі методи формування MAC-кодів;

дослідити та порівняти основні характеристики MAC-кодів;

дослідити властивості стійкості MAC-кодів за допомогою пакету NIST STS822, вибрати найкращий та обґрунтувати свій вибір.

Об’єктом дослідження є процес забезпечення автентичності БіН в АВС.

Предметом дослідження є методи ключового гешування для забезпечення автентичності БіН в АВС.

Практичним результатом є створення програмного застосунку, який дозволяє проводити дослідження сучасних MAC-кодів на стійкість.

Апробація результатів магістерської роботи. Окремі результати роботи доповідались на VIII науково-технічній конференції “Інформаційні моделі, системи та технології”, Тернопіль, ТНТУ, 9 – 10 грудня 2020 р.

РОЗДІЛ 1 АНАЛІЗ СУЧАСНИХ МЕХАНІЗМІВ ЗАБЕЗПЕЧЕННЯ АВТЕНТИЧНОСТІ ТА ЦІЛІСНОСТІ БАНКІВСЬКОЇ ІНФОРМАЦІЇ В АВТОМАТИЗОВАНИХ БАНКІВСЬКИХ СИСТЕМАХ

1.1 Аналіз послуг і механізмів забезпечення безпеки інформації відповідно до міжнародних стандартів ISO 7498, ISO / IEC 10181

Сучасний розвиток держави пов’язаний з розвитком цифрової економіки основою якої є національна система масових електронних платежів (НСМЕП) – складна багаторівнева система централізованого управління, що забезпечує якісний стратегічно важливий канал проведення фінансових транзакцій [7; 13], яка наведена на рис. 1.1.

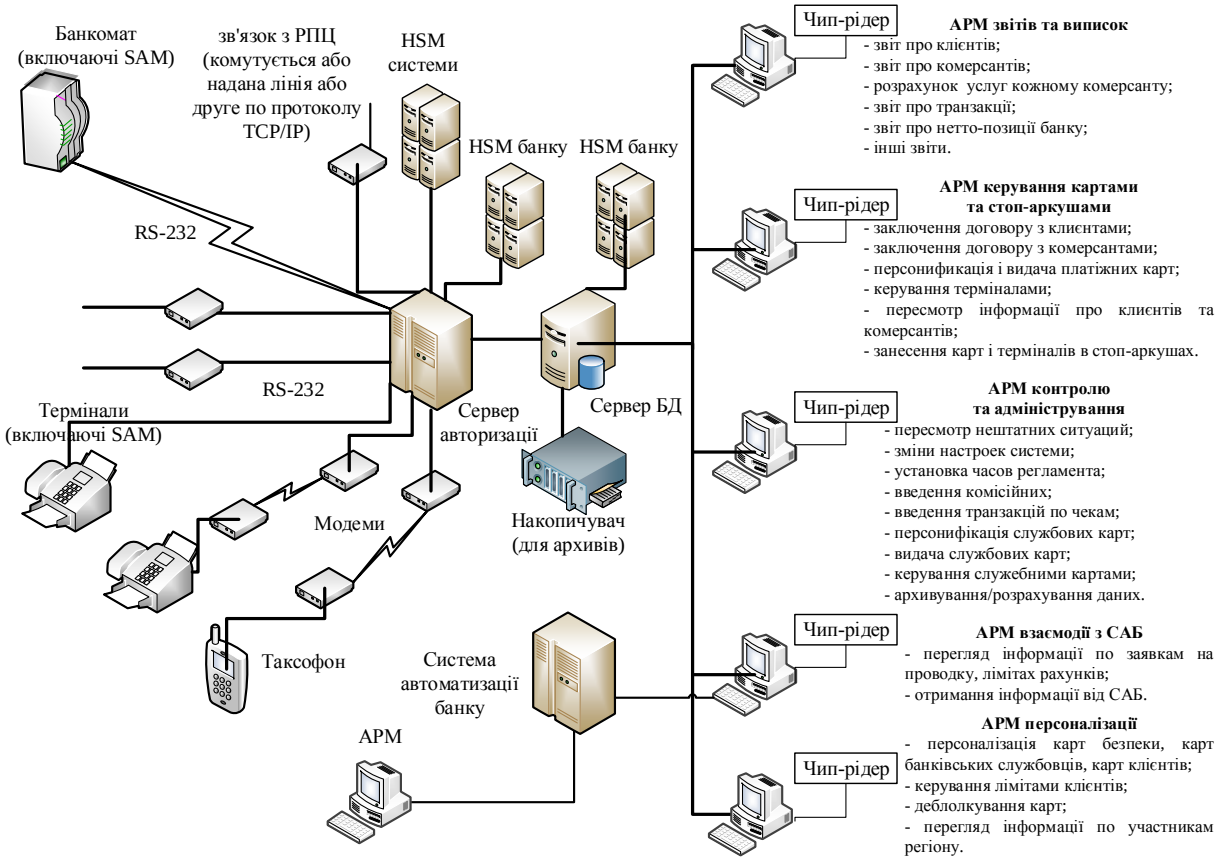

Рисунок 1.1 – Структурна схема національної платіжної системи

Умовні позначення до рис. 1.1:

– РПЦ – регіональні процесингові центри в обласних керівництвах НБУ або в комерційних підприємствах (до 25 РПЦ на всю Україну);

– банки-емітенти й еквайери НСМЕП зі своїми банківськими підсистемами, торговельною інфраструктурою й інфраструктурою сфери послуг;

– BST – банківські термінали;

– POS – модулі безпеки;

– HSM-системи – системний модуль безпеки;

– АРМ – автоматизоване робоче місце.

Аналіз сучасного стану розвитку елементів НСЕМП показав, що стрімкий розвиток обчислювальних технологій, Інтернет-послуг дозволяє поширювати кількість банкоматів і POS-терміналів [2 – 5; 16], що дозволяє поєднувати централізовані і децентралізовані системи цифрової економікт. На рис. 1.2 наведена діаграма росту кількості банкоматів у світі, на рис. 1.3 – в Україні [17].

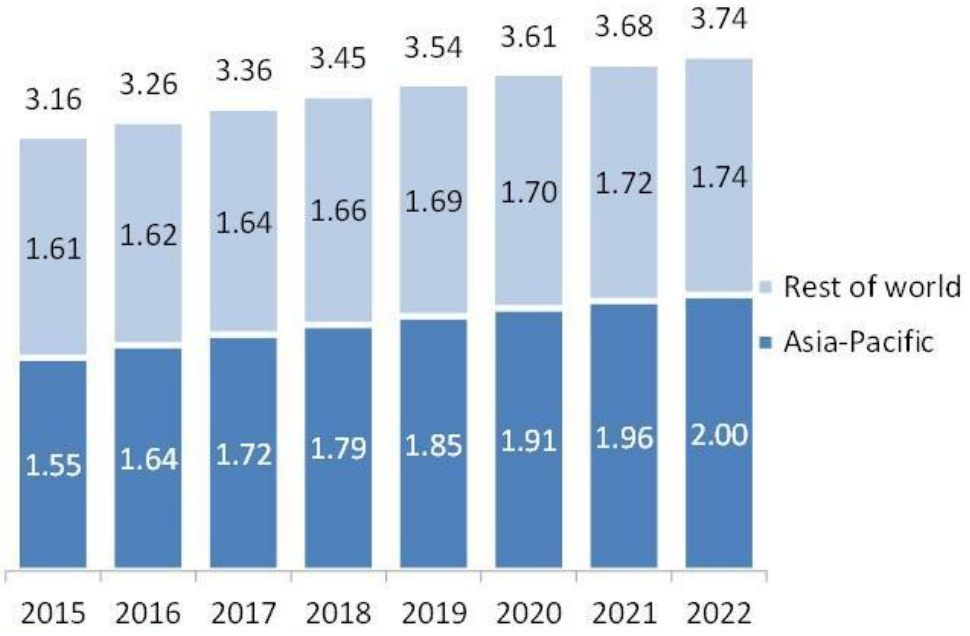

Рисунок 1.2 – Зростання кількості банкоматів у світі (млн. пристроїв)

Формування електронної економіки дозволяє використовувати кіберпростір для виконання головних завдань, які пов’язані з наданням основних

послуг електронного банкінгу, на рис. 1.4 представлені результати аналізу грошового товарообігу через банкомати [17].

Рисунок 1.3 – Зростання числа банкоматів в Україні

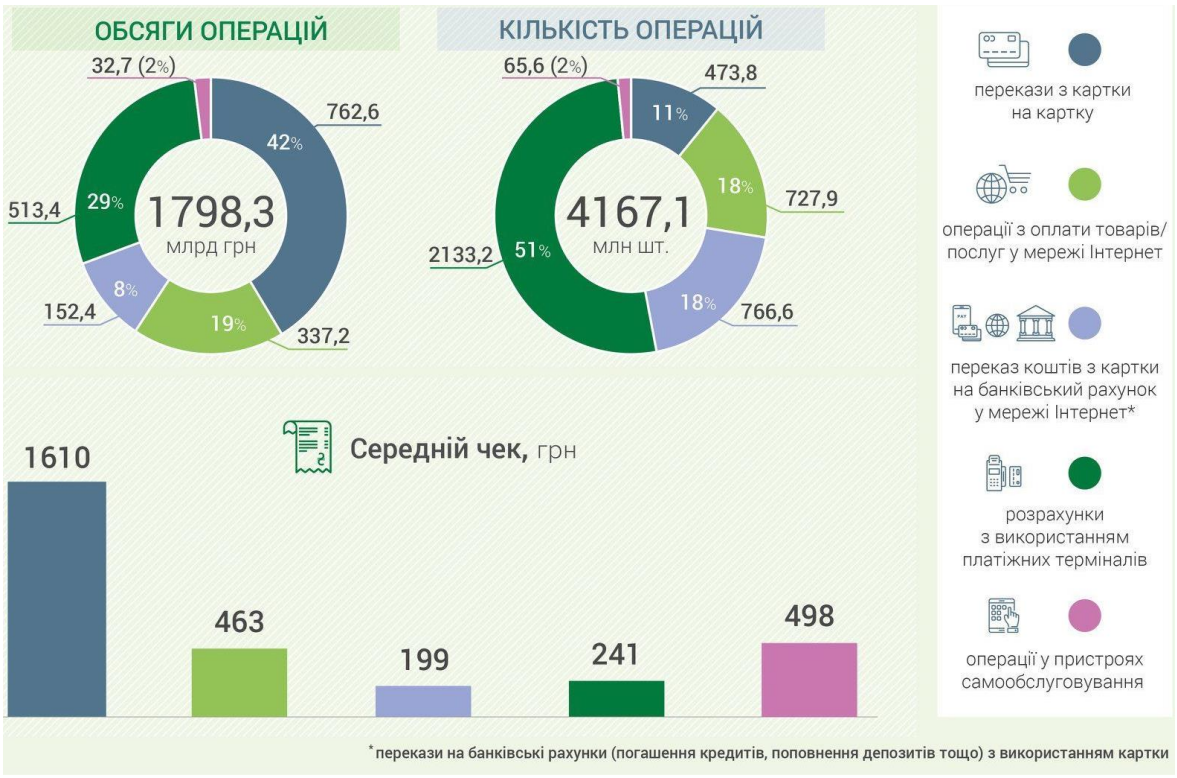

Рисунок 1.4 – Зростання товарообігу через банкомати в Україні

Аналіз рис. 1.2–1.4 стверджує поступовий розвиток електронного банкінгу, формування основних елементів сучасної цифрової економіки.

Однак проведений аналіз сучасних загроз показує, що вони набувають ознак гібридності та синергізму. На рис. 1.5 наведена синергічна модель сучасних загроз на АБС.

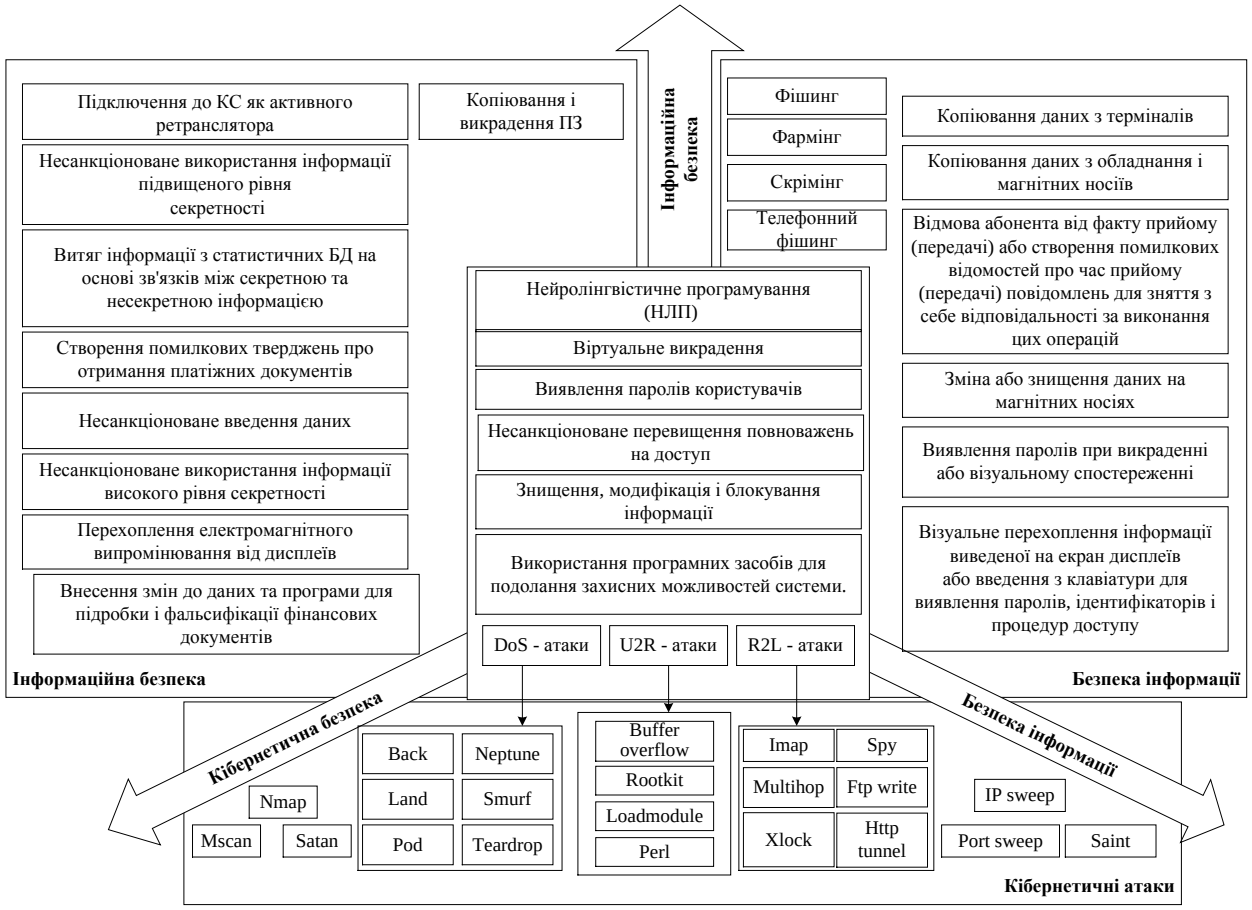

Рисунок 1.5 – Синергетична модель загроз безпеці БіН

Особливістю розглянутої моделі (рис. 1.5) є відображення на ній у вигляді логічних зв’язків взаємодії загроз для різних профілів безпеки. Тому синергетична модель загроз безпеці БІР закладає необхідні і достатні умови для розробки нового методологічного базису, спрямованого на досягнення синергетичного ефекту в сфері забезпечення безпеки державних і приватних систем банківського захисту.

Прикладами програмної реалізації механізмів безпеки в АБС є програмні засоби криптографічного захисту інформації “Грифон-Б” і “Грифон-Л” [9, 10,

23]. Процедури, що входять до складу бібліотеки криптографічного захисту інформації “Тайфун - *PKCS* # 11” реалізують:

– шифрування / розшифрування даних за алгоритмом ГОСТ 28147-2009;

– формування / перевірку імітовставки за алгоритмом ГОСТ 28147-2009;

– формування / перевірку ЕЦП за алгоритмами ДСТУ 4145-2002, ГОСТ 34.310-95, 34.311-95;

– формування ключів шифрування за схемою Діффі-Геллмана (використовується відкритий розподіл ключів відповідно до вимог *ISO 11166-94*).

Швидкісні характеристики програмних засобів, що реалізують алгоритми криптографічних перетворень (для ПК на базі *Intel Celeron* 2,4 ГГц):

– швидкість шифрування / розшифрування даних в режимі простої заміни БСШ ГОСТ 28147-2009 не менше 8 Мбайт/с;

– швидкість обчислення геш-функції даних відповідно до ГОСТ 34.311 – 95 не менше 3 Мбайт/с;

– формування ЕЦП відповідно до ГОСТ 34.310-95 при довжині ключа 512 біт не більше 0,003 с;

– час перевірки ЕЦП відповідно до ГОСТ 34.310-95 при довжині ключа 512 біт не більше 0,006 с;

– формування ЕЦП відповідно до ГОСТ 34.310-95 при довжині ключа 1024 біт не більше 0,01 с;

– час перевірки ЕЦП відповідно до ГОСТ 34.310-95 при довжині ключа 1024 біт не більше 0,02 с;

– формування ЕЦП (з обчисленням предпідпису) згідно з ДСТУ 4145-2002 для основного поля степені 163 не більше 0,0068 с;

– при перевірці ЕЦП згідно з ДСТУ 4145-2002 для основного поля степені 163 не більше 0,013 с.

Правління Національного банку України визначило основні механізми забезпечення послуг безпеки:

криптографічні алгоритми симетричної криптографії (ГОСТ-28147-2009, “Калина-256”, *AES*, з довжиною ключа не менше 128 біт;

несиметричної криптографії (алгоритми Діффі-Геллмана, алгоритм Ель-Гамаля, як звичайні, так і на еліптичних кривих, алгоритм *RSA*, з довжиною простих чисел 2048 біт – забезпечення обміну ключами);

MAC-коди“Купина” (ДСТУ 7564), *SHA*-224, *SHA*-256, *SHA*-384, *SHA*-512; методи суворої автентифікації;

електронний цифровий підпис (ЕЦП) ДСТУ – 4145.

1.2 Аналіз та порівняльні дослідження механізмів забезпечення автентичності БіН в АБС

Для забезпечення автентичності та/або цілісності БіН в АБС як правило використовуються методи симетричної та несиметричної криптографії, ЕЦП, методу суворої (строгої) автентифікації 2FA, а також MAC та MDC-коди, які наведені на рис.1.6.

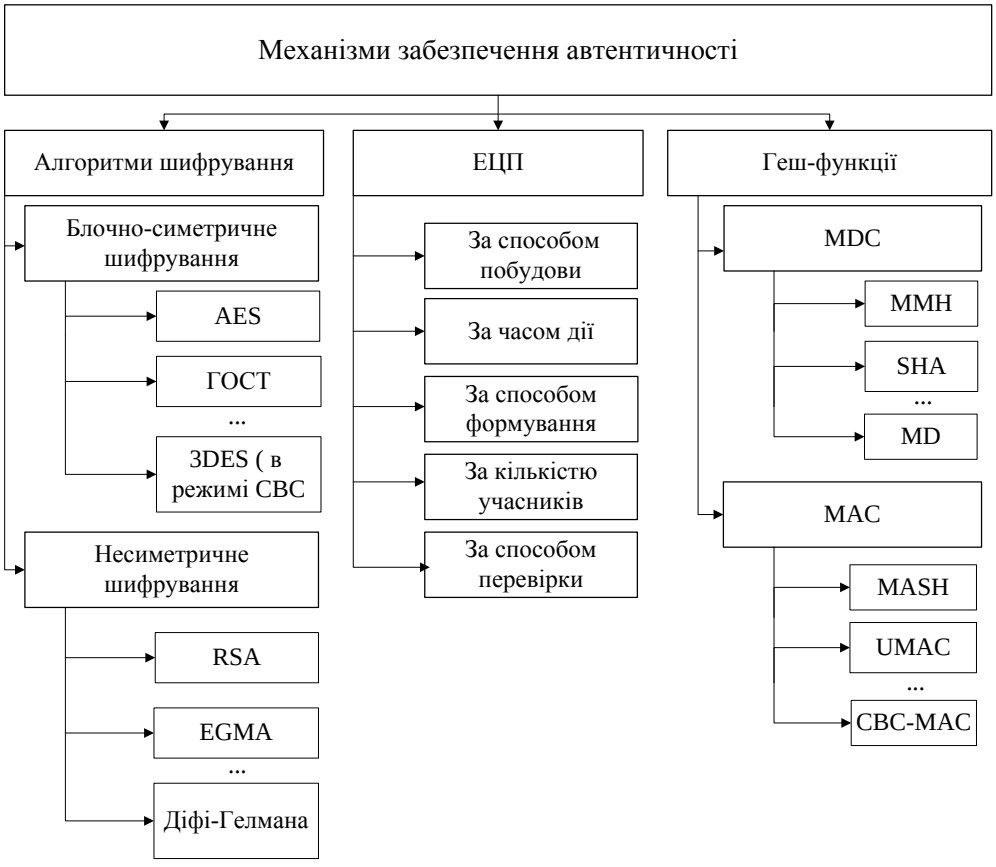

Рисунок 1.6 – Механізми перевірки автентичності інформації

Серед механізмів автентичності особливе місце займають механізми формування геш-кодів, загальна класифікація яких наведена на рис. 1.7:

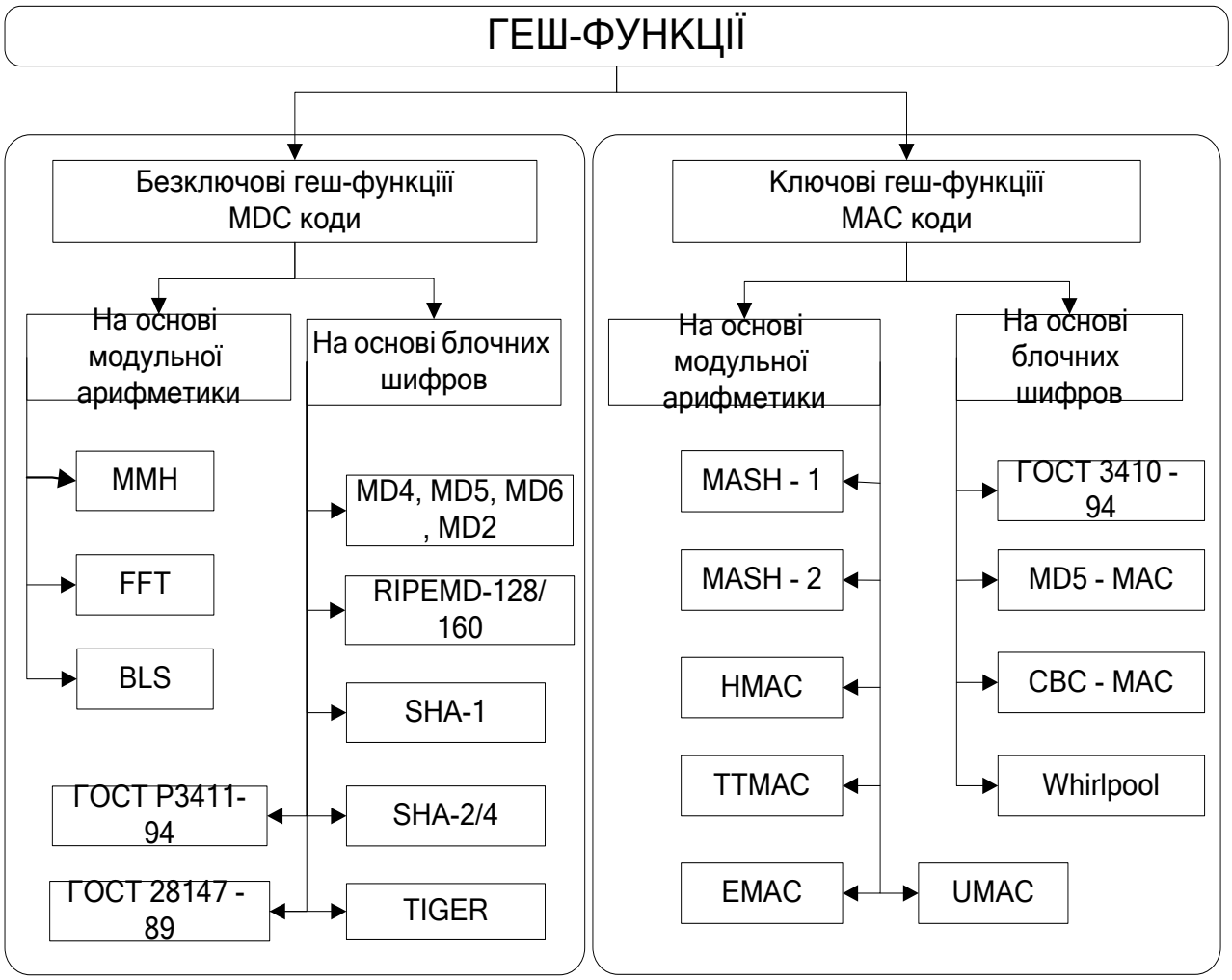

Рисунок 1.7 – Класифікація геш-функцій

Основні технічні ознаки сучасних геш-алгоритмів наведені в табл. 1.2.

Таблиця 1.2 – Порівняльна характеристика безключових геш-функцій

Характеристика	MD5	MD6	SHA-1	SHA-2 (256/512)	ГОСТ 28147-89	RIPEMD-128	RIPEMD - 160
Довжина дайджесту, біт	128	512	160	256/512	256	128	160
Розмір блоку обробки, біт	512	512	512	512/1024	512	512	512
Кількість ітерацій	64	168	80	64/80	32	128	160
Кількість елементарних логічних функцій	4	4	3	6/6	8	5	5
Кількість додаткових констант	64	-	4	64/64	-	4	9
Швидкість роботи на Pentium III, Мбіт/с	574,6	-	344,4	135,5 / 68,7	315,27	63.8	39.8

Аналіз табл. 1.2 свідчить про необхідність поєднання швидкості перетворень при формуванні геш-коду, та забезпечення стійкості отриманого дайджеста. Як правило, для забезпечення стійкості використовуються процедури симетричного шифрування. В табл. 1.3 наведено аналіз тестування швидкості роботи алгоритмів гешування.

Таблиця 1.3 -Аналіз тестування швидкості роботи алгоритмів гешування

Функція гешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
ГОСТ 34311-95	-	C + ASM	49,408 Мбіт/с	83,056 Мбіт/с
UMAC	-	C C + ASM	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с
ГОСТ 28147-89 (OFB)	16	C + ASM	189,559 Мбіт/с	315,270 Мбіт/с

Аналіз табл. 1.3 підтверджує рішення конкурсу NESSIE, щодо алгоритму UMAC, який забезпечує формування геш-коду зі швидкістю10^9 біт/с. Для забезпечення стійкості на третьому шарі цього алгоритму використовується підложка за допомогою AES.

1.3 Висновки до розділу 1

Аналіз механізмів та протоколів забезпечення автентичності і цілісності інформації показав, що вони не задовольняють сучасним потребам користувачів при обробленні великих обсягів інформації за короткий час. Найбільш широковживаним на даний момент є механізм, у якому використовується електронно-цифровий підпис і коди автентифікації повідомлення, але їх оперативність та криптостійкість також потребує оновлення.

Проведені дослідження дозволяють зробити висновок, що для автентичності і цілісності інформації слід використовувати MAC-коди, які не тільки формують геш-код повідомлення, але й забезпечують належний рівень криптостійкості для протистояння атакам зловмисника. Перспективним напрямком розвитку сучасних механізмів та протоколів є використання MAC-кодів, які дозволяють (при рівних умовах з MDC-кодами) формувати геш-код з достатньою швидкістю для забезпечення автентичності та цілісності інформації в автоматизованих банківських системах.

РОЗДІЛ 2 ПОРІВНЯЛЬНІ ДОСЛІДЖЕННЯ МЕТОДІВ ГЕШУВАННЯ ЗАБЕЗПЕЧЕННЯ АВТЕНТИЧНОСТІ БАНКІВСЬКОЇ ІНФОРМАЦІЇ

2.1 Вибір критеріїв та показників ефективності гешування

На основі міжнародних стандартів ISO 7498-2 і ISO/IEC 10181 визначено, що оцінку геш-кодів можливо провести на основі досліджень можливості прояви колізій (формування однакового геш-коду від випадкових двох відкритих текстів). Тому для проведення статистичного дослідження колізійних властивостей формованих елементів використаємо методику в основі якої лежить емпірична оцінка максимумів кількості ключів (правил гешування) при яких:

1. Для довільних $x_1, x_2 \in A$, $x_1 \neq x_2$ виконується рівність

$$h(x_1) = h(x_2)\ ; \tag{2.1}$$

2. Для довільних $x_1 \in A$ та $y_1 \in B$ виконується рівність

$$h(x_1) = y_1; \tag{2.2}$$

3. Для довільних $x_1, x_2 \in A$, $x_1 \neq x_2$ та $y_1, y_2 \in B$ виконуються рівності

$$h(x_1) = y_1, h(x_2) = y_2\,. \tag{2.3}$$

Оцінка за першим критерієм відповідає перевірці здійсненності умови для універсального класу геш-функцій, оцінка по другому і третьому критерію – умов для строго універсального класу геш-функцій.

Введемо наступні позначення:

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

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

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

Перший показник $n_1(x_1, x_2)$ характеризує число правил гешування, при яких для заданих $x_1, x_2 \in A$, $x_1 \neq x_2$, виконується рівність (2.1), тобто число ключів, при яких існує колізія (збіг геш-кодів) для двох вхідних послідовностей x_1 і x_2.

Другий показник $n_2(x_1, y_1)$ характеризує число правил гешування, при яких для заданих $x_1 \in A$, $y_1 \in B$, виконується рівність (2.2), тобто число ключів, при яких для вхідної послідовності x_1 значення геш-коду y_1 не змінюється.

Третій показник $n_3(x_1, x_2, y_1, y_2)$ характеризує число 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 (1.3), тобто число ключів, при яких для двох вхідних послідовностей x_1 та x_2 відповідні їм значення геш-кодів y_1 і y_2 не змінюються.

Кількість ключів, при яких можуть виконуватися рівності (2.1–2.3), не повинне перевершувати відповідних їм значень $P_{кол} \cdot |H|$, $|H|/|B|$ та $P_{кол}|H|/|B|$.

Таким чином, визначим статистичні характеристики максимумів цих величин, а потім порівняємо отримані результати з числом $P_{кол} \cdot H$ (для першого критерію), з кількістю $|H|/|B|$ (для другого критерію) і кількістю $P_{кол}|H|/|B|$ (для третього критерію).

В якості статистичних показників оцінки колізійних властивостей використовуємо:

- математичні очікування $m(n_1)$ $m(n_2)$ та $m(n_3)$ максимумів кількості правил гешування, при яких виконуються рівності (2.1–2.3), відповідно;

- дисперсії, $D(n_1)$, $D(n_2)$ та $D(n_3)$, що характеризують розсіювання значень кількості правил гешування, при яких виконуються рівності (2.1–2.3), щодо їх математичних сподівань, $m(n_1)$, $m(n_2)$ та $m(n_3)$, відповідно.

Оцінку колізійних властивостей за наведеними критеріями будемо виробляти в середньостатистичному сенсі.

Оцінка *m* випадкової величини *X* є середнє арифметичне її значень X_i (або статистичне середнє):

$$\tilde{m} = \frac{1}{N}\sum_{i=1}^{N} X_i ,$$

де *N* – кількість реалізацій випадкової величини *X*.

Оцінка дисперсії випадкової величини *X* визначається виразом:

$$\tilde{D} = \frac{1}{N-1}\sum_{i=1}^{N} (X_i - \tilde{m})^2 ,$$

У силу центральної граничної теореми теорії ймовірностей при великих значеннях кількості реалізацій *N* середнє арифметичне буде мати розподіл, близький до нормального з математичним очікуванням:

$$m[\tilde{m}] \approx \tilde{m} ,$$

і середнім квадратичним відхиленням:

$$\sigma[\tilde{m}] \approx \frac{\sigma}{\sqrt{N}} ,$$

де σ – середнє квадратичне відхилення оцінюваного параметру.

При цьому вірогідність того, що оцінка відхилиться від свого математичного сподівання менше ніж на (довірча ймовірність), дорівнює:

$$P(|\tilde{m} - m| < \varepsilon) \approx 2\Phi\left(\frac{\varepsilon}{\sigma[\tilde{m}]}\right) , \qquad (2.4)$$

де $\Phi(x)$ – функція Лапласа, визначається виразом

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_0^x e^{-\frac{t^2}{2}} dt .$$

Таким чином, при проведенні експериментальних досліджень колізійних властивостей враховуємо:

1. З генеральної сукупності випадкової величини *X* сформуємо вибірку обсягу *N: X_1, X_2, ..., X_N*:

- для середньостатистичної оцінки математичного сподівання $m(n_1)$ і дисперсії $D(n_1)$ в якості випадкової величини виступає максимум $n_1(x_1, x_2)$ по всім $h(x_1) = h(x_2)$ для заданих x_1 і x_2, отже, вибірку сформуємо відбором з N пар елементів, $x_1, x_2 \in A$, $x_1 \neq x_2$;

- для середньостатистичної оцінки математичного сподівання $m(n_2)$ і дисперсії $D(n_2)$ в якості випадкової величини виступає максимум по всім $y_1 = h(x_1)$, отже, вибірку сформуємо відбором з N пар елементів $x_1 \in A$, $y_1 \in B$;

- для середньостатистичної оцінки математичного сподівання $m(n_3)$ і дисперсії $D(n_3)$ в якості випадкової величини виступає максимум $n_3(x_1, x_2, y_1, y_2)$ по всім парам $y_1 = h(x_1)$ та $y_2 = h(x_2)$, отже, вибірку сформуємо відбором з N четвірок елементів $x_1, x_2 \in A$, $x_1 \neq x_2$, $y_1, y_2 \in B$.

2. При експериментальних дослідженнях колізійних властивостей гешування:

- за першим критерієм будемо оцінювати середнє арифметичне $\tilde{m}(n_1)$ спостережуваних значень максимумів $n_1(x_1, x_2)$ і дисперсію $\tilde{D}(n_1)$;

- за другим критерієм будемо оцінювати середнє арифметичне $\tilde{m}(n_2)$ спостережуваних значень максимумів $n_2(x_1, y_1)$ і дисперсію $\tilde{D}(n_2)$;

- за третім критерієм будемо оцінювати середнє арифметичне $\tilde{m}(n_3)$ спостережуваних значень максимумів $n_3(x_1, x_2, y_1, y_2)$ і дисперсію $\tilde{D}(n_3)$.

3. Обґрунтуємо достовірність отриманих середньостатистичних оцінок. Для цього зафіксуємо точність ε і розрахуємо значення функції Лапласа, яка, відповідно до виразу (1.4), дає відповідні довірчі ймовірності:

$$P\left(\left|\tilde{m}(n_1) - m(n_1)\right| < \varepsilon\right) \approx 2\Phi\left(\frac{\varepsilon}{\sigma\left[\tilde{m}(n_1)\right]}\right),\ \sigma\left[\tilde{m}(n_1)\right] \approx \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}};$$

$$P\left(\left|\tilde{m}(n_2) - m(n_2)\right| < \varepsilon\right) \approx 2\Phi\left(\frac{\varepsilon}{\sigma\left[\tilde{m}(n_2)\right]}\right),\ \sigma\left[\tilde{m}(n_2)\right] \approx \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}};$$

$$P(|\tilde{m}(n_3) - m(n_3)| < \varepsilon) \approx 2\Phi\left(\frac{\varepsilon}{\sigma[\tilde{m}(n_3)]}\right),\ \sigma[\tilde{m}(n_3)] \approx \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}}.$$

При зворотній постановці завдання, тобто для фіксованої довірчої ймовірності P_∂ при обсязі вибірки N довірчий інтервал визначимо наступним чином:

$$\tilde{m} - t_\rho \cdot \sigma[\tilde{m}] < m < \tilde{m} + t_\rho \cdot \sigma[\tilde{m}],$$

де t_ρ – корінь рівняння $2\Phi(t_\rho) = P_\partial$.

Іншими словами, в цьому випадку межі довірчого інтервалу будуть відповідати заданій довірчій ймовірності P_∂, а точність оцінок визначається як,

$$\varepsilon = t_\rho \cdot \sigma[\tilde{m}].$$

Для нашого випадку при заданій імовірності P_∂ маємо:

$$\tilde{m}(n_1) - t_\rho \cdot \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}} < m(n_1) < \tilde{m}(n_1) + t_\rho \cdot \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}},\ \varepsilon = t_\rho \cdot \frac{\sqrt{\tilde{D}(n_1)}}{\sqrt{N}};$$

$$\tilde{m}(n_2) - t_\rho \cdot \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}} < m(n_2) < \tilde{m}(n_2) + t_\rho \cdot \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}},\ \varepsilon = t_\rho \cdot \frac{\sqrt{\tilde{D}(n_2)}}{\sqrt{N}};$$

$$\tilde{m}(n_3) - t_\rho \cdot \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}} < m(n_3) < \tilde{m}(n_3) + t_\rho \cdot \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}},\ \varepsilon = t_\rho \cdot \frac{\sqrt{\tilde{D}(n_3)}}{\sqrt{N}}.$$

На основі описаної методики будуть проводитись дослідження статистичної безпеки ключових геш-функцій.

2.2. Порівняльні дослідження методів гешування, які представлені на конкурс NESSIE

Європейський конкурс NESSIE проводився в Бельгії в 2000–2004 роках.

Головними критеріями відбору алгоритмів-конкурсантів серед методів симетричної та несиметричної криптографії, формування MAC- та MDC-кодів були:

- безпека – найважливіший критерій, тому що безпека алгоритмів шифрування - головне призначення цих алгоритмів. Процес оцінки безпеки враховував і вплив подій поза проектом NESSIE (таких як нові напади або методи аналізу);

- ринкові вимоги пов’язані з потребою в алгоритмі, його зручності і простоті використання, можливості міжнародного використання;

- продуктивність алгоритму шифрування на певному обладнанні, на основі сучасних процесорів;

- гнучкість алгоритму.

У фінальному звіті міжнародного криптографічного конкурсу NESSIE відзначена функція гешування Whirlpool (на основі власного убудованого блоково-симетричного шифру) і відібрані чотири кращі алгоритми формування MAC-кодів: UMAC, Two-Track-MAC, CBC-MAC, HMAC [14, 38].

UMAC – каскадний алгоритм MAC-коду, який розроблений Тедом Кроветцом (Ted Krovetz), Джоном Блеком (John Black), Шаї Халеві (Shai Halevi), Хьюго Кравцеком (Hugo Krawczyk) і Пилипом Рогевеєм (Phillip Rogaway) [13; 14]. Існує багато версій UMAC, але найбільш відомими є дві специфічні версії: UMAC-32 і UMAC-16. Головною різницею між двома версіями є довжина внутрішнього блоку, яка рівна 32 біта для UMAC-32 і 16 біт для UMAC-16. Довжина типової хеш-коду цих алгоритмів рівна 64 бітам, але є можливість отримати хеш довжиною до 32, 64, 96 або 128 біт для того, щоб забезпечити різні рівні стійкості. Для різних версій UMAC схема роботи алгоритму універсального хешування виглядає таким чином, як показано на рис. 2.1.

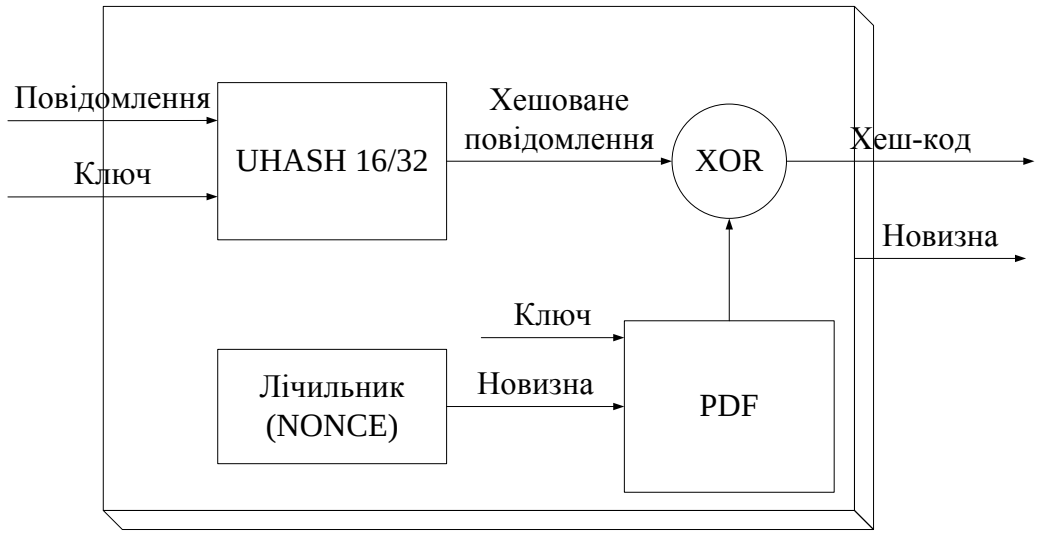

Рисунок 2.1 – Загальна схема роботи універсальних хеш-функцій UMAC-16/32

MDX-MAC, які можуть бути засновані на MD5, SHA, RIPEMD або аналогічних функціях хешу. Тут основна геш-функція перетворена в MAC невеликими модифікаціями, включаючи секретний ключ на початку, в кінці або в кожній ітерації функції хешу. Це досягається ключозалежною модифікацією початкової величини і константи значення, що додається, використовується хеш-функцією, і вихідним перетворенням, залежним від ключа. Безпека MDX-MAC може бути доведена на основі припущення, що основна функція стискування псевдовипадкова.

TTMAC (Two-Track-MAC) має найвищий рівень захисту для MAC примітивів, представлених NESSIE і за підсумковими показниками є переважним. Алгоритм TTMAC заснований на хеш-функціях RIPEMD-160 з невеликими модифікаціями. Алгоритм має певні переваги в швидкодії, особливо у випадку коротких повідомлень, і оптимальну довжину ключових даних [3]. Алгоритм працює на блоках 512 біт, розділених на слова по 32 біти, використовує секретний ключ 160 біти, і MAC код довжиною до 160 біт. Великий розмір внутрішнього стану (320 біт) в Two-Track-MAC дає алгоритму високий рівень захисту від атак, заснованих на внутрішніх колізіях. Схема роботи алгоритму Two-Track-MAC наведено на рис. 2.2.

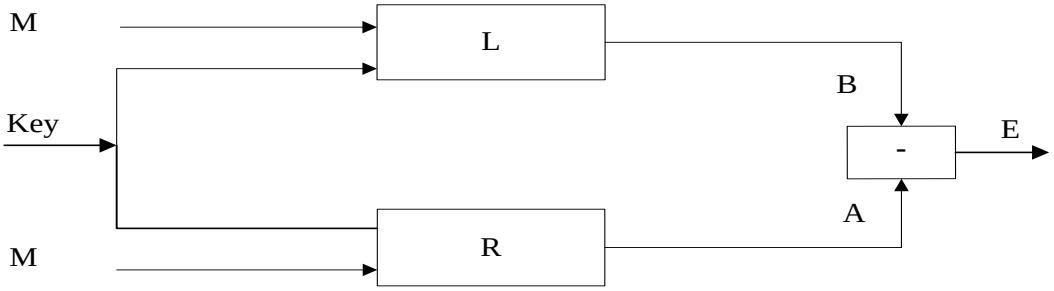

Рисунок 2.2 – Схема роботи алгоритму Two-Track-MAC

CBC-MAC (ISO/IEC 9797-1), до яких відносяться EMAC та RMAC.

Блоковий шифр E є функцією E : $K_E \times \{0,1\}^n \to \{0,1\}^n$, де кожна *E (K, •) = Eк* (•) є перестановкою на $\{0,1\}^n$, K_E це множина можливих ключів, а *n* довжина блоку.

CBC MAC найбільш простий і добре відомий алгоритм, щоб зробити MAC з блочного шифру *E*. Нехай *M* = *M* [1] ◦ *M* [2] ◦ … ◦ *M* [*m*] буде рядком повідомлення, де | *M* [1] | = | *M* [2] | = … = | *M* [*m*] | = *n*. Тоді CBC_K *(M),* CBC MAC з *M* з ключами *K,* , визначається як *Y* [*m*], де *Y* [*i*] = E_K *(M [i]⊕ Y[i –1]),* для *i* = 1,..., *m* і *Y* [0] = 0^n.

В існуючих реалізаціях MAC-алгоритмів CBC MAC Bellare, Kilian, і Rogaway запропонували EMAC.

Обчислення EMAC для секретного ключа K і повідомлення X – поділеного (після заповнення) на 128-бітові блоки X_1, X_t – оброблених таким чином (E_K позначає шифрування з використанням 128-бітового блокового шифру AES на ключі *K*):

1. Обчислити $H_1 = Ek_1(X_1)$.

2. Для *i* = 2, *t*: обчислити $H_i = Ek_i(X_i \oplus H_{i-1})$.

3. Обчислити вихідне перетворення: *Hout* = $Ek_2(Ht)$. Ключ K_2 може бути похідній від K_1, отриманою в ході наступної процедури: $K_2 = K_1 \oplus f0f0...f0_x$.

4. Для того, щоб отримати величину *m*-бит MAC, виберіть найлівіші *m* біти *Hout*.

RMAC є випадковим варіантом цієї схеми, що забезпечує кращу протидію атаці на основі внутрішніх помилок. Єдина відмінність – у вихідному перетворенні, де можна кодувати на ключі, отриманому порозрядним доповненням K_2 і R: для RMAC, ключі K_1 і K_2 можуть бути незалежними або похідними від одного головного ключа стандартним чином. R є значенням r біт завдовжки і повинно бути заповнено 0 – бітами якщо воно менше, ніж K_2. П’ять різних наборів параметрів було визначено для розмірів m і r.

Безпека EMAC може бути доведена на базі припущення, що основний блоковий шифр – псевдо-випадковий, у такому разі Rijndael, який був схильний до аналізу впродовж тривалого часу в течію і після процесу AES, реалізує вимоги безпеки.

Основна перевага випадкового варіанту RMAC – те, що надається краща протидія атаці, заснованій на внутрішніх помилках. З іншого боку, RMAC необхідні міцні основи для доказу безпеки шифру; наприклад, основний блоковий шифр повинен бути безпечним проти атаки зчепленого ключа.

Схема роботи алгоритму CBC-MAC наведено на рис. 2.3.

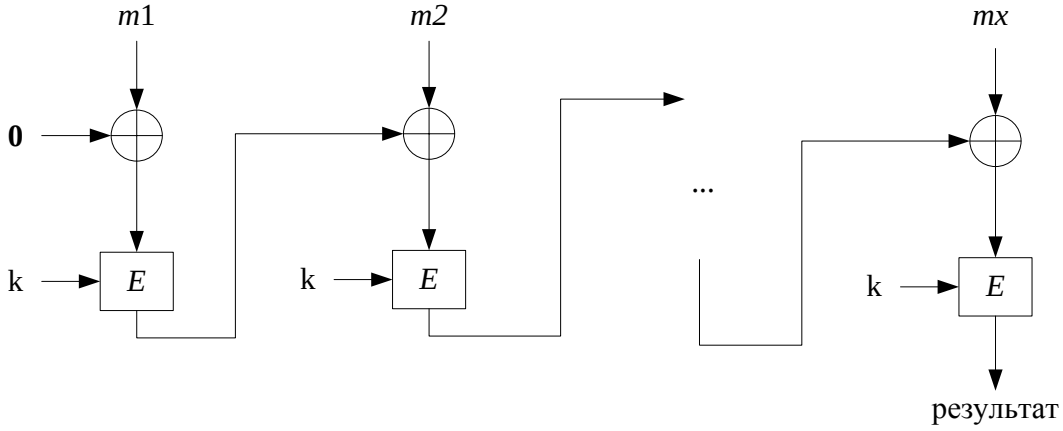

Рисунок 2.3 – Схема роботи алгоритму CBC-MAC

HMAC (ISO/IEC 9797-1).

Формування MAC-кодів за допомогою HMAC:

$$HMAC(K,M) = h\big((K \oplus opad) \parallel h(K \oplus ipad) \parallel M\big),$$

де h — геш-функція, К — секретний ключ, доповнений нулями до розміру блоку, М —повідомлення для ідентифікації, ІІ – конкатенація,

opad — 0x5c5c..5c (довжина дорівнює розміру блока)

ipad — 0x3636..36 (довжина дорівнює розміру блока)

Ключ К спочатку заповнюється нульовими бітами, і opad і ipad – постійні величини. Беллейр (Bellare) [22] довів безпеку цієї конструкції наступними припущеннями: основна функція хешу є стійкою проти помилок для секретної початкової величини; функціональною по ключу стискування початкової величиною – безпечний алгоритм MAC (для повідомлень одного блоку); функція стискування є слабкою псевдо-випадковою функцією.

Схема роботи алгоритму HMAC наведено на рис. 2.4.

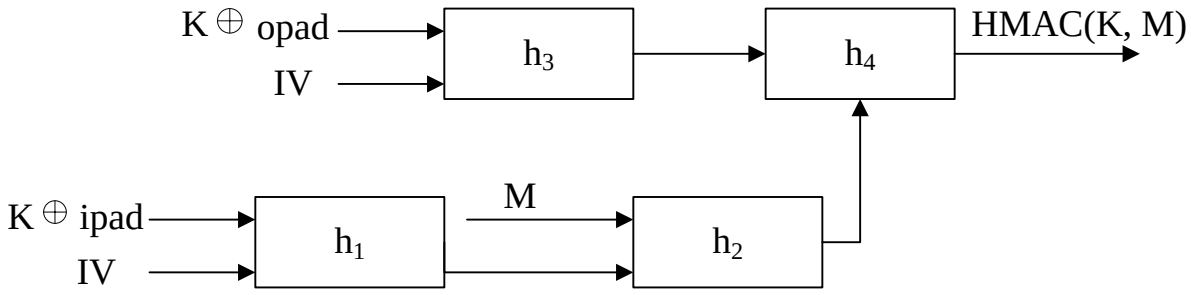

Рисунок 2.4 – Схема роботи алгоритму HMAC

В табл. 2.1. наведено аналіз швидкості роботи алгоритмів гешування.

Таблиця 2.1 – Аналіз тестування швидкості роботи алгоритмів гешування

Функція гешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
Whirlpool	10	C	28,013 Мбіт/с	46,961 Мбіт/с
SHA-2 (512)	80	C	41,159 Мбіт/с	68,701 Мбіт/с
SHA-2 (256)	64	C	81,308 Мбіт/с	135,557 Мбіт/с
ГОСТ 34311-95	-	C+Assembler	49,408 Мбіт/с	83,056 Мбіт/с
HAVAL	96(128, 160)	C	337,842 Мбіт/с	564,809 Мбіт/с
SHA-1	80	C, Assembler	206,285 Мбіт/с 361,581 Мбіт/с	344,433 Мбіт/с 605,558 Мбіт/с
RIPEMD-160	160	C	147,465 Мбіт/с	246,568 Мбіт/с
MD5	64	C	278,715 Мбіт/с	574,635 Мбіт/с
MD4	48	C	344,086 Мбіт/с	467,793 Мбіт/с
UMAC	-	C C+Assembler	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с
ГОСТ 28147-89 (OFB)	16	C+Assembler	189,559 Мбіт/с	315,270 Мбіт/с

Як видно з аналізу табл. 2.1 Two-Track-MAC уступає за швидкістю тільки UMAC алгоритму, приблизно в 3–7 разів, але має в 2,5 рази більше розмір MAC коду. В табл. 2.2 наведено можливі значення довжин ключа і геш-коду для різних алгоритмів сімейства MAC.

Таблиця 2.2 – Довжина ключа k і довжина геш-коду n для MAC-кодів.

Алгоритм	k (ключ)	n (геш-код)
UMAC	128	64
TTMAC	160	≤ 160
EMAC-AES	128,192,256	≤ 128
RMAC-AES	128,192,256	≤ 128
HMAC-SHA-1	≤ 512	≤ 160

У табл. 2.3 наведено основні результати оцінки швидкодії Two-Track-MAC алгоритму для різних операційних платформ. Швидкість обчислень визначається кількістю циклів процесора, затрачуваних на один байт повідомлення.

Таблиця 2.3 – Швидкодія MAC алгоритмів

Алгоритм	Довжина MAC-коду (біт)	Довжина ключа (біт)	Тип ПЕВМ				
			Pentium2	PIII/Linux	Pentium4	Xeon	AMD
TTMAC	160	160	21	21	40	37	21
UMAC-16	64	128	6.1	6.0	6.2	6.1	6.2
UMAC-32	64	128	2.5	2.9	6.7	6.6	1.9
HMAC-Whirlpool	512	512	86	72	98	103	100
HMAC-MD4	128	512	4.7	4.7	6.4	6.4	4.7
HMAC-MD5	128	512	7.2	7.3	9.4	9.4	7.4
HMAC-RIPE-MD	160	512	23	18	27	26	21
HMAC-SHA-0	160	512	16	15	23	23	13
HMAC-SHA-1	160	512	16	15	25	24	12
HMAC-SHA-2	256 384 512	512	40 84 84	39 84 84	40 124 124	39 132 132	33 72 72
HMAC-Tiger	192	512	24	21	28	26	20
CBC MAC-Rijndael (EMAC)	128	128	24	26	26	27	31
CBC MAC-DES	64	56	62	61	72	69	54
CBC MAC-Shacal	512	160	31	31	67	74	29

Аналіз табл. 2.3 підтверджує можливість використання алгоритма UMAC, як самого швидкісного алгоритму формування MAC-коду..

Проведений аналіз табл. 2.1–2.3 показав, що на сьогоднішній час ключовими критеріями відбору є швидкодія перетворень, стійкість до сучасних загроз. Таким чином сучасні алгоритми гешування повинні забезпечувати [2, 3, 10]:

- складність знаходження колізії $2^{n/2}$;
- складність відновлення прообразу 2n;
- складність знаходження другого прообразу не менше 2n;
- стійкість до атак length extension;
- стійкість до усічених колізій;
- стійкість до атак мультиколлізій;
- відсутність атак розпізнавання для генераторів псевдовипадкових послідовностей, що використовують HMAC, побудованих на базі геш-функції зі складністю, меншою ніж знаходження другого прообразу і кількістю запитів до генератора не менше $2^{n/2}$;
- надійність математичної бази.

2.3 Порівняльні дослідження методів безпечного гешування, які представлені на конкурс SHA-3

У 2008 році у зв'язку з можливістю зламу на основі квантового комп'ютера геш-кодів, які використовуються в сучасних АБС Національним інститутом стандартів і технологій США (NIST) був відкритий конкурс SHA-3 [3].

Основними вимогами конкурсу SHA-3 є:

розмір вихідного блоку 224, 256, 384 і 512 бітів,

довжина геш-коду повинна перевищувати 160-біт;

колізійна стійкість;

стійкість до знаходження прообразу і другого прообразу;

потоковий режим обчислення "за один прохід".

У табл. 2.4 наведені основні характеристики алгоритмів гешування, які представлені на конкурс.

Таблиця 2.4 – Характеристики алгоритмів гешування кандидатів 2 раунду конкурсу

Назва алгоритму	Платформа	Стійкість алгоритму	Швидкодія (Мбіт/с)
BLAKE	8, 32, 64	є достатньо стійким	44,37
Blue Midnight Wish	8, 32, 64	нестійкий до псевдо атак – колізій і 2х прообразів	62,05
CubeHash	8, 32, 64		22,13
ECHO	32, 64	Стійких до всіх видів атак	9,88
Fugue	32, 64		18,27
Gröstl		Нестійкий до атак “напіввільний початок”	7,98
Hamsi	32, 64	низька алгебраїчна ступінь у висновках функції стиснення	8,84
JH	8, 32, 64	низька алгебраїчна ступінь у висновках функції стиснення	10,6
Keccak	32, 64	кількість раундів, необхідне для захисту від атак – 18	8,05
Luffa	8, 32, 64	Стійкість до знаходження прообразів	24,77
Shabal	32, 64	Стійкий до всіх видів атак	128,5
SHAvite-3	32, 64	Нестійкий до знаходження псевдо-прообразів	28,32
SIMD	32, 64	Достатня	1,58
Skein	8, 32, 64	Захищений від атак – підбору подовжених повідомлень і псевдо колізій	39,23

Аналіз табл. 2.4 підтверджує основі вимоги: безпеку, надійність та швидкість .

У 2 раунд були відібрані алгоритми гешування, в яких не була порушена будь-яким чином безпеку. У деяких випадках, представлений варіант був занадто повільним чи вимагав занадто багато пам’яті, але NIST вважає, що в деяких випадках, настроюються параметри можуть бути відкоректовані, щоб дати прийнятний рівень продуктивності без шкоди для безпеки [3].

Короткі характеристики кандидатів, відібраних на 2 тур, наведені у дод. А.

Попереднє вивчення алгоритмів-учасників показало, що кожен з них запозичив деякі принципи побудови у алгоритму AES для забезпечення стійкості геш-кодів.

Перспективним напрямком подальших досліджень є аналіз статистичної стійкості алгоритмів-конкурсантів 2 раунди, їх реалізація на різних платформах з метою оцінки продуктивності та адаптації їх застосування в банківських транзакціях документообігу внутріплатіжних банківських систем

2.4 Обґрунтування вибору методу гешування для забезпечення автентичності БіН в АБС

Одним із основних додатків застосування функцій гешування є електронний цифровий підпис. У зв’язку з широким впровадженням ЕЦП практично в усі інформаційно-телекомунікаційні системи (ІТС) та необхідності обробки інформації в реальному часі до функцій гешування зросли вимоги, перше за все в частині забезпечення криптографічної стійкості та прийнятної складності (швидкості) гешування [2–5].

Для перевірки стійкості геш-кодів пропонується використовувати Методику тестування NIST STS 822, яка дозволяє визначити міру випадковості на виході алгоритму гешування. Основні вимоги NIST STS 822:

1. Для кожного геш-коду необхідно оцінити та прийняти рішення про те, що він формує випадкові двійкові послідовності. Алгоритм гешування формує двійкову послідовність $S = \{s_1, s_2, \ldots, s_n\}$, $s_i \in \{0,1\}$ довільної довжини n.

2. Для фіксованого значення n формують множину з m двійкових послідовностей: $S_1 = \{s_1, s_2, \ldots, s_n\} \| S_2 = \{s_1, s_2, \ldots, s_n\} \| \ldots \| S_m = \{s_1, s_2, \ldots, s_n\}$.

Загальна кількість геш-кодів складає $N = m \times n$.

3. Кожну послідовність перевіряють з використанням пакету NIST STS. У результаті формується статистичний портрет генератора.

Статистичний портрет генератора – матриця розмірністю $m \times q$, де m – кількість двійкових послідовностей, що перевіряються, а q – кількість статистичних тестів, що використовуються для тестування кожної послідовності. Елементи матриці $P_{ij} \in [0,1]$, де $i = \overline{1,m}$, $j = \overline{1,q}$ представляють собою значення ймовірності, яка отримана у результаті тестування i-ої послідовності j-им тестом.

4. За отриманим статистичним портретом визначають долю послідовностей, що пройшли кожний статистичний тест. Для цього задають рівень значимості $\alpha \in [0{,}001, 0{,}01]$ та здійснюють підрахунок значень ймовірностей $P - value$, що перевищують встановлений рівень α для кожного з q тестів, тобто визначають коефіцієнт [19].

$$r_j = \frac{\#\left\{ P_{ij} \geq \alpha \middle| i = 1,2,\ldots,m \right\}}{m}.$$

У результаті формується вектор коефіцієнтів $R = \{r_1, r_2, \ldots, r_q\}$, елементи якого характеризують, у відсотках, проходження послідовності S_i всіх статистичних тестів.

5. Здійснюється статистичний аналіз статистичного портрету. Отримані значення імовірностей P_{ij} повинні підкорятися рівноймовірному закону розподілу на інтервалі [0, 1] [19]. Для кожного вектору-стовпцю статистичного портрету будується гістограма частоти F_k потраплянь значень P_{ij} у кожний з

k = 1, 2, …, 10 під інтервалів, на які розбитий інтервал [0, 19]. Рівноймовірність розподілу значень ймовірностей P_{ij} перевіряється з використанням критерію χ^2. Для цього розраховується статистика виду:

$$\chi_j^2 = \sum_{k=1}^{10} \frac{\left(F_k - m/10\right)^2}{m/10},$$

яка підкоряється розподілу χ^2 с дев'ятьма ступенями волі.

Вважається, що генератор G пройшов тестування за j-им тестом, якщо виконується умова $P(\chi_j^2) > 0,0001$.

6. Прикінцеве рішення приймають у відповідності з правилом: вважається, що генератор G пройшов статистичне тестування пакетом NIST STS, якщо значення коефіцієнтів r_j для всіх $j = \overline{1,q}$ знаходяться в межах довірчого інтервалу [r_{max}, r_{min}] та виконується умова $P(\chi_j^2) > 0,0001$ для всіх $j = \overline{1,q}$.

Завдяки значній кількості тестів, що використовуються, та гнучкості їх використання методика статистичного тестування NIST STS є досить гнучким та ефективним інструментом випробувань.

2.5 Висновки до розділу 2

Критерії безпеки, продуктивності і можливості роботи на великому числі платформ відносяться до основних NIST-критеріїв відбору, саме це дозволило керівникам конкурсу проводити відбір серед кандидатів. NIST алгоритми, що мають рівень продуктивності зіставний або краще, ніж у SHA-2.

Перспективним напрямком подальших досліджень є аналіз статистичної стійкості алгоритмів-претендентів конкурсу з відбору стандартного алгоритму SHA-3, їх реалізація на різних платформах з метою оцінки продуктивності та адаптації їх застосування в системах документообігу внутріплатіжних банківських систем, дозволить вибрати переможця, який буде задовольняти всім критеріям.

РОЗДІЛ 3 ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ ГЕШУВАННЯ

3.1 Розробка програмної реалізації методу гешування

Розроблений програмний продукт призначений для реалізації методу безпечного гешування для забезпечення цілісності і автентичності інформації в автоматизованих банківських системах.

Для розробки програмної реалізації було обрано мову програмування Java. Це об’єктно-орієнтована мова програмування, що розроблена компанією Sun Microsystems. Програми Java компілюються в спеціальний байт-код, тому вони можуть працювати на будь-який віртуальній Java-машині (JVM) незалежно від комп’ютерної архітектури. У дод. В наведені діаграми бізнес варіантів та варіантів використання розробленого програмного модулю.

До функціональних можливостей програмного продукту можна віднести наступні:

- можливість гешування повідомлення (довжиною $< 2^{64}$) будь яким з 14 обраних алгоритмів (причому повідомлення можна вводити як вручну, так і завантажувати з файлу);

- проводити перевірку повідомлення на автентичність;

- генерувати дані, які будуть у подальшому використовуватись в якості вхідних для програмного пакету NIST STS;

- завантажувати файл-звіт, що надає програмний пакет NIST STS після обробки даних, та будувати статистичний портрет;

- проводити різні налаштування пов’язані з параметрами графіка.

Результати роботи програмного продукту наведені на рис. 3.1 – 3.9. Лістинг програми наведено у дод. Г.

Головне вікно програми показано на рис. 3.1, воно складається з трьох вкладок, кожна з яких в свою чергу реалізує описаний вище функціонал.

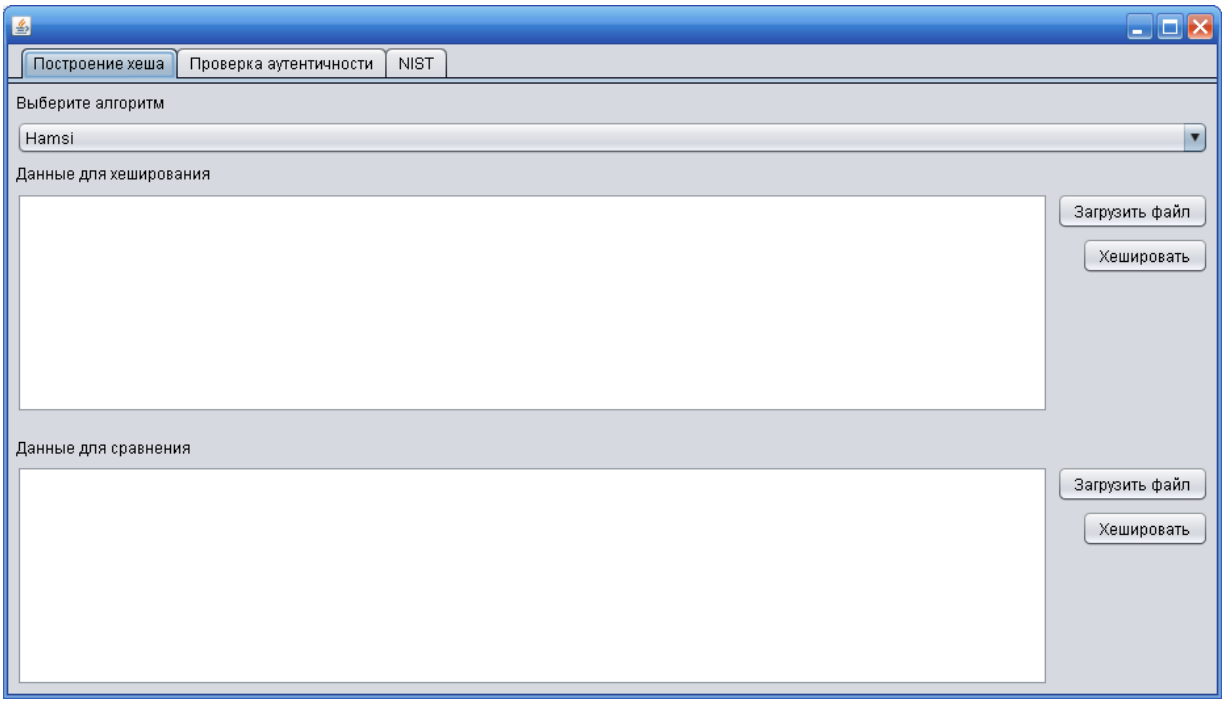

Рисунок 3.1 – Головне вікно програми

На вкладці “Построение хеша” можна обрати один з 14 вищеописаних алгоритмів (рис. 3.2).

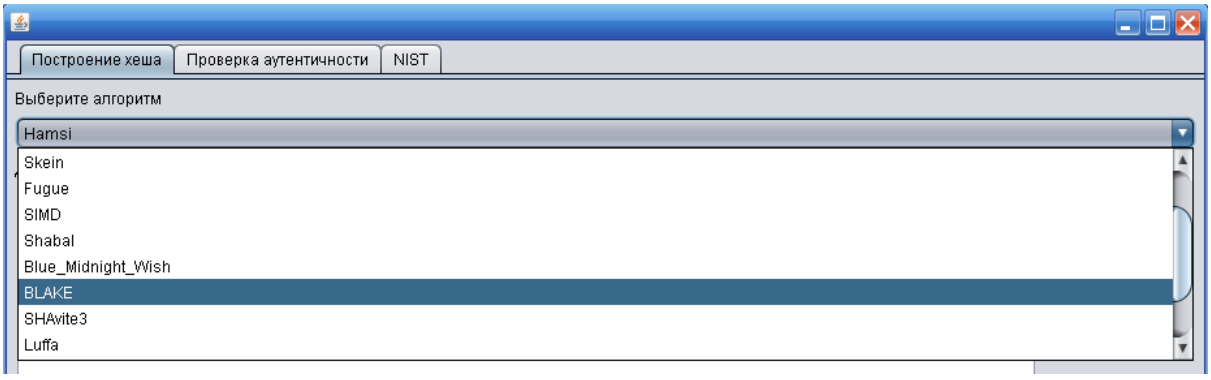

Рисунок 3.2 – Вибір алгоритму для гешування повідомлення

Вхідне повідомлення можна вводити як вручну в область “Данные для хеширования” або завантажувати файл. Для того щоб завантажити дані необхідно натиснути на кнопку “Загрузить файл” та в діалоговому вікні обрати потрібний файл. Дане діалогове вікно має стандартний windows інтерфейс, тому ніяких проблем при роботі з ним не повинно виникати.

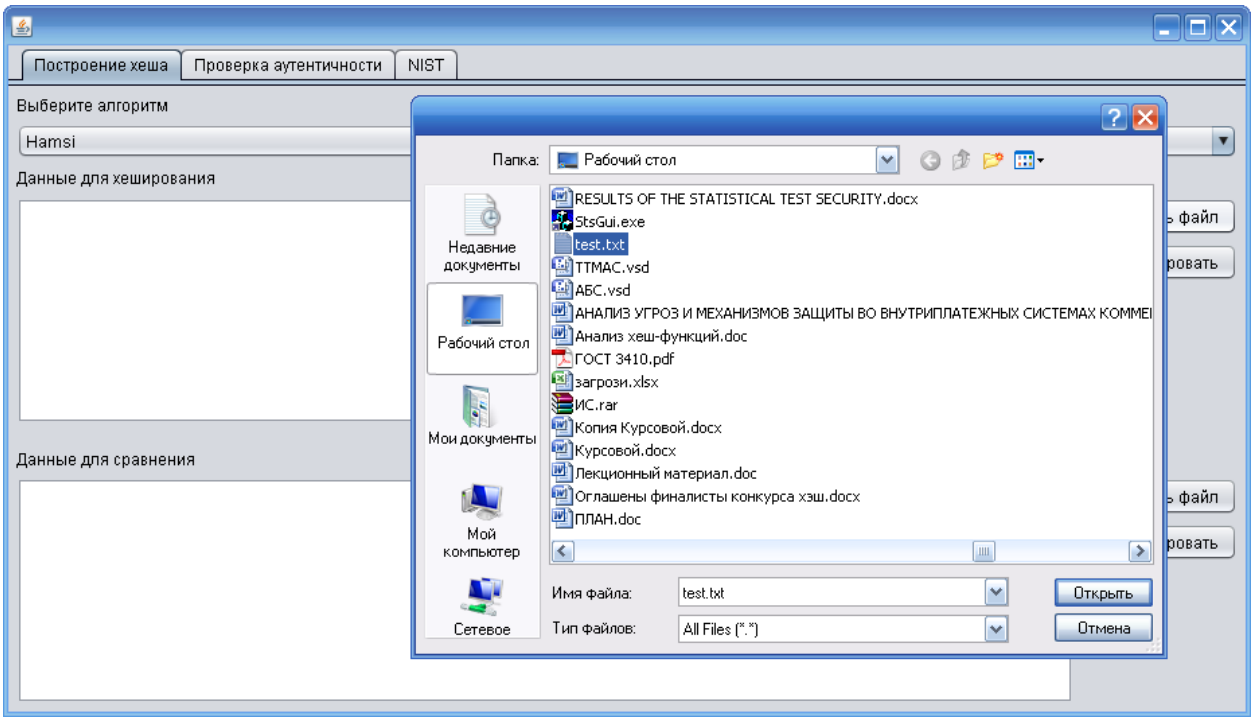

Рисунок 3.3 – Завантаження файлу з повідомленням

Щоб отримати геш повідомлення необхідно натиснути на кнопку “Хешировать”, а результат гешування буде відображено на вкладці “Проверка аутентичности”.

На вкладці “Проверка аутентичности” можна побачити геш введених повідомлень (області “Хеш 1” та “Хеш 2”), зберегти їх в окремий файл (кнопки “Сохранить в файл”) а також виконати їх порівняння (кнопка “Сравнить”) результат чого відобразиться в області “Результат сравнения”.

Для того щоб виконати перевірку на автентичність необхідно спочатку на вкладці “Построение хеша” в область “Данные для хеширования” та “Данные для сравнения” ввести вручну або завантажити файл з повідомленням, а потім в області “Данные для сравнения” внести зміни. Після виконання цих дій необхідно загешувати повідомлення (рис. 3.4).

Для того щоб побачити як зміна повідомлення вплинула на його геш виконаємо їх порівняння. Треба перейти на вкладку “Проверка аутентичности” и натиснути на кнопку “Сравнить”. На рис. 3.5 показаний результат порівняння повідомлень.

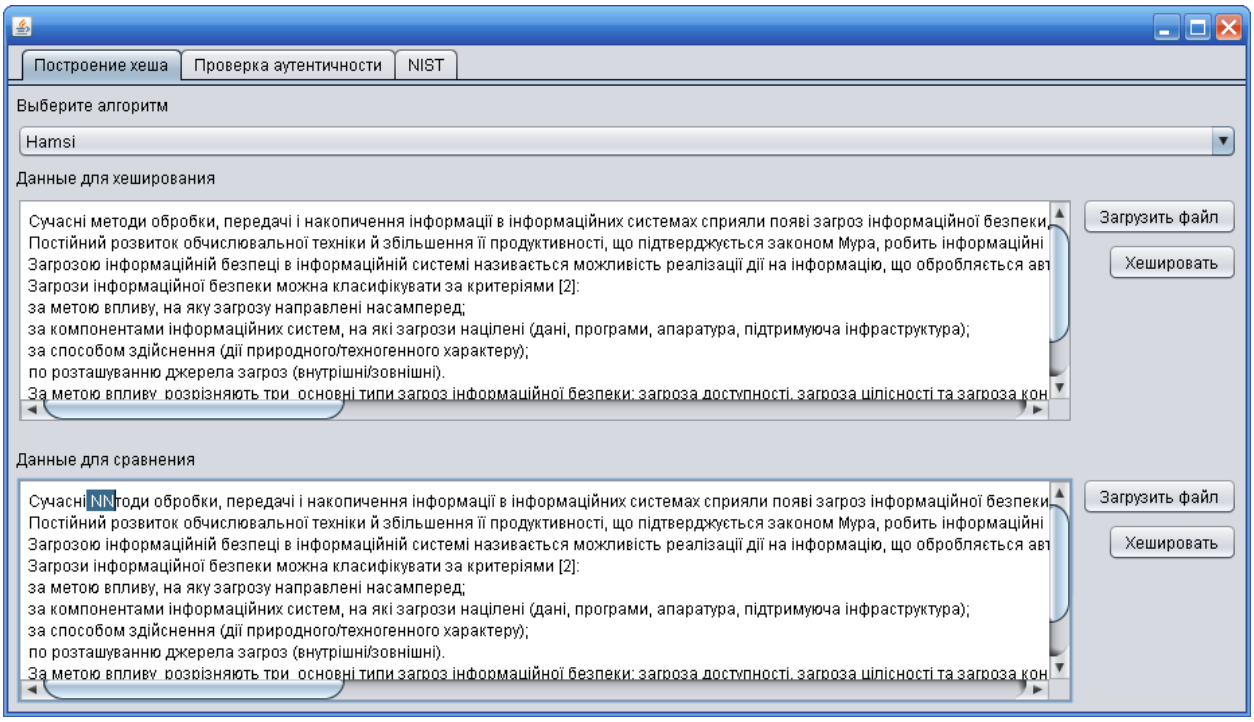

Рисунок 3.4 – Перевірка повідомлення на автентичність

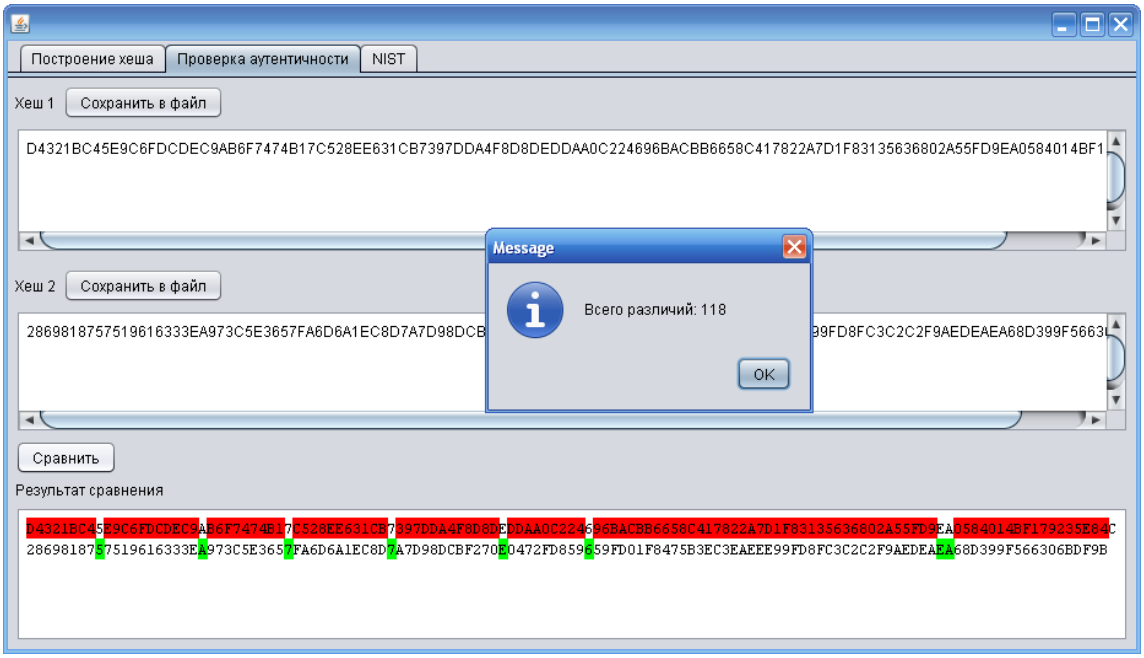

Рисунок 3.5 – Результат порівняння повідомлень

Як показано на рис. 3.5 зміна 2 бітів повідомлення спричинила 118 відмінностей, в області “Результат сравнения” більш детально показано, чим саме відрізняються отримані геші повідомлень (зеленим кольором показані співпадіння, а червоним – відмінності).

На вкладці “NIST” виконується генерація данних, які в якості вхідних приймає програмний пакет NIST STS.

Щоб отримати необхідний набір даних необхідно з випадаючого списку обрати алгоритм, задати розмір вхідного повідомлення в байтах, та задати кількість гешувань. Так. Як відомо, що програмний пакет NIST STS приймає в якості вхідних даних послідовність кратну 10^6 бітам, ми сформуємо 100 таких послідовностей. Для цього нам необхідно загешувати 195 313 різних повідомлень ((100*10^6) / 512). Розмір вхідного повідомлення оберемо довільним чином з єдиною умовою, щоб він не перевищував 2^{64} біт. На рис 3.6 показано формування даних для пакету NIST STS.

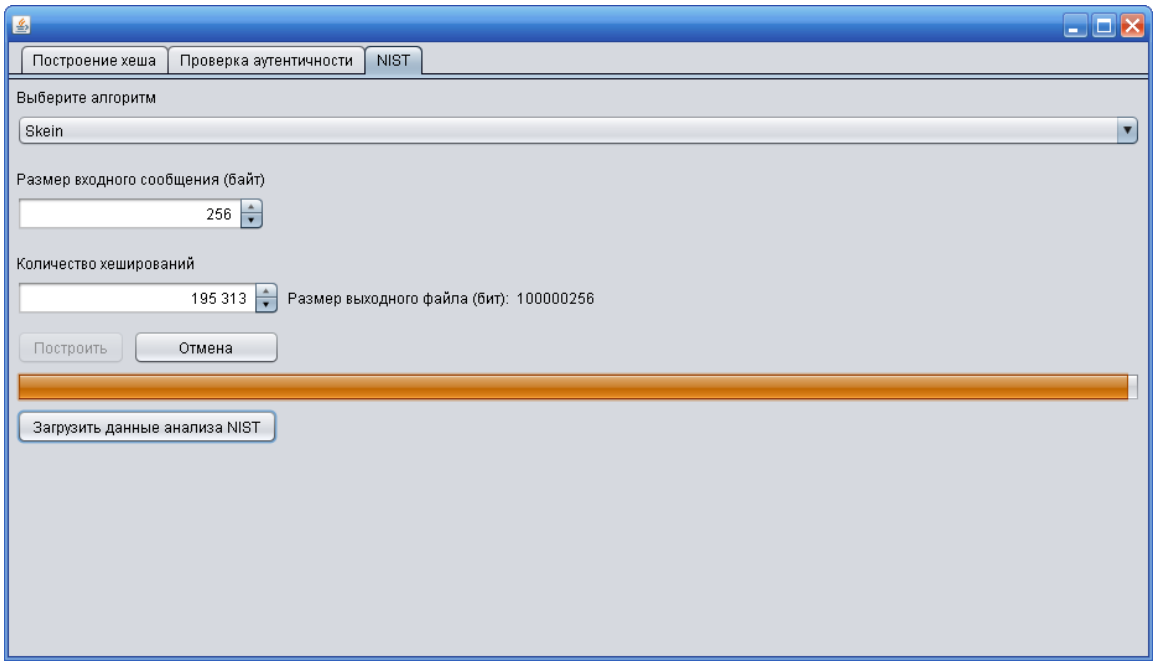

Рисунок 3.6 – Формування даних для пакету NIST STS

Після того як дані згенеровано, програма запропонує зберегти їх до окремого файлу. Після виконання цих дій отримані дані відкриваються за допомогою програмного пакету NIST STS і проводиться тестування. Згодом отриманий за допомогою програмного пакету NIST STS файл-звіт можна відкрити, для чого необхідно натиснути на кнопку “Загрузить данные для анализа NIST” та в діалоговому вікні обрати необхідний файл (рис. 3.7).

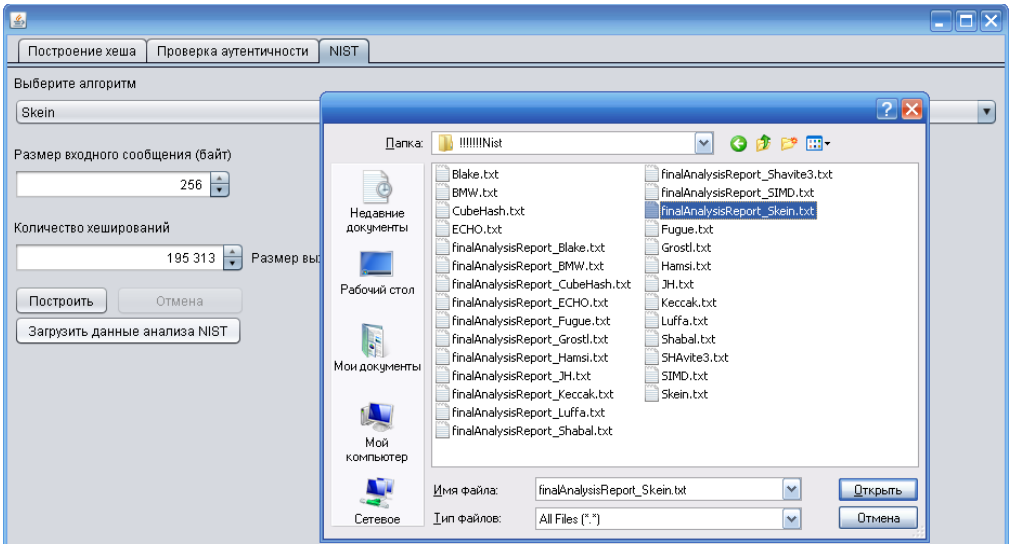

Рисунок 3.7 – Завантаження файлу-звіту отриманого за допомогою програмного пакету NIST STS

Після завантаження даних будується графік, що являє собою статистичний портрет програмної реалізації алгоритму (рис. 3.8).

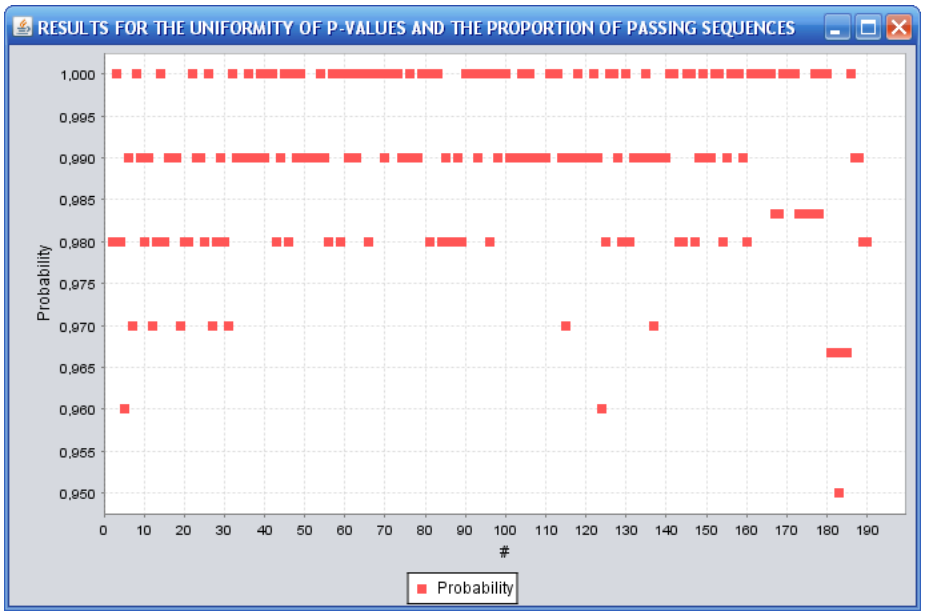

Рисунок 3.8 – Статистичний портрет програмної реалізації алгоритму

У вікні з графіком за допомогою контекстного меню можна виконувати різні налаштування параметрів області побудови графіку, а саме підписувати осі,

виконувати масштабування рисунку, задавати фон, зберегти графік у окремий файл та ін. (рис. 3.9).

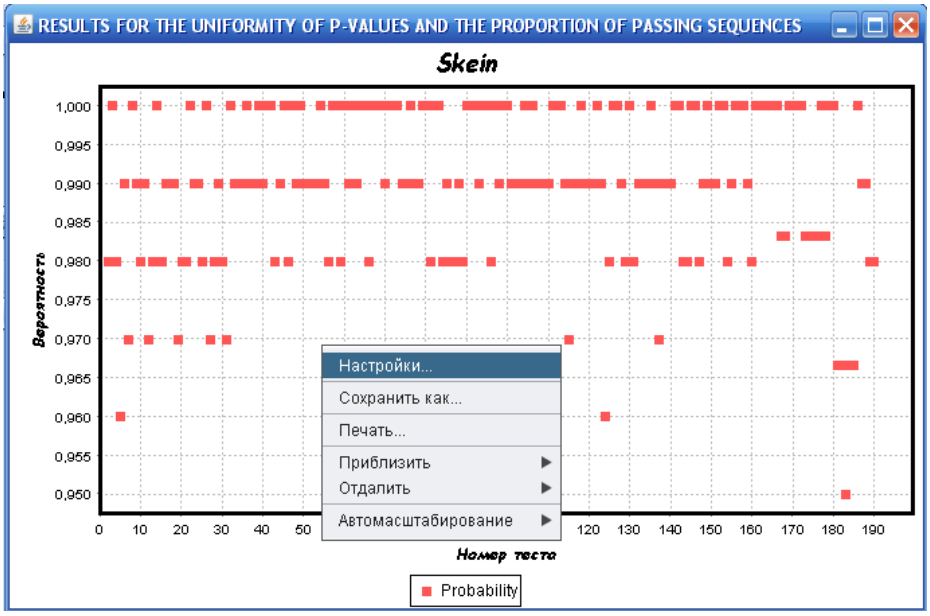

Рисунок 3.9 – Налаштування параметрів графіку

За допомогою розробленого програмного продукту у подальшому будуть проведені статистичні дослідження властивостей всіх алгоритмів.

3.2 Експериментальні дослідження швидкості алгоритмів гешування з використанням розробленої програмної реалізації

До основних критеріїв у конкурсі з відбору стандартного геш алгоритму SHA-3 керівники NIST віднесли продуктивність і можливість роботи на великому числі платформ, що дозволило "відсіяти" кілька кандидатів (алгоритми Vortex, LUX і т.д.). Тому алгоритми, що мали рівень продуктивності зіставний або краще, ніж у SHA-2, були відібрані до 2-го раунду. Продуктивність представляє собою пропускну здатність для довгих повідомлень, без урахування часу, необхідного для читання першого блоку повідомлення, ініціалізації, завершення і запису геш-значення в пам’ять. Продуктивність визначається за наступною формулою:

$$Thr = \frac{Block\_size}{T \cdot (HTime(N+1) - HTime(N))},$$

де *block_size* є розміром блоку повідомлення, характерного для кожної геш-функції, *HTime(N)* це загальне кількість тактів необхідне для гешування повідомлення з N – блоків, T – тактовий період, різний і характерні для кожної апаратної реалізації конкретної геш-функції.

На рис. 3.10, 3.11 показані показники продуктивності алгоритмів-претендентів на різних платформах в порівнянні з існуючим стандартом SHA-2.

Xilinx Virtex 5 FPGAs надвелика інтегральна схема програмованої логіки (FPGA), виконане з дотриманням норм 65-нм технологічного процесу – Virtex-5.

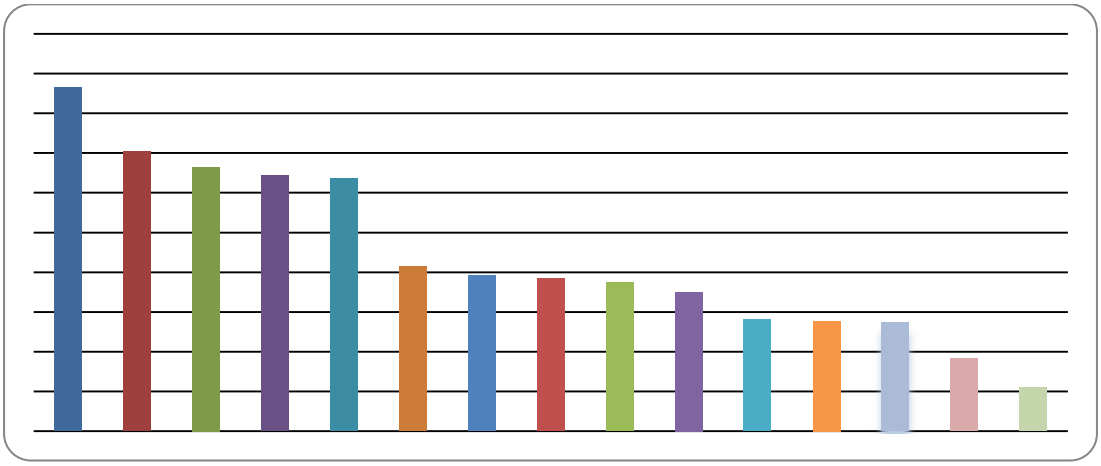

Рисунок 3.10 – Показники продуктивності (в Мбіт / с.) алгоритмів претендентів SHA-3 (512 бітний варіант) реалізований на Xilinx Virtex 5 FPGAs

Проведений аналіз рис. 3.10 показав що, алгоритми Hamsi та Fugue за своєю продуктивністю поступаються існуючому стандарту SHA-2, а найвищі показники має алгоритм BMW.

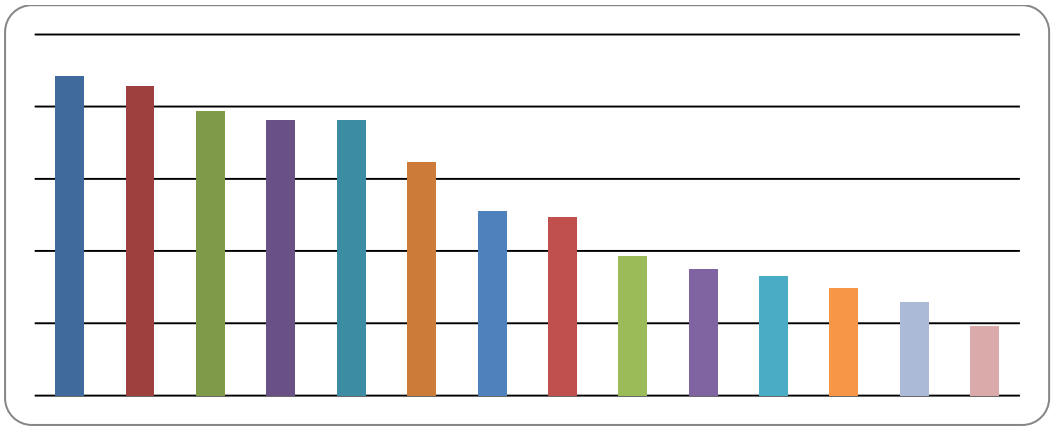

Рисунок 3.11 – Показники продуктивності (в Мбіт / с.) алгоритмів претендентів SHA-3 (512 бітний варіант) реалізований на Altera Stratix III FPGAs

Проведений аналіз рис. 3.11 показав що, алгоритми Hamsi та Shabal за своєю продуктивністю поступаються існуючому стандарту SHA-2, а найвищі показники має алгоритм Groestl.

З огляду на те, що програмний продукт реалізовано на Java, проведемо дослідження продуктивності на Java (апаратна конфігурація: Intel x86 Q6600, 2.4 GHz, 32-бітний режим) (рис. 3.12).

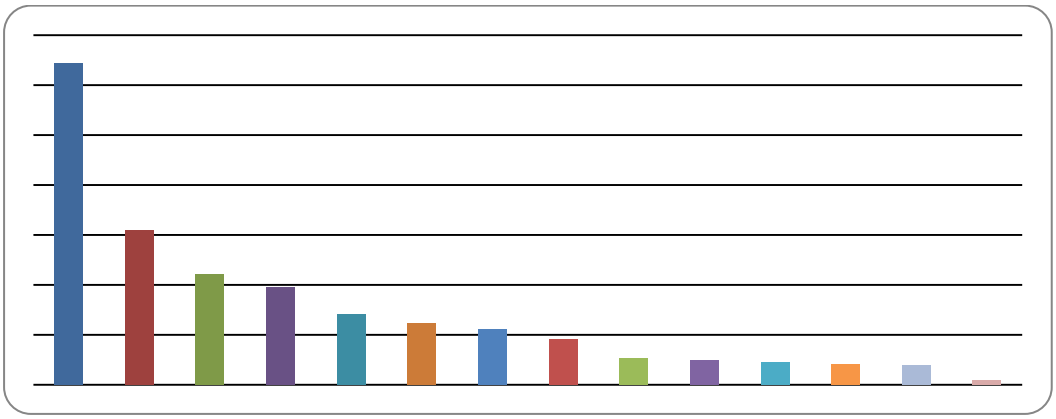

Рисунок 3.12 – Показники продуктивності (в Мбіт / с.) алгоритмів претендентів SHA-3 (512 бітний варіант) реалізований на Intel x86 Q6600

Проведений аналіз рис. 3.12 показав, що найнижчі показники, при даній апаратній конфігурації, має алгоритм SIMD, а найкращі Shabal.

У табл. 3.1 наведено показники продуктивності роботи (у порядку спадання) алгоритмів-претендентів конкурсу з відбору стандартного геш-алгоритму SHA-3 на різних платформах.

Таблиця 3.1 – Показники продуктивності роботи алгоритмів-претендентів на різних платформах

Найменування алгоритму	Продуктивність (Мбіт/с)		
	Xilinx Virtex 5 FPGAs	Altera Stratix III FPGAs	Intel x86 Q6600
BMW	8655,87	7618,56	62,05
Luffa	7043,81	8576,64	24,77
Grostl	6361,09	8841	7,98
ECHO	6430,88	7872	9,88
Keccak	6644,52	6470,64	8,05
Skein	2811,72	7618,56	39,23
SIMD	4138,55	4935,68	1,58
JH	3917,97	5104,92	10,6
SHAvite3	3834,56	3869,28	28,32
Blake	3743,28	3298,34	44,37
CubeHash	3508,77	3488,8	22,13
Shabal	2770,94	2589,49	128,5
Hamsi	1828,05	1932,37	8,84
Fugue	1107,88	1650,16	18,27

Подальшими діями в даному напрямку буде проведення дослідження статистичних властивостей алгоритмів претендентів.

3.3 Експериментальні дослідження статистичної безпеки гешування з використанням пакету NIST STS

Дослідження стійкості геш алгоритмів-претендентів конкурсу по відбору стандартного геш алгоритму SHA-3 проводилося відповідно до методики тестування NIST SP 800-22, рекомендованої Національним інститутом по стандартизації і технологіям США.

Статистичні портрети програмних реалізацій алгоритмів претендентів конкурсу з відбору стандартного геш-алгоритму SHA-3 наведені у дод. Б.

Таблиця 3.2 – Результати тестування геш алгоритмів-претендентів

Назва алгоритму	Кількість тестів, у яких тестування пройшло 99% послідовностей	Кількість тестів, у яких тестування пройшло 96% послідовностей	Кількість тестів, в яких значення ймовірності $P \leq 0{,}01$	Кількість тестів, в яких значення ймовірності $P \leq 0{,}001$	Кількість тестів, у яких значення ймовірності $P \leq 0{,}05$	Припустиме значення проходження тесту вибірки розміром 100 двійкових послідовностей
Blake	130	189	2	0	10	0.960150
BMW	144	188	2	0	7	0.960150
CubeHash	137	189	2	1	9	0.960150
ECHO	139	189	2	0	13	0.960150
Fugue	142	188	0	0	4	0.960150
Grostl	140	189	4	0	8	0.960150
Hamsi	134	187	1	0	11	0.960150
JH	129	187	3	0	14	0.960150
Keccak	134	187	5	1	10	0.960150
Luffa	129	189	3	0	10	0.960150
Shabal	123	188	0	0	9	0.960150
SHAvite3	135	187	1	0	15	0.960150
SIMD	131	188	3	0	8	0.960150
Skein	137	188	1	0	9	0.960150

Аналіз табл. 3.2 показує, що не всі алгоритми конкурсанти забезпечують безпеку, так алгоритми BMW, Fugue, Hamsi, JH, Kessak, Shabal, SHAvite3, SIMD, Skein не пройшли тестування і можуть бути зламані відповідними атаками.

3.4 Обґрунтування практичних рекомендацій щодо реалізації гешування

У грудні 2010 року Національний інститут стандартів і технології США (NIST) оголосив список претендентів, які вийшли у фінальну стадію конкурсу з відбору стандартного геш-алгоритму SHA-3: BLAKE (Jean-Philippe Aumasson), Grøstl (Lars Ramkilde Knudsen), JH (Hongjun Wu), Keccak (Joan Daemen), Skein (Брюс Шнайер). У табл. 3.3 зведено дані з продуктивності та статистичної безпеки алгоритмів-претендентів по відбору стандартного алгоритму SHA-3.

Таблиця 3.3 – Дані з продуктивності та статистичної безпеки алгоритмів-претендентів (їх 512-бітний варіант) конкурсу по відбору стандартного алгоритму SHA-3

Найменування алгоритму	Кількість тестів, у яких тестування минуло 99% послідовностей	Кількість тестів, у яких тестування минуло 96% послідовностей	Продуктивність (Мбіт/с)		
			Xilinx Virtex 5 FPGAs	Altera Stratix III FPGAs	Intel x86 Q6600
Blake	130	189	3743,28	3298,34	44,37
BMW	142	186	8655,87	7618,56	62,05
CubeHash	137	189	3508,77	3488,8	22,13
ECHO	138	188	6430,88	7872	9,88
Fugue	142	187	1107,88	1650,16	18,27
Grostl	140	189	6361,09	8841	7,98
Hamsi	134	187	1828,05	1932,37	8,84
JH	129	187	3917,97	5104,92	10,6
Keccak	134	187	6644,52	6470,64	8,05
Luffa	129	189	7043,81	8576,64	24,77
Shabal	123	188	2770,94	2589,49	128,5
SHAvite3	135	187	3834,56	3869,28	28,32
SIMD	131	188	4138,55	4935,68	1,58
Skein	137	188	2811,72	7618,56	39,23

Проведений аналіз табл. 3.3 показав, що найбільш продуктивними алгоритмами з фіналістів на платформі Xilinx Virtex 5 FPGAs являються Keccak, Altera Stratix III FPGAs – Grostl, Intel x86 Q6600 – Blake. Але найкращі результати по швидкості на всіх трьох апаратних платформах показав алгоритм фіналіст Grostl.

У табл. 3.4 зведені остаточні результати тестування програмних реалізацій алгоритмів-претендентів конкурсу з відбору стандартного геш-алгоритму SHA-3, тестової послідовності генератора псевдовипадкових чисел BBS і програмної реалізації алгоритму блокового симетричного шифрування FIPS 197 в режимі лічильника.

Таблиця 3.4 – Результати тестування програмних реалізацій алгоритмів-претендентів конкурсу з відбору стандартного геш-алгоритму SHA-3

Генератор	Кількість тестів, у яких тестування пройшло 99% послідовностей	Кількість тестів, у яких тестування пройшло 96% послідовностей
BBS	134 (71%)	189 (100%)
FIPS 197	126 (67%)	189 (100%)
Blake	130 (69%)	189 (100%)
BMW	144 (77%)	188 (99,47%)
CubeHash	137 (73%)	189 (100%)
ECHO	139 (74%)	189 (100%)
Fugue	142 (76%)	188 (99,47%)
Grostl	140 (75%)	189 (100%)
Hamsi	134 (71%)	187 (98,94%)
JH	129 (69%)	187 (98,94%)
Keccak	134 (71%)	187 (98,94%)
Luffa	129 (69%)	189 (100%)
Shabal	123 (66%)	188 (99,47%)
SHAvite3	135 (72%)	187 (98,94%)
SIMD	131 (70%)	188 (99,47%)
Skein	137 (73%)	188 (99,47%)

Як видно з представлених даних в табл. 3.4 генератори на основі алгоритмів-претендентів по гешуванню, Grostl (140) володіє найкращими статистичними властивостями і має хороші показники продуктивності в порівнянні з іншими алгоритмами-фіналістами (з урахуванням роботи на різних платформах).

3.5 Висновки до розділу 3

1. Продуктивність відноситься до одного з найважливіших критеріїв роботи геш-алгоритмів, а можливість роботи на великому числі платформ визначає гнучкість роботи алгоритму. Не всі алгоритми-претенденти конкурсу з відбору стандартного геш-алгоритму SHA-3 мали рівень продуктивності зіставний або краще, ніж у SHA-2 тому цей показник дозволив відсіяти кілька конкурсантів. Проведений аналіз серед алгоритмів, що потрапили до 3 раунду

конкурсу з відбору стандартного геш-алгоритму SHA-3 показав, що найкращі показники з продуктивності на різних платформах має геш-функція Grostl.

2. Згідно з проведених досліджень по двом критеріям продуктивність та статистична безпека найкращим алгоритмом серед фіналістів 3-го раунду є геш-функція Grostl. Механізм забезпечення цілісності та автентичності інформації у автоматизованих банківських системах побудований на геш-функції Grostl дозволить вирішити протиріччя та забезпечити оперативне (своєчасне) формування MAC-коду для повідомлень довільної довжини.

РОЗДІЛ 4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Забезпечення та підтримання необхідних санітарно-гігієнічних умов праці має великий вплив на умови нормальної життєдіяльності людини, на створення комфортних, нормальних умов праці для підтримки високовиробничої праці робітника на робочому місці.

Охорона праці представляє собою систему правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів і засобів, спрямованих на збереження здоров'я і працездатності людини в процесі праці на промислових підприємствах. Охорона праці виявляє і вивчає можливі причини виробничих нещасних випадків, професійних захворювань, аварій, вибухів, пожеж з метою ліквідації цих причин і створення безпечних і сприятливих для людини умов праці.

Приміщення відділення знаходиться в п’ятиповерховому будинку на першому поверсі з орієнтацією вікон на південь. Загальна площа приміщення складає $S = 8{,}00 * 4{,}75 = 38$ м^2, висота приміщення 2,70 м, об’єм приміщення – $V = 8{,}00 * 4{,}75 * 2{,}70 = 102{,}6$ м^3. Площа та об’єм приміщення, які приходяться на одного працівника (всього штат працівників відділення “Райффайзен Банк Аваль” складає 5 чоловік) відповідно складає $S_1 = \frac{38}{5} = 7{,}60$ м^2/чол, $V = \frac{102{,}6}{5} = 20{,}52$ м^3/чол. Відповідно зі стандартом найменше допустиме значення приміщення складає: площі − 6,0 м^2 та об’єму – 20 м^3, з чого можна зробити висновки, що приміщення відповідає стандарту.

Істотний вплив на стан робітника, його працездатність має мікроклімат у виробничих приміщеннях, під яким розуміються умови внутрішнього середовища цих приміщень, які впливають на тепловий обмін працюючих з оточенням. Ці умови визначаються температурою повітря, відносною вологістю повітря, швидкістю руху повітря на робочому місці, атмосферний тиск.

Для підтримки нормального мікроклімату у приміщенні застосовується вентиляція, кондиціювання повітря та опалення.

Найбільш важливою характеристикою місцевих кондиціонерів, яку необхідно оцінювати при встановленні кондиціонерів в приміщенні є потужність охолодження. Від цієї величини залежить площа, на яку він розрахований.

Для оцінки охолоджуючої потужності кондиціонера необхідно розраховувати теплонадходження в приміщенні наступним чином:

$$Q_{\text{заг.}} = Q_1 + Q_2 + Q_3, \quad (4.1)$$

де $Q_{\text{заг.}}$ – загальні теплонадходження в приміщення, Вт;

Q_1– теплонадходження від стін, пола, стелі та вікон, Вт;

Q_2– теплонадходження від устаткування, Вт;

Q_3 – теплонадходження від людей, Вт.

Теплонадходження від стін, пола, стелі та вікон розраховують за формулою:

$$Q_1 = S \times \square \times q, \quad (4.2)$$

де S – площа приміщення, м^2; h – висота приміщення, м; q – коефіцієнт, який дорівнює у випадку південної орієнтації вікон в приміщенні – 40 Вт/м^3, у випадку північної орієнтації вікон в приміщенні – 30 Вт/м^3, в інших випадках – 35 Вт/м^3.

$$Q_1 = 38 * 2{,}70 * 40 = 4104(Вт).$$

Теплонадходження від устаткування, Вт:

$$Q_2 = N \times n_{\text{заг}}, \quad (4.3)$$

де N – потужність одиниці устаткування (для офісної техніки в середньому N = 250…300 Вт), Вт;

$n_{заг}$– кількість одиниць устаткування, шт.

$$Q_2 = 250 * 6 = 1500(Вт).$$

Теплонадходження від людей, Вт:

$$A_{людей} = W \times n_л, \quad (4.4)$$

де $n_л$ – кількість людей;

W – енерговитрати людини в залежності від категорії робіт, які вона виконує.

$$A_{людей} = 160 \times 5 = 800(Вт).$$

$$Q_{заг} = 4104+1500+800= 6404 \text{ (Вт)}.$$

На підставі розрахованого значення Qзаг можна зробити висновок, що у даному приміщенні необхідно встановити кондиціонер потужністю не менш, ніж 6,4 кВт.

Робоче приміщення має один кондиціонер LG Maestro inverter 2010S18LHU потужністю 7,0 кВт, що свідчить про достатнє кондиціонування повітря в приміщенні відділення.

Вентиляція повітря в приміщенні відбувається природнім шляхом (провітрюглвання приміщення) та існує витяжна система вентиляції.

Приміщення має автономну систему опалення. У закладі є батареї.

Шум у приміщенні може створюватися через роботу кондиціонера, принтерів, сканерів, комп’ютерів, а також мобільних телефонів. Норма допустимого рівня шуму на робочих місцях не повинна перевищувати 50-65 дБА, а реальний рівень шуму складає 55 - 65 дБА, шум відповідає санітарним нормам.

Природне освітлення здійснюється падінням прямих променів сонця через 2 вікна, площа кожного з яких становить 6 $м^2$ (ширина – 3 м, висота − 2 м).

Достатність природнього освітлення визначається за формулою (4.5):

$$K = \frac{S_{вікон}}{S_{підлоги}} \geq 0,2 \div 0,166, \qquad (4.5)$$

де $S_{вікон}$ – площа вікон, м^2;

$S_{підлоги}$ – площа підлоги, м^2.

$$S_{вікон} = (2*3)*2 = 12(м^2),$$

$$S_{підлоги} = 38(м^2).$$

$$К = \frac{12}{38} = 0,32.$$

Таким чином, К = 0,32.

Коефіцієнт природнього освітлення даного приміщення (К = 0,32) відповідає нормативному значенню.

Для штучного освітлення приміщення відділення використовуються лампи типу ЛД–60 з ФЛ = 3570 лм у кількості 4 шт., розташовані у 3 ряди перпендикулярно площині вікон.

Розраховуємо штучне освітлення в приміщенні за формулою:

$$E_ф = \frac{N \times Ф_л \times n \times \eta}{S \times z \times k_з}, \qquad (4.6)$$

де N = 4 – кількість світильників;

ФЛ = 3570 лм – світловий потік лампи;

n = 3 – кількість ламп у світильнику;

η – коефіцієнт використання освітлюваної установки;

S = 38 м^2 – площа освітлюваного приміщення,

z = 1,15 – коефіцієнт нерівномірності освітлення для люмінесцентних ламп;

$k_з$ – коефіцієнт запасу, враховуючий зниження освітленості через

забруднення та старіння лампи, згідно СНиП II–4–79 значення $k_з$ = 1,5.

Визначимо індекс приміщення за формулою:

$$i = \frac{A \times B}{h_n (A + B)}, \tag{4.7}$$

де А – довжина приміщення;

В – ширина приміщення;

Н – висота приміщення;

h_c – висота підвісу, яка визначається висотою приміщення та висотою умовної робочої поверхні ($h_р$ = 0,8 м) за формулою:

$$h_c = H - h_р. \tag{4.8}$$

Висота підвісу дорівнює 2,1 м.

Таким чином:

$$i = \frac{5{,}00 \times 7{,}25}{2{,}1 \times (5{,}00 + 7{,}25)} = 1{,}4.$$

Згідно з тим, що стеля і стіни пофарбовані в синій колір, а підлога – коричневого кольору, коефіцієнти відображення світлового потоку, відповідно дорівнюють $р_п$= 70 %, $р_с$= 50 %, $р_{пола}$= 30 %.

Нормативне освітлення приміщення дорівнює $Е_н$ = 300 лк.

Згідно із знайденим індексом приміщення (і = 1,4), також коефіцієнтом відображення світлового потоку від стелі, стін і підлоги та типом світильника, отримуємо коефіцієнт використання освітлюваної установки рівний η = 0,49.

Підставляємо всі знайдені величини у формулу (4.2) і отримуємо:

$$E_{ф} = \frac{4 \times 3570 \times 2 \times 0{,}49}{38 \times 1{,}15 \times 1{,}5} = \frac{20991{,}6}{65{,}55} = 320{,}24(лк).$$

Виходячи з того, що отримане значення ($E_ф$ = 320,24 лк) більше, ніж нормативне ($E_н$ = 300 лк), то робимо висновок, що освітлення приміщення відділення “Райффайзен Банк Аваль” відповідає нормативному значенню.

Санітарно-побутові приміщення в будівлі розташовані відповідно до нормативних вимог. Туалетна кімната суміщена з умивальниками знаходиться на першому та другому поверхах будівлі. У приміщеннях проводиться прибирання раз на день, в обідню перерву.

Значення параметрів, які характеризують санітарно-гігієнічні умови праці в приміщенні, наведені в табл. 4.1.

Таблиця 4.1 -Параметри, які характеризують санітарно-гігієнічні умови праці в приміщенні відділення “Райффайзен Банк Аваль”

Параметри	Фактичне значення	Норматив по ДНАОП або ГОСТу	Відповідність параметрів нормі ДНАОП або ГОСТу
Температура повітря в приміщені, в холодний період, ºС	20-23	21-25	відповідає
Відносна вологість повітря, %	35-50	40-60	відповідає
Шум, дБА	55-65	50-65	відповідає
Освітленість загальна, лк	320,24	300-400	відповідає
Швидкість руху повітря, м/с	0,1-0,2	0,1-0,2	відповідає

Зважаючи на дані табл. 4.1, можна констатувати, що санітарно-гігієнічні умови праці в приміщенні відділення “Райффайзен Банк Аваль” відповідають нормативам.

4.2 Безпека в надзвичайних ситуаціях

Техніка безпеки.

Приміщення установи за електробезпекою, відноситься до приміщень без підвищеної небезпеки поразки струмом, тому що в ньому відсутня висока вологість і температура, немає струмопровідного та хімічно активного середовища, підлога не проводить електричний струм [32].

Обладнання та організація робочих місць користувачів ПК в відділенні, забезпечують відповідність конструкцій всіх елементів робочого місця та їх взаємного розташування ергономічним вимогам з урахуванням характеру і особливостей трудової діяльності відповідно до ДСанPin 3.3.2.007-98.

У відділенні проведена внутрішня електропроводка, обладнана розетками. У відділенні є 5 комп'ютерів. Для захисту працівників від електроструму в приміщенні зроблене заземлення окремим контуром, автоматичне відключення. Відстань розеток від підлоги – 0,6 м.

У відділенні електропроводка улаштована правильно, з застосуванням захисних заходів, що відповідають ГОСТ 12.1.030-81.

Безпека робочого місця відповідає вимогам виробничої санітарії, техніки безпеки й протипожежній техніці.

На підприємстві з охорони праці, техніки безпеки організоване навчання і інструктаж всіх працівників. Для новоприбулих працівників проводиться вступний інструктаж, потім первинний інструктаж на робочому місці. Повторний інструктаж для всіх працівників проводиться 1 раз в квартал.

Як наслідок за роки роботи відділенні не виникло ні одного нещасного випадку, тому можна сказати, що техніка безпеки знаходиться на достатньому рівні.

Пожежна безпека

Будівля, в якій знаходиться в "Райффайзен Банк Аваль", по вогнестійкості відноситься до II рівня. Приміщення по пожежній безпеці відноситься до категорії В, тому що у приміщенні є горючі матеріали та меблі. Вибухонебезпечних парів і концентратів немає [34].

З точки зору пожежної профілактики, в відділенні можливими причинами виникнення пожежі може бути неакуратне поводження з електроприладами та коротке замикання.

В установі в кожному коридорі є схематично зображені плани евакуації працівників у разі виникнення пожежі. Евакуаційні виходи чітко позначені і знаходяться в легкодоступних місцях. Ширина евакуаційного шляху відповідає СНиП 2.01.02-85.

Відповідно до правил пожежної безпеки будівля організації оснащена засобами пожежогасіння і протипожежного інвентарю, а також електронною протипожежною системою з датчиками, реагуючими на підвищену задимленість. Наявні такі засоби пожежогасіння: вогнегасники ВВ-3 в приміщенні, в коридорах пожежні крани з рукавами по 20 м, пожежні відра, також є пожежний щиток на котрому знаходиться пожежний інструмент (гаки, ломи та пісок). Вогнегасники та пожежний інвентар мають червоне пофарбування.

Відповідно до Закону України “Про охорону праці” у випадку ушкодження здоров’я внаслідок нещасного випадку, пов’язаного з виконанням трудових обов’язків, чи загибелі працівника, здійснюються належні відшкодування. Адміністрація, при розв’язанні питань охорони праці і техніки безпеки керується вимогами законодавства.

На мою думку, публічне акціонерне товариство “Райффайзен Банк Аваль” має достатньо засобів пожежної безпеки для забезпечення нормальної роботи співробітників.

ВИСНОВКИ

1Аналіз швидкодії формування MAC-кодів показав, що алгоритми UMAC є найшвидшими. Однак застосовувані методи універсального гешування не дозволяють забезпечити криптографічну стійкість до атак зловмисника. Практично всі функції універсального гешування застосовуються в композиції з алгоритмами шифрування, в результаті чого забезпечується стійкість, але втрачається властивість універсальності.

2. Критерії безпеки, продуктивності і можливості роботи на великому числі платформ відносяться до основних NIST-критеріїв відбору, саме це дозволило керівникам конкурсу проводити відбір серед кандидатів. NIST алгоритми, що мають рівень продуктивності зіставний або краще, ніж у SHA-2. У роботі проведений аналіз статистичної стійкості алгоритмів-претендентів конкурсу з відбору стандартного алгоритму SHA-3, їх реалізація на різних платформах з метою оцінки продуктивності та адаптації їх застосування в системах документообігу внутріплатіжних банківських систем, дозволить вибрати переможця, який буде задовольняти всім критеріям.

4. Для оцінки того, наскільки близько примітиви апроксимують генератори “випадкових” послідовностей, використовувались статистичні тести запропоновані NIST (Американським Національним Інститутом Стандартів). Пакет тестів NIST STS для тестування генераторів випадкових або псевдо випадкових чисел є однім з підходів реалізації задачі оцінки статистичної безпеки криптографічних примітивів. Дослідження, що були проведені у магістерській роботі за допомогою пакету тестів NIST STS підтвердили стійкість алгоритмів-претендентів конкурсу з відбору стандартного алгоритму гешування SHA-3 до відомих атак крипто аналітиків. Алгоритм гешування Grostl необхідно відзначити окремо в силу того, що він володіє найкращими статистичними властивостями і має хороші показники продуктивності в порівнянні з іншими алгоритмами-фіналістами (з урахуванням роботи на різних платформах). Перспективним напрямком подальших досліджень є

оцінка колізійних властивостей геш-алгоритмів, що потрапили до 3 раунду конкурсу.

СПИСОК ЛІТЕРАТУРИ

1. Информационная технология. Криптографическая защита информации. Функция хеширования : ГОСТ 34.311-95. − [Действующий с 1998-04-16] − Киев. Госстандарт Украины, 1998. – 1 с. – (Национальный стандарт Украины).

2. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма : ГОСТ Р 34.10-94.– [Действующий с 1995-01-01] − Москва. Госстандарт России, 1998. – 23 с. – (Национальный стандарт России).

3. Информационная технология. Криптографическая защита информации. Функция хеширования : ГОСТ Р 34.11-94. – [Действующий с 1994-05-23] − Москва. Госстандарт России, 1998. – 16 с. – (Национальный стандарт России).

4. Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 2 : ГОСТ 7498-2-99 − [Действующий с 2000-01-01] − Москва. Госстандарт России, 1998. – 32 с. – (Национальный стандарт России)..

5. Аволио Ф. М. Защита информации на предприятии // Сети и системы связи. / Ф. М. Аволио, Г. Шипли. – 2000. − №8 – С. 91−99.

6. Вервейко В. Н. Функции хеширования: классификация, характеристика и сравнительный анализ / В. Н. Вервейко, А. И. Пушкарев, Т. В. Цепурит. – ХНУРЕ : Харьков, 2002. – 68 с.

7. Горбенко И.Д. Анализ каналов уязвимости системы RSA // Безопасность информации. / И.Д. Горбенко, В.И. Долгов, А.В. Потий. – 1995.-№2. – С. 22–26.

8. Диффи У. Защищенность и имитостойкость // Введение в криптографию / У. Диффи, М. Хеллман.– 1979. – №3 − С. 79–109.

9. Долгов В.И. О некоторых подходах к построению безусловно стойких кодов аутентификации коротких сообщений // Управление и связь. / В.И. Долгов, В.Н. Федорченко. – 1996. – №4– С. 47-51.

10. Домарев В.В. Защита информации и безопасность компьютерных систем. – Киев : Издательство "ДиаСофт", 1999. – 480 с.

11. Євсеєв С. П. Механизмы обеспечения аутентичности банковских данных во внутриплатежных системах комерческого банка : збірник наукових статей ХНЕУ. / С. П. Євсеєв, В. Е. Чевардин, С. А. Радковский. – Харків : ХНЕУ, 2008. – Вип. 6. – 40–44 с.

12. Задірака В. К. Методи захисту банківської інформації. / В. К. Задірака, О. С. Олесюк, Н. О. Недашковський. – Киів : Вища школа, 1999. – 264 с.

13. Иванов М.А. Криптографические методы защиты информации в компьютерных системах и сетях / М.А. Иванов. – М. : КУДИЦ-ОБРАЗ, 2001. – 368 с.

14. Конеев И. Р. Информационная безопасность предприятия / И. Р. Конеев, А. В. Беляев. – Спб. : БХВ-Петербург, 2003. – 752 с.

15. Кузнецов А. А. Анализ механизмов обеспечения безопасности банковской информации во внутриплатежных системах коммерческого банка : Матеріали з міжнародної науково-практичної конференції «Безпека та захист інформації в інформаційних і телекомунікаційних системах» : зб. наук. статей «Управління розвитком». / А. А. Кузнецов, О. Г. Король, А. М. Ткачов. − ХНЕУ. № 6 – Х. : 2008. – 28 – 35 с.

16. Романец Ю.В. Защита информации в компьютерных системах и сетях / Ю. В. Романец, П. А. Тимофеев, В. Ф. Шаньгин. – 2-е изд., перераб. и доп. – М. : Радио и связь, 2001. – 376 с.

17. Ростовцев А. Г. Подпись и шифрование на эллиптической кривой: анализ безопасности и безопасная реализации. Проблемы информационной безопасности : компьютерные системы / А. Г. Ростовцев, Е. Б. Маховенко – 2003 – №1 − С. 64 – 73.

18. Соколов А. В. Защита от компьютерного терроризма. / Соколов А. В., О. М. Степанюк. – БХВ-Петербург Арлит, 2002. – 496 с.

19. Термінологія в галузі захисту інформації в комп’ютерних системах від несанкціонованого доступу. НД СТЗІ 1.1-003-99. − Чинний від 28.04.1999. − К. : Держстандарт України, 1999. – 24 с.

20.Чмора А. Л. Современная прикладная криптография. – Москва, 2002. – 508 с.

21. Шеннон К.Э. Теория связи в секретных системах. Работы по теории информации и кибернетике / К.Э. Шеннон. – М. : 1963. – 333 – 402 с.

22. Шипли Г. Основы безопасности ИТ // Сети и системы связи. / Г. Шипли. – М., 2003 – №4 – С. 78 – 82.

23. 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 – Pp. 331–342.

24. Carter J. L. Universal classes of hash functions / J.L.Carter, M.N.Wegman // Computer and System Scince. – 1979 – №18 – Pp. 143–154.

25. Diffi W. The first Ten Years of Public-Key Cryptography/ W. Diffi/ Computer Since. – 1988 – №5 – P. 21.

26. Krawczyk H. LFSB-based Hashing and Authenticator./ H. Krawczyk/ Proceedings of CRYPTO Notes in Computer Science. – 1994 – №80 – P. 129 – 139.

27. McElliece R.J. A public-key cryptosystem based on algebraic coding theory/ R.J. McElliece - NY: Springer-Verlag, 1978. – 116 p.

28.Pieprzyk J.P. On public-key cryptosystems built using polynomial rings. In Advanced in Cryptography-Eurocrypt/ J.P.Pieprzyk./ – NY: Springer-Verlag, 1985 – 80 p.

29. Preneel B., Biryukov A., «New European Schemes for Signature, Integrity and Encryption» Final report of European project number IST-1999-12324, NESSIE, April 2004. – Pp. 487– 623

30. Simmons G.J. Authentication theory/coding theory in Cryptology/ G.J.Simmons/ Computer Science. – 1985 – №96 – Pp. 411–431.

31. Simons G.J. An impersonation-proof identity verification scheme/ G.J.Simons. / Computer Science. – 1988 – №87 – Pp. 211–215.

32. Smid M.E. The Past and Future / M.E.Smid, D.K.Branstad. / Computer Science. – 1988 – №76 – Pp. 122–132.

33. Wegman M. N. New hash functions and their use in authentication and set equality / M. N. Wegman, J. L. Carter. /Computer and System Scince. – 1981 – № 22 – Pp. 265–279.

34. Суханова Н.С. Современные методы криптоанализа алгоритмов хеширования. / Н. С. Суханова // Міжнародна науково-практична конференція молодих вчених, аспірантів та студентів «Актуальні проблеми науки та освіти молоді: теорія, практика, сучасні рішення», 21-22 квітня 2011 р.: тези доповідей. Том І. – Х.: ХНЕУ, 2011. – С. 208 –210.

35. Суханова Н. С. Аналіз загроз та механізмів захисту у внутриплатіжних системах комерційного банку. / Н. С. Суханова // «Збірник наукових праць студентів спеціальностей «Інформаційні управляючі системи і технології», «Комп'ютерний еколого-економічний моніторинг» »: / редкол.: В.С. Пономаренко [та ін.]. – Харків: ХНЕУ, 2010. – С. 208 – 210.

36. Межбанковские расчеты на Украине [Електронний ресурс] — Режим доступу до ресурсу : http://www.nbuv.gov.ua/articles/2003/03klinko.html.

37. Программное средство криптографической защиты информации "Грифон-Б" [Електронний ресурс] — Режим доступу до ресурсу : http://www.banksoft.com.ua/index.php?id=28

38. Программное средство «Библиотека функций криптографической защиты информации "Грифон-Л" [Електронний ресурс] — Режим доступу до ресурсу : http://www.banksoft.com.ua/ index.php?id=27

39. Логинов А.А. Общие принципы функционирования электронных платежных систем и осуществление мер безопасности при защите от злоупотребления и мошенничества // А.А. Логинов, Н.С. Елхимов – Конфидент. – 1995. – Сс.48-54

ДОДАТКИ

Додаток А. Короткі характеристики алгоритмів-претендентів конкурсу з відбору стандартного алгоритму хешування SHA3 відібраних на 2 раунд

BLAKE Алгоритм стиснення, функція якого заснована на використанні ключової підстановки в конструкції Davies-Meyer. Ключова підстановка заснована на внутрішній організації потокового шифру ChaCha. Отримав свою не лінійність з накладенням модульного складання і операцій XOR. Сама інноваційна частина BLAKE – своя ключова перестановка.

Структурна схема алгоритму наведено на рис. А.1.

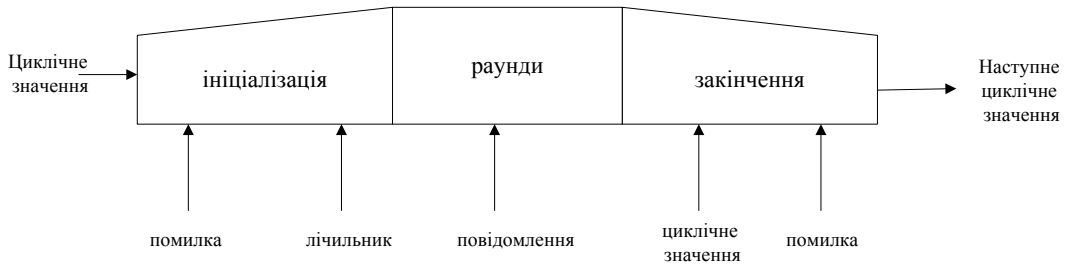

Рисунок А.1 – Структурна схема алгоритму BLAKE

Blue Midnight WISH (BMW) – ширококанальна конструкція геш Merkle-Damgard з нетрадиційною функцією стиснення, де нелінійність отримана накладенням модульного підсумовування та операцій XOR. Найбільш інноваційні частині дизайну – конструкція функції стиснення і дизайн перестановок; велика частина дизайну нова і унікальна серед кандидатів другого кола.

Структурна схема алгоритму наведено на рис. А.2.

CubeHash – подібний до губки алгоритм гешування, який заснований на фіксованій перестановці. Перестановка надзвичайно проста і елегантна, що використовує тільки доповнення, XORs, і обертання в фіксованому та простому зразку. Вся нелінійність в алгоритмі гешування отримана з накладання модульних доповнень і операцій XOR. Нова частина CubeHash – фіксована перестановка.

Продовження дод. А

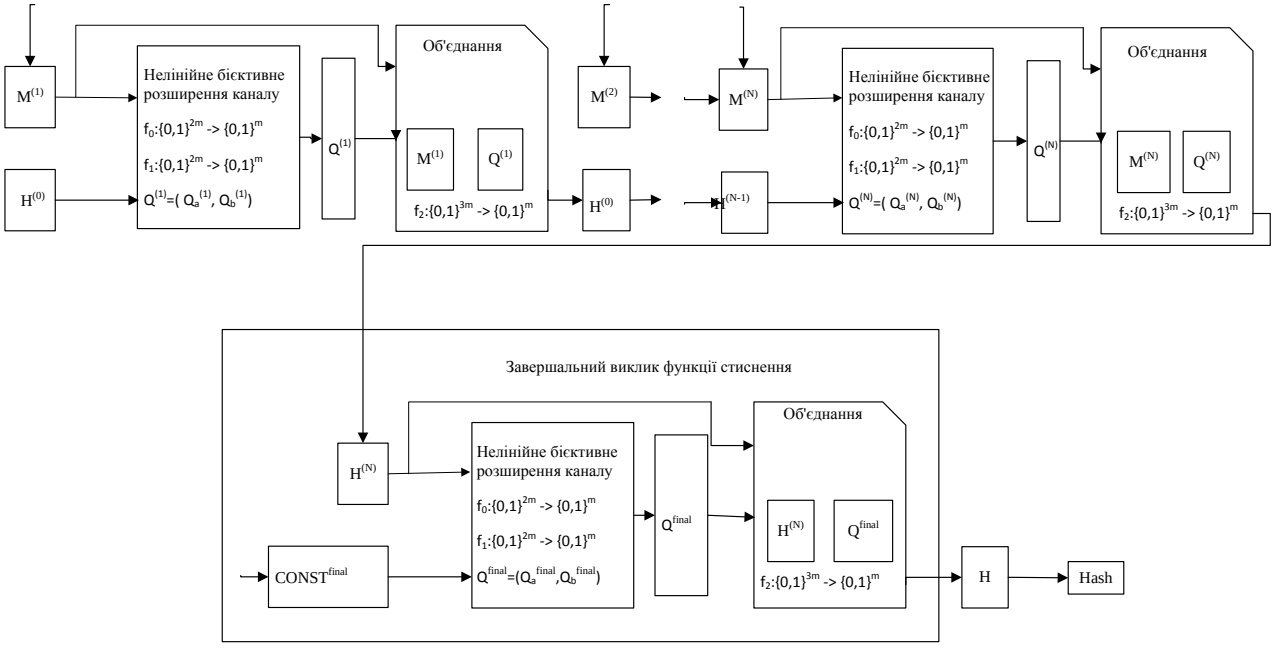

Рисунок А.2 – Структурна схема алгоритму BMW

Структурна схема алгоритму наведено на рис. А.3.

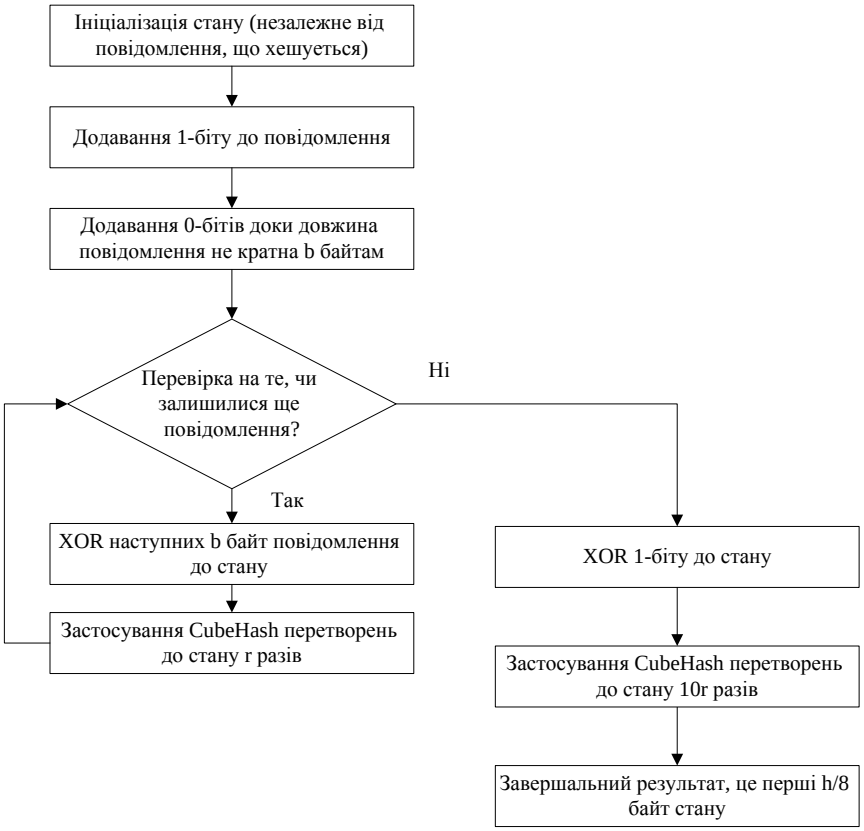

Рисунок А.3 – Структурна схема алгоритму CubeHash

Продовження дод. А

ECHO – алгоритм гешування широкого каналу після конструкції HAIFA. Його функція стиснення використовує ключову перестановку; лічильник і помилки використовуються в якості ключа, а також повідомлення і послідовні значення використовуються як вводів у підстановці. Підстановка дуже нова, використовує 2048-бітові блоки AES – як S-бокс шифру AES. S-бокс і забезпечує всю нелінійність в цьому алгоритмі гешування.

Структурна схема алгоритму наведено на рис. А.4.

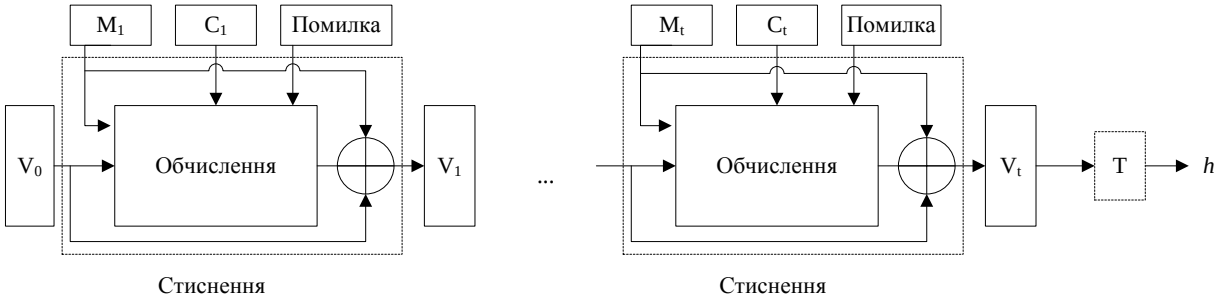

Рисунок А.4 – Структурна схема алгоритму ECHO

де V_i – поточне циклічне значення, M_i – поточний блок повідомлення, C_i – загальне число не доповнених біт повідомлення гешованого в кінці ітерації.

Fugue – варіант конструкції "sponge", функція стиснення заснована на нелінійному регістрі зсуву. Регістр зсуву включає посилений варіант AES-функції; всі інші операції лінійні. Таким чином, вся не лінійність в цьому алгоритмі заснована на S-боксі AES.

Структурна схема алгоритму наведено на рис. А.5.

Продовження дод. А

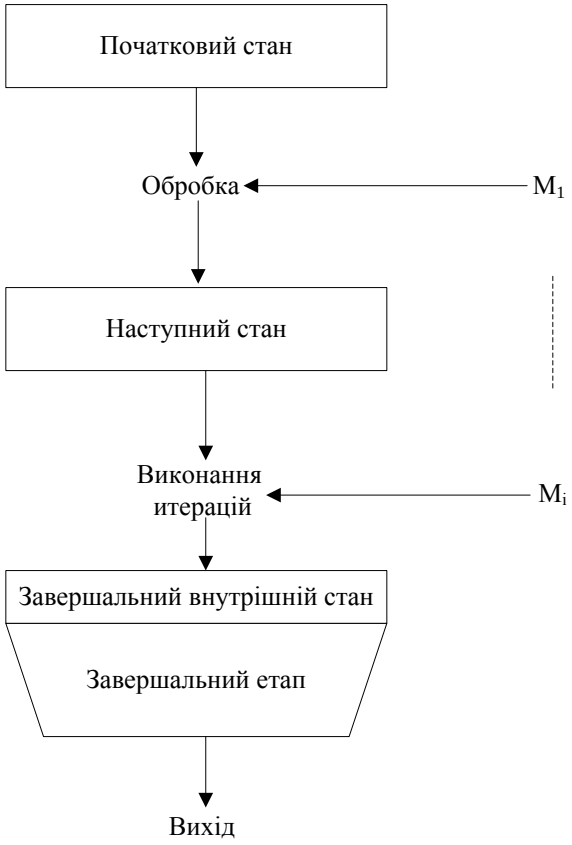

Рисунок А.5 – Структурна схема алгоритму Fugue

Grostl – ширококанальна конструкція геш Merkle-Damgård. Його функція стиснення використовує два S-боксу AES як блоки-підстановки. Вся нелінійність в алгоритмі отримана на основі S-боксу AES.

Структурна схема алгоритму наведено на рис. А.6.

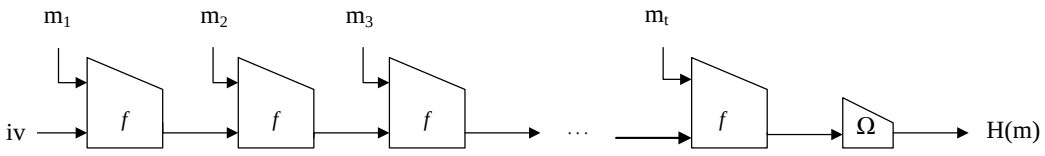

Рисунок А.6 – Структурна схема алгоритму Grostl

де f – функція стиснення, m – поточний блок повідомлення, Ω - вихідні перетворення.

Продовження дод. А

Hamsi – конструкція геш Merkle-Damgård. Функція стиснення створена на блоці-підстановці; повідомлення розширено, використовуючи код з виявленням помилок, щоб заповнити половину вхідного блоку до блоку підстановки з іншою половиною, заповненої значенням зчеплення геш. Вся нелінійність в алгоритмі отримана з одного S-боксу.

Структурна схема алгоритму наведено на рис. А.7.

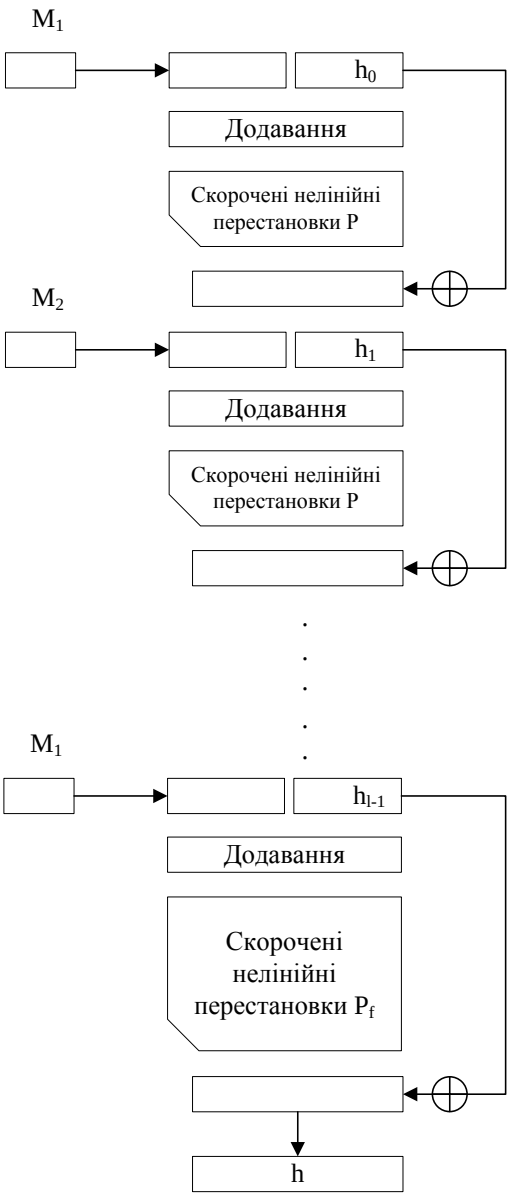

Рисунок А.7 – Структурна схема алгоритму Hamsi

Продовження дод. А

JH – використовує нову конструкцію, що дещо нагадує конструкцію “sponge” для вбудовування алгоритму геш в блок-підстановку. Блок-підстановка – комбінація з двох 4-бітових S-боксу з низкою лінійних операцій змішування і розрядних підстановок. Вся нелінійність у цьому дизайні отримана з S-боксів.

Структурна схема алгоритму наведено на рис. А.8.

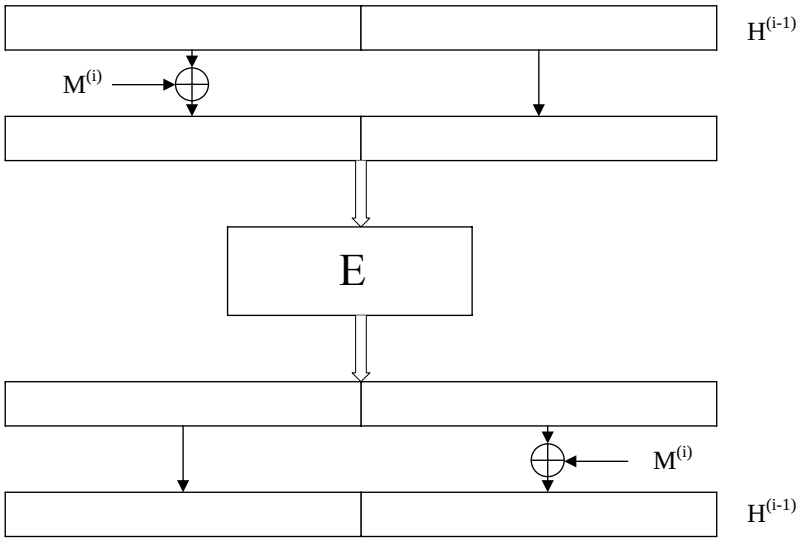

Рисунок А.8 – Структурна схема алгоритму JH

Keccak – використовує конструкцію “sponge” і блок-підстановку. Підстановка може бути реалізована на основі 5-бітових S-блоків або на комбінації лінійної і нелінійної операціях змішування.

Структурна схема алгоритму наведено на рис. А.9.

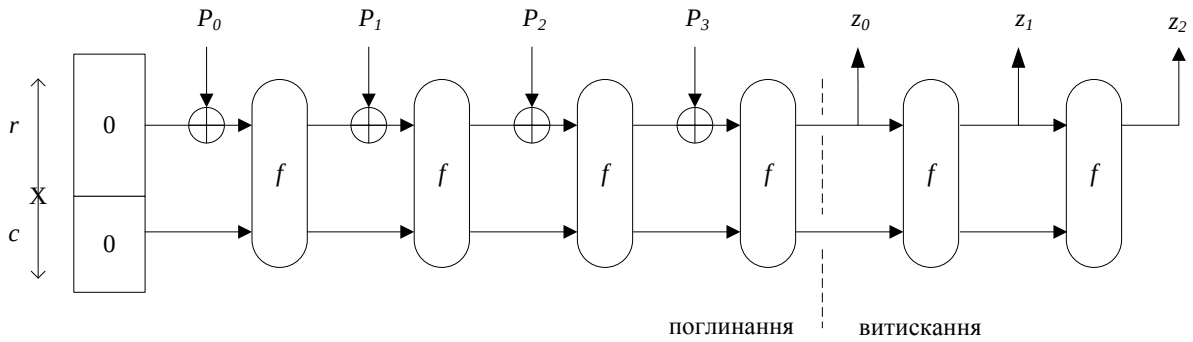

Рисунок А.9 – Структурна схема алгоритму Keccak

Продовження дод. А

Luffa – варіант конструкції “sponge”, що використовує лінійну операцію змішування і кілька 256-бітових блоків-підстановок замість однієї нелінійної підстановки. Блоки-підстановки комбінують лінійні операції змішування з єдиним 4-бітовим S-боксом, і цей S-бокс забезпечує всю нелінійність в алгоритмі.

Структурна схема алгоритму наведено на рис. А.10.

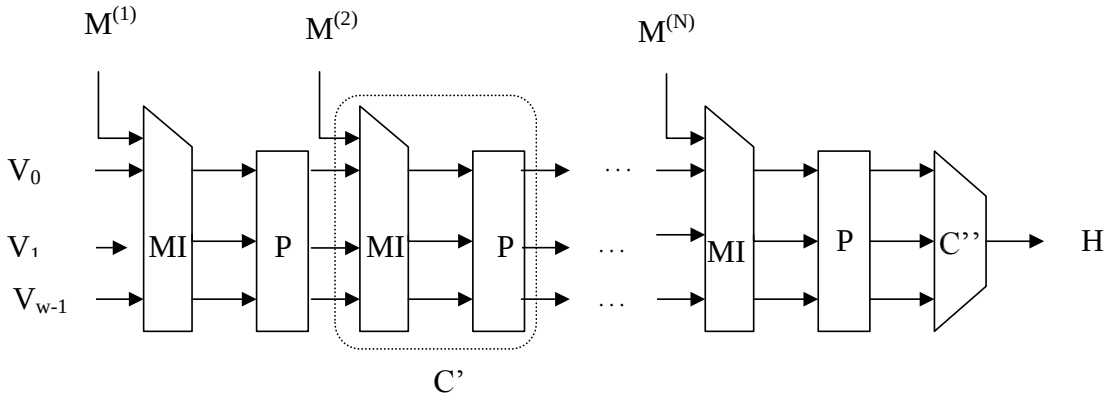

Рисунок А.10 – Структурна схема алгоритму Luffa

де V_i – поточне циклічне значення, C’’ – функція округлення, P – перестановки.

Shabal – використовує новий послідовний режим, як варіант шірококанальної конструкції геш Merkle-Damgard. Його функція стиснення заснована на конструкції регістра зсуву зі зворотним зв’язком, яка об’єднує декілька вводів, ефективно забезпечених послідовним режимом. Нелінійність отримана накладенням XOR, модульного підсумовування, і порозрядних операцій AND.

Структурна схема алгоритму наведено на рис. А.11(а, б).

Продовження дод. А

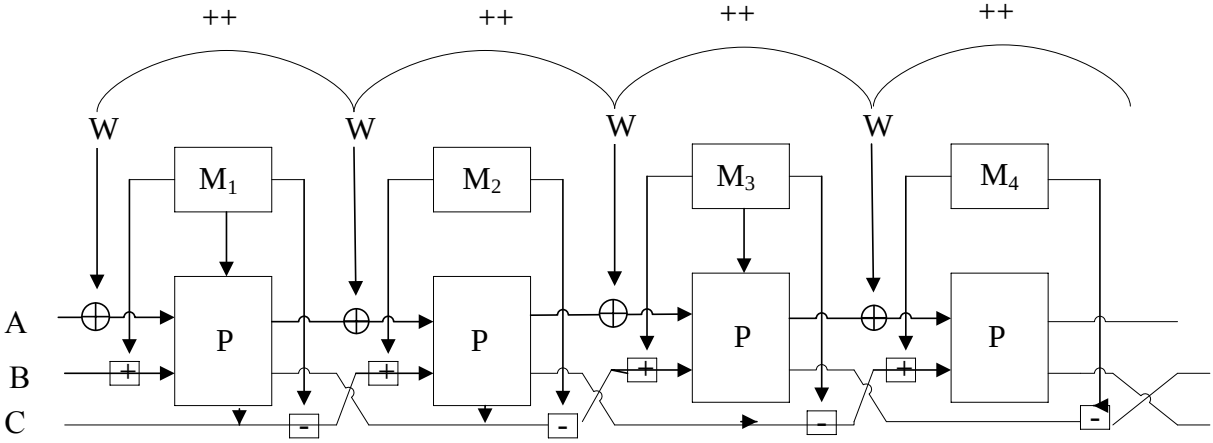

Рисунок А.11 (а) – Структурна схема алгоритму Shabal (Режим роботы: Повідомлення раундів)

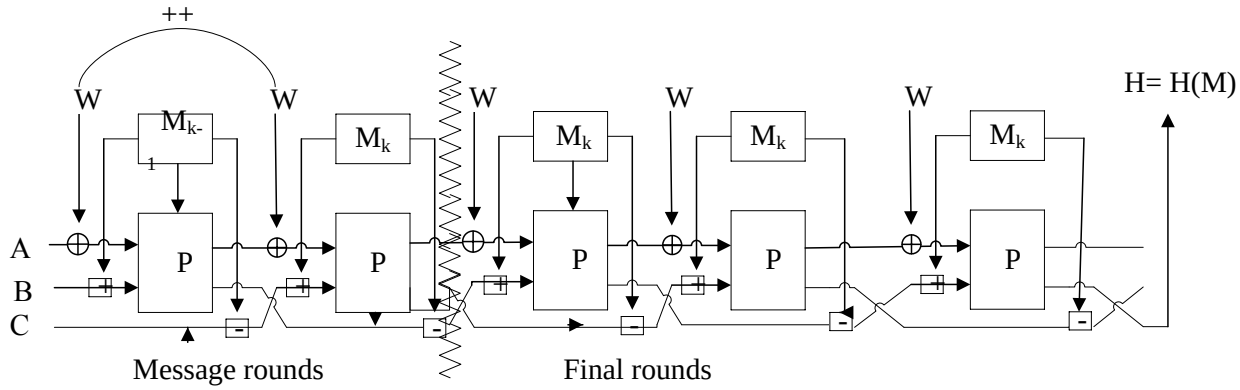

Рисунок А.11 (б) – Структурна схема алгоритму Shabal (Останні раунди)

SHAvite-3 – алгоритм геш HAIFA. Функція стиснення – ключова підстановка, яка використовується в конструкції Девіс-Мейєра. Ключова підстановка – збалансована мережа Feistel (для 256-бітового випадку), або пара переплетених збалансованих мереж Feistel (для 512-бітового випадку), з F-функцією, створеної з AES-функції. Вся нелінійність у цілої конструкції заснована на S-боксі. Структурна схема алгоритму наведено на рис. А.12.

Продовження дод. А.

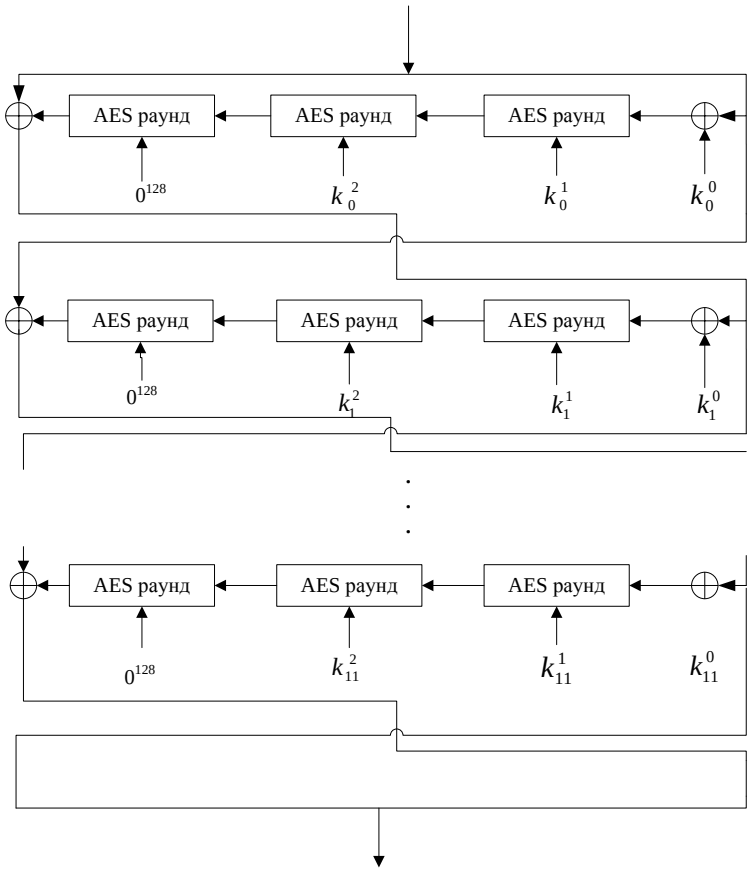

Рисунок А.12 – Структурна схема алгоритму SHAvite-3

SIMD – ширококанальна конструкція геш Merkle-Damgard. Функція стиснення створена з ключовою підстановки, у варіанті конструкції Девіс-Мейєра. Ключова підстановка використовує лінійний код з доказовими властивостями поширення, як “ключової список”, і використовує чотири незбалансовані мережі Feistel, які нагадують MD4 і MD5-функції.

Нелінійність в цьому алгоритмі забезпечена накладенням модульного підсумовування, операцій нееквівалентності і порозрядних нелінійних функцій.

Структурна схема алгоритму наведено на рис. А.13.

Закінчення дод. А.

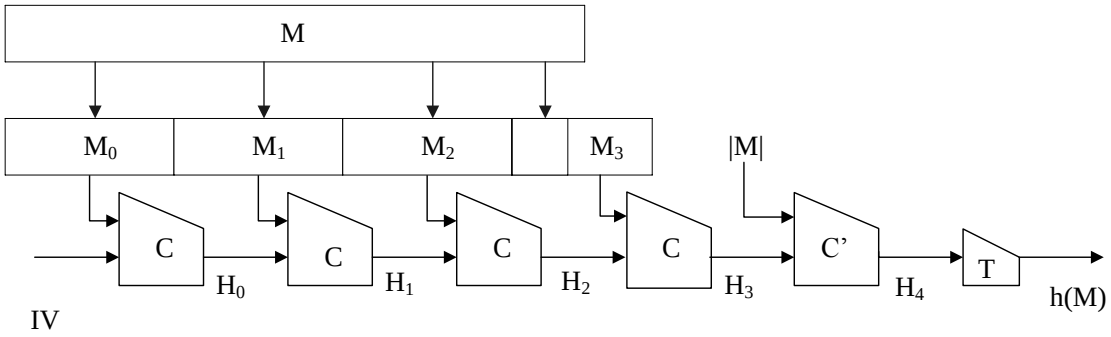

Рисунок А.13 – Структурна схема алгоритму SIMD

Skein – варіант конструкції геш Merkle-Damgard, заснована на блочному шифрі в режимі CBC. Функція стиснення використовується у варіанті конструкції Matyas-Meyer-Oseas. Безпека конструкції ґрунтується на стійкості БСШ "Threefish", створеного на основі великої кількості простих раундів, і використовує тільки три 64-бітових модульних операції підсумовування, порозрядної XOR, і ітерацію. Вся нелінійність в алгоритмі геш забезпечена накладенням модульного підсумовування та операцій нееквівалентності.

Структурна схема алгоритму наведено на рис. А.14.

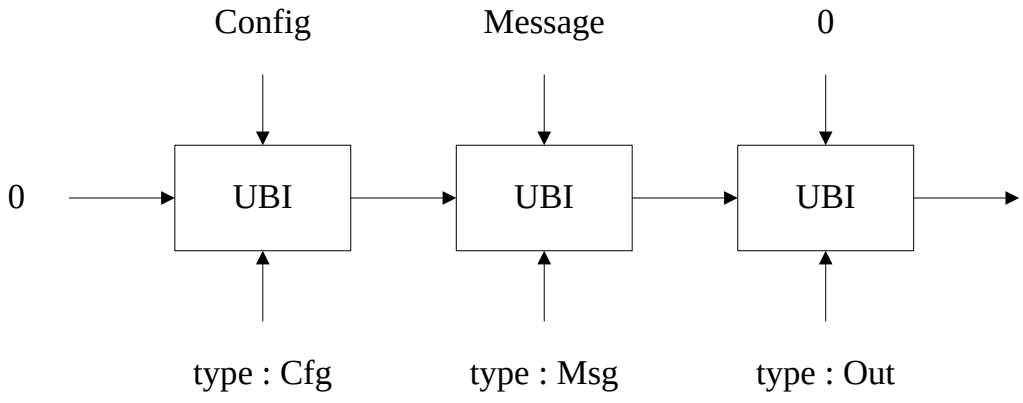

Рисунок А.14 – Структурна схема алгоритму Skein

Додаток Б

Статистичні портрети програмних реалізацій алгоритмів претендентів конкурсу з відбору стандартного геш-алгоритму SHA-3

На рис. Б.1 представлено статистичний портрет програмної реалізації алгоритму-претендента Blake. Статистичний портрет представляє із себе діаграму ймовірностей проходження відповідних статистичних тестів. З рис. Б.1 видно, що статистичний портрет програмної реалізації алгоритму-претендента Blake відповідає пропонованим вимогам – по 130 тестах позитивно пройшло більше 99 % послідовностей.

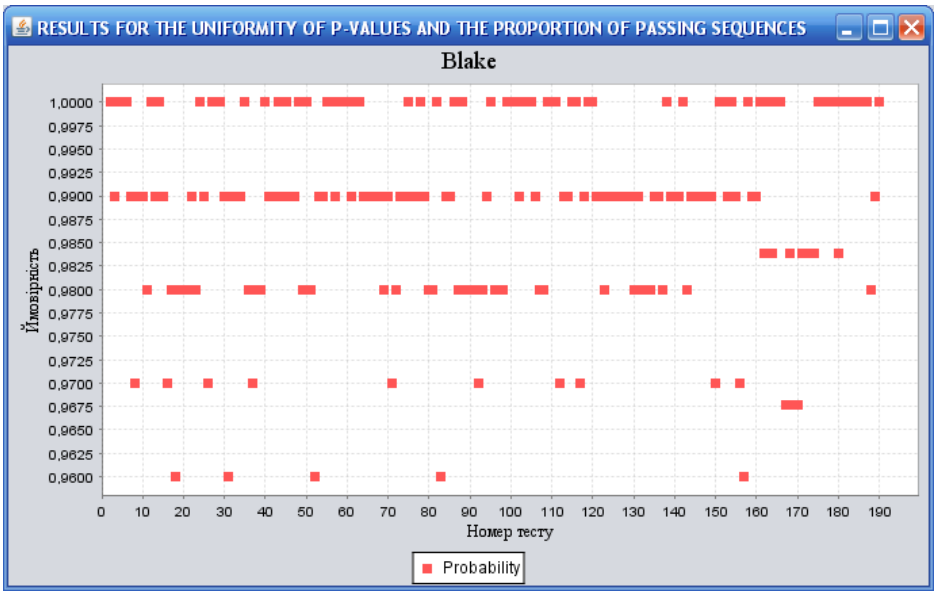

Рисунок Б.1 – Статистичний портрет програмної реалізації алгоритму-претендента Blake.

На рис. Б.2 представлено статистичний портрет програмної реалізації алгоритму-претендента BMW. З рис. Б.2 видно, що статистичний портрет програмної реалізації алгоритму-претендента BMW відповідає пропонованим вимогам – по 144 тестах позитивно пройшло більше 99% послідовностей, однак одна атака на даний алгоритм дозволяє отримати позитивний результат.

Продовження дод. Б

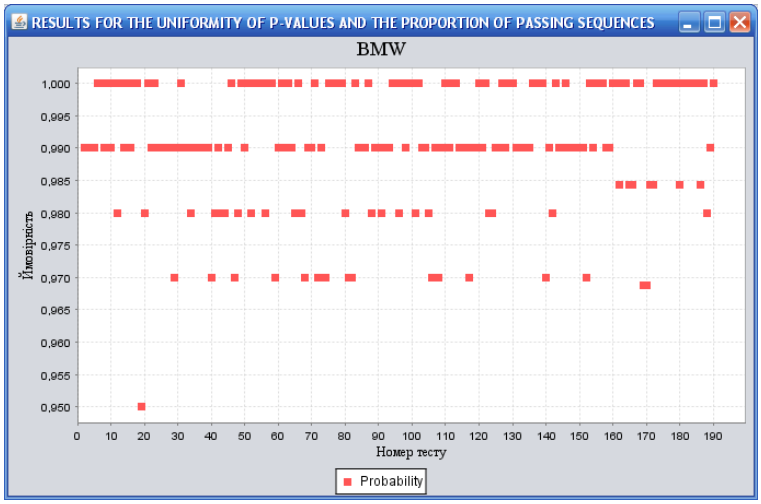

Рисунок Б.2 – Статистичний портрет програмної реалізації алгоритму-претендента BMW

На рис. Б.3 представлено статистичний портрет програмної реалізації алгоритму-претендента CubeHash. З рис. Б.3 видно, що статистичний портрет програмної реалізації алгоритму-претендента CubeHash відповідає пропонованим вимогам – по 137 тестах позитивно пройшло більше 99% послідовностей.

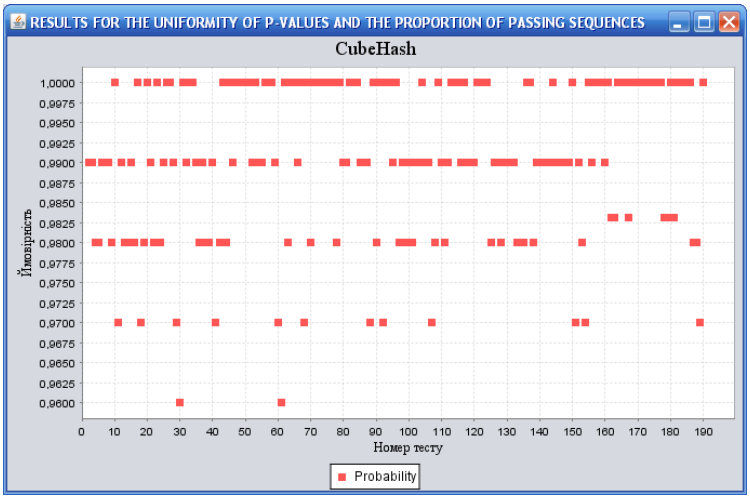

Рисунок Б.3 – Статистичний портрет програмної реалізації алгоритму-претендента CubeHash

На рис. Б.4 представлено статистичний портрет програмної реалізації алгоритму-претендента ECHO. З рис. Б.4 видно, що статистичний портрет програмної реалізації алгоритму-претендента ECHO

Продовження дод. Б

відповідає пропонованим вимогам – по 139 тестах позитивно пройшло більше 99% послідовностей.

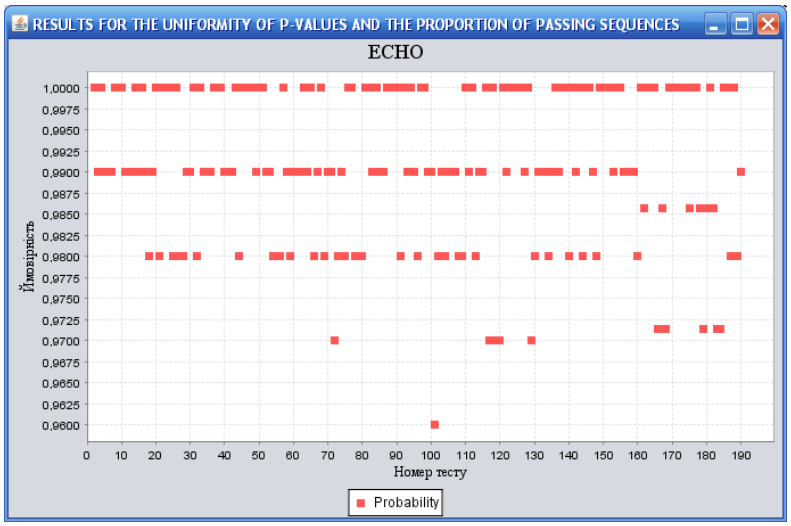

Рисунок Б.4 – Статистичний портрет програмної реалізації алгоритму-претендента ECHO

На рис. Б.5 представлено статистичний портрет програмної реалізації алгоритму-претендента Fugue. З рис. Б.5 видно, що статистичний портрет програмної реалізації алгоритму-претендента Fugue відповідає пропонованим вимогам – по 142 тестах позитивно пройшло більше 99% послідовностей, однак одна атака на даний алгоритм дозволяє отримати позитивний результат.

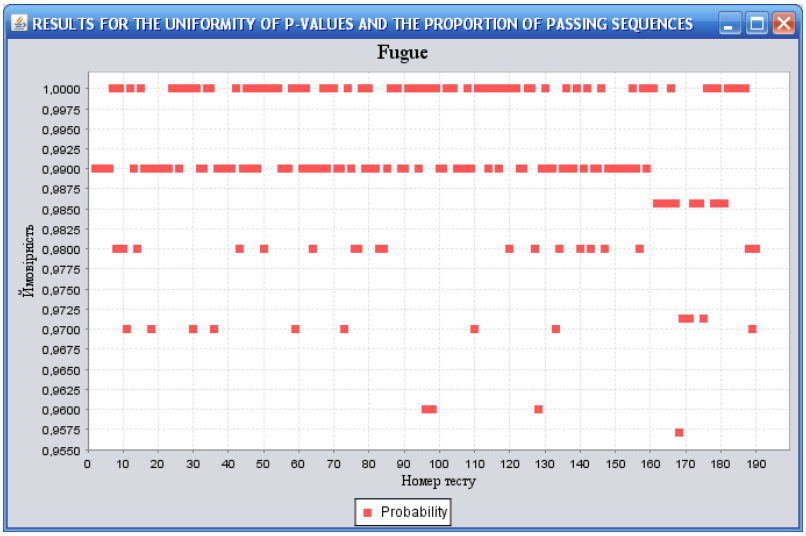

Рисунок Б.5 – Статистичний портрет програмної реалізації алгоритму-претендента Fugue.

Продовження дод. Б

На рис. Б.6 представлено статистичний портрет програмної реалізації алгоритму-претендента Grostl. З рис. Б.6 видно, що статистичний портрет програмної реалізації алгоритму-претендента Grostl відповідає пропонованим вимогам – по 140 тестах позитивно пройшло більше 99% послідовностей.

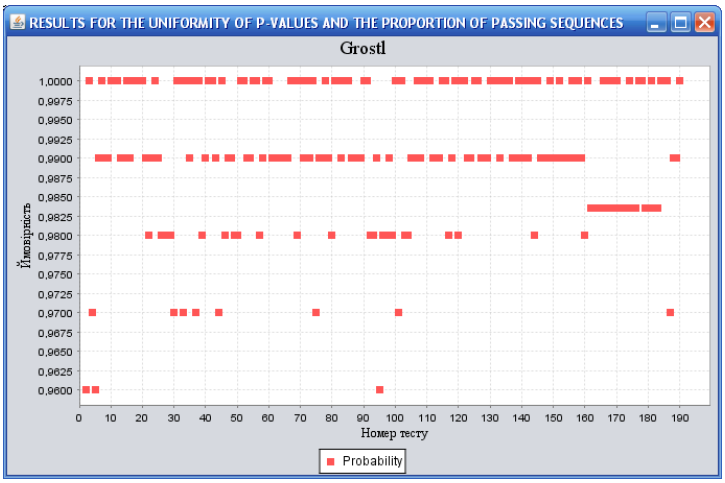

Рисунок Б.6 – Статистичний портрет програмної реалізації алгоритму-претендента Grostl

На рис. Б.7 представлено статистичний портрет програмної реалізації алгоритму-претендента Hamsi. З рис. Б.7 видно, що статистичний портрет програмної реалізації алгоритму-претендента Hamsi відповідає пропонованим вимогам – по 134 тестах позитивно пройшло більше 99% послідовностей, однак дві атаки на даний алгоритм дозволяють отримати позитивний результат.

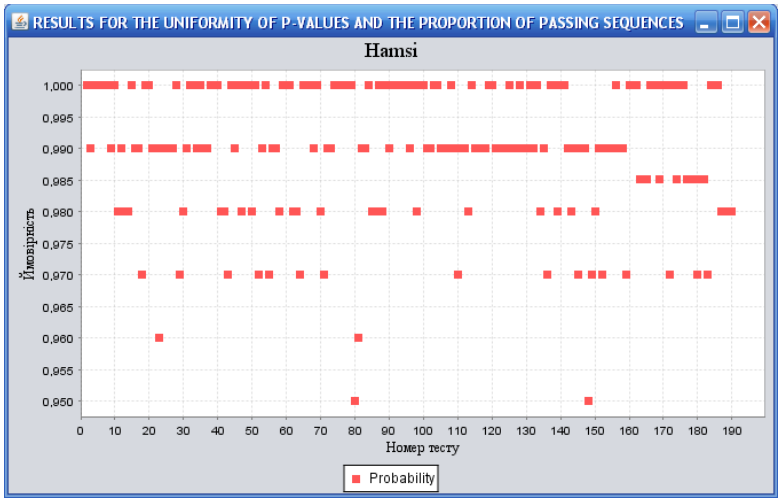

Рисунок Б.7 – Статистичний портрет програмної реалізації алгоритму-претендента Hamsi

Продовження дод. Б

На рис. Б.8 представлено статистичний портрет програмної реалізації алгоритму-претендента JH. З рис. Б.8 видно, що статистичний портрет програмної реалізації алгоритму-претендента JH відповідає пропонованим вимогам – по 129 тестах позитивно пройшло більше 99% послідовностей, однак дві атаки на даний алгоритм дозволяють отримати позитивний результат.

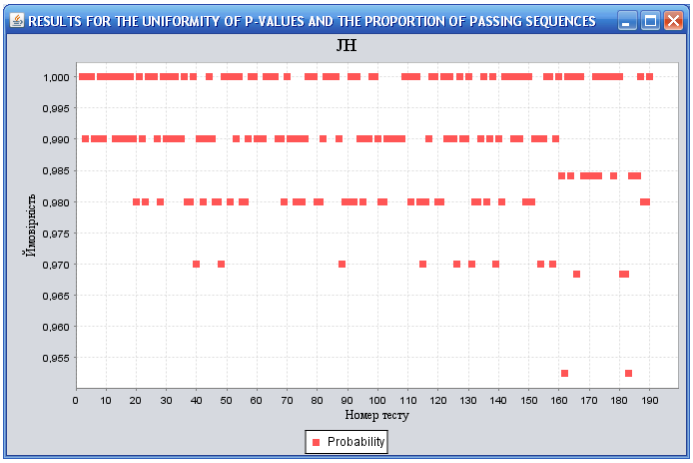

Рисунок Б.8 – Статистичний портрет програмної реалізації алгоритму-претендента JH

На рис. Б.9 представлено статистичний портрет програмної реалізації алгоритму-претендента Keccak. З рис. Б.9 видно, що статистичний портрет програмної реалізації алгоритму-претендента Keccak відповідає пропонованим вимогам – по 134 тестах позитивно пройшло більше 99% послідовностей, однак дві атаки на даний алгоритм дозволяють отримати позитивний результат.

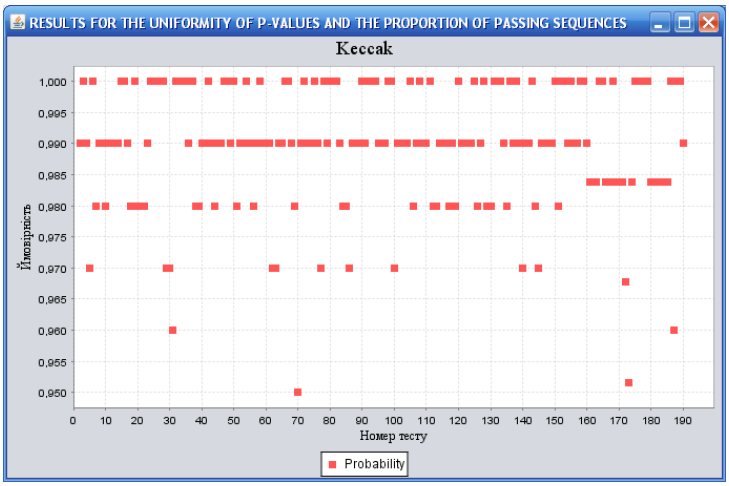

Рисунок Б.9 – Статистичний портрет програмної реалізації алгоритму-претендента Keccak

Продовження дод. Б

На рис. Б.10 представлено статистичний портрет програмної реалізації алгоритму-претендента Luffa. З рис. Б.10 видно, що статистичний портрет програмної реалізації алгоритму-претендента Luffa відповідає пропонованим вимогам – по 129 тестах позитивно пройшло більше 99% послідовностей.

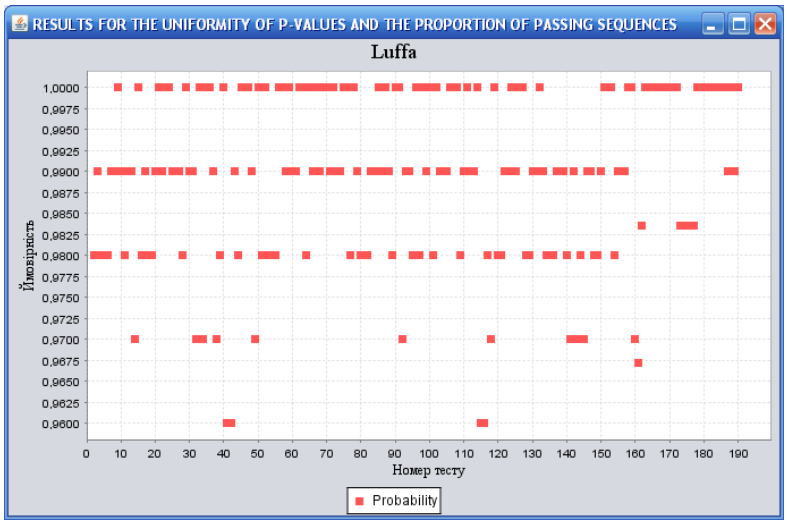

Рисунок Б.10 – Статистичний портрет програмної реалізації алгоритму-претендента Luffa.

На рис. Б.11 представлено статистичний портрет програмної реалізації алгоритму-претендента Shabal. З рис. Б.11 видно, що статистичний портрет програмної реалізації алгоритму-претендента Shabal відповідає пропонованим вимогам – по 123 тестах позитивно пройшло більше 99% послідовностей, однак одна атака на даний алгоритм дозволяє отримати позитивний результат.

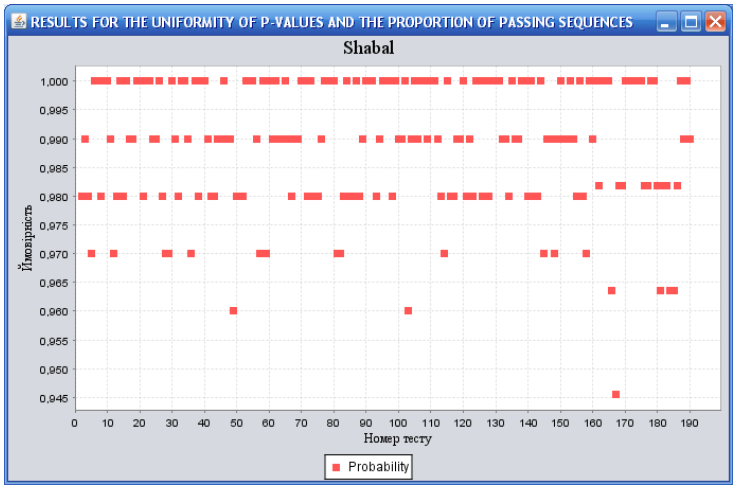

Рисунок Б.11 – Статистичний портрет програмної реалізації алгоритму-претендента Shabal.

Продовження дод. Б

На рис. Б.12 представлено статистичний портрет програмної реалізації алгоритму-претендента SHAvite3. З рис. Б.12 видно, що статистичний портрет програмної реалізації алгоритму-претендента SHAvite3відповідає пропонованим вимогам – по 135 тестах позитивно пройшло більше 99% послідовностей, однак дві атаки на даний алгоритм дозволяють отримати позитивний результат.

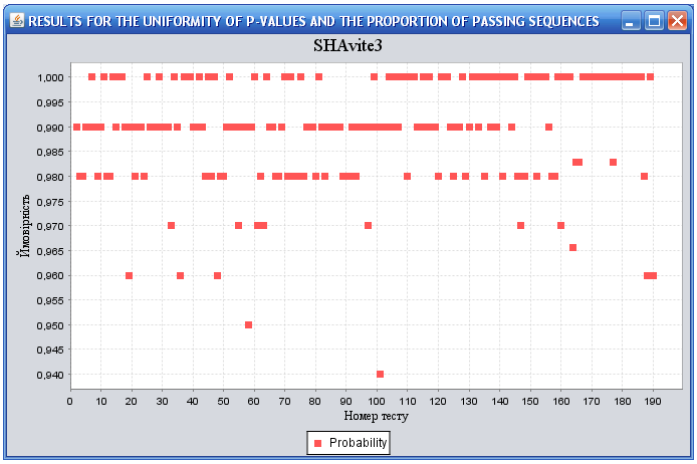

Рисунок Б.12 – Статистичний портрет програмної реалізації алгоритму-претендента SHAvite3.

На рис. Б.13 представлено статистичний портрет програмної реалізації алгоритму-претендента SIMD. З рис. Б.13 видно, що статистичний портрет програмної реалізації алгоритму-претендента SIMD відповідає пропонованим вимогам – по 131 тестах позитивно пройшло більше 99% послідовностей, однак одна атака на даний алгоритм дозволяє отримати позитивний результат.

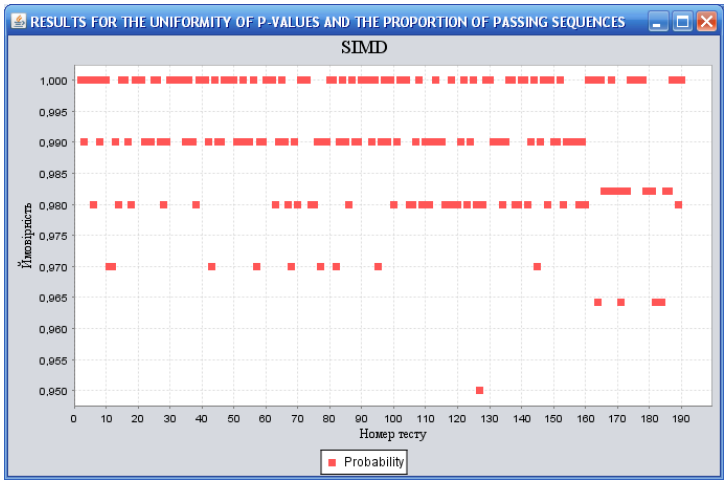

Рисунок Б.13 – Статистичний портрет програмної реалізації алгоритму-претендента SIMD.

Закінчення дод. Б

На рис. Б.14 представлено статистичний портрет програмної реалізації алгоритму-претендента Skein. З рис. Б.14 видно, що статистичний портрет програмної реалізації алгоритму-претендента Skein відповідає пропонованим вимогам – по 137 тестах позитивно пройшло більше 99% послідовностей, однак одна атака на даний алгоритм дозволяє отримати позитивний результат.

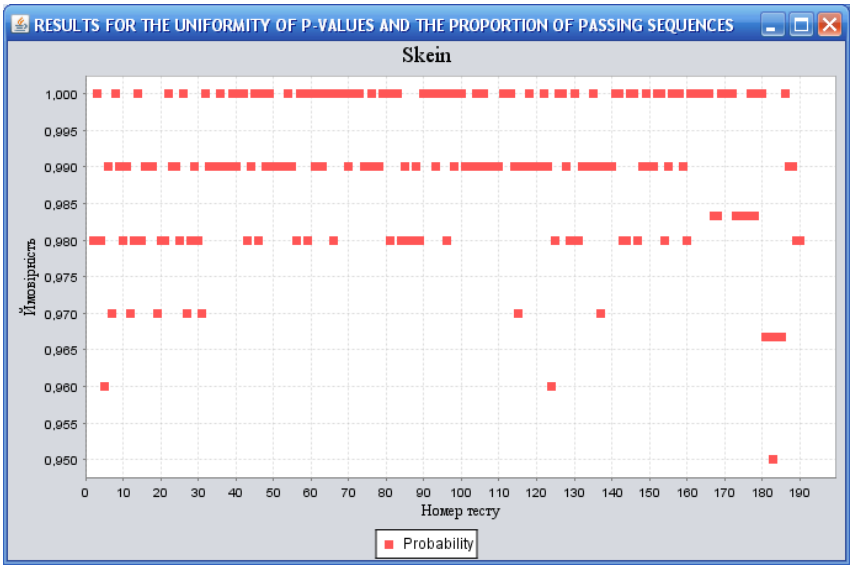

Рисунок Б.14 – Статистичний портрет програмної реалізації алгоритму-претендента Skein.

Додаток В

Діаграми бізнес-варіантів та варіантів використання програмного модулю

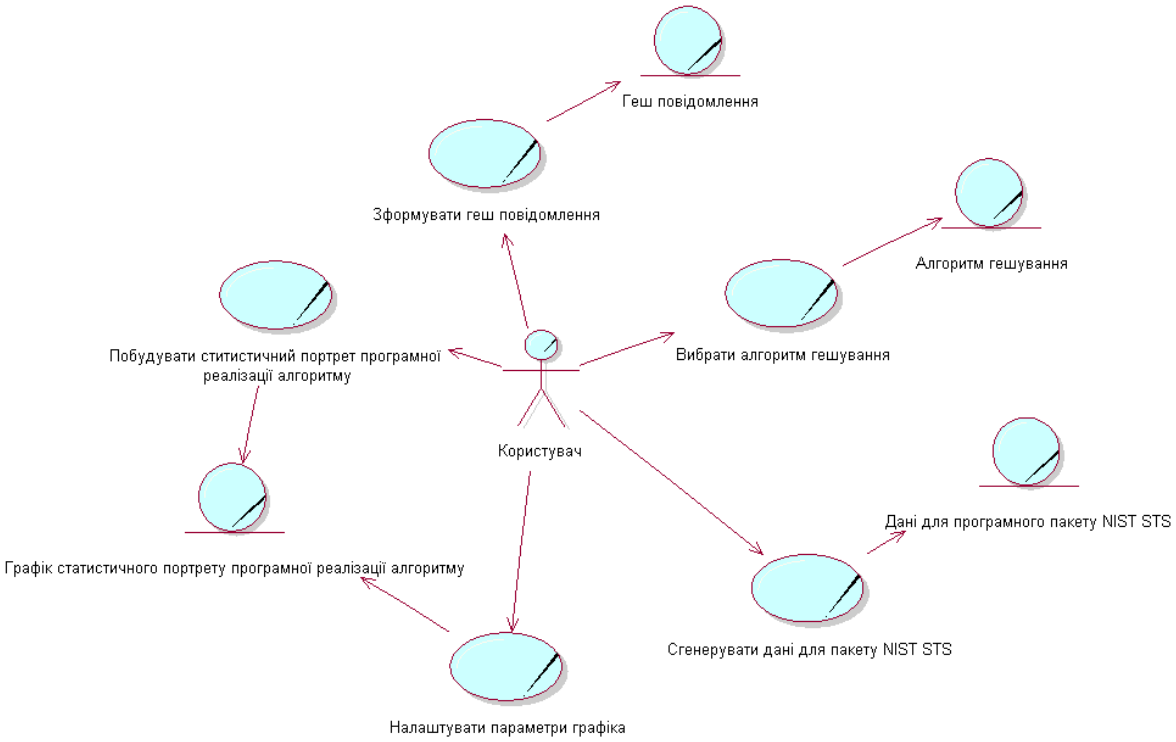

Рисунок В.1 – Діаграма бізнес-варіантів програмного модулю

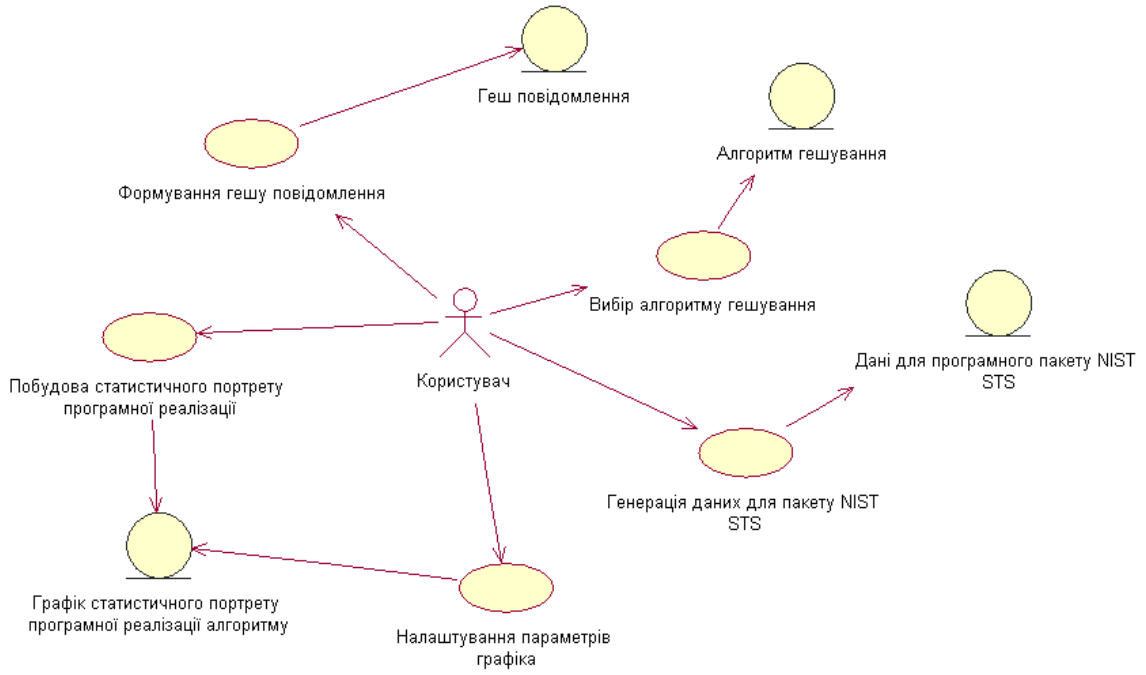

Рисунок В.2 – Діаграма варіантів використання програмного модулю

Додаток Г

Лістинг програми

```
package org.edu.ksuo.HashAnalyzer;
import java.io.InputStream;

/**
 *
 */
public interface HashAlgorithm {

   public static class HashingError extends Exception
   {
      public HashingError()
      {
      }

      public HashingError(Exception exc) {
         super(exc);
      }
   }

   HashValue compute(InputStream source) throws HashingError;
}


package org.edu.ksuo.HashAnalyzer;

import java.nio.charset.Charset;
/**
 *
 */
public class HashValue {
   private final byte[] digest;
   public HashValue(byte [] digest)
   {
      this.digest = digest;
   }
   @Override
   public String toString() {
      StringBuilder sb = new StringBuilder();
      if (digest.length % 2 != 0) sb.append("0");
      for (byte b: digest)
      {
         sb.append(String.format("%X", b));
      }
      return sb.toString();
   }
   public byte[] getRawData()
   {
      return digest;
   }
   public String toAsciiString()
   {
      return new String(digest, Charset.forName("US-ASCII"));
   }
}

package org.edu.ksuo.HashAnalyzer;

import fr.cryptohash.BLAKE512;
```

Продовження дод. Г

```
import fr.cryptohash.BMW512;
import fr.cryptohash.CubeHash512;
import fr.cryptohash.ECHO512;
import fr.cryptohash.Fugue512;
import fr.cryptohash.Groestl512;
import fr.cryptohash.Hamsi512;
import fr.cryptohash.JH512;
import fr.cryptohash.Keccak512;
import fr.cryptohash.Luffa512;
import fr.cryptohash.SHAvite512;
import fr.cryptohash.SIMD512;
import fr.cryptohash.Shabal512;
import fr.cryptohash.Skein512;
import java.util.HashMap;
import java.util.Map;
import org.edu.ksuo.HashAnalyzer.gui.MainForm;
import org.edu.ksuo.HashAnalyzer.impl.SaphirAlgo;
public class Main {
   public static void main(String [] args)
   {
      final Map<String, HashAlgorithm> algorithms = new HashMap<String, HashAlgorithm>();
      algorithms.put("BLAKE", new SaphirAlgo(new BLAKE512()));
      algorithms.put("Blue_Midnight_Wish", new SaphirAlgo(new BMW512()));
      algorithms.put("CubeHash", new SaphirAlgo(new CubeHash512()));
      algorithms.put("ECHO", new SaphirAlgo(new ECHO512()));
      algorithms.put("Fugue", new SaphirAlgo(new Fugue512()));
      algorithms.put("Grostl", new SaphirAlgo(new Groestl512()));
      algorithms.put("Hamsi", new SaphirAlgo(new Hamsi512()));
      algorithms.put("JH",    new SaphirAlgo(new JH512()));
      algorithms.put("Keccak", new SaphirAlgo(new Keccak512()));
      algorithms.put("Luffa", new SaphirAlgo(new Luffa512()));
      algorithms.put("SHAvite3", new SaphirAlgo(new SHAvite512()));
      algorithms.put("SIMD", new SaphirAlgo(new SIMD512()));
      algorithms.put("Shabal", new SaphirAlgo(new Shabal512()));
      algorithms.put("Skein", new SaphirAlgo(new Skein512()));
      // Show main form
      java.awt.EventQueue.invokeLater(new Runnable() {
         @Override
         public void run() {
            new MainForm(algorithms).setVisible(true);
         }
      });
   }
}
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package org.edu.ksuo.HashAnalyzer.impl;
import fr.cryptohash.Digest;
import java.io.IOException;
import java.io.InputStream;
import org.edu.ksuo.HashAnalyzer.HashAlgorithm;
import org.edu.ksuo.HashAnalyzer.HashAlgorithm.HashingError;
import org.edu.ksuo.HashAnalyzer.HashValue;
/**
 *
 * @author Natali
 */
```

Закінчення дод. Г

```
public class SaphirAlgo implements HashAlgorithm
{
   private final Digest d;
   public SaphirAlgo(Digest d) {
      this.d = d;
   }

   @Override
   public HashValue compute(InputStream in) throws HashingError
   {
      try
      {
         return new HashValue(d.digest(Helper.readAll(in)));
      }
      catch (IOException ioExc)
      {
         throw new HashingError(ioExc);
      }
   }
}

package org.edu.ksuo.HashAnalyzer.impl;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.jar.Pack200;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 *
 * @author Natali
 */
public class NistResultsLoader {
   private final static Pattern PATTERN = Pattern.compile("([ ]+[0-9]+[ ]+){10}([0-9]+\\.[0-9]+)[ ]+([0-9]+\\.[0-9]+)[
*]+[\\-a-zA-Z]+");

   public static List<Double> load(InputStream in)
   {
      final List<Double> res = new ArrayList<Double>();
      final Scanner scanner = new Scanner(in);
      while (scanner.hasNext())
      {
         final String line = scanner.nextLine();

         Matcher m = PATTERN.matcher(line);
         if (m.matches())
         {
            res.add(Double.parseDouble(m.group(3)));
         }
      }

      return res;
   }}
```