НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ім. ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

Кафедра обчислювальної техніки
(повна назва кафедри)

До захисту допущено
Завідувач кафедри
_____ С.Г. Стіренко
(підпис) (ініціали, прізвище)
«4» червня 2021 р.

Дипломний проект
на здобуття ступеня бакалавра

з напрямку підготовки **121 «Комп'ютерна інженерія»**
(код і назва)

на тему: Метод криптографічно строгої автентифікації та програмні засоби його реалізації

Виконав: студент 4 курсу, групи ІП-73
(шифр групи)
Сидоренко Костянтин Миколайович _____
(прізвище, ім'я, по-батькові) (підпис)

Керівник доцент, к.т.н. Марковський О.П. _____
(посада, вчене звання, науковий степінь, прізвище та ініціали) (підпис)

Консультант нормоконтроль д.т.н., проф. Сімоненко В.П. _____
(назва розділу) (вчені ступінь та звання, прізвище, ініціали) (підпис)

Рецензент декан ФПМ, д.т.н, проф. Дичка І.А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2021 р.

Національний технічний університет України

«Київський політехнічний інститут

ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень **бакалавр**

Напрям підготовки 121 **«Комп'ютерна інженерія»**

ЗАТВЕРДЖУЮ

Завідувач кафедри

Стіренко С.Г. _____

(прізвище, ініціали) (підпис)

« 4 » червня 2021 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Сидоренка Костянтина Миколайовича

(прізвище, ім'я, по-батькові)

1. Тема проекту Метод криптографічно строгої автентифікації та програмні засоби його реалізації

керівник проекту (роботи) Марковський Олександр Петрович, к.т.н., доцент,

(прізвище, ім'я, по-батькові, науковий ступінь,вчене звання)

затверджені наказом по університету від «*23*» *квітня* 2019 року № *1180-с*

2. Термін здачі студентом закінченого проекту (роботи) *5 червня 2021 р.*

3. Вихідні дані до проекту (роботи) *див. технічне завдання*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Аналіз проблеми ефективності ідентифікації віддалених абонентів в сучасних умовах. Огляд нині існуючих методів ідентифікації користувачів. Розробка програми прискореної реалізації хеш-трансформацій. Методи використання незворотних булевих перетворень для прискорення обчислювальної реалізації строгої ідентифікації користувачів.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) *Блок-схема алгоритму, структурна схема, схема методу строгої ідентифікації з використанням хеш-перетворень.*

6. Консультанти проекту (роботи), з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Норм. контроль	Сімоненко В.П проф., д.т.н.		

7. Дата видачі завдання 27.11.2020

КАЛЕНДАРНИЙ ПЛАН

№ /п п	Найменування етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітки
1	*Затвердження теми роботи*	*10.12.2020-15.12.2020*	
2	*Вивчення та аналіз завдання*	*15.12.2020-15.03.2021*	
3	*Розробка архітектури та загальної структури системи*	*15.03.2021-25.03.2021*	
4	*Розробка структур окремих підсистем*	*25.03.2021-05.04.2021*	
5	*Програмна реалізація системи*	*05.04.2021-15.04.2021*	
6	*Оформлення пояснювальної записки*	*15.04.2021-20.05.2021*	
7	*Захист програмного продукту*	*25.04.2021*	
8	*Передзахист*	*29.05.2021*	
9	*Захист*	*19.06.2021*	

Студент-дипломник _____

(підпис)

Керівник роботи _____

(підпис)

Анотація

В бакалаврському дипломному проекті розглянуто проблему строгої ідентифікації користувачів з використанням концепції нульових знань та запропоновані, обгрунтовані та протестовані методи прискорення даного процесу.

Програма дозволяє більш швидко вираховувати хеш-сигнатуру за стандартизованим алгоритмом для використання у розроблених методах. Програмний продукт був створений на мові C++ у візуальному середовищі Visual Studio 2019.

Експериментальні результати підтвердили ефективність запропонованого у роботі методу.

Annotation

The bachelor's thesis project considers the problem of strict identification of users using the concept of zero knowledge conception and proposed, substantiated and tested methods to accelerate this process.

The program allows you to calculate the hash signature more quickly according to a standardized algorithm for usage in developed earlier method. The software product was created in C ++ in the visual environment Visual Studio 2019.

Experimental results confirmed the effectiveness of the proposed method.

справки	Формат	*Значення*	*Найменування*	Кіл. листів	№ екземпляр	*Додаток*
			Документація загальна			
			Знову розроблена			
	А4	***ІАЛЦ.468243.002 ТЗ***	Метод криптографічно	4		
			строгої автентифікації та			
			програмні засоби			
			його реалізації			
			Технічне завдання			
	А4	***ІАЛЦ.468243.004 ПЗ***	Метод криптографічно	56		
			строгої автентифікації та			
			програмні засоби			
			його реалізації			
			Пояснювальна записка			
	А4	***ІАЛЦ.468243.005 Д1***	Метод криптографічно	1		
			строгої автентифікації та			
			програмні засоби			
			його реалізації			
			Блок-схема алгоритма			
	А4	***ІАЛЦ.468243.006 Д2***	Метод криптографічно	1		
			строгої автентифікації та			
			програмні засоби			
			його реалізації			
			Структурна схема			
	А4	***ІАЛЦ.468243.007 Д3***	Метод криптографічно	1		
			строгої автентифікації та			
			програмні засоби			
			його реалізації			
			Схема методу строгої			
			ідентифікації з використанням			
			хеш-перетворень			
	А4	***ІАЛЦ.468243.008 Д4***	Метод криптографічно	8		
			строгої автентифікації та			
			програмні засоби			
			його реалізації			
			Лістинг програми			

	Лист	***№ докум.***	***Підп.***	***Дата***

ІАЛЦ. 468243.001 ОА

Розроб	Сидоренко К.М		
Перев	Сімоненко В. П.		

Метод криптографічно строгої автентифікації та програмні засоби його реалізації
Опис альбому

Літ.	Аркуш	Аркушів
	1	1

«КПІ ім. Ігоря Сікорського», ФІОТ, ІП-73

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Технічне завдання поширюється на розробку методу криптографічно строгої ідентифікації користувачів та програмній його реалізації.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня “бакалавр комп’ютерної інженерії”, затверджене кафедрою спеціалізованих комп’ютерних систем Національного технічного університету України “Київський політехнічний інститут”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою проекту є підвищення ефективності ідентифікації віддалених абонентів в сучасних умовах, а також розробка кращого способу ідентифікації користувача у віддалених системах.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерела, що були використані для розробки є науково-технічна література з відновлення даних та корекції помилок, публікації в періодичних виданнях та публікації в Інтернеті.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

Розроблений метод має виконувати такі вимоги:

- Має забезпечувати темп формування сеансових паролей не більше 1.3мс;
- має бути орієнтованим на використання сучасних криптопроцесорів та криптоприскорювачів;
- розрядність сеансового паролю повинна бути не менше 150 і вони не повинні повторюватись

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 10
- Visual Studio 2019
- C++11

5.3. Вимоги до апаратної частини

- Процесор рівня Intel i5 і вище.
- Оперативна пам’ять не менше 500 МБ.
- Вільне місце на жорсткому диску не менше 100 МБ.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення та аналіз дипломного завдання	15.12.2020-15.03.2021
Створення та узгодження технічного завдання	15.03.2021-25.03.2021
Дослідження існуючих методів ідентифікації	25.03.2021-05.04.2021
Аналіз структури методу, що розробляється	05.04.2021-15.04.2021
Програмна реалізація системи	15.04.2021-5.05.2021
Відлагодження та виправлення помилок	15.05.2021
Оформлення документації дипломного проекта	06.06.2021

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SHA	(Secure Hash Algorithm) алгоритм криптографічного хешування
RIPEMD	(Race Integrity Primitives Evaluation Message Digest) Криптографічна хеш функція
ШБ	шифроблок
ПЗ	пряме заміщення
ЗЗ	зворотне заміщення
ВБС	відновлення байтів стовпця
DES	(Data Encryption Standard) симетричний алгоритм шифрування даних
ПСР	прямий зсув рядків
FFSIS	(Feige Fiata Shamir Identification Scheme) - схема ідентифікації
АЛП	Арифметично-логічний пристрій

ВСТУП

Наразі мережа Інтернет стрімко розвивається і знаходить абсолютно нові ніші в усіх аспектах сучасного суспільства. Вона планомірно знижує вплив телебачення, замінює собою телефонний зв'язок. Разом з тим все більше стає актуальною задача надійного та швидкодійного способу захисту персональної інформації користувачів.

В наш час відбулось значне розширення можливостей для зловмисників для доступу до конфіденційних інформаційних ресурсів за рахунок порушення порядку процедур ідентифікації. Основною причиною такого процесу є дуже стрімке зростання обчислювальних потужностей за рахунок використання сучасних технологій розподілених та особливо хмарних обчислень.

Також значного поширення набула процедура перехоплення шахраями пароля справжнього користувача, а також його заміна після проведення сеансу ідентифікації, не кажучи про методи порушення процесу віддаленої ідентифікації, які можна назвати банальними: вплив на персонал або вірусну атаку.

Способи якнайякіснішої протидії зловмисникам з застосуванням найменшої кількості ресурсів є як ніколи популярною темою для реалізації в складних системах, які використовуються одночасно великою кількістю людей.

РОЗДІЛ 1

ОГЛЯД СУЧАСНИХ МЕТОДІВ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

В сучасному інформаційному суспільстві , коли інформаційна безпека є обов'язковою для будь-кого, технології захисту даних, розподілення доступу до них, потреба щодо забезпечення їх автентичності та цілісності стає все більш і більш актуальною для спеціалістів.

В останні десятиліття прогрес засобів захисту інформації відбувається в сторону ускладнення алгоритмів через збільшення розрядності чисел, які в них застосовуються, і це зрозуміло, бо складність цих самих обчислень зростає по експоненті зі збільшенням розрядності. До останніх років це не було проблемою, але зараз помітні проблеми, пов'язані зі збільшенням складності реалізації цих алгоритмів.

Дуже критично зараз ця проблема стоїть для способів ідентифікування користувачів великих розподілених систем. Абсолютний захист конфіденційності може бути досягнутий лише в рамках використання криптографічно строгих методів ідентифікації, що мають за основу теоретичну концепцію "нульових знань". Суттєвою перешкодою широкому застосуванню цієї прогресивної технології є складність її практичної реалізації, яка потребує значних часових ресурсів. Таким чином, наукова задача підвищення ефективності ідентифікації віддалених користувачів інтегрованих систем за рахунок розробки методів прискорення її реалізації, є актуальною та практично важливою.

1.1. Аналіз сучасних вимог до систем ідентифікації

Станом на сьогоднішній день, сервіси безпеки у розподілених складних мережах можуть бути атаковані зловмисниками, як з мережевого середовища, так і з локального. Типи загроз можна класифікувати наступним чином[14]:

1. Здійснення несанкціонованого фізичного доступу зловмисником до обчислювальної системи, в якій працює сервіс безпеки;

2. Спроби обходу зловмисником методів захисту, котрі направлені на отримання прямого незаконного шляху в систему;

3. Підміна користувача – процес, коли злочинець отримує доступ до системи під виглядом привілегійованого користувача, а саме системного адміністратора. Здійснення цього можливо попередньо перехопивши дані для ідентифікації сисадміном;

4. Підміна серверу – злочинець за допомогою сторонніх програм підміняє шлях до віддаленого серверу та імітує його зі сторони. Дана дія надає зловмисникові можливість маніпулювання користувачем, для отримання від нього конфіденційної інформації;

5. Несанкціонований доступ до конфіденційної інформації, зокрема маніпулювання та підміна даних, заважання процесу їх шифрування або дешифрування;

6. Зміна чи підміна інформації, яка захищена криптографічними механізмами.

Також виділяються 2 стадії процесу ідентифікації користувача у віддаленій системі:

1. Стадія ініціалізації – під час цієї фази користувач отримує деяку секретну інформацію або самостійно її генерує. Ця інформація, або її перетворення в майбутньому будуть давати йому доступ у систему.

2. Стадія перевірки легальності доступу – після ініціалізації система періодично перевіряє, що даний абонент дійсно знає отриману секретну інформацію.

Забезпечення неможливості сторонньому користувачу доступу в систему ідентифікації віддалених абонентів систем обробки інформації є найголовнішою вимогою до будь-якої системи такого роду. Але, важливо пам’ятати, що хоч і обчислювальні можливості такої системи у сучасних реаліях є майже нескінченні і ми можемо розробити дуже ресурсоємку систему ідентифікації, вона також частково завжди буде виконуватись на стороні абонента, що може становити певного роду проблеми для користувачів з застарілим обладнанням.

Отже і тут ми повинні знайти баланс між складністю алгоритму, його надійністю та швидкодією.

1.2. Огляд існуючих систем ідентифікації

Переважна більшість систем безпеки, які використовуються у наш час у мережі, застосовують механізм ідентифікації в основі якого лежить схема ідентифікації користувач-пароль. В останні декілька років до цієї системи додався принцип двохфакторної аутентифікації через мобільний телефон або/і електронну почту та механізми зламу залишилися принципово такими ж, адже дуже часто достатньо зламати лише одну ланку для знівельовування двохфакторної системи через користувачів, які ставлять собі однакові пароль та через сучасну систему, яка дозволяє вислати новий пароль на почту чи телефон. Окремо стоять системи на основі біометичної ідентифікації користувача, і, хоч вони і є дуже надійними, адже підробити щось людське набагато важче, до їх недоліків можна віднести:

1) Майже повна неможливість використання віддалено

2) Велика вартість таких установок

3) Велика кількість помилок другого роду - відмова у доступі вже зареєстрованому користувачеві.

В бакалаврській роботі будуть розглядатись більш “старомодні” системи на основі принципу користувач-пароль.

Способи ідентифікації, які мають за основу використання користувачем імені та паролю для реєстрації, є достатньо зручною для середнього користувача. Кожному користувачеві системи присвоюється унікальний ідентифікатор з заданим відповідно паролем. Ідентифікація за допомогою паролю є дуже простою, тому, на даний момент, вона найчастіше використовується в мережі. Проходить пошук введеного суб’єктом реєстраційного імені у відповідному реєстрі, а пароль порівнюється зі збереженим. Доступ у систему надається, якщо вони співпадають, інакше доступ забороняється. Проте, описана автентифікація доволі проста та має багато недоліків:

1) Зловмисник має можливість дізнатися пароль абонента зі спеціального файлу з паролями на комп’ютері користувача або з його збереженої копії;

2) Злочинець, використовуючи мімкрію під адміністратора ресурсу, може за допомогою соціальної інженерії надати свій пароль користувачу або абонента змінити на вказаний ним пароль - інакше кажучи, вразливість до соціальної інженерії;

3) Злочинець здійснює спроби отримати доступ до системи використовуючи ім’я користувача жертви та підібрати потрібний пароль базуючись на отриманих особистих даних про неї;

4) Для визначення імен користувачів та їх паролів, злочинець може здійснювати аналіз мережевого трафіку від клієнта до сервера, який був ним перехоплений.

Враховуючи все вищенаведене, можна сказати, що такі системи є дуже ненадійними саме через наведені вище причини. Класичним засобом протидії зловмиснику являється періодичне повторне проведення сеансів ідентифікації в процесі взаємодії системи з абонентом. Для цього процедура ідентифікації повинна виконуватися достатньо швидко, образно кажучи, на льоту.

В першому наближенні процедура ідентифікації має задовольняти таким вимогам [1]:

1) Структура зберігання секретної ідентифікаційної інформації має бути наступною: одна її частина зберігається у системі, а інша – у користувача. Якщо зловмисник дізнається одну частину інформації, він не повинен вирахувати її цілком, тобто кожна з цих частин не була б достатньою для підробки доступу до ресурсів системи.

2) Інформація, яку висилає абонент для ідентифікації у системі, повинна змінюватися.

3) Об’єм секретної інформації для ідентифікації, що знаходиться у системі і використовується під час процесу має бути якомога меншим для того, щоб надати можливість зберігання такої інформації в спеціально захищеній на апаратному рівні пам’яті для додаткового захисту системи від зловмисників.

Всі сучасні протоколи ідентифікації абонентів можна розділити на два класи: з використанням паролів, що перевіряються системою шляхом порівняння (“слабка” ідентифікація) та на основі теоретичної концепції “нульових знань” (“строга” ідентифікація) [3].

Типовим представником концепції “слабкої ідентифікації” є ідентифікація користувача у Unix-подібних мережах.

ID – індивідуальний ідентифікатор, котрий надає змогу отримати з множини всіх зареєстрованих користувачі *i*-го користувача системи;

C – інформація, яка додатково підтверджує аутентичність *i*-го абонента системи.

Основні властивості протоколу ідентифікації UNIX у загальному вигляді такі :

1. Користувач зобов'язаний представити власний ідентифікатор ID;

2. Система здійснює перевірку в пам'яті існуючих ідентифікаторів для перевірки факту існування подібного ідентифікатора. За умови успіху суб'єкт, котрий іменується *i*-м суб'єктом доступу, здійснив ідентифікацію. В протилежному випадку в отриманні доступу буде відмовлено;

3. Система відправляє запит для того, щоб абонент ввів інформацію для початку процесу ідентифікації C;

4. Система генерує значення $Z= \xi(ID,C)$;

5. Система порівнює значення Z і K.

Суб'єкт доступу здійснює ідентифікацію за умови, якщо значення збігаються. Інакше користувач не отримає доступ в Unix-подібну систему.

Ще одна частина протоколів ідентифікації орієнтована на застосовуванні приватного спільного ключа. Розглянемо нижче схему ідентифікації користувачів, яка базується на застосуванні приватного спільного ключа, що може також використовуватися також у протоколах типу вигук-відгук [6]. Базовою концепцією для цього протоколу слугує паттерн, який застосовується у класичних ідентифікаційних протоколах : з одної сторони відсилається випадкове число іншій, яка перетворює це число за допомогою функціонального перетворення, а потім повертає назад результат.

1. Системі надсилається власний особистий ідентифікатор користувача;

2. Для додаткового підтвердження істинності, система випадковим чином генерує число (дуже велике) та відсилає в якості відгуку для додаткової перевірки ;

3. За допомогою приватного ключа, який знають сторони, користувач здійснює шифрування цього повідомлення та надсилає результат у систему;

4. Система обов'язково перевіряє повідомлення для того, щоб переконатися, що його шифрування здійснювалося за допомогою приватного ключа, який знає лише користувач;

5. Але, на кожному кроці користувач повинен бачити, що він спілкується із системою, а не з її підробкою. Отже, наступні кроки протоколу циклічні: користувач надсилає відгук, а система має кожен раз відповісти на нього щоб не розірвати сеанс зв'язку.

Дана модифікація, з одного боку, дуже швидка при реалізації, але з іншого, не виключає можливість здійснення «дзеркальної атаки» на систему[7]. Видаючи себе за звичайного користувача зловмисник відправляє системі вигук. Система, зі свого боку, відповідає особистим вигуком. Далі зловмисник відкриває другий сеанс повідомленням 3, та надсилає системі її власний вигук з 2-го повідомлення. При такому сценарії, коли зловмисник матиме можливість одночасно здійснювати декілька сеансів зв'язку з системою, він зможе втрутитись в саму роботу протоколу.

1.3. Аналіз можливостей підвищення ефективності криптографічно строгої ідентифікації

Звичайно, на ринку присутні й деякі алгоритми, які втілюють в собі ідею строгої ідентифікації. Найвідоміші з них це схема Шнорра , схема Гіллоу-Квіскватера та FFSIS[2] (Feige Fiata Shamir Identification Scheme).

FFSIS забезпечує високу надійність процедури ідентифікації. Проте ця схема має свої недоліки, основним з яких є сильне перевантаження ліній

передач, так як FFSIS необхідна велика кількість обміну даними в процесі ідентифікації. В випадку, коли є проблеми з перевантаженням ліній передач, більш доцільним є використання алгоритмів Гіллоу-Квіскватера та Шнорра.

Всі вищеназвані алгоритми мають в суті операцію модулярного експоненціювання, що робить їх реалізацію не лише складною чисто технічно, а й дуже ресурсоємкою - при зростанні числа, над яким проводяться такі маніпуляції, складність обчислень зростатиме експоненційно. Для прикладу розгланемо детальніше схему Гіллоу-Квіскватера.

Генерація ключів в схемі Гіллоу-Квіскватера виконується наступним чином:

Абонент отримує відкритий пароль J, що на практиці являє собою хеш-сигнатуру символьного рядка деякого значення, прив'язаного до імені абонента. Відкритий ключ у системі також містить у собі деяке число n – добуток двох наших простих чисел *p* і *q*. Ці числа вже не є відомими, абсолютно так як і просте число *v*.

Для алгоритму існує рівність *((J*B^(v)) mod n)*, у відповідності до якої підбирається секретний ключ B. Наприклад, p=31 і q=29. Сам процес є доволі розповсюдженим в криптографії, зокрема приблизно по такому принципу працює знаменитий RSA.

Процес ідентифікації ж зводиться до наступної послідовності дій:

1. Абонент формує випадкове число r таке, що 1< r < (n-1), потім обчислює значення *T=((r^v) mod n)*, та відсилає T в систему.

2. Система генерує випадкове d, яке має знаходитися на проміжку 0 < d < (n-1). Це значення також пересилається віддаленому абоненту.

3. Абонент обчислює та відправляє обчислений код *D = (r*(B^d) mod n)* в систему.

4. В системі вираховується T_1 *= ((D^v)*(J^d) mod n)* якщо T_1 *= T*, то результат ідентифікації виходить позитивним. Далі процедура повторюється, звичайно ж, ітеративно.

Схема Шнорра [3] являє собою альтернативу схемам FFSIS та Гіллоу-Квіскватера[4]. Надійність схеми базується на складності обчислення дискретного логарифму. Схема Шнорра дозволяє проводити попередні розрахунки, що зручно при незначних обчислювальних ресурсах. Фактично передається лише три повідомлення – це дозволяє зменшити взаємодію в мережах з невеликою пропускною здатністю. Ця схема використовується на державному рівні в Білорусії та Південній Кореї.

Абонент обирає два простих числа p і q, причому останнє повинно бути множником (p-1). Наприклад q=5, тоді значення обчислюється як добуток на будь-яке парне число: 2n*(p-1). Наприклад, p=41.

Далі необхідно вибрати значення *a* таке, щоб задовольняло умову *(a^q) mod p =1*. Для нашого прикладу можна взяти *a=10*. Далі вибирається випадкове число *s<q*, наприклад s=1,обчислюється.

Число s є закритим ключем. Для генерації відкритого ключа потрібно розрахувати *v = (a^(-s)) mod p.*

Процес ідентифікації звичайного абонента можна представити як наступну послідовність:

1. На стороні абонента вибирається випадкове r<q , обчислюємо *x=a*r mod*. Потім посилаємо це число в систему.

2. Система генерує випадкове *e < (2^t - 1)* та посилає це значення абоненту. В рамках розглянутого прикладу.

3. Наступним кроком для абонента є обчислення значення *y = (r + s*e) mod q* та відправлення його в систему.

4. Система перевіряє виконання рівності. В такому підтвердження ідентифікація може вважатися успішною.

З викладеного випливає, що основною обчислювальною операцією майже всіх відомих схем строгої ідентифікації віддалених абонентів є модулярне експоненціювання над числами, довжина яких значно перевищує

розрядність процесору. Також, варто взяти до уваги, що даним алгоритмам вже пішов третій десяток, що свідчить про те, що їх треба замінювати на більш вдалі. В умовах сучасних трендів до збільшення розрядності чисел для протидії порушникам, обчислювальна складність реалізації даного типу операцій збільшується експоненційно, значно випереджаючи темпи зростання продуктивності існуючих комп’ютерних систем.

Висновки до розділу 1.

В результаті проведених досліджень, котрі були направлені на здійснення аналізу нинішнього стану проблеми ефективної ідентифікації віддалених абонентів розподілених систем обробки інформації, розгляду наявних протоколів та методів ідентифікації можна виділити наступні висновки:

Хоча на даний момент ідентифікація та аутентифікація за допомогою системи паролів є не найнадійнішою, вона безумовно є найкращою для віддалених розподілених систем.

1. Головним недоліком ідентифікації, яка базується на паролях є недостатній рівень захисту від спроб здійснення несанкціонованого доступу, різноманіття типів якого в теперішніх умовах різко зростає. Описаний недолік визначений відсутністю в системах ідентифікації, які базуються на паролях, розвинених криптографічних механізмів, які надають можливість сеансової модифікації ідентифікуючої інформації, вразливість від отримання несанкціонованого доступу з боку системи даних, які застосовуються під час ідентифікації, недотриманням одного з головних принципів захисту інформації – присутності лише одного єдиного власника секретних даних.

2. Також було розглянуто декілька популярних протоколів захисту процесу ідентифікації, які можна вважати “слабкими”. Всі вони виявились недостатньо надійними, незважаючи на повсякмісне використання.

3. Алгоритми ідентифікації віддаленого абонента, які не використовують концепцію строгої ідентифікації, є абсолютно ненадійними у сучасних реаліях.

РОЗДІЛ 2

МЕТОДИ ВИКОРИСТАННЯ НЕЗВОРОТНИХ БУЛЕВИХ ПЕРЕТВОРЕНЬ ДЛЯ ПРИСКОРЕННЯ ОБЧИСЛЮВАЛЬНОЇ РЕАЛІЗАЦІЇ СТРОГОЇ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

Теоретична концепція строгої ідентифікації або концепція “нульових знань” в теоретичному плані найбільш повною мірою на даний момент відповідає вимогам щодо системи ідентифікації абонентів. В сучасних умовах і у найближчій перспективі достатній рівень надійності ідентифікації може бути забезпечений лише методами, що виконані на основі строгої ідентифікації.

“Нульові знання” - ідея, суть якої полягає у використанні для збереження даних незворотних математичних перетворень. Це означає, що існує можливість перетворення в прямому напрямку, та, водночас, принципово неможливим є аналітичне знаходження алгоритму оберненого перетворення. В майже всіх існуючих схемах ідентифікації на основі даної концепції для практичної реалізації перетворень використовують принципово нерозв’язувані аналітично на даний момент проблеми, особливо популярною є задача дискретного логарифмування.

Через це базовою обчислювальною операцією більшості схем строгої ідентифікації віддалених абонентів використовується модулярне експоненціювання над великими числами, довжина яких дуже велика в порівнянні з розрядністю процесору. В умовах постійного зростання розрядності чисел, обчислювальна складність реалізації вказаного типу операцій збільшується також експоненційно, випереджаючи темпи зростання продуктивності комп’ютерних систем.

Одним з найкращих шляхів покращення швидкодійності реалізації строгої ідентифікації користувачів є перехід до булевої алгебри. В основі

незворотних булевих перетворень лежить принцип нерозв'язуваності систем нелінійних булевих рівнянь аналітичними методами. Він також лежить в основі значної частини сучасних механізмів криптографічного захисту інформації: хеш-перетворень, поточних шифрів, алгоритмів симетричного шифрування. Основною перевагою використання незворотних булевих перетворень вважається [8] висока швидкість обчислювальної реалізації. Через це, логічним є дослідження можливостей використання цих перетворень для реалізації криптографічно строгої ідентифікації віддалених користувачів.

На даний момент відомі два основні способи використання незворотних булевих перетворень: без зміни вихідного коду або зі зміною, за певним законом вихідного вектору.

Незворотні булеві перетворення можуть бути спеціальним чином зімітовані користувачем, або бути реалізованими використанням вже існуючих, які присутні в стандартизованих алгоритмах (SHA, RIPEMD) чи шифроблоках симетричного шифрування. Реалізація незворотних перетворень в другому варіанті ефективніша, оскільки незворотність стандартизованих алгоритмів для булевих перетворень багаторазово перевірена та протестована. Також на ринку існує багатий вибір криптопроцесорів та криптоакселераторів, які можуть апаратно реалізувати згадані вище блоки криптографічних булевих перетворень.

При позначенні через $\phi(h)$ перетворення, які виконуються системою для ідентифікації користувача по паролю h, то при незмінному коді Y ідентифікації користувача потрібно реалізувати наступне булеве перетворення $f(h)$, для якого користувач отримає сукупність паролів $h_1, h_2,\ldots,h_n$ $\forall h_i$, $i=1,2,\ldots,n$: $f(h_i) = Y$. Дане перетворення теоретично може бути представлене як хеш-перетворення з вбудованими в нього колізіями. Система, маючи на вході незворотне перетворення $\phi(h)$ та вихідний вектор Y, не здатна

отримати будь-який з кодів паролів h_1, h_2,…, h_m для яких $f(h_j) = Y$. Дослідженню задачі побудови такого перетворення присвячено останній підрозділ поточного розділу.

Інший варіант реалізації строгої ідентифікації користувачів незворотних перетворень полягає в тому, що код Y ідентифікації користувача змінюється на кожній ітерації, тобто при кожному зверненні його до системи. Для більш простого варіанту в якості коду Y ідентифікації користувача на поточному сеансі використовується лише попередній пароль.

Такий варіант реалізації строгої ідентифікації бажано реалізовувати з використанням стандартизованих хеш-перетворень.

2.1. Метод ідентифікації на основі стандартизованих хеш-перетворень

Стандартизований хеш-перетворювач (Н) – сертифікований відповідними органами алгоритм незворотного перетворення інформаційного блоку довільної довжини в хеш-сигнатуру фіксованої розрядності *h*. Найбільш відомими є хеш-перетворювачі SHA та RIPEMD, які формують хеш-сигнатуру довжиною 8 байт (*h*=160). Також використовується варіант хеш-перетворення збільшеної розрядності SHA-256, (*h*=256). Найважливішою характеристикою хеш-перетворювачів є незворотність – практична неможливість віднаходження інформаційного блоку, хеш-сигнатура якого дорівнює заданій. Запропонований метод визначає процедуру ініціалізації та сеансової ідентифікації для кожного наступного звернення до системи зі сторони користувача.

При реєстрації користувача виконується наступні дії :

1) Визначається кількість *n* циклів ідентифікації (на стороні користувача).

2) Випадковим чином генерується *n*–тий сеансовий пароль P_n.

3) Обчислюються *n*-1 паролів, причому кожен *i*-тий пароль P_i, $i=n-1,\ldots,0$

обчислюється як хеш-перетворення $H(x)$ від конкатенації паролю з попередньої ітерації та номера сеансу: $P_i = H(P_{i+1} \| j)$.

4) У систему відсилається, P_0 ,перший пароль, зашифрований її відкритим ключем.

Послідовність дій для *i*–того сеансу ідентифікації має вигляд:

1) Відкритим ключем системи шифрується *i*-тий сеансовий пароль (P_i) і відсилається в систему.

2)Система виконує хеш-перетворення над конкатенацією отриманого паролю та номера сеансу: $\xi = H(P_i \| i)$ і порівнює результат з паролем з попереднього сеансу P_{i-1} : якщо $\xi = P_{i-1}$ то надається доступ.

Без даних, що ми передаємо, система, маючи в розпорядженні лише попередній пароль P_{i-1} не здатна сама генерувати наступний пароль P_j : це зводиться до злому стандартизованого хеш-алгоритму. Це унеможливлює застосування для цього інших методів крім перебору. При цьому, в середньому, для цього алгоритму потрібно виконати 2^{h-1} прорахунків хеш-перетворення, що для стандартизованого SHA-1 становить 2^{159} реалізацій алгоритму і майже завжди виходить за рамки практичної доцільності.

Виходячи з цього, можна вважати, що задача підбору пароля як стороннім користувачем так і деякою системою потребує ресурси, що виходять за рамки практичної доцільності.

Структурна схема перетворень запропонованого методу показана на рис. 2.1.

Основною перевагою запропонованого вище методу є висока швидкість обчислювальної реалізації при строгій ідентифікації системою, яка може обслуговувати в реальному часі до декількох сот тисяч користувачів.

Основним же недоліком методу є те, що всі паролі будуть генеруватися користувачем відразу і зберігатися в пам’яті. Також згенеровані паролі

мають жорстку прив’язку до номеру сеансу ідентифікації, тобто повинні використовуватися лише послідовно.

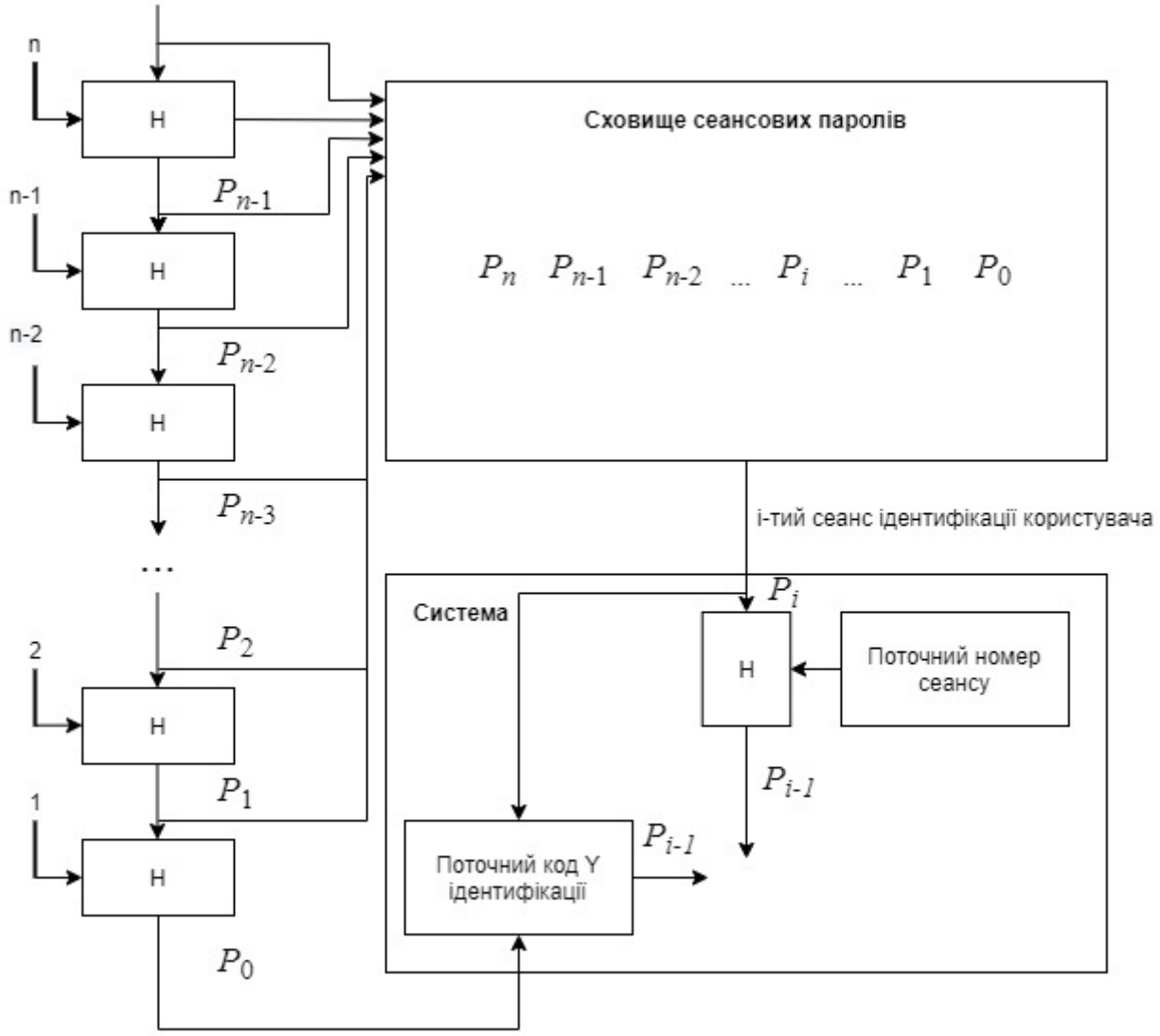


Рис 2.1. Структурна організація строгої ідентифікації з використанням ланцюжка хеш-перетворень

Беручи за основу вищезгаданий метод, на його базі були розроблені більш складні методи строгої ідентифікації користувачів розподілених систем з імплементуванням хеш-перетворювачів та шифроблоків.

2.2. Строга ідентифікація з використанням стандартизованих шифроблоків.

Нелінійні булеві функціональні перетворення лежать в основі значної кількості способів криптографічного захисту інформації, як блокові алгоритми, хеш-функції, потокові схеми шифрування. Такі типи перетворень є базою систем нелінійних булевих функцій, а також володіють властивостями, які використовуються для забезпечення ефективності засобів захисту побудованих на їх основі [22]. Одним з таких прикладів є стандартизований шифроблок, пристрій для симетричного шифрування даних.

Стандартизований шифроблок (ШБ) - сертифікований відповідними державними органами алгоритм симетричного, тобто з використанням однакового ключа, шифрування-дешифрування блоку даних фіксованої довжини.

У межах поставленої задачі пропонується метод строгої ідентифікації віддалених користувачів, що програмно реалізує концепцію “нульових знань” з використанням як основи стандартизованих шифроблоків.

Структурно шифроблок в режимі шифрування можна розглядати як функціональний перетворювач F, на вхід якого подаються n -розрядний блок D даних та m –розрядний ключ K. На виході ж отримується n –розрядний блок C з зашифрованими даними: $C = F(D,K)$. Для шифроблоку Rijndael запрограмовано шифрування блоків, довжиною 128, 192 або 256 розрядів. В запропонованому методі нами використовується режим шифрування блоків для 256 бітів ($n = 256$). Відповідно, в режимі дешифрування шифроблок реалізує зворотне перетворення: на його входи подаються n -розрядний блок C зашифрованих даних та m–розрядний ключ K, а на виході відновлюється n -розрядний блок даних $D= F(C,K)$.

Симетричні криптографічні шифроблоки можуть бути побудовані на базі двох основних принципів: в основу першого з них покладена ідея К. Шеннона, яка передбачає формування шифротекста (С) як порозрядної суми за модулем 2 коду (М) повідомлення і результату деякого функціонального перетворення G (M, K) над кодом ключа, що може залежати і від повідомлення. Процес дешифрування, згідно з цією ідеєю полягає у повторному порозрядному додаванні по модулю 2 коду шифротексту з результатом функціонального перетворення над кодом ключа (іноді даних) - G (M, K):

$$C = M \oplus G(M, K)$$
$$M = C \oplus G(M, K) \quad . \tag{2.1}$$

Отже, оборотність криптографічних шифроблоків, в основі яких лежить ідея Шеннона базується на операції додавання по модулю 2, а саме на її оборотності. До цього типу відноситься більшість симетричних криптографічних шифроблоків, в тому числі DES, ГОСТ 28147-89, IDEA, SAFER, RC-6, MARS. Такі алгоритми отримали назву лінійних, оскільки текст вихідного повідомлення входить в шифротекст лінійно.

Нелінійні симетричні алгоритми базуються на використанні спеціальних оборотних фунцкій ϕ(К), що С = ϕ (К, М) і М = ϕ (К, С). В якості таких функцій використовуються оборотні симетричні перетворення на полях Галуа [11] GF (2^n). Найбільш характерним симетричним криптографічним алгоритмом розглянутого класу є алгоритм Rijndael, в якому здійснюється для шифрування багаторазове оборотне, залежне від ключа функціональне перетворення на кінцевих полях Галуа для дешифрування з метою відновлення вихідного тексту, ці ж функціональні перетворення виконуються в зворотному порядку[20].

Найпоширенішим на практиці є алгоритм Rijndael. Цей алгоритм оперує з блоком даних , байти якого формують матрицю станів, що складається з 4-

х рядків та 4, 6 або 8 стовпців. Згідно цього алгоритм може працювати відповідно з блоками інформації в 128, 192 або 256 біт. Довжина ж ключа, якає незалежною від довжини блоку даних може ,в свою чергу, бути довжиною 128, 192 або 256 біт. Виконання алгоритму проходить в 10, 12 або 14 циклів, довжина котрих залежить від вибору певного інформаційного блоку та ключа. З ініціюючого блоку ключів відповідною процедурою видається необхідне число циклічних ключів, використовуваних на кожному циклі. При апаратній реалізації прийнято, що довжина блоку і даних, і ключа становить 192 біта, а, кількість циклів, відповідно, повинна бути рівною 10. Після криптографічного перетворення матриця станів

$$M = \begin{bmatrix} m_{00} m_{01}, m_{02}, m_{03} \\ m_{10}, m_{11}, m_{12}, m_{13} \\ m_{20}, m_{21}, m_{22}, m_{23} \\ m_{30}, m_{31}, m_{32}, m_{33} \end{bmatrix}$$

перетвориться в матрицю С станів шифрованого представлення:

$$C = \begin{bmatrix} c_{00}, c_{01}, c_{02}, c_{03} \\ c_{10}, c_{11}, c_{12}, c_{13} \\ c_{20}, c_{21}, c_{22}, c_{23} \\ c_{30}, c_{31}, c_{32}, c_{33} \end{bmatrix}$$

При дешифруванні матриця С дасть на виході М, матрицю вихідного інформаційного блоку. Rijndael використовує чотири базових операції: пряме і інверсне байтове заміщення, циклічний зсув рядків матриці станів, переміщення стовпців, сума по модулю 2 поточної матриці станів з певною матрицею розширення ключа.

В якості прямого заміщення (ПЗ), яке виконується над байтом G використовується операція отримання мультиплікативної інверсії $R=G^{-1} \bmod p(x)$ байта на полі Галуа за поліномом $p(x)=x^8+x^3+x+1$, з наступним кроком - лінійним перетворенням отриманої інверсії R шляхом множення її на

фіксовану матрицю M_d і підсумовування по модулю 2 з фіксованим байтом B={01100011} : D= S(G)= (M_d×R)⊕B. На практиці ж ця 8-бітова опера виконується шляхом табличної заміни: всі 256 можливих варіантів мультиплікативної інверсії наперед підраховуються і зберігаються в постійній табличній пам'яті.

За допомогою математичних перетворень ЗЗ байта D зводиться до виконання зворотного лінійного перетворення за допомогою матриці M_r: $G=S^{-1}(D)=(M_r\times(D\oplus B))^{-1} \bmod p(x)$. Матриця прямого M_d і зворотного M_r перетворення і байт B використовуються в алгоритмі Rijndael[16] мають вигляд:

$$M_d = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \qquad M_r = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Пряме і зворотне байтове заміщення ілюструється наступним прикладом. Нехай G=6Fh, $R=G^{-1} \bmod p(x)$=3Bh, D=(R×M_d)⊕B=0A8h. (D ⊕ B) × M_r = 3Bh, $(3Bh)^{-1} \bmod p(x)$ = 6Fh.

В матриці станів переміщення байтів виконується через лінійне перетворення Y = P (X) байтів стовпця X={x_0,x_1,x_2,x_3}:

$$y_0 = 2\cdot x_0 \oplus 3\cdot x_1 \oplus x_2 \oplus x_3$$
$$y_1 = x_0 \oplus 2\cdot x_1 \oplus 3\cdot x_2 \oplus x_3$$
$$y_2 = x_0 \oplus x_1 \oplus 2\cdot x_2 \oplus 3\cdot x_3$$
$$y_3 = 3\cdot x_0 \oplus x_1 \oplus x_2 \oplus 2\cdot x_3$$

де - Y = {y0, ..., y3}-вихідний стовпець матриці станів, y_0, y_1, y_2, y_3-байти його складові, а операції множення 8-розрядних кодів виконуються за законами множення на полях Галуа (використаний поліном $x^8+x^4+x^2+x+1$).

Операція відновлення $X=P^{-1}(Y)$ байтів стовпця (ВБС) $X=\{x_0,x_1,x_2,x_3\}$ за кодом $Y=\{y_0,\ldots,y_3\}$ полягає в лінійному перетворенні:

$$x_0 = 0Eh \cdot y_0 + 0Bh \cdot y_1 + 0Dh \cdot y_2 + 09 \cdot y_3$$
$$x_1 = 09 \cdot y_0 + 0Eh \cdot y_1 + 0Bh \cdot y_2 + 0Dh \cdot y_3$$
$$x_2 = 0Dh \cdot y_0 + 09 \cdot y_1 + 0Eh \cdot y_2 + 0Bh \cdot y_3$$
$$x_3 = 0Bh \cdot y_0 + 0Dh \cdot y_1 + 09 \cdot y_2 + 0Eh \cdot y_3$$

Операція ПСР являє собою зсув вліво j-того рядка матриці станів на j байтових позицій, який виконується у циклі, а суть операції ЗСР представляється в циклічному зсуві вправо j-того рядка на j байтових позицій.

Послідовність вищеназваних операцій наведена на рис.2.2.

Матриці розширення ключів, які мають розмірності рівні матрицям станів, а число яких дорівнює кількості циклів обчислюються тільки один раз і використовуються для шифрування-дешифрування всіх блоків інформації сполучення. Відповідно, час формування розширення ключа не є критичним і практично не позначається на швидкості шифрування. Тому в більшості апаратних реалізацій Rijndael [16] розширення ключів виконується на центральному процесорі.

Оборотність криптографічного перетворення алгоритму Rijendael досягається виконанням інверсних базових операцій в зворотному порядку. При дешифруванні, як самі базові операції, так і їх порядок виявляються відмінними, що потребує виділення додаткових операційних коштів для шифрування і дешифрування, а також відмінної топології зв'язків між ними.

Одна з найбільших переваг стандартизованих шифроблоків та хеш-перетворювачів полягає в тому, що їх криптографічні властивості не підлягають сумніву, бо були багаторазово перевірені спеціальними тестуваннями та в процесі застовування на практиці.

Інша важлива перевага використання для ідентифікації стандартизованих шифроблоків заключається у тому, що для них до теперішнього часу створено і серійно випускаються спеціалізовані апаратні засоби. Так для апаратної реалізації широко відомого алгоритма DES випускається понад 20-ти типів спеціалізованих НВІС.

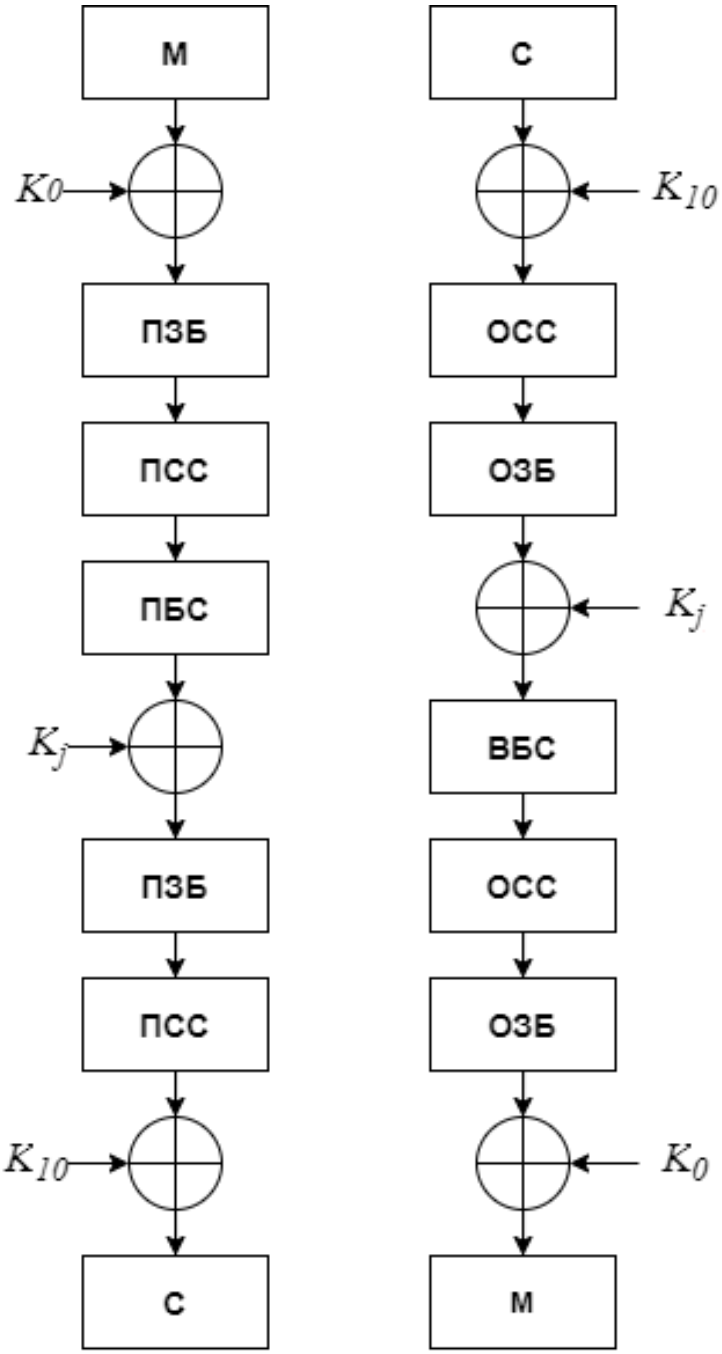


Рис. 2.2. Структурна схема виконання операцій в алгоритмі Rijndael при прямому і зворотньому перетворенні

З моменту прийняття алгоритму Rijendael в якості стандарту у США симетричного шифрування було запропоновано ряд архітектурних рішень для його апаратної реалізації та приведені оцінки їх ефективності з урахуванням використання елементів бібліотек ASIS і FPGA. Проте всі вони відтворюють лише базові операції алгоритму і тому є надто складними, щоб реалізувати Rijndael на лише одному програмованому кристалі. Основні складності апаратної реалізації пов'язані з тим, що процеси шифрування і дешифрування реалізовані в цьому алгоритмі ньому в зворотній послідовності і використовують відмінні базові перетворення. Внаслідок цього необхідність використання роздільних апаратних засобів для шифрування і дешифрування. Через це бажаними для ефективної реалізації Rijndael є рішення, в яких допустиме використання тих самих апаратних компонент як для шифрування, так і для дешифрування. Більш того, Rijndael здійснює одночасну обробку всього блоку даних, в той як більшість симетричних обробляють лише половину розрядів блоку даних в один момент часу.

Доволі поширеним на сьогоднішній день криптопроцесором є IBM 1752. Він дозволяє реалізацію алгоритмів DES, RSA, El-Gamal і DSA, причому останні 2 тільки у форматі 1024 розряду, що є недостатнім для сучасних вимог. Також цей варіант криптопроцесора не здатен реалізовувати деякі алгоритми сімейства AES.

Процесор здатний виконувати завдання модулярної арифметики, а також генерацію та операції з випадковими числами, інші побічні завдання базуються на центральному процесорі пристрою. 6500 можна сміливо назвати дуже достойним агрегатом, якщо помітити розрядність робочих регістрів

(банк регістрів даних) і АЛП (арифметико-логічного пристрою) - 1024 біти та їх організацію - кожному регістру за роботою відповідає автоматично оновлюваний запасний регістр, який по розміру рівний слову, записаному в даний регістр. Крім того, в процесорі реалізовані шість шин такої ж розрядності з тактовою частотою 100 MHz. Також нетиповим є і процес генерації випадкових чисел - замість звичних генераторів шумової напруги і подальшого аналого-цифрового перетворення застосовано неочевидне, але дуже елегантне з інженерної точки зору рішення - формування випадкового числа на основі порівняння фаз сигналів стабільного тактового генератора процесора і нестабільного автономного генератора, що входить до складу генератора випадкових чисел. При тактовій частоті 100 MHz швидкість формування випадкового потоку бітів досягає 3 Mbps.

Список команд процесора максимально примітивний, кожна команда може задіяти чотири довільних регістри даних - регістр результату (РР), два регістри операндів (РО) і регістр величини модуля (РВМ), за яким виконуються операції. Однак самі команди ніяк не настільки нетривіальні, і ні в одному поширеному мікропроцесорі не реалізовані - по-перше, всі вони мають арифметичну структуру (базові: додавання, віднімання, множення та ділення, а також зведення до степеню), по-друге - виконуються по модулю, який знаходиться в РВМ, а також виконуються відносно швидко (за 4 такти, тобто при тактова частота виходить на рівні 100 MHz). Також на кристалі можна вмістити інтерфейс ITA-шини з мікроконтролером, що дозволить записувати і швидко виконати типову 32-командну програму. Таким чином, результати використання 6500, особливо в Internet-орієнтованих продуктах, дуже достойні: так, наприклад, ми маємо підтримку до 300 шифрованих звернень Internet Key Encryption до сервера за секунду, а також виконання понад 80 операцій аутентифікації з 1024-бітовими ключами в 1 секунду. Незважаючи на порівняно високу вартість 6500, його застосування повністю

доцільно в спеціалізованих системах, а також в великих серверах для криптопідтримки потреб Internet-комерції.

Також зарубіжними компаніями розроблені і запущені в масове виробництво модулі криптографічних співпроцесорів на базі цього сімейства. Заслуговує окремої уваги модуль DX1845, який дозволяє вигідно та швидко реалізувати операції шифрування і ідентифікації. Однак він дорожчий - в середньому 450 $ і не перевірений на наявність у ньому технології "чорного ходу". Тому доводиться застосувати компромісний підхід - самим розробити та спроектувати систему на базі зарубіжних мікросхем.

Структурна схема всіх криптографічних перетворень, які реалізуються в запропонованому методі ідентифікації реалізованого на основі концепції нульових знань показана на рис.2.3.

Запропонований вище метод регламентує процедуру ініціалізації та сеансової ідентифікації на кожній ітерації звернення користувача до системи.

1. Користувачеві пересилається ідентифікуючий код U.
2. Визначається кількість n сеансів ідентифікації(з боку користувача).
3. Випадково генерується код p_n - сеансовий пароль на останньому, n-тому циклі ідентифікації. Індексу i присвоюється значення n-1 : $i = n$-1.
4. Користувачем обчислюється $q_i = F(i|U, p_i)$, де $i|U$ - конкатенація номера сеансу ідентифікації та його ідентифікуючого коду.
5. Обчислюється $p_{i-1} = F(p_i, q_i)$ (з боку користувача).
6. Здійснюється декрементація індексу i: $i = i$ -1. Якщо $i > 0$, виконується повторно п.4.
7. Код p_0 пересилається системі користувачем.

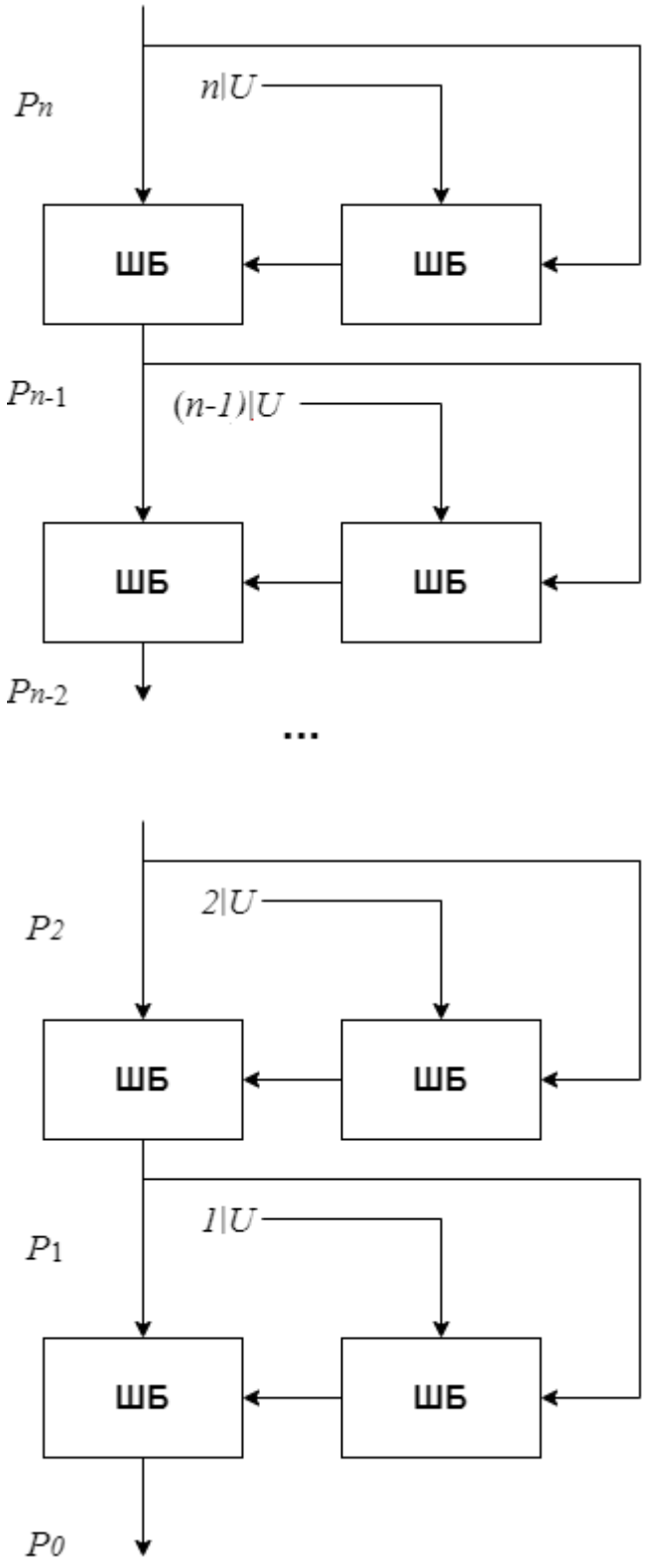


Рис. 2.3. Структурна схема криптографічних перетворень з використанням шифроблоків, що реалізуються у запропонованому методі.

Коди сеансових паролей p_0, p_1,…,p_n , обчислені за допомогою даного методу, зберігаються в пам'яті машини користувача.

Процедура j–то сеансу ідентифікації абонента полягає у виконанні наступної послідовності дій.

1. Системі надсилається j-й сеансовий пароль p_j .
2. Система обчислює $d = F(j|U, p_j)$.
3. Система обчислює $\eta=F(p_j, d)$. Якщо $\eta = p_{j-1}$, то ідентифікація віддаленого абонента проведена успішно, користувачу надається доступ до ресурсу системи.

Степінь захищеності, яка забезпечується вищезгаданим методом, може бути оцінена наступним чином. Зрозуміло, що пароль p_j пов'язаний з попереднім паролем p_{j-1} через функціональне рівняння виду: : $p_{j-1} = F(p_j, d) = F(p_j, F(\alpha, p_j))$, $\alpha = j|U$. А для зловмисника зі сторони злам системи зводиться до отримання наступного j–го паролю p_j на базуючись на попередніх паролях p_{j-1}, p_{j-2},…,p_0. Підбір ж двох компонентів шифроблоку унеможливлює вирішення такої проблеми. Об'єм перебору в такому методі при експлуатації шифроблоку Rijndael-256 з ключем довжиною 256 біт складає $2^{256+L(U)}$, де $L(U)$- розрядність U.

Система також не в змозі практично генерувати наступний сеансовий пароль p_j при заданні їй коду U та паролів p_{j-1}, p_{j-2},…,p_0. Через те, що системі відомо значення $\alpha = j|U$, то для генерації пароля вона повинна знати й p_j, для якого справедливо: $p_{j-1} = F(p_j, F(\alpha, p_j))$. Об'єм ресурсів для виконання такої задачі є еквівалентним злому шифроблоку, тобто не є практично доцільним.

Найбільш важлива перевага запропонованого методу реалізації ідентифікації на основі "нульових знань" порівнюючи з існуючими на даний момент концепціями(схеми Fiat-Shamir, Schnorr) полягає в тому, що він забезпечує значне зменшення обчислювальної складності процедури

ідентифікації. Основна більшість існуючих способів ідентифікації віддалених користувачів, що реалізують концепцію “нульових знань” беруть за основу незворотні перетворення теорії чисел, які здійснюють модулярне експоненціювання над числами великої розрядності – 2048 або 4096. За оцінкою [17] експоненціювання для розрядності 2048 на 3-4 порядки програє в складності обчислень будь-якому виду шифрування з використанням стандартизованого шифроблоку.

Так як розроблений у даній роботі метод шифрування пристосований до використання у ньому стандартизованих шифроблоків, це дає йому широкі можливості для практичної реалізації його з використанням криптопроцесорів, що виробляються промисловістю провідних країн світу і на апаратному рівні будуть реалізовувати обчислення.

На відміну від відомих методів, для одного сеансу ідентифікації використовується лише одне пересилання пароля через потенційно відкриті канали передачі даних. Отже, в результаті проведених досліджень розроблено метод, який відповідає концепції “нульових знань” для ідентифікації віддалених абонентів і пристосований до використання у ньому стандартизованих шифроблоків. Використання добре досліджених та протестованих, випробованих часом шифроблоків забезпечує високий рівень надійності захисту від спроб порушення ідентифікації.

Середня обчислювальна складність операцій в сучасних шифроблоках в середньому три-чотири порядки менша за складність операцій модулярного експоненціювання такої ж розрядності. Це означає, що запропонований метод дозволяє на порядки прискорити процедуру ідентифікування віддалених юзерів системи, що може бути критично важливо для роботи систем з сотнями тисяч абонентів.

2.3. Строга ідентифікація користувачів на основі комбінованого використання шифроблоків та хеш-перетворень

З ціллю досягнення запланованої мети після відповідних досліджень був обраний метод строгої ідентифікації віддалених користувачів, на основі концепції “нульових знань” з виправданим використанням стандартизованих шифроблоків та хеш-перетворень.

Схематично в режимі шифрування звичайний шифроблок можна розглядати як функціональний трансформатор φ, на вхід якому потрапляють *m*-розрядний блок даних D та *k*–розрядний ключ *K*, а на виході отримується *m*–розрядний блок C зашифрованих даних: $C = \varphi(D,K)$. В режимі дешифрування ж шифроблок може також виконувати обернене перетворення φ^{-1}: на його входи подаються *m*-розрядний блок *C* зашифрованих даних разом з *k*–розрядним ключем K, а на виході буде отриманий *m*-розрядний блок даних $D = \varphi^{-1}(C,K)$. У випадку типового шифроблоку Rijndael може бути виконано шифрування блоків розрядністю 128, 192 чи 256 бітів. В запропонованому способі розглядалося шифрування блоків розрядністю 256.

Стандартизований хеш-перетворювач (H) – сертифікована відповідними органами схема перетворення блоку даних довільної довжини в незворотну закодовану хеш-сигнатуру довжини *h*. Найважливішою властивістю хеш-перетворювачів є їх незворотність – майже повна практична неможливість знаходження інформаційного блоку, хеш-сигнатура якого дорівнює заданій. Найпоширенішими є хеш-перетворювачі SHA-1 та RIPEMD-160, які формують на виході хеш-сигнатуру довжиною 20 байт (*h*=160).

Для ефективної реалізації концепції нульових знань стосовно ідентифікації віддалених абонентів пропонується використання шифроблоку, довжина *m* блоку даних якого набагато більша за розрядність хеш-сигнатури

(*h)*. Зокрема, для практичного застосування методу пропонується використання шифроблоку з розрядністю блоку даних 256 біт (*m*=256) та хеш-перетворювачів з розрядністю хеш-сигнатури 160 (*h*=160). Різниця розрядностей цих блоків далі буде позначатися як *d=m-h*.

На рис.2.4 представлено процеси, які запускаються при реєстрації користувача та при кожному і-тому сеансі його ідентифікації.

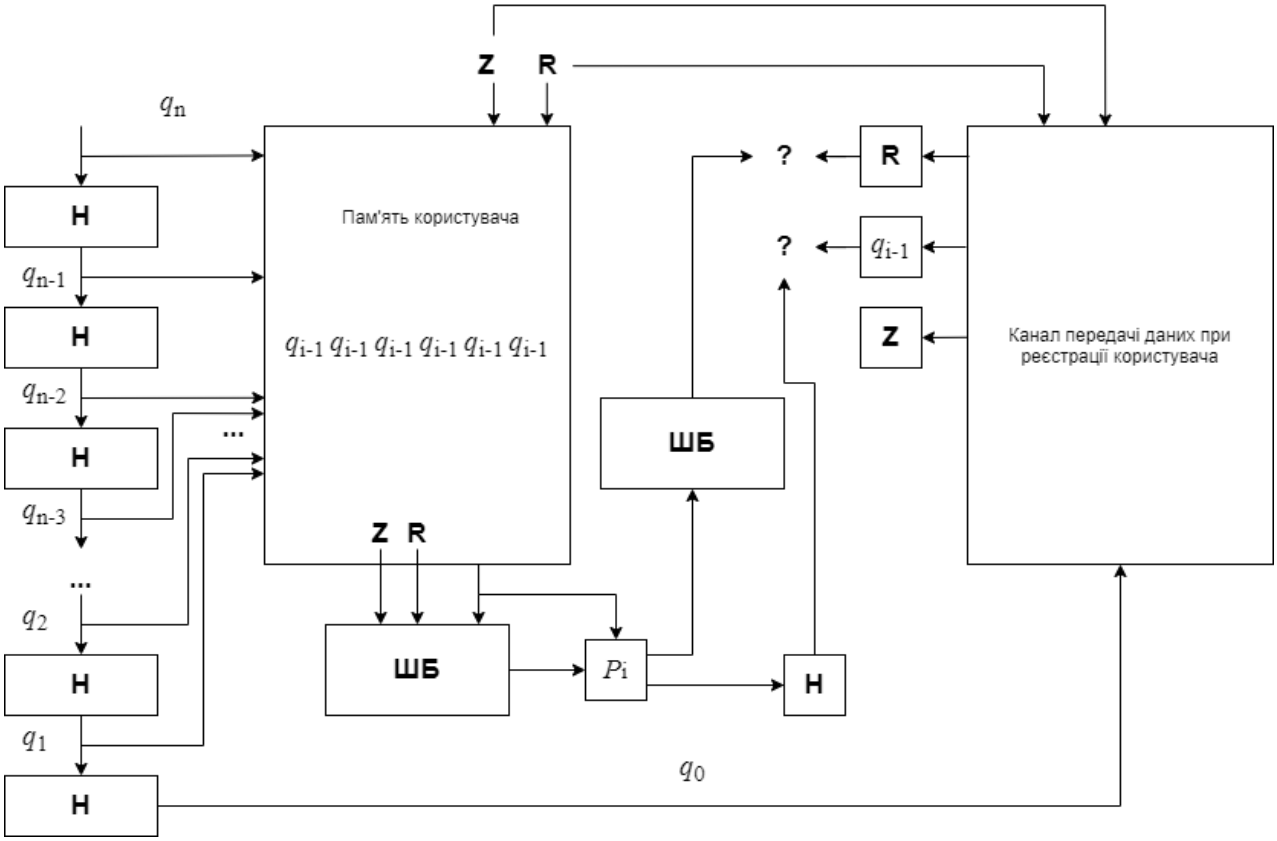


Рис 2.4. Структурна організація формування сеансових паролів, реєстрації та і-го сеансу ідентифікації користувача системою з використанням шифроблоків та хеш-перетворювачів.

Головна перевага використання стандартизованих шифроблоків та хеш-перетворювачів полягає в тому, що їх криптостійкість прискіпливо перевірена спеціальними тестуваннями та в процесі практичного їх застосування.

Запропонований метод визначає процедури ініціалізації та сеансової ідентифікації при кожному зверненні абоненту до системи.

Ініціалізація проходить в 2 фази:

1) На стороні абонента випадково формується d–розрядний код z, а також m–розрядний код R на стороні користувача. Отримані коди z та R в заносяться до системи.

2) На стороні абонента обирається h-розрядний код q_n, а потім, по ланцюжку, з використанням стандартизованого хеш-перетворення H, обчислюється n-1 h-розрядних кодів $q_{n-1}, q_{n-2},\ldots,q_2, q_1$ так, що наступний код є хеш-сигнатуру попереднього, в нашому випадку, $q_i = H(q_{i+1})$. Сформовані таким чином коди зберігаються лише у абонента.

Процедура n-ної ідентифікації абонента виглядає наступним чином:

1)Абонент формує m-розрядний код сеансового ключа K_i як конкатенацію n–розрядного коду q_i та d-розрядного коду z: $K_i = q_i|z$.

2)Абонентом формується m-розрядний код U_i який отримується зворотнім шифруванням коду R ключем K_i : $U_i = \varphi^{-1}(R,K_i)$. Пара кодів q_i та U_i утворюють (h+m)-бітовий i-тий сеансовий пароль: $P_i = < q_i, U_i >$.

3) В систему передається черговий сеансовий пароль P_i.

4) Сеансовий пароль отриманий в п.3 розкладається на складові q_i та U_i.

5) Перевіряється вірність переданого коду q_i перевіркою того, що результат обчислення хеш-перетворення q_i рівний коду q_{i-1} останнього сеансу ідентифікації, інакшими словами задовольняється умова: $q_{i-1} = H(q_i)$. Якщо ця рівність не виконується, то сеанс ідентифікації переривається.

6) Система за допомогою операції конкатенації коду q_i та секретного коду z формує сеансовий ключ K_i : $K_i = q_i|z$.

7) За допомогою отриманого сеансового ключа виконується шифрування m-розрядного коду U_i : $Y= \varphi(U_i, K_i)$.

8) Якщо виведений в процесі шифрування код Y дорівнює коду доступу R абонента, тобто $Y=R$, то ідентифікація абонента вважається успішною.

Таким чином, розроблений метод забезпечує ідентифікацію віддалених абонентів з використанням паролей, що змінюються при кожному сеансі. При цьому система не здатна генерувати пароль, а може лише перевіряти його коректність. Перевірка виконується не через порівняння, а шляхом використанням незворотних перетворень. Відповідно, запропонований метод реалізує ідентифікацію, що цілком вкладається в рамки теоретичної концепції “нульових знань”.

Запропоновані вище ідеї можуть бути доволі просто модифіковані для реалізації автентифікації віддалених абонентів для роботи розподілених систем.

2.4. Оцінка ефективності реалізації строгої ідентифікації з використанням стандартизованих криптографічних примітивів

Задля виконання порівняльної оцінки ефективності вищезгаданих способів ідентифікації віддалених користувачів з використанням стандартизованих криптографічних механізмів та існуючих схем строгої ідентифікації були взяті за основу наступні критерії:

- ступінь захищеності від атак зловмисника з ціллю отримати доступ до інформаційних ресурсів як ззовні, так і зі сторони осіб, які мають доступ до внутрішньої пам’яті системи;

- обчислювальна складність процедури ідентифікації;

- можливості апаратної реалізації ідентифікації системою, яка дозволяє збільшити рівень захищеності та прискорити процедуру ідентифікації;

-використання потенційно відкритих каналів обміну даними: ступінь використання таких маршрутів при ідентифікації підвищує шанси на незаконний доступ до ресурсів системи.

Поданий вище метод втілює в собі концепцію строгої ідентифікації віддалених абонентів з імплементуванням також стандартизованих криптографічних примітивів.

В запропонованому методі при кожному новому зверненні до системи користувач використовує новий варіант паролю.

Для кожного сеансу ідентифікації використовується пароль, який складається з двох частин: q_i та U_i. У випадку перехоплення послідовності паролей P_1, P_2,…,P_i стороннім користувачем, йому потрібно буде відтворити P_{i+1} для вже наступного, (i+1)-го сеансу ідентифікації. Цей пароль складається також з двох частин: h-розрядного коду q_{i+1} та m-розрядного коду U_{i+1}. Код q_{i+1} можна отримати з коду q_i через формулу: $q_i = H(q_{i+1})$, отже, для отримання q_{i+1} зловмиснику потрібно зімітувати зворотнє хеш-перетворення. Для стандартизованих хеш-перетворень доступним способом виконання такого зворотного перетворення є банальний перебір. Це означає, що для вирішення задачі реконструкції q_{i+1} по значенню попереднього паролю q_i методом перебору, в середньому, потрібно виконати 2^{h-1} підборів потрібного хеш-перетворення. Для прикладу, стандартизований хеш-алгоритму SHA-1 потребує для злому, в середньому, 2^{159} варіантів перебору.

Для встановлення коду U_{i+1} зловмиснику зі сторони буде потрібно підібрати такий код, щоб виконувалася умова: $U_{i+1} = \varphi^{-1}(R,K_{i+1})$, при невідомих значеннях m-розрядних кодів R та K_{i+1}. Технологічно, підбір доцільно виконувати по відомому значенню U_i, з використанням рівняння $\varphi(U_i, K_i) = (U_{i+1}, K_{i+1})$. Якщо до цього сторонній зловмисник теоретично зможе реконструювати код q_{i+1}, то йому потрібно підібрати m-розрядний код U_{i+1} для кожного з можливих значень d-розрядного коду z – закритої компоненти ключа K. Таким чином, об’єм перебору становить $2^{m+d} = 2^{2 \cdot m-h}$. При використанні в якості шифроблоку алгоритму Rijndael, а в якості хеш-перетворення SHA-1, m=256 і h=160, тобто обєм перебору для реконструкції

коду U_{i+1} може буди оцінений як 2^{352} реалізацій процедури шифрування за алгоритмом Rijndael, що виходить за рамки сучасних технічних можливостей.

Для спроби зімітувати абонента самою системою або спроби незаконного доступу при становленні відомими злочинцю кодів R та z, які сховані в системі, треба встановити значення коду q_{i+1} по попередньому коду q_i. Але, зрозуміло, що імплементація стандартизованого хеш-перетворення робить єдиним валідним в цій ситуації метод підбору. В такій ситуації, в середньому, зловмисник змушений буде перепробувати 2^{h-1} варіантів підрахунку хеш-перетворення, що для випадку SHA-1 дасть в результаті 2^{159} спроб і не впишеться в економічно доцільні рамки.

Також метод може бути розвинений: ступінь захищеності методу строгої ідентифікації може бути підвищена за рахунок банального збільшення розрядності у шифроблоку, а також хеш-сигнатур в рамках стратегії “нульових знань”.

Найбільша ж перевага даного методу реалізованого в парадигмі “нульових знань” в порівнянні з існуючими на даний момент методами (схеми Fiat-Shamir, Schnorr) полягає в тому, що він гарантує зменшення в рази обчислювальної складності процесу ідентифікації користувачів. Більшість способів ідентифікації, які втілюють в собі ідею “нульових знань”, використовують незворотні перетворення в області теорії чисел, зокрема операцію модулярного експоненціювання над числами величезної розрядності – 2048 чи 4096, що погано впливає на швидкодію. За оцінкою, середня обчислювальна складність експоненціювання чисел розрядності 2048 на 3-4 порядки перевищує обчислювальну складність шифрування з використанням стандартизованого шифроблоку. Це означає, що обчислювальна складність реалізації строгої ідентифікації на більш ніж три порядки менша в порівнянні з відомими втілюючими концепцію “нульових знань” методами ідентифікування. На практиці це означає, що

запропонований метод дозволяє в багато разів скоротити час ідентифікації абонента, що критично важливо для роботи систем з великим навантаженням.

Щоб оцінити прискорення строгої ідентифікації, яке досягається розробленими методами було змодульовано два варіанти реалізації: апаратна та програмна.

Під час програмній реалізації модулярного експоненціювання n-разрядних чисел на деякому r–розрядному процесорі n-розрядні коди розділяються на k фрагментів ($k=n/r$). На сьогоднішній день найбільш ефективним алгоритмом модулярного експоненціювання вважається революційний алгоритм Монтгомері, який дозволяє не використовувати ділення, а обійтись звичайними зсувами. Але навіть у ньому число операцій множення рівне $3 \cdot n \cdot r(r+1)$, а число операцій суми - $4 \cdot n \cdot r(r+1)$. Для сучасних процесорів операція додавання чи зсуву потребують лише 1 машинний такт, в той час як операція множення – 10 тактів. Таким чином, загальна кількість T_M тактів для експоненціювання по методу Монтгомері рівна: $T_M = 34 \cdot n \cdot k \cdot (k-1)$.

Вираховано в ході досліджень, що для реалізації типового шифроблоку AES-256 виконано 1400 логічних операцій XOR, більш ніж 300 операцій звернення пам’яті у табличній формі, 40 операцій зсуву, що в сумі становить приблизно 1700 процесорних команд.

На основі наведеної вище інформації проведені розрахунки кількості машинних тактів для реалізації AES-256 та модулярного експоненціювання. Результати дослідження приведено в таблиці 2.1. [13].

Табл. 2.1. Порівняльний аналіз часу програмної реалізації шифроблока AES-256 та операції модулярного експоненціювання

Розрядність чисел	Розрядність процесора	Кількість машинних тактів для реалізації модулярного експоненціювання	Співвідношення кількості машинних тактів для реалізації модулярного експоненціювання та шифроблоку
1024	32	36760000	5340
	64	9460000	1370
2048	32	289660000	21000
	64	73530000	5340

Значимою на практиці перевагою запропонованого методу в порівнянні з відомими є також те, що шифроблоки та хеш-перетворення побудовані на логічних операціях, які просто та ефективно можуть бути реалізовані апаратними засобами. Що не менш важливо, апаратна реалізація операцій модулярного експоненціювання над числами великої розрядності, що лежать в основі відомих методів строгої ідентифікації, суттєво більш складна і менш ефективна.

Враховуючи факт, що метод заточений на використання стандартизованих шифроблоків та хеш-перетворень, для його реалізації ідеально можуть підійти криптопроцесорні мікросхеми, які випускаються промисловістю найрозвиненіших країн світу. Вони можуть легко на апаратному рівні реалізувати як хеш-перетворення так і стандартизовані шифроблоки.

В сучасні часи стрімкого розвитку інтегральної технології невпинно збільшується питома вага виконання обчислень, пов'язаних з криптографічним захистом інформації спеціалізованими апаратними засобами – криптопроцесорами або криптоакселераторами. Останні найчастіше використовуються не окремо, а у вигляді конструкції, вбудованої в процесор. Криптопроцесори частіше ж експлуатують як спеціалізовані плати сопроцесорів, які підключаються до шини PCI.

Якщо в першу декаду та початок другої декади третього тисячоліття криптопроцесори входили до складу апаратного комплексу серверів, то нинішні сопроцесори, які здатні підтримувати криптообчислення, входять до складу всіх ноутбуків та навіть термінальних мікроконтролерів. На даний момент, всі мікроконтролери AVR родини XMega фірми Atmel оснащені криптоакселераторами шифроблоків DES і AES. Апаратна реалізація шифроблоку AES також передбачена в мікроконтролерах родини MSP432P4xx фірми Texas Instruments та мікроконтролерах родини SiM3U1xx/SiM3C1xx фірми Silicon Laboratories.

Використання криптоакселераторів для підняття швидкості обчислень дасть потенційне збільшення цієї швидкості у 10–20 разів для 8/16-бітових МК та до 10 порядків – для 32-бітових МК, в порівнянні з програмно реалізованими алгоритмами шифрування. Через це, ефективність криптоакселераторів для більшості структур є цілком достатньою на практиці, а у випадку 32-бітових ядерних систем отримана продуктивність на рівні сотень Мбайт/сек є навіть нераціонально через надефективність, оскільки ці мікроконтролери, як правило, не використовують такі потужні інтерфейси аналогового чи цифрового вводу-виводу даних, щоб забезпечити безперервний потік даних. Найбільш продуктивним для шифрування алгоритмом AES є мікроконтролер SAME70 фірми Atmel з тактовою

частотою 300 МГц, він може забезпечити швидкість обробки до 400 Мбайт/сек.

Проведений аналіз ринку криптопроцесорів та криптоакселераторів показав, що всі вони спроможні апаратно реалізувати шифроблок типу DES або AES, приблизно на 70% можуть бути реалізовані стандартизовані перетворення типу SHA , але лише 40% можуть апаратно підтримати алгоритмів з відкритим ключем RSA або DSA.

Зважаючи на проведений аналіз, можна зробити висновок, що:

- При аналізі ефективності запропонованих методів строгої ідентифікації для максимальної акселерації її обчислювальної реалізації прийнято рішення щодо орієнтування на реалізацію апаратно.

- З висоти проведеного дослідження можна стверджувати, що використання запропонованих методів для імплементації строгої ідентифікації за допомогою стандартизованих шифроблоків та хеш-перетворювачів дозволяє збільшити сферу можливостей апаратної швидкісної реалізації.

- Оцінка рівня прискорення обчислювальної реалізації строгої ідентифікації, що досягається запропонованими методами у порівнянні з відомими методами може бути реалізована через порівняння часових характеристик стандартизованих шифроблоків AES та модулярного експоненціювання (алгоритму RSA) за існуючими типами криптопроцесорів.

Зокрема, як згадано вище, на ринку присутній широкий вибір крипто-процесорів, якими можна реалізувати шифроблок типу AES, стандартизоване перетворення SHA або алгоритм RSA. Характеристики типового для ринку криптопроцесора 7955 наведені нижче в таблиці 2.2:

Табл 2.2. Характеристики криптопроцесора 7955

Криптографічний алгоритм	Швидкість шифрування Мбіт/с	
AES - 256	550	
SHA	325	
RSA	Час шифруваня мс	
	Розрядність 2048	Розрядність 1024
	82.75	11.90
	Оцінка прискорення обчислювальної реалізації строгої ідентифікації	
Співвідношення часу реалізації RSA та AES	21450	6380
Співвідношення часу реалізації RSA та SHA	13130	3775

На основі даних з таблиці 2.2 можемо помітити, що швидкість шифрування 1024 бітів алгоритмом AES – 256 криптопроцесором 7955 становить 0.00186 мс, а 2048 бітів – 0.00372 мс. З цього можна зробити висновок, що маючи розрядність 1024 шифроблок AES виконується у більш ніж 6000 разів швидше за модулярне експоненціювання, при розрядності 2048 – більш ніж в 20000 разів ефективніше.

Якщо порівнювати з іншими відомими методами, користувач відправляє лише один пароль за сеанс для ідентифікації через канали передачі даних, які можуть бути під контролем зловмисника.

Висновки по розділу 2.

1**.** Використання незворотних перетворень на основі булевої алгебри для строгої ідентифікації ефективно вирішує проблему прискорення процесу обчислювальної реалізації ідентифікації користувачів. Розглянені принципово можливі варіанти використання незворотних булевих перетворень, а саме: попередня генерація користувачем сеансових паролів, які перевіряються в зворотному порядку; використання незворотного, спеціально підібраного хеш-перетворення з програмованими колізіями, які будуть виступати як сеансові паролі.

2.Теоретично обгрунтована можливість застосування незворотних булевих перетворень в рамках "нульових знань" для строгої ідентифікації користувачів. В рамках реалізації даної можливості розроблено новий метод криптографічно строгої ідентифікації користувачів з застосуванням в якості сеансових паролей кодів, що утворюються в результаті послідовних хеш-перетворень з використанням стандартизованих хеш-алгоритмів, який дозволяє прискорити процес ідентифікації за рахунок того, що обчислювальна реалізація цих алгоритмів, основаних на незворотних булевих перетвореннях, виконується швидше в порівнянні з операціями модулярного експоненціювання, які використовуються в багатьох відомих методах строгої ідентифікації. Використання хеш-перетворень, надійність яких доведена, дозволяє знизити ризики доступу сторонніх осіб до ресурсів розподілених систем до мінімуму та використовувати для цього криптопроцесори, які мають вбудовані апаратні засоби для реалізації стандартизованих хеш-перетворень.

3. Досліджено, обґрунтовано та розроблено метод строгої ідентифікації віддалених користувачів на основі застосування стандартизованих шифроблоків, за допомогою яких користувачем формується послідовність

сеансових паролів, перевірка яких виконується системою в зворотному порядку, що дозволяє зменшити час ідентифікації користувача системою за рахунок того, що обчислювальна складність шифроблоків, основаних на незворотних булевих перетвореннях, на порядки менша складності операцій модулярного експоненціювання чисел великої розрядності.

4. Проведений аналіз ефективності поданих методів строгої ідентифікації користувачів з використанням стандартних криптографічних примітивів: шифроблоків та хеш-перетворень. Показано, що при реалізації запропонованих і розроблених раніше методів строгої ідентифікації використовуючи криптопроцесори досягається великий виграш в прискоренні реалізації процедури користувацької ідентифікації, що може істотно збільшити ефективність в сучасних умовах стрімкого зростання навантаження на віддалені розподілені системи.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМИ ПРИСКОРЕНОЇ РЕАЛІЗАЦІЇ ХЕШ-ПЕРЕТВОРЕНЬ

Програма реалізує алгоритм криптографічної хеш-функції ГОСТ Р.34.10-94. Для будь-якого по об'єму вхідного повідомлення генерується 160-розрядне хеш-значення, яке будемо називати дайджестом повідомлення. Обробка даних виконується блоками по 512-біт [16].

Програма виконана у середовищі Visual Studio 2019 за допомогою мови програмування C++.

Числові константи, які використовуються на всіх кроках алгоритму в програмі виражені за через директиву мови C++ #define наступним чином(рис 3.1):

Макрос #define F1(X,Y,Z) виконує сумування за модулем двох аргументів: X,Y та Z.

Макрос #define F2(X,Y,Z) виконує операцію суми за модулем двох значень Y та Z. Після цього виконується операція логічне «І» для отриманого результату та нашого значення X. Завершальною операцією, втіленою у даному макросі, є сума за модулем два отриманого результату та перемінної Z

```
#define K1_0  0x00000000
#define K1_1  0x5A827999
#define K1_2  0x6ED9EBA1
#define K1_3  0x8F1BBCDC
#define K1_4  0xA953FD4E
#define K2_0  0x50A28BE6
#define K2_1  0x5C4DD124
#define K2_2  0x6D703EF3
#define K2_3  0x7A6D76E9
#define K2_4  0x00000000
```

Рисунок 3.1 - числові константи

Ініціалізація слів дайджесту представлена так:

```
#define INIT_DATA_H0    0x67452301
#define INIT_DATA_H1    0xEFCDAB89
#define INIT_DATA_H2    0x98BADCFE
#define INIT_DATA_H3    0x10325476
#define INIT_DATA_H4    0xC3D2E1F0
```

Рисунок 3.2 - Ініціалізація слів дайджесту

Задання розміру блока:

```
#define RMD160_BLOCK_SIZE  16 // 16 32-розрядних слів у блоці
```

Макрос #define F1(X,Y,Z) виконує сумування за модулем двох аргументів: X,Y та Z.

Макрос #define F2(X,Y,Z) виконує операцію суми за модулем двох значень Y та Z. Після цього виконується операція логічне «І» для

отриманого результату та нашого значення X. Завершальною операцією, втіленною у даному макросі, є сума за модулем два отриманого результату та перемінної Z.

Макрос #define F3(X,Y,Z) виконує інвертування значень розрядів Y та після цього виконує операцію логічного «АБО» для змінних Y та X. Після цього виконується сума за модулем два для значення отриманого результату зі змінною Z.

Макрос #define F4(X,Y,Z) виконує суму за модулем двох значень Y та X після цього виконує операцію логічного «І» отриманого результату та значення Z. Завершальною операцією є виконання суми за модулем два отриманого результату із змінною Y.

Макрос #define F5(X,Y,Z) виконує інвертування значень розрядів Z та після цього виконує операцію логічного «АБО» змінних Z та Y. Після цього виконується сума за модулем два значення отриманого результату із змінною X.

Макрос TRANSFORM(F,A,B,C,D,E,X,S,N) викликає задану в параметрах F функцію, яка може виявитись однією з п’яти описаних вище, на її вхід подаються значення змінних B,C та D. Після цього виконується арифметичне додавання отриманого результату роботи функції із значеннями змінних A,N та pD[X], отримана сума зміщується циклічно вліво на S розрядів.

Функція void RMD160U(unsigned char * pData, unsigned int pLength, RMD160CONTEXT c) виконує основну частину алгоритму, котра складається із десяти ітерацій обробки даних по п’ять ітерацій кожна. Всі 10 ітерацій мають подібну структуру, але в кожному застосовується своя власна функція. Їх позначення f_1, f_2, f_3, f_4 та f_5, відповідні цим функціям макроси F1, F2, F3, F4 та F5 були описані раніше. На наступній стадії обробки ці функції використовуються в оберненому порядку. В програмі

замість показаних на малюнку значень ключів ми можемо побачити задані директивами значення: K1_0=$К_1$, K1_1=$К_2$, K1_2=$К_3$, K1_3=$К_4$, K1_4=$К_5$, K2_0=$К'_1$, K2_1=$К'_2$, K2_2=$К'_3$, K2_3=$К'_4$, K2_4=$К'_5$,

Макроси ROUND1_1, ROUND1_2, ROUND1_3, ROUND1_4, ROUND1_5, ROUND2_1, ROUND2_2, ROUND2_3, ROUND2_4, ROUND2_5 реалізують раунди алгоритму. Логіка роботи одного 512-ти бітового блоку повідомлення полягає в тому, що кожен раунд складається із послідовності 16-ти кроків котрі реалізуються за допомогою описаного вище макросу TRANSFORM(F,A,B,C,D,E,X,S,N). Порядок, в котрому обробляються п'ять слів (A, B, C, D, E), в результаті виконує зсув вправо на одне слово для кожного кроку на рівні слів. Перетворення на кожній ітерації використовують різні функції та різні ключі, так у макросі ROUND1_1 використовується функція F1 та ключ K1_0, у макросі ROUND1_2 використовується функція F2 та ключ K1_1, у макросі ROUND1_3 використовується функція F3 та ключ K1_2, у макросі ROUND1_4 використовується функція F4 та ключ K1_3, у макросі ROUND1_5 використовується функція F5 та ключ K1_4, у макросі ROUND2_1 використовується функція F1 та ключ K2_0, у макросі ROUND2_2 використовується функція F2 та ключ K2_1, у макросі ROUND2_3 використовується функція F3 та ключ K2_2, у макросі ROUND2_4 використовується функція F4 та ключ K2_3, у макросі ROUND2_5 використовується функція F5 та ключ K2_4,

Вибір 32-бітних слів з повідомлення здійснюється в наступному порядку

- r(j) = j при $(0 \le j \le 15)$
- r(16..31) = 6; 3; 12; 1; 9; 5; 14; 2; 11; 0; 8; 4; 2; 13; 10; 7
- r(32..47) = 4; 11; 15; 5; 10; 15; 9; 1; 3; 8; 1; 7; 14; 12; 6; 13
- r(48..63) = 2; 10; 12; 11; 1; 9; 13; 5; 14; 4; 8; 14; 13; 6; 7; 3

- r(64..79) = 5; 1; 6; 10; 8; 13; 5; 11; 15; 2; 4; 9; 12; 7; 13; 12
- r'(0..15) = 4; 13; 6; 1; 8; 1; 10; 3; 12; 5; 13; 7; 2; 11; 2; 11
- r'(16..31) = 5; 10; 4; 8; 1; 12; 6; 11; 13; 14; 7; 13; 5; 10; 2; 5
- r'(32..47) = 14; 6; 2; 4; 6; 13; 5; 8; 10; 7; 11; 1; 9; 2; 3; 12
- r'(48..63) = 9; 7; 5; 3; 4; 12; 14; 1; 6; 11; 1; 14; 10; 5; 9; 13
- r'(64..79) = 11; 13; 8; 6; 3; 5; 11; 6; 5; 3; 14; 13; 2; 4; 8; 11

Вирахування кількості розрядів для циклічного побітового зсуву вліво виконувалась згідно до наступного порядку [21].

- s(0..15) = 10; 13; 14; 11; 4; 7; 6; 8; 10; 12; 13; 14; 5; 6; 8; 7
- s(16..31) = 6; 5; 7; 12; 10; 8; 6; 14; 6; 11; 14; 8; 10; 6; 12; 11
- s(32..47) = 10; 12; 5; 6; 13; 8; 12; 14; 13; 7; 12; 5; 4; 11; 6; 4
- s(48..63) = 10; 11; 13; 14; 13; 14; 8; 7; 7; 13; 4; 5; 7; 5; 4; 11
- s(64..79) = 8; 14; 4; 10; 5; 7; 12; 11; 4; 11; 12; 13; 10; 7; 4; 5
- s'(0..15) = 7; 8; 8; 10; 12; 14; 14; 4; 6; 6; 7; 10; 13; 13; 11; 5
- s'(16..31) = 8; 12; 14; 6; 11; 7; 7; 10; 6; 6; 11; 6; 5; 14; 12; 10
- s'(32..47) = 7; 6; 14; 10; 7; 5; 5; 13; 11; 12; 4; 13; 12; 12; 6; 4
- s'(48..63) = 14; 4; 7; 10; 13; 13; 5; 13; 5; 6; 11; 8; 11; 4; 14; 7
- s'(64..79) = 7; 4; 11; 8; 11; 4; 13; 5; 7; 12; 5; 4; 14; 12; 10; 10

Програма передбачає два варіанти вводу повідомлення для прорахунку кінцевого значення хеш-сигнатури : ввід з клавіатури чи сканування файлу з текстовим розширенням на його вміст .

Приклад вводу повідомлення та отримання хеш- сигнатури для нього наведений на рисунку 3.3.

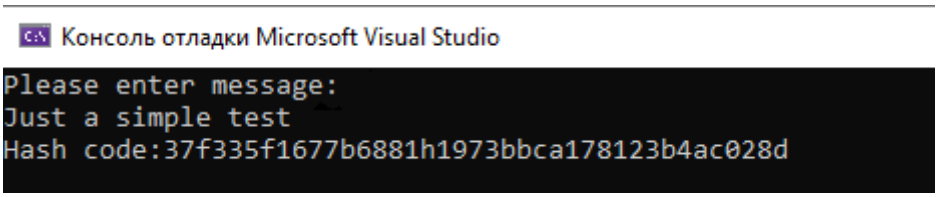


Рисунок 3.3 – Приклад отримання хеш-сигнатури для повідомлення

Було проведене багаторазове тестування на наявність багів при зміні початкового повідомлення.. Приклад такої зміни, навіть незначної, проілюстрований на рис 3.4.

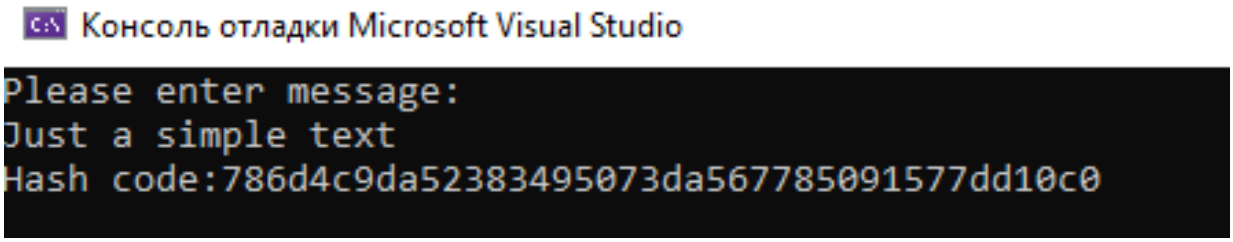


Рисунок 3.4. – Хеш-сигнатура після зміни початкових даних

Приклад отримання хеш-сигнатури для файлу наведений на рисунку 3.5.

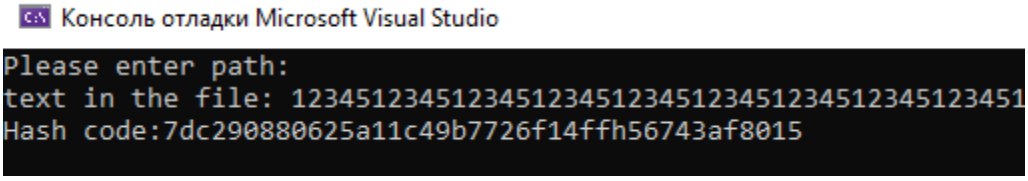


Рисунок 3.5 – Хеш-сигнатура для тестового файлу file1.txt

Висновки до розділу 3

В результаті виконання робіт, що складають даний розділ можна отримати наступні висновки:

1) Розглянуте та розроблене швидке обчислення хеш-сигнатур, розроблене за хеш-алгоритмом, було організовано в програму на мові С++. Основним джерелом підвищення швидкості обчислення хеш-сигнатур стало використання спеціально підібраних паттернів функціональних перетворень, що дає нам можливість тільки один раз виконувати обчислення, які використовуються на кожному із циклів хеш-перетворення.

2) Описаний спосіб організації даних, які використовуються в процесі розробки програми.

3) Описаний принцип дії головних функцій програмної розробки на мові С++, які були створені для реалізації нами обговореного алгоритму ідентифікації сторонніх абонентів.

4) Проаналізовані результати експериментального застосування розробленого програмного продукту: зроблені висновки на основі практичних експериментів, що використання нами вибраних шаблонів при обчисленні хеш-сигнатур за алгоритмом дозволяє на більш ніж 40 % покращити швидкість обчислень в порівнянні з аналогічними обчисленнями таких же хеш-сигнатур, виконаних за допомогою стандартного алгоритму.

ВИСНОВКИ

Під час виконання бакалаврської роботи виконане дослідження методів, за допомогою яких можна здійснити виграш у швидкості процесу строгої ідентифікації віддалених користувачів.

У дослідженні використовуються шифроблоки та хеш-перетворення для реалізацій запропонованих методів.

Метою роботи було дослідити можливості використання альтернативних способів строгої ідентифікації та порівняння їх ефективності з модулярним експоненціюванням, що зараз широко використовується.

Було проведено дослідження методів ідентифікації користувача, починаючи з найпримітивніших та закінчуючи сучасними розробками та обгрунтоване використання методу, який базується на концепції “нульових знань”.

Для вирішення поставленої задачі було запропоновано кілька способів прискорення обчислень. всі вони ґрунтуються на математичних твердженнях, що були проілюстровані та доведені у роботі. На основі цих доведень та теоретичних положень був розроблений алгоритм ідентифікації на основі хеш перетворень та шифроблоків, який реалізує концепцію нульових знань.

Для дослідження ефективності було проведено об’ємні дослідження та розроблено програмні засоби для тестування запропонованого методу. Також були змодульовані результати з використанням запропонованого методу строгої ідентифікації та модулярним експоненціюванням. Запропонований метод дозволив покращити швидкість ідентифікації на кілька порядків, якщо порівнювати зі стандартними алгоритмами.

У рамках проектного дослідження було також розроблено програмне забезпечення для набагато більш швидкого обчислення хеш-сигнатур та обґрунтовано використання цього алгоритму для імплементації у розроблені в роботі методи на практиці

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Марковський О.П., Захаріудакіс Л., Федотов М.Ф. Метод строгої ідентифікації відалених абонентів на основі стандартних шифроблоків та хеш-перетворень. // Телекомунікаційні та інформаційні технології, 2016- С 161-165.
2. Schneier B. Applied Cryptography. Protocols. Algorithms and Source codes in C. -Ed.John Wiley, 1996 - 758 pp
3. Feige U. Zero knowledge proofs of identity / Feige U.,Fiat A.,Shamir A.// Journal of Cryptology.- v.1.- №.2.- 1988, pp.77-94.
4. Schnorr C.P. Efficient Signature Generation for Smart Cards // Journal of Cryptology, Vol. 4, No.3.- 1991.- pp.161-174.
5. Марковский А.П. Организация идентификации абонентов многопользовательских систем / Марковский А.П., Азаде Герами, Ияд Мохд Маджид Ахмад Шахрури. // Труды 9-й ”Международной конференции ”Современные информационные и электронные технологии”. Одесса.-2008.- С.129.
6. Wang W. On the Relation Between Identifiability , differential Privacy and Mutual Informational Privacy / Wang W., Ying I., Zhang J. // IEEE Trans. Inf. Theory.- Vol. 62.- 2016.- № 9.- P. 5018-5029.
7. Мухін В.Є. Метод ідентифікації віддалених абонентів на основі концепції “нульових знань” / В.Є. Мухін, Лефтеріс Захаріудакіс, Ю.Н. Герасименко, М.С. Козерацький // Телекомунікаційні та інформаційні технології, 2017 –№1.– С.50-57. рг.-2005.- 286 с.
8. Markovskyy, O., Bardis, N. and Doukas, N. (2010), “Fast subscriber identification based on the zero knowledge principle for multimedia content distribution”, International Journal of Multimedia Intelligence and Security, no. 1, pp. 78-82.

9. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.:”Триумф”. - 2002.-816 с.

10. Nikolaychuk, Ya.M. (2012), Kody polya Galua: teoriya ta zastosuvannya [Galois field codes: theory and applications], TzOV Ternograf, Ternopil, Ukraine.

11. Николайчук Я.М. Коди полів Галуа: теорія і застосування. Тернопіль.-Вид-во ТНУ.-2012.- 576 с.

12. Брэй Б. Микропроцессоры Intel. Архитектура, программирование и интерфейсы. - СПб.: Изд-во “БХВПетрербург”.- 2005.- 1328 С

13. Самофалов К.Г. Повышение эффективности идентификации удаленных абонентов многопользовательских систем / Самофалов К.Г., Тубольцев А.А., Ияд Мохд Маджид Ахмад Шахрури // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 3(14).- К.,НАУ.- 2008.- С.129-136.

14. Markovskyy, O., Bardis, N. and Doukas, N. (2010), “Fast subscriber identification based on the zero knowledge principle for multimedia content distribution”, International Journal of Multimedia Intelligence and Security, no. 1, pp. 78-82.

15. Марковський О. П. Використання алгебри полів Галуа для реалізації концепції нульових знань при ідентифікації та автентифікації віддалених користувачів /О.П.Марковський, Захаріудакіс Лефтеріс., В.Р.Максимук // Электронное моделирование. 2017.- № 6.- С.96-110.

16. Харин Ю.С., Берник В.И., Матвеев Г.В. Математические основы криптологии / Ю.С. Харин, В.И. Берник, Г.В. Матвеев //.-Минск.: Изд-во БГУ, 1999.- 319 с.

17. Марковський О.П. Метод строгої ідентифікації абонентів в телекомунікаційних системах / О.П. Марковський, Лефтеріс Захаріудакіс, М.Ф. Зм. Арк. № докум. Підпис Дата Арк. ДП 4682 74 .03.000 ПЗ Федотов //

XI Міжнародна науково-технічна конференція “Проблеми телекомунікації” ПТ-2017: Збірник матеріалів конференції КПІ ім. Ігоря Сікорського, 2017.- С.334-337.

18. Bellekens X. Cyber-PhysicalSecurity Model for Safety-Critical IoT Infrastructures /X. Bellekens, A. Seeam, K. Nieradzinska, C. Tachtatzis, A. Cleary, R. Atkinson, and I. Andonovic // Proceeding Wireless World Research Forum Meeting 35 (WWRF35), Copenhagen, Danemark, 2015.- P.114-118.

19. Расторгуев С.П. Программные методы защиты информации в компьютерах и сетях. М: Изд - во “ Яхтсмен “, 1993. - 188 с.

20. Самофалов К.Г., Марковский А.П., Гаваагийн Улзисайхан, Бардис Н., Метод получения булевых балансных SAC-функций для систем защиты информации. // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка.-1998.- № 31.- С.131-140.

21. Марковский А.П. Получение булевых преобразований специальных классов для построения эффективных алгоритмов защиты информации / А.П. Марковский, А.А. Зюзя, В.Д. Шерстюк // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. К.,"ВЕК++",- 2008.- № 49.- С.7-13

22. Столлингс В. Криптография и защита сетей: принципы и практика. М.: Изд.дом.”Вильямс”.- 2001-621 с.

ДОДАТОК А

Метод криптографічно строгої автентифікації та програмні засоби його реалізації

Блок-схема алгоритму

Аркушів 1

Київ – 2021 р.

					ІАЛЦ 468243.005 Д1	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		СидоренкоК.М			*Метод криптографічно строгої автентифікації та програмні засоби його реалізації* Блок-схема алгоритму	Літ. / Аркуш 1 / Аркушів 1
Перевірив		Марковський О.П.				
Н. Контр.		Сімоненко В.П.				«КПІ ім. Ігоря Сікорського», ФІОТ, ІП-73
Затвердив						

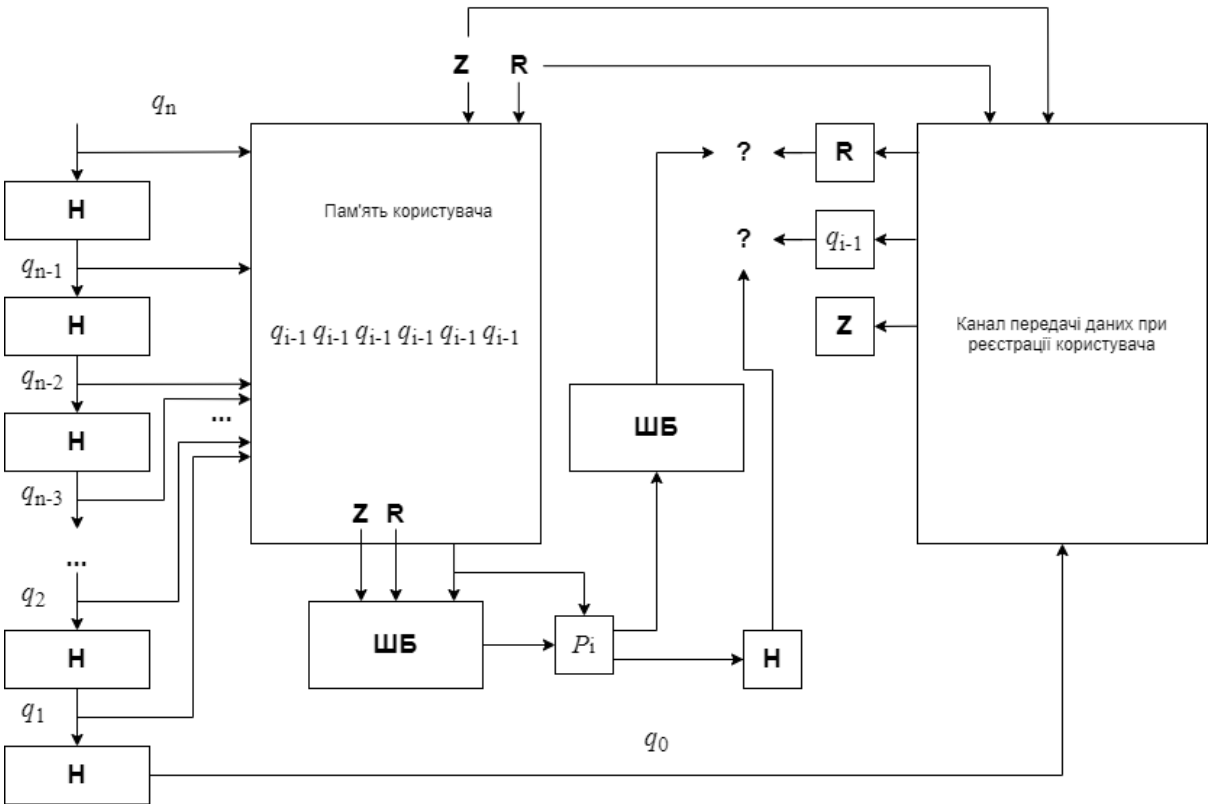
ДОДАТОК Б

Метод криптографічно строгої автентифікації та програмні засоби його реалізації

Структурна схема

Аркушів 1

Київ – 2021 р.



Z та R -спеціальні коди, які формуються на стороні користувача при ініціалізації

H - хеш-перетворення

P_i - і-тий сеансовий пароль

ШБ - шифроблок

q_n, q_{n-1}, q_{n-2},...,q_2, q_1 - коди, хеш сигнатури якого обчислюються на стороні користувача

					ІАЛЦ 468243.006 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Сидоренко К.М			*Метод криптографічно строгої автентифікації та програмні способи його реалізації* Структурна схема	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Марковський О.П.					
Н. Контр.		Сімоненко В.П.				«КПІ ім. Ігоря Сікорського», ФІОТ, ІП-73	
Затвердив							

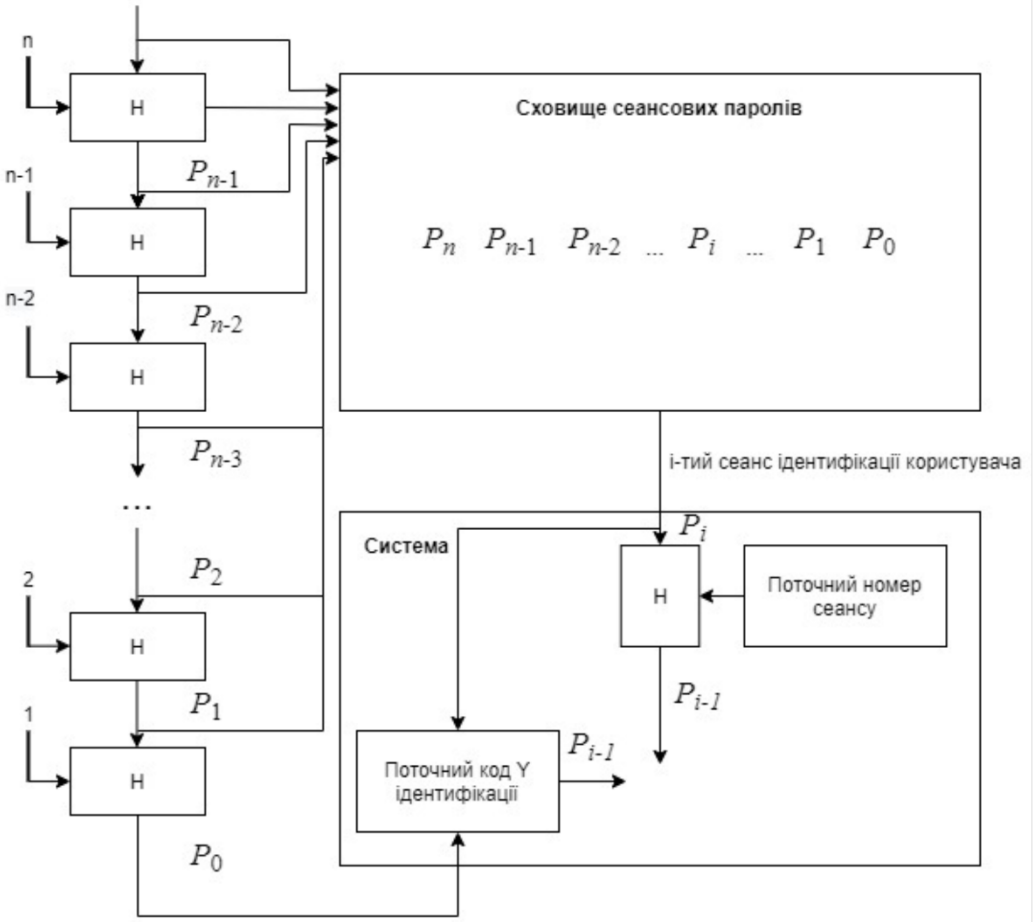
ДОДАТОК В

Метод криптографічно строгої автентифікації та програмні засоби його реалізації

Схема методу строгої ідентифікації з використанням хеш-перетворень

Аркушів 1

Київ – 2021 р.



H - хеш-перетворення

P_i - і-тий сеансовий пароль

					ІАЛЦ 468243.006 Д3		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Сидоренко К.М			*Метод криптографічно строгої автентифікації та програмні способи його реалізації* Схема методу	Літ.	Аркуш 1 / Аркушів 1
Перевірив		Марковський О.П.					
Н. Контр.		Сімоненко В.П.				«КПІ ім. Ігоря Сікорського», ФІОТ, ІП-73	
Затвердив							

ДОДАТОК Г

Метод криптографічно строгої автентифікації та програмні засоби його реалізації

Лістинг програми

Аркушів 1

Київ – 2021 р.

```
#include "fileHasher.h",
#include "multithreadHasher.h"
#include <fstream>
#include <stdexcept>
#include <thread>
#include <future>
#include <mutex>
#include <chrono>
#include <iostream>
#define K1_0 0x00000000
#define K1_1 0x5A827999
#define K1_2 0x6ED9EBA1
#define K1_3 0x8F1BBCDC
#define K1_4 0xA953FD4E
#define K2_0 0x50A28BE6
#define K2_1 0x5C4DD124
#define K2_2 0x6D703EF3
#define K2_3 0x7A6D76E9
#define K2_4 0x00000000
#define INIT_DATA_H0 0x67452301
#define INIT_DATA_H1 0xEFCDAB89
```

```
#define INIT_DATA_H2 0x98BADCFE
#define INIT_DATA_H3 0x10325476
#define INIT_DATA_H4 0xC3D2E1F0
const int NUM_THREAD = 4;,
std::atomic<bool> is_done[NUM_THREAD] = { true, true, true, true };
std::mutex mtx;
#define RMD160_BLOCK_SIZE 16 // 16 32-розрядних слів у блоці
fileHasher::fileHasher(std::string filePathIn, std::string filePathOut, int chunkSize)
{
_filePathIn = filePathIn;
_filePathOut = filePathOut;
_chunkSize = chunkSize;
}

fileHasher::~fileHasher()
{
}

size_t fileHasher::CalculateHash()
{
std::hash<std::string> hash_fn;
std::ifstream fileStream;
fileStream.open(_filePathIn);
```

```
if (fileStream.is_open()) {
fileStream.seekg(0, fileStream.end);
auto length = fileStream.tellg();
fileStream.seekg(0, fileStream.beg);
std::string buffer(_chunkSize, ' ');
size_t result = 0;
for (auto i = 0; i < length; i = i + _chunkSize) {
fileStream.read(&buffer[0], _chunkSize);
result ^= hash_fn(buffer);
}
return result;
}
else {
throw std::invalid_argument("received bad input file.");
}
}

void CalculateHashThread(size_t &result, std::string buffer, int num_of_thread) {
std::hash<std::string> hash_fn;
auto p = hash_fn(buffer);
mtx.lock();
result ^= p;
mtx.unlock();
is_done[num_of_thread] = true;
```

```
}


size_t fileHasher::CalculateHashMultithread()
{
std::ifstream fileStream;
fileStream.open(_filePathIn);
if (fileStream.is_open()) {
fileStream.seekg(0, fileStream.end);
auto length = fileStream.tellg();
fileStream.seekg(0, fileStream.beg);
std::string buffer(_chunkSize, ' ');
size_t result = 0;
std::thread threads[NUM_THREAD];
int curr_thread = 0;
for (auto i = 0; i < length; i = i + _chunkSize) {
fileStream.read(&buffer[0], _chunkSize);
while (true) {
curr_thread++;
if (curr_thread == 4) {
curr_thread = 0;
}
if (is_done[curr_thread]) {
if (threads[curr_thread].joinable()) {
threads[curr_thread].join();
```

```
}
is_done[curr_thread] = false;
threads[curr_thread] = std::thread(CalculateHashThread, std::ref(result), buffer, curr_thread);
break;
}
}
}
for (auto i = 0; i < NUM_THREAD; i++) {
if (threads[i].joinable()) {
threads[i].join();
}
}
return result;
}
else {
throw std::invalid_argument("received bad input file.");
}
}

size_t fileHasher::CalculateHashMultithreadWithClass() {
multithreadHasher multithreadHasherInstance(NUM_THREAD, _filePathIn, _filePathOut, _chunkSize);
return multithreadHasherInstance.CalculateHash();
```

```
}

#include "multithreadHasher.h",
#include <string>

multithreadHasher::multithreadHasher(int numberOfThreads, std::string inputFilePath, std::string outputFilePath, int chunkSize) :
_numberOfThreads(numberOfThreads), _currThread(0), _isDone(numberOfThreads, true),
_result(0), _chunkSize(chunkSize), _threads(numberOfThreads) {
_inputFileStream.open(inputFilePath);
_outputFileStream.open(outputFilePath);
}

multithreadHasher::~multithreadHasher()
{
_inputFileStream.close();
_outputFileStream.close();
}

void multithreadHasher::CalculateHashThread(std::string buffer, int num_of_thread) {
std::hash<std::string> hash_fn;
auto p = hash_fn(buffer);
_mtx.lock();
```

```
_result ^= p;
_mtx.unlock();
_isDone[num_of_thread] = true;
}

size_t multithreadHasher::CalculateHash()
{
if (!_outputFileStream) {
throw std::invalid_argument("cannot create output file.");
}
if (_inputFileStream.is_open()) {
_inputFileStream.seekg(0, _inputFileStream.end);
auto length = _inputFileStream.tellg();
_inputFileStream.seekg(0, _inputFileStream.beg);
std::string buffer(_chunkSize, ' ');
for (auto i = 0; i < length; i = i + _chunkSize) {
_inputFileStream.read(&buffer[0], _chunkSize);
while (true) {
_currThread++;
if (_currThread == 4) {
_currThread = 0;
}
if (_isDone[_currThread]) {
if (_threads[_currThread].joinable()) {
```

```
_threads[_currThread].join();
}
_isDone[_currThread] = false;
_threads[_currThread] = std::thread(&multithreadHasher::CalculateHashThread, this, buffer, _currThread);
break;
}
}
}
for (auto i = 0; i < _numberOfThreads; i++) {
if (_threads[i].joinable()) {
_threads[i].join();
}
}
_outputFileStream << std::hex << _result;
return _result;
}
else {
throw std::invalid_argument("received bad input file.");
}
}
```