НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

«На правах рукопису»
УДК _____

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
«___»_____2023 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем» зі спеціальності 121 “Інженерія програмного забезпечення” на тему: «Метод захищеної реалізації криптографічних алгоритмів на термінальних пристроях»

Виконала:
студентка ІІ курсу, групи ІМ-21мп
Щелконогова Марина Борисівна _____

Керівник:
доц., к.т.н.
Марковський Олександр Петрович
_____

Консультант з нормоконтролю:
проф. каф. ОТ, д.т.н., професор
Жабін Валерій Іванович
_____

Рецензент:
Декан ФПМ, проф., д.т.н., Дичка І.А. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Обчислювальної техніки*
(повна назва)

Рівень вищої освіти – другий (магістерський)

Спеціальність *121. Інженерія програмного забезпечення*
(код і назва)

Освітня програма: *Інженерія програмного забезпечення комп'ютерних систем*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Сергій СТІРЕНКО*
(підпис) (ініціали, прізвище)
« » 2023 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Щелконоговій Марині Борисівні
(прізвище, ім'я, по батькові)

1. Тема дисертації *Метод захищеної реалізації криптографічних алгоритмів на термінальних пристроях*

Науковий керівник дисертації *доц., к.т.н. Марковський О.П.*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «07» листопада 2023 р. № 5168-с

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження *обчислювальні процеси базової операції криптографії з відкритим ключем – модулярного експоненціювання, що здійснюється над числами, довжина яких на порядок перевищує розрядність процесора, з залученням віддалених комп'ютерних систем*

4. Предмет дослідження *метод розподілення обчислень модулярної експоненти між термінальною та віддаленими комп'ютерними платформами, який забезпечує високу швидкість реалізації криптографічних алгоритмів в системах*

віддаленого управління на базі технологій IoT зі збереженням високого рівня захищеності

5. Перелік завдань, які потрібно розробити: *дослідити властивості та спроєктувати схему криптографічного захисту в системах віддаленого моніторингу та управління об'єктами реального світу, побудованих на базі технологій IoT..*

6. Консультанти розділів дисертації:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	проф., д.т.н. Жабін В.І.		
1	доц., к.т.н. Марковський О.П.		
2	доц., к.т.н. Марковський О.П.		
3	доц., к.т.н. Марковський О.П.		

7. Дата видачі завдання 1.09.2023

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1.	Затвердження темироботи	14.05.2023	
2.	Дослідження літературита документації	15.05.2023-01.06.2023	
3.	Аналіз існуючих систем та підходів	01.06.2023-10.06.2023	
4.	Аналіз сучасних вимог до засособів віддаленого шифрування даних	10.06.2023-01.07.2023	
5.	Аналіз моделей віддаленого шифрування даних	01.07.2023-15.07.2023	
5.	Написання пояснювальноїзаписки	15.07.2023-20.09.2023	
6.	Розрахунки стартапу	20.09.2023-30.09.2023	
7.	Захист дисертації	17.01.2024	
8.	Затвердження темироботи	14.05.2023	

Студент Марина ЩЕЛКОНОГОВА
(підпис)

Науковий керівник дисертації Олександр МАРКОВСЬКИЙ

РЕФЕРАТ

Актуальність теми визначається динамічним розширенням використання технологій IoT в системах управління віддаленими об’єктами реального світу, з підвищеними вимогами до безпеки.

Мета досліджень полягає в підвищенні ефективності розподіленого обчислення базової операції криптографії з відкритим ключем – модулярного експоненціювання на термінальних мікроконтролерах систем управління на базі технологій IoT з залученням хмарних обчислень за рахунок зменшення обчислювального навантаження на термінальну платформу, що дозволяє прискорити реалізацію криптографічних алгоритмів зі збереженням високого рівня захищеності.

Об'єкт дослідження – обчислювальні процеси базової операції криптографії з відкритим ключем – модулярного експоненціювання, що здійснюється над числами, довжина яких на порядок перевищує розрядність процесора, з залученням віддалених комп’ютерних систем.

Предмет дослідження – метод розподілення обчислень модулярної експоненти між термінальною та віддаленими комп’ютерними платформами, який забезпечує високу швидкість реалізації криптографічних алгоритмів в системах віддаленого управління на базі технологій IoT зі збереженням високого рівня захищеності.

Наукова новизна отриманих результатів визначається наступним:

Теоретично обґрунтовано, розроблено та досліджено метод організації розподіленого обчислення базової операції з відкритим ключем – модулярного експоненціювання на термінальному мікроконтролері систем управління на базі IoT з залученням хмарних обчислень, який відрізняється тим, що організовано передачу проміжних результатів з відданих обчислювальних платформ на термінальну, що дозволило зменшення обчислювального навантаження на термінальну платформу, за рахунок чого досягнуто прискорення реалізації криптографічних алгоритмів зі збереженням високого рівня захищеності.

Практична значимість отриманих результатів визначається тим, що їх використання дозволяє реалізувати в реальному часі складні механізми криптографічного захисту в системах віддаленого моніторингу та управління об’єктами реального світу, побудованих на базі технологій IoT.

Магістерська дисертація складається з чотирьох розділів.

В першому розділі виконано аналіз захисту інформації в системах віддаленого моніторингу та управління об’єктами реального світу, побудованих на базі технологій IoT. Визначено вимоги до часових характер реалізації механізмів захисту інформації в таких системах. Здійснено критичний огляд існуючих підходів і методів до швидкої реалізації и базової операції криптографії з відкритим ключем – модулярного експоненціювання, що здійснюється з залученням віддалених комп’ютерних систем.

В другому розділі виконано теоретичне дослідження розподіленого обчислення модулярної експоненти з урахуванням безпекових аспектів, обґрунтовано та викладено метод захищеного обчислення модулярної експоненти з залученням віддалених комп’ютерних систем.

В третьому розділі виконано теоретичне та експериментальне дослідження ефективності запропонованого методу метод захищеного обчислення модулярної експоненти з залученням віддалених комп’ютерних систем.

В четвертому розділі виконано стартапну розробку проєкту.

Представлена магістерська дисертація складається з 96 сторінок, містить 5 рисунків, 8 таблиць та лістинги програм.

ABSTRACT.

The relevance of the topic is determined by the dynamic expansion of the use of IoT technologies in control systems for remote real-world objects with increased security requirements.

The purpose of the research is to increase the efficiency of distributed computation of the basic operation of public key cryptography - modular exponentiation on terminal microcontrollers of control systems based on IoT technologies using cloud computing by reducing the computational load on the terminal platform, which allows accelerating the implementation of cryptographic algorithms while maintaining a high level of security.

The object of research is the computational processes of the basic operation of public key cryptography - modular exponentiation performed on numbers whose length is an order of magnitude greater than the processor bit capacity, with the involvement of remote computer systems.

The subject of the study is a method of distributing modular exponent calculations between terminal and remote computer platforms, which provides a high speed of implementation of cryptographic algorithms in remote control systems based on IoT technologies while maintaining a high level of security.

The scientific novelty of the obtained results is determined by the following:
A method for organizing the distributed computation of the basic operation with a public key - modular exponentiation on the terminal microcontroller of IoT-based control systems using cloud computing is theoretically substantiated, developed and investigated, which differs in that the transfer of intermediate results from remote computing platforms to the terminal one is organized, which allowed reducing the computational load on the terminal platform, thereby accelerating the implementation of cryptographic algorithms for preserving

The practical significance of the obtained results is determined by the fact that their use allows to implement complex cryptographic protection mechanisms in real-time in systems of remote monitoring and control of real-world objects built on the basis of IoT technologies.

The master's thesis consists of four chapters.

The first chapter analyzes the protection of information in systems of remote monitoring and control of real-world objects based on IoT technologies. The requirements for the time character of the implementation of information security mechanisms in such systems are determined. A critical review of existing approaches and methods to the rapid implementation and basic operation of public key cryptography - modular exponentiation performed on numbers whose length is an order of magnitude greater than the processor bit capacity, with the involvement of remote computer systems is carried out.

In the second section, a theoretical study of the distributed calculation of the modular exponent is carried out, taking into account security aspects, and a method of secure calculation of the modular exponent using remote computer systems is substantiated and presented.

In the third section, a theoretical and experimental study of the effectiveness of the proposed method of securely calculating the modular exponent using remote computer systems is performed.

The fourth chapter describes the startup development of the project.

The presented master's thesis consists of 96 pages, contains 5 figures, 8 tables, and program listings.

ЗМІСТ

ВСТУП

Результатом розвитку технологій Інтернет та якісного поліпшення характеристик інтегральних мікросхем стало динамічне розширення використання технологій Інтернету речей ((Internet of Things –IoT) для створення систем віддаленого моніторингу стану та управління різноманітними об’єктами реального світу. Тобто в останні роки технології IoT виходять далеко за рамки побутових пристроїв: фактично вже сьогодні вони широко застосовуються для управляння технологічними процесами, транспортними засобами, в системах відео спостереження та охоронній сигналізації.

Основним вузлом в технологіях IoT виступають мікроконтролери з розвиненою периферією і вбудованим радіомодемом, який дозволяє здійснювати інтерфейс з глобальною мережею Інтернет. При цьому такі мікроконтролери через Інтернет виконують функції контролю стану та управління різноманітними віддаленими об'єктами реального світу і при цьому здатні взаємодіяти із різними системами обробки інформації. Обмін даними через потенційно відкрите середовище Інтернет створює загрози несанціонованого втручання у роботу мікроконтролерів, які безпосередньо взаємодіють в об'єктах реального світу. Для більшості практичних застосувань наявність загроз стороннього втручання в роботу термінальних мікроконтролерів, що контролюють реальні об'єкти та керують ними, є неприйнятною. Тому при обміні даними через глобальну мережу Інтернет необхідно підтримувати мережові протоколи інформаційної безпеки. Переважна більшість цих протоколів використовує криптографію з відкритим ключем, базовою операцією якої є модулярне експоненціювання, яке виконується над числами довжина яких становить 2048 з перспективою подальшого збільшення до 4096 і більше. Відповідно, виконання цієї операції малопотужними портативними мікроконтролерами потребує значних витрат часу. Разом з тим, більшість вбудованих мікроконтролерів повинні виконувати функції контролю та стану реальних об'єктів та управління ними в реальному часі. Це диктує практичну необхідність пошуку можливостей швидкої реалізації

операції модулярного експоненціювання над числами великої розрядності на малопотужних мікроконтролерах.

У сучасних умовах широкого використання хмарних технологій класичним підходом вирішення проблеми нестачі обчислювальних потужностей полягає у залученні віддалених потужних комп'ютерних систем. Застосування цього підходу до прискорення обчислення модулярної експоненти на термінальних мікроконтролера систем управління потребує збереження в секреті значень компонент цієї операції. Для цього необхідно організувати розподілене обчислення модулярної експоненти таким чином, щоб ці значення не передавалися в хмару. При цьому треба розділити процес обчислення модулярної експоненти на дві нерівні частини, з яких більша за об'ємом обчислень виконується з застосуванням хмарних обчислень, а менша реалізується термінальним мікроконтролером.

Таким чином наукова задача створення ефективною організації розподіленого швидкого обчислення модулярної експоненти з залученням хмарних технологій для систем управління об’єктами реального світу на базі технологій Інтернету речей є важливою і актуальною з огляду за зазначені вище тенденції розвитку інформаційних технологій.

РОЗДІЛ 1
АНАЛІЗ ЗАДАЧІ ШВИДКОЇ РЕАЛІЗАЦІЇ КРИПТОГРАФІЇ З ВІДКРИТИМ КЛЮЧЕМ В СИСТЕМАХ УПРАВЛІННЯ НА БАЗІ IoT ТА ОГЛЯД ІСНУЮЧИХ МЕТОДІВ ЇЇ ВИРІШЕННЯ

Динамічне розширення використання технологій Інтернету речей (Internet of Things) в системах комп'ютерного управління об'єктами реального світу різного призначення створило задачу ефективного захисту даних при застосуванні потенційно вразливого середовища їх передачі – глобальної мережі Інтернет [1]. Для створення ефективного захисту потрібно застосовувати увесь спектр методів та засобів криптографічного захисту, передбаченими існуючими протоколами мережевої безпеки. Значна частина цих засобів, зокрема криптографія з відкритим ключем, базується на ресурсоємких в обчислювальному плані операціях модулярного експоненціювання, які здійснюються над числами, довжина яких на порядки перевищує розрядність термінальних комп'ютерних платформ систем комп'ютерного управління. Мікроконтролери, які використовуються в якості таких платформ, виходячи із специфіки їх призначення, мають низьку продуктивність, зумовленою вимогам до низького споживання потужності, компактності, малої вартості. Відповідно, об'єктивно виникають труднощі реалізації складних обчислень модулярного експоненціювання на таких платформах в режимі реального часу.

Для віднаходження шляхів вирішення вказаної наукової задачі потрібно виконати аналіз особливостей операції криптографії з відкритим ключем – модулярного піднесення до степеню на обладнаних радіомодемом термінальних обчислювальних платформах моніторингу стану та управління віддаленими об'єктами реального світу з використанням технологій Інтернету речей. Важливим наступним етапом є критичний аналіз методів прискореного виконання модулярного експоненціювання на різних типах обчислювальних платформ. Потрібно також здійснити аналіз можливостей використання інших алгебраїчних базисів. Нарешті, враховуючи, що термінальні платформи систем управління на базі IoT обладнані вбудованим радіомодемом, що відкриває

можливості підключення до глобальної мережі з залученням для прискорення обчислень хмарних технологій. Останній з накреслених напрямків виглядає як найбільш перспективний , хоча потребує спеціальних засобів захисту для виключення можливості відновлення секретного коду експоненти на віддалений і, відповідно, непідконтрольній, комп’ютерній платформі, яка безпосередньо здійснює операції модулярного експоненціювання.

1.1. Аналіз задач захисту інформації в системах комп’ютерного управління на базі IoT.

Базовими задачами захисту інформації в сучасних системах комп’ютерного управління віддаленими об’єктами реального світу, інформаційна взаємодія з якими здійснюється через глобальні мережі є:

- Протидія несанкціонованого доступу (читання) до інформаційних повідомлень, якими обмінюються термінальні мікроконтролери та центральний комп'ютер;
- Забезпечення цілісності та автентичності інформаційних повідомлень, якими обмінюються термінальні мікроконтролери системи моніторингу стану та управління об’єктами реального світу і центральний комп'ютер, тобто захист від несанкціонованих змінень даних в процесі обміну та підміни відправника цих даних;

- Забезпечення захисту від підміни учасника інформаційної взаємодії в системах комп’ютерного моніторингу стану та управління об’єктами реального світу, тобто автентифікація термінальних пристроїв таких систем та їх базового комп'ютера.

Для вирішення цих завдань використовуються системи органічно пов'язаних організаційних, алгоритмічних, технічних і програмних засобів [3] які утворюють систему захисту захисту інформації. [4].

Аналіз практики захисту інформації в системах комп’ютерного управліннями віддаленими об’єктами реального світу на базі технологій IoT засвідчує, що найбільш важливою задачею з наведених вище є друга, а саме:

забезпечення цілісності та автентичності інформаційних повідомлень, якими обмінюються термінальні мікроконтролери системи моніторингу стану та управління об’єктами реального світу і центральний комп'ютер, тобто захист від несанкціонованих змінень даних в процесі обміну та підміни відправника цих даних. Більшість атак на такі системи пов’язані з фальшуванням даних про стан віддалених об’єктів або посилка фальшованих команд управління ними. Технічно це може бути виконано шляхом підміни автора або безпосереднього фальшування даних в процесі їх пересилок між компонентами системи управління.

Автентичність та цілісність даних, що передаються між компонентами системи комп’ютерного управління віддаленими об’єктами реального світу забезпечується криптографічними механізмами цифрового підпису DSA та ГОСТ Р34.10-94. В основі цих механізмів лежать операції модулярного експоненціювання, а основною операцією є $A^E \bmod M$.

Цифровий підпис інформаційної посилки - це криптографічний механізм, що гарантує: по - перше, цілісність - тобто те, що дані не були змінені в процесі передачі, а по - друге: авторство, тобто те, що дані сформовані і підписані певним відправником: термінальним пристроєм системи управління або його базовим компютером [5]. Початок практичного використання цифрового підпису в банківських установах було покладено створенням в 1978 році механізмів криптографії з відкритим ключем [6].

Перші механізми цифрового підпису мали за основу широко відомий алгоритм несиметричного шифрування даних - RSA. З часом стало зрозуміло, що використання алгоритмів несиметричного шифрування для формування і перевірки автентичності та цілісності блоків даних не є ефективним, оскільки пов'язане з використанням надлишкових обчислювальних ресурсів для реалізації цієї цілі.

Почалися роботи по створенню ефективних спеціалізованих механізмів формування та перевірки цифрового підпису. Етапним моментом на цьому шляху стало прийняття в серпні 1991 року стандарту цифрового підпису DSS

(Digital Signature Standard) [7]. Цей стандарт передбачає процедуру перевірки автентичності та цілісності електронних документів, яка регламентується алгоритмом DSA (Digital Signature Algorithm). Через кілька років в Росії був прийнятий у якості стандарту практично ідентичний за закладеними в його основу математичними принципами алгоритм формування і перевірки цифрового підпису ГОСТ Р.34.10-94 [8].

Алгоритм цифрового підпису DSA передбає три режими його роботи:

- Генерація закритий і відкритих ключів;
- Формування цифрового підпису блоку даних: підписування даних;
- Перевірка правильності блоку даних та його цифрового підпису на стороні приймача інформаційного блоку.

На етапі генерації ключів виконується наступна послідовність дій:

1.Генерується просте h-розрядне число p. Наприклад, при h =6 можливим значенням p може бути $p = 59_{10} = 111011_{2.}$

2. Довільним чином віднаходиться просте число q розрядності f ≈ h/2, таке, що воно націло ділиться на число p -1. Наприклад, якщо p = 59, то p -1 = 58 = 2 · 29 і тоді q = 29.

3. Випадковим чином обирається ціле число w, таке, що мають виконуватися дві умови: $w<p$-1 та $w^{(p-1)/q} \bmod p > 1$. Наприклад, якщо обрати число w = 37, то $37^2 \bmod 59 = 12$.

4. Обчислюється $g = w^{(p-1)/q} \bmod p$. Наприклад при p = 59, q = 29 і w = 5, g = 14.

5. Випадковим чином обирається число $x < q$, наприклад, x = 16.

6. Обчислюється $y = g^x \bmod p$. У рамках поточного прикладу значення $y = 14^{16} \bmod 59 = 41$.

Отримані описаним способом цілі числа p, q, g і y - є компонентами відкритого ключа, а x - закритий ключ.

Формування цифрового підпису під конкретним блоком даних *D* виконується DSA в наступній послідовності:

При передачі даних *D* формується його хеш-сигнатура *H (D)* фіксованої довжини. Стандартом для формування хеш-сигнатури встановлений хеш-алгоритм SHA-1 [8]. Нехай, наприклад *H (M)* = 47. Пристій (термінальний мікроконтролер чи базовий комп'ютер системи управління) виконує такі дії:

1. Випадковим чином обирає ціле число *k* таке, що *t* <*q;* наприклад *t* = 17.

2. Обчислює $r = (g^t \bmod p) \bmod q$. У рамках прикладу, який розглядається значення *r* обчислюється у вигляді: *r* =(14^{17} mod 59) mod 29 = 43 mod 29 = 14.

3. Визначається мультиплікативна інверсія t^{-1} тобто таке число, для якого $t \cdot t^{-1} \bmod q = 1$. Наприклад, при *t*= 17 значення $k^{-1} = 11$.

4. Обчислюється $s = (t^{-1} \cdot (H(D) + x \cdot r) \bmod q$. Для розглянутого прикладу значення s = (11 · (47 +14 · 14)) mod 29 = 2683 mod 29 = 5.

5. Цифровий підпис, який складається з 2-х компонент ***r*** і ***s*** надсилається на сторону приймача відправником(термінальним мікроконтролером чи базовим комп'ютером системи управління) разом з блоком даних.

Сторона, що приймає блок даних та цифровий підпис перевіряє автентичність та цілісність з використанням цифрового підпису в такому порядку:

1. Для отриманого блока даних D 'згідно з обраним хеш-алгоритмом обчислюється хеш-сигнатура *H (D').* Якщо дані D в процесі передачі через мережу Інтернет не зазнали змін, тобто D =D' і, відповідно виконується *H (D) = H (D').* Наприклад, якщо *H (D)* = 47 і блок даних *D* не зазнав змін при передачі, то *H (D*') = 47.

2. На стороні приймача віднаходиться код мультиплікативної інверсії, тобто такой код s^{-1} , що $s \cdot s^{-1} \bmod q = 1$. Якщо *q*= 29, *s* = 5 то $s^{-1} = 3$.

3. На стороні приймача (термінального мікроконтролера або базового комп’ютера системи управління віддаленими об'єктами) обчислюється перевірочний код $u_1 = (H(D') \cdot s^{-1}) \bmod q$. Для прикладу u_1 = (47·3) mod 29 = 1123 mod 29 = 16.

4. На стороні приймача (термінального мікроконтролера або базового комп’ютера системи управління віддаленими об'єктами) обчислюється другий перевірочний код $u_2 = (r \cdot s^{-1}) \bmod q$, для розглянутого прикладу u_2 = (4·3)mod mod 29=12 mod 29 = 12.

5.Обчислюється $v = ((w^{u_1} \cdot y^{u_2}) \bmod p) \bmod q$. Якщо $v = r$, то вважається, що дані не зазнали змін при передачі від передавача до приймача і передавач при цьому не підмінений, тобто дані є цілісними і автентичними.

Головним недоліком розглянутого стандартизованого алгоритму цифрового підпису є велика обчислювальна складність. Основний обсяг обчислень займає виконання модулярного експоненціювання, тобто обчислення $r = (g^k \bmod p) \bmod q$ при формуванні цифрового підпису блоку даних та $v = ((g^{u_1} \cdot y^{u_2}) \bmod p) \bmod q$ при перевірці коректності підпису.

Наразі використання хмарних обчислень розповсюджується з великою швидкістю і в різних сферах. Ці прогресивні технології надають широкому кругу користувачів доступ до значних за обсягом обчислювальним ресурсам віддалених комп’ютерних систем і суперкомп’ютерів. Проте, як засвідчує практика [4] їх можуть використовувати не тільки для захисту інформації, а й для порушення безпеки, зокрема для підбору ключів механізмів та алгоритмів криптографічного захисту інформації. .

Широке розповсюдження розподілених обчислень і, зокрема технології хмарових обчислень, пов’язане з можливостями для зловмисників концентрації для вирішення задачі підробки підпису або зміни документу на їх користь значно більших обчислювальних потужностей. Для підвищення стійкості

механізмів цифрового підпису об'єктивно потрібно збільшувати розрядність чисел до 2048 і, в перспективі до 4096. Особливо велике значення має збільшення розрядності показника ступеню *E,* оскільки саме воно є найбільш важливим та таким, що потребує захисту.

Зазначена тенденція збільшення розрядності має наслідком експоненціальне зростання обчислювальної складності реалізації цифрового підпису, причому це зростання суттєво випереджає темпи збільшення швидкодії комп'ютерів. Проблема зростання обчислювальної складності комп'ютерної реалізації цифрового підпису особливо гостро стоїть для термінальних пристроїв - малорозрядних контролерів, які підтримують мережеві протоколи захисту даних. Для таких обчислювальних платформ неможливо використувати спеціалізовані криптопроцесори, які на апаратному рівні реалізують операцію модулярного експоненціювання і забезпечують при цьому прискорення її виконання приблизь на два порядки в порівнянні з програмною реалізацією.

Тому актуальною та важливою є проблема підвищення продуктивності програмної та апаратної реалізації механізмів цифрового підпису за рахунок, зокрема, пошуку шляхів прискорення обчислювальної реалізації операції модулярного експоненціювання.

Для цього у нас є такі можливості:

- шукати засоби прискореного обчислення процедур цифрового підпису саме для операції модулярного експоненціювання за рахунок більш раціональної організації обчислень;
- здійснити перехід до іншої алгебри, в якій ця операція реалізується швидке. Мова йде про алгебру кінцевих полів Галуа, в якій визначена операція експоненціювання $A|^{E}$ rem P для обраного утворюючого поліному P(x);
- залучити до обчислень віддалені комп'ютерні системі з використанням розвинений хмарних технологій. Така можливість зумовлена тим, що термінальні мікроконтрели систем комп'ютерного управління на бізі технології

IoT обладнані вбудованими радіо модемами, які дозволяють здійснити підключення до мережі Інтернет.

Якщо перейти до експоненціювання на полях Галуа можна значно скоротити обсяг обчислень і прискорити формування та перевірку цифрового підпису. Можна побачити, що в механізмі цифрового підпису можна виділити дві групи операцій - це операції над показниками ступеня та операції зведення в ступінь. Операції першого типу на порядки менш трудомісткі і в запропонованій модифікації виконуються у звичайній арифметиці. А операції модулярного зведення в ступінь замінюються операціями експоненціювання на кінцевих полях.

На кінцевих полях Галуа адитивні операції представлені лише однією операцією додавання, яка позначається як ’⊕’ і зводиться до побітової реалізації логічної операції логічного додавання, яке відповідає логічній функції "виключне АБО". Редукція на полях Галуа, або знаходження залишку від поліноміального ділення поліному $A(x)$ ступені більшого або рівному n, яке співвідноситься з числом А, на утворюючий поліном $P(x)$, позначається як A rem P з тим, щоб відрізняти цю операцію від модулярної редукції A mod M в традиційній алгебрі. Основним чинником прискорення обчислення експоненти на кінцевих полях Галуа є те, що квадрат числа А на кінцевому полі Галуа GF(2^n) можна представити в вигляді логічної суми його зсунутих двійкових розрядів. При цьому i-тий розряд зсувається на $2 \cdot i$ розрядів. Іншими словами, операція піднесення до квадрату числа A^2 rem P на кінцевих полях Галуа зводиться до вставки нулів між його двійковими розрядами з подальшим віднаходженням поліноміального залишку від ділення на утворюючий поліном поля Галуа.

До теперішнього часу запропоновані варіанти цифрового підпису, які використовують вказані властивості кінцевих полів Галуа [9]. Фактично ці схеми цифрового підпису являють собою модифікації стандартизованих і розглянутих вище алгоритмів формування цифрового підпису з урахуванням особливостей алгебри кінцевих полів Галуа. В літературі [9] є посилання на те,

що такі схеми дозволяють прискорити формування цифрового підпису на порядок при використанні модифікованої редукції Монтгомері. Проте аналіз показує, що реалізація експоненціювання на кінцевих полях Галуа з використанням в якості обчислювальної платформи малопотужних термінальних мікроконтролерів не забезпечує можливості кардинального прискорення, яке б дозволила реалізувати криптографічні механізми цифрового підпису в режимі реального часу. Це зумовлено тим, що архітектура сучасних мікроконтролерів погано пристосована для виконання операцій специфічної алгебри кінцевих полів Галуа. Зокрема, нема можливості застосовувати вбудовані команди множення цілих чисел для реалізації множення на полях Галуа.

1.2. Аналіз підходів до прискорення виконання модулярного експоненціювання.

Важливою задачею підвищення ефективності реалізації криптографічних механізмів захисту даних з відкритим ключем в реальному часі є пошук резервів прискорення операцій модулярного експоненціювання.

Обчислення модулярної експоненти $A^E \bmod M$ виконується за одним із двох класичних алгоритмів [10]. Обидва ці алгоритми організовані у вигляді n циклів, у кожному з яких обробляється один поточний розряд коду експоненти E. Вказані алгоритми модулярного експоненціювання відрізняються порядком обробки розрядів експонентів. В алгоритмі, в якому розряди E обробляються, починаючи з молодшого, передбачено використання двох змінних D і R, в які до початку циклу заносяться значення $D=A$ і $R=1$. У кожному n із циклів, якщо поточний розряд експоненти E дорівнює одиниці, то виконується модулярне множення $R=R\cdot D \bmod M$ і завжди виконується модулярне зведення до квадрату $D=D^2 \bmod M$. В алгоритмі модулярного експоненціювання зі старших розрядів коду експоненти E використовується лише одна змінна R, початкове значення якої дорівнює одиниці: $R=1$. У кожному n із циклів над цією змінною виконується операція модулярного зведення до квадрату: $R=R^2 \bmod M$ і, якщо

поточний біт коду експоненти E дорівнює одиниці – здійснюється модулярне множення на A: $R = R \cdot A \bmod M$.

Очевидно, що обидва алгоритми мають строго послідовний характері і не можуть бути розпаралелені глобально. Проте алгоритм модулярного експоненціювання з молодших розрядів коду експоненти може потенційно бути організований у вигляді двох гілок: в рамках однієї з них здійснюється обчислення і зміна змінної D, а в рамках другої - здійснюється обчислення і зміна змінної R. Обчислювальне навантаження на кожну з гілок не рівне, адже операція модулярного піднесення до квадрату виконується в кожному циклі, а операція модулярного множення – в кожному другому циклі.

Відповідно, основним резервом їхнього прискорення є зменшення часу виконання їх базових операцій. Для обох алгоритмів середня кількість операцій модулярного множення дорівнює $1.5 \cdot n$ [1].

Аналіз алгоритмів обчислення модулярної експоненти показує, що існують певні резерви їх прискорення незважаючи на те, що вони мають послідовних характер і не можуть бути розпаралелені. Одним із таких потенційних резервів є те, що 2/3 складових операцій модулярного експоненціювання по обох алгоритмах посідає модулярне зведення до квадрату. Тому актуальним напрямком прискорення комп'ютерної реалізації операції модулярного експоненціювання є пошук резервів прискорення модулярного зведення до квадрату. Добре відомо, що ця операція має певну операційну надлишковість, яка потенційно можу бути усунена і за рахунок цього сама операція піднесення до квадрату (без редукції результату) може виконуватися вдвічі швидше. Окремо може бути розроблена процедура модулярного піднесення до квадрату. При виконанні цієї операції існує можливість зменшення об’єму обчислень за рахунок виключення дублювання процесорних множень симетричних секцій множимого та множника.

Тому дослідження способів усунення операційної надмірності при модулярному піднесенні до квадрату є важливою та актуальною для забезпечення безпеки систем віддаленого управління на базі технологій IoT.

Базова операція модулярного експоненціювання – модулярне множення *n*-розрядних чисел складається з двох під операцій: власне множення *n*-розрядних чисел та модулярної редукції, тобто обчислення залишку від ділення отриманого добутку на модуль. В обчислювальному плані друга з наведених операцій значно складніша. Метод прискоренного множення Карацуби [3] дозволяє при організації множення чисел, які складаються з s окремих секцій зменшити кількість процесорних множень з s^2 до. $s^{\log_2 3} = s^{1.585}$. Для організації модулярного редукції використовуються три технології:

- метод Баретта [4], в якому операція модулярної редукції здовиться до двох операцій модулярного множення *n*-розрядних чисел. Відповідно, при використанні цієї технології операція модулярного множення зводиться до виконання трьох операцій звичайного множення та низки адитивних операцій над 2·*n*- розрядними числами. суттєвого переваго використання технології Баретта є те, що можна повною мірою задіяти апартні засоби вбудованої операції множення цілих чисел. Це, як показали дослідження, забезпечує прискорення виконання множення в 10-15 раз в порівнянні з бітовим множенням. Недоліком технології є те. що послідовно виконуються операції множення з устовореням чисел подвійної розрядності, що збільшує обчислювальну складність її реалізації на мало розрядних вбудованих термінальних мікроконтролера;

- метод Монтромері [5], який передбачає побітове множення, яке поєднується з редукцією. на кожному кроці оброблюється один біт множника. Відповідно, до суми часткових добутків додається або не додається множене. Наступний крок полягає в виконанні редукції, яка виконується зсувом чум часткових добутків праворуч. В залежності від значення молодшого біту суми здійснюється корекція – додавання до суми часткових добутків коду модуля.

- метод використання для редукції таблиць передобчислень [6], які дозволяють акумулювати в собі результати, що залежать лише від модуля, який для переважної більшості систем криптографічного захисту систем управління віддалними об’єктами реального світу є незмінним.

Організація множення відразу на декілька розрядів множника є традиційним підходом до прискорення множення. Цілком виправданим може розглядатися такий підхід стосовно модулярного множення і модулярного експоненціювання.

Основним чинником прискорення при обробці декількох розрядів множника виступає використання передобчислень, в рамка яких попередньо обраховується доданок для суми часткових добутків для всіх можливих 2^k значень групи з k бітів множника.

1.3. Аналіз методів розподіленого обчислення модулярної експоненти з залученням хмарних технологій.

Здійснення операції модулярного експоненціювання $A^E \bmod M$ на термінальному мікроконтролері з використанням хмарних технологій має бути організоване таким чином, щоб практично унеможливити реконструкцію секретних компонентів A та E за даними, що передаються на віддалену комп'ютерну систему, а також прискорити обчислення для реалізації цифрового підпису в режимі реального часу [10].

Згідно з викладеним вище, чисельними компонентами операції модулярного експоненціювання $A^E \bmod M$ в криптографічних застосуваннях виступають: інформаційна складова A, закритий ключ E та значення модуля M, який є частиною відкритого ключа. Це означає, що рівень секретності кожної із розглянутих компонентів суттєвим чином різниться. Очевидно, що модуль M не є секретним і може передаватися в хмару в відкритому вигляді. Разом з тим, обробка секретного ключа E на віддалених непідконтрольних і тому потенційно небезпечних комп'ютерних системах вимагає значно вищого рівня захищеності, ніж значення A. Це зумовлено тим, що закритий ключ E використовується багатократно протягом тривалого часу, тобто отримання до нього доступу надає можливості для його подальшого несанкціонованого використання з метою підробки даних на протязі тривалого часу. Разом з тим інформаційна компонента A змінюється в кожному сеансі шифрування даних і тому швидко

втрачає актуальність. Крім того, для багатьох застосувань, зокрема для цифрового підпису, інформаційна компонента *A* використовується у відкритому вигляді, зокрема в механізмах цифрового підпису пакетів даних з термінальних мікроконтролерів. Виходячи з викладеного, очевидно,що основні зусилля на забезпечення секретності мають концентруватися на виключення практичної можливості відновлення секретного коду експоненти E на віддалених комп'ютерних системах, які здійснюють їх безпосередню обробку.

Ефективність захищеної реалізації модулярного експоненціювання з залученням віддалених комп'ютерних систем в рамках хмарних обчислень визначається двома базовими критеріями [11]:

- рівнем захищеності, який оцінюється обсягом ресурсів, які потрібно витратити для реконструкції секретних компонентів операції модулярної експоненти за кодами, що передаються у відкриту хмару і далі на обчислювальну платформу, яка здійснює безпосереднє обчислення модулярної експоненти;
- прискоренням обчислювальної реалізації операції модулярного експоненціювання за рахунок залучення хмарних обчислень тобто організації розподілених обчислень, порівняно з виконанням цієї операції тільки на термінальному мікроконтролері системи комп'ютерного управління об'єктами реального світу.

На практиці рівень підвищення продуктивності найпростіше оцінити шляхом підрахунку кількості операцій модулярного множення над *n*-розрядними числами. Класичні алгоритми модулярного експоненціювання полягають в послідовній обробці кожного з *n* розрядів коду експоненти E. Обробка одного розряду складається з обов'язкової операції модулярного піднесення до квадрату попереднього результату і модулярного множення результату на A, якщо поточний біт коду експоненти дорівнює одиниці. Якщо припустити приблизно рівну кількість одиниць і нулів в коді експоненти E, що справедливо при великих значеннях *n*, то очевидно, що середня кількість операцій модулярного множення становить $1.5 \cdot n$.

Виходячи з того, що операція модулярного експоненціювання можна розглядати з точки зору теорії як частково гомоморфною в частині множення, практично всі відомі методи захищеної реалізації цієї операції базуються на представлення секретного коду експоненти E у вигляді адитивної комбінації компонентів $v_1,v_2,\ldots,v_l$: $E = v_1+v_2+\ldots+v_l$, частина яких оброблюється на віддалених комп'ютерних системах, а частина – на термінальних мікроконтролерах [13].

Для захисту числової компоненти A операції модулярної експоненти використовуються два підходи. Перший з них пов'язаний з використанням спеціального множника, який пов’язаний з модулем і самоліквідується при виконанні піднесення до ступеню *E*. Зазначений підхід передбачає віднаходження спеціальної пари цілих чисел *G* та *R* таких, що $G^R \bmod M = 1$. В більшості протоків мережевої безпеки, в тому числі алгоритму формування та перевірки цифрового підпису модуль *M* утворюється у вигляді добутку двох простих чисел *p* та *q*. Наприклад, якщо $p=59$ а $q=11$, то $M=649$. Відповідно в рамках цього прикладу можуь бути обрані числа $G=129$ и $R=116$, такі, що $129^{116} \bmod 649=1$. Підбір чисел *G* та *R* більш ефективно може здійснювати на основі малої теореми Ферма [14].

При цьому ця задача може бути вирішена за рахунок поділення розподіленого обчислювального процесу модулярного експоненціювання на три фази:

- виконання попередніх обчислень на термінальному мікроконтролері з формуванням спеціальних кодів *G*, *U*, що залежать від *A* та *E*, але так, що значення останніх не можуть бути отримані на основі наданих системі значень G і *U*;

- передачі кодів *G* і *U* до протилежного учасника інформаційної взаємодії, який здійснює обчислення $R= G^U \bmod M$ з використанням хмарних обчислень і повертає результати на термінальний мікроконтролер системи комп’ютерного управління віддаленими об’єктами реального світу;

- здійснення гомоморфного дешифрування на термінальному мікроконтролері, в результаті яких з отриманого коду R формується коректний результат обчислення $A^e \bmod M$.

При цьому, рівень захищеності оцінюється складністю відновлення секретних кодів А та E при наявності у розпорядженні зловмисника кодів G та U. Обчислювальна ефективність організації віддаленого обчислення модулярної експоненти оцінюється через співвідношення складності обчислень, що здійснюються на другій фазі та сумарної складності обчислень, що реалізуються на першій та другій фазах.

Усі запропоновані на цей час методи захищеного модулярного експоненціювання можна розділити на два класи. До першого належать методи, які виходять із того, що відомі параметри генерації криптосистеми з відкритим ключем [27]. Відповідно, другий клас складають методи, які припускають, що такі параметри невідомі. Для цих методів функція φ гомоморфного шифрування компоненти закритого ключа Е має вигляд адитивного або адитивно-мультиплікативного розкладання коду E. У найпростішому випадку адитивно-мультиплікативного розкладання коду E має вигляд $E = (L + Q) \cdot Y$.

З іншого боку, рівень захисту від незаконного відновлення кодів A та E на віддаленій системі визначається обсягом перебору можливих значень компонент Q та Y, який дорівнює. Це означає, що у відомих методах тимчасова ефективність використання хмарних обчислень жорстко обмежена рівнем захищеності.

Теоретично захист інформаційної компоненти А може бути виконано за рахунок її адитивного або мультиплікативного перетворення. Аналіз показує, що застосування адитивних перетворень неефективно в силу того, що це призводить до представлення експоненти у вигляді біному Ньютона і, відповідно, має наслідком суттєве зростання обчислення обчислювальної складності. В цьому плані більш перспективним є використання мультиплікативних перетворень. Технологічно такі перетворення A можуть бути виконані в наступних формах:

- розкладання A на множники, тобто представлення $A = A_1 \cdot A_2 \cdot \ldots \cdot A_p$. Цей варіант видається доволі складним;

- накладання на інформаційну компоненту A мультиплікативної маски F в формі добутку : G = A·F. Таким чином, обчислення модулярної експоненти проводяться з кодом G. Отримання з цього коду істинного значення A практично неможливо, оскільки це класична важкорозв'язувана задача теорії чисел: розкладення числа на прості множники [14]. Тобто, при використанні мультиплікативної маски умова неможливості відновлення справжнього значення секретного коду A виконується. Для зняття мультиплікативної маски може бути використано результат передобчислень H= F^E mod M. Іншим варіантом, більш ефективним в плані обчислювальної складності може розглядатися вибір маски F таким чином, щоб вона взагалі не впливала на кінцевий результат, тобто щоб виконувалася умова $F^E \bmod M = 1$. Теоретично це можливо за умови, що маска може бути розкладена на певні множники, по відношенню до яких код M є кратним [15].

У роботі [16] запропоновано метод захищеного модулярного експоненціювання у хмарах на основі випадкового поділу коду експоненти E на адитивні компоненти. Це дозволяє організувати обчислення модулярної експоненти $A^E \bmod M$ у вигляді модулярного добутку часткових експонентів, які можуть обчислюватися незалежно на віддалених обчислювальних потужностях хмар. Формування кінцевого результату $A^E \bmod M$ шляхом модулярного множення результатів часткових обчислень модулярної експоненти виконується на термінальному мікроконтролері. Перевагою цього підходу є використання можливостей організації великої кількості паралельних обчислювальних процесів в хмарних системах. З іншого боку, цілком очевидно, що значна частина обчислень фактично дублюється, що знижує ефективність такого рішення. В роботі [17] викладений метод адитивного розкладення секретного коду експоненти комбінується з мультиплікативним захистом інформаційної компоненти A.

Теоретично та експериментально доведено, що метод дозволяє приблизно втричі підвищити продуктивність виконання модулярного експоненціювання. Для виключення можливості відновлення секретного коду експоненти на віддалених системах застосовується адитивне її розкладення на дві компоненти, одна з яких оброблюється на термінальному мікроконтролері, а інша – на віддаленій системі. Рівень захищеності, при цьому визначається кількістю розрядів коду експоненти, які оброблюються на термінальній обчислювальній платформі. Цим чинником обмежується часова ефективність використання хмарних технологій в рамках викладеного методу.

Інший відомий метод [17] захищеного модулярного експоненціювання з використанням віддалених комп'ютерних систем передбачає знаходження пари чисел G і R таких, що G^RmodM =1. При обчисленні модулярної експоненти A^E mod M спочатку обчислюється значення B=G·A mod M. таким чином здійснюється гомоморфне шифрування інформаційної компоненти A. Випадковим чином вибирається число *W*, розрядність якого значно менше розрядності *n* і обчислюється *V=E-W*. Числа *B*, *V*, *M* передаються в хмару, де обчислюється $C=B^V$ mod M і отримане значення повертається на термінальний пристрій, де обчислюється $F=B^{R-V}$mod M та $D=A^{E-R}$ mod M. Після отримання з хмари числа C результат остаточний експоненціювання обчислюється у вигляді: *R=C·F·D* mod M. Таким чином, у цьому методі захист секретного коду експоненти E досягається за рахунок її адитивного розкладення у вигляді E = V + R – V + E – R, у якому лише дві останні компоненти передаються у хмару. Захист інформаційної компоненти A досягається шляхом того, що в хмару передається модульний добуток цієї компоненти на секретний ключ B.

У роботі [18] запропоновано метод захищеного обчислення модулярної експоненти в хмарах на основі поетапного розкладання компоненти A. Такий підхід дозволяє розкласти завдання обчислення AE mod M на ряд експонентів, які оперують з модифікованими даними і можуть обчислюватися паралельно на віддалених обчислювальних потужностях. Метод дозволяє приблизно у 3-4 рази прискорити процес модулярного експоненціювання. Ще один цікавий підхід

запропонований у роботі [20].Основний акцент у цих дослідженнях зроблено на тому, щоб користувач міг не тільки віддалено виконати модулярне експоненціювання в закритому режимі, але й опосередковано контролювати правильність виконаних операцій.

Прискорення β обчислення модулярної експоненти визначається співвідношенням розрядності n експоненти E до сумарної розрядності r адитивних компонентів експоненти, що обчислюється на термінальному пристрої: $\beta = n/r$. При цьому обсяг ресурсів для реконструкції секретного коду E зловмисником пропорційний 2^r.

Один із можливих варіантів ефективного вирішення компромісу між підвищенням рівня захищеності та прискоренням обчислення модулярної експоненти за рахунок збільшення кількості розрядів коду експоненти, що оброблюються на термінальному мікроконтролері полягає в зменшенні на нього обчислювального навантаження. Варто зазначити, що обчислювальна потужність віддаленої комп'ютерної системи значно перевищує можливості термінального мікроконтролера, Це може бути досягнуто шляхом розділення обчислювального процесу на дві частини, одна з яких, безпосередньо пов'язана з секретними розрядами коду експоненти, оброблюється на термінальній обчислювальній платформі, а інша – на віддалених комп'ютерних платформах. У відомих рішеннях такий розподіл обчислювального процесу модулярного експоненціювання реалізується лише на рівні циклів модулярного експоненціювання. Очевидно, що можуть існувати такі схеми розподілення обчислень, при застосуванні яких термінальний мікроконтролер виконує операції, лише безпосередньо пов'язані з секретними кодами.

Характерною особливістю розглянутих методів розподіленого обчислення модулярної експоненти є те, що обмін даними між віддаленими системами та термінальним мікроконтролером відбувається лише один раз: з хмари передаються результати обчислення часткової модулярної експоненти. При цьому невикористаним резервом підвищення ефективності є організація більш широкого обміну даними. Це, в перспективі дозволить зменшити

обчислювальне навантаження на термінальний мікроконтролер і за рахунок цього досягти кращої часової ефективності розподілених обчислень.

Висновки до розділу 1.

За результатами виконання досліджень, які складають перший розділ магістерської дисертації і присвячені аналізу задачі швидкої реалізації криптографічних алгоритмів з відкритим ключем на термінальних пристроях систем комп'ютерного управління на базі технології Інтернету речей, а також огляду відомих методів її вирішення можна зробити наступні висновки:

1. Використання технологій Інтернету речей, яке базується на застосування в якості середовища обміну даними Інтернет дозволяє значно підвищити ефективність широкого класу систем моніторингу стану об'єктів реального світу, а також управління ними. Разом з тим, використання потенційно вразливого середовища глобальних мереж створює загрози стороннього втручанні в роботу згаданих систем. Для її вирішення потрібно застосовувати повний спектр засобів криптографічного захисту даних, передбаченими протоколами мережевої безпеки. Частина цих засобів базується на складних в обчислювальному плані операціях модулярного експоненціювання, що виконуються над числами, розрядність яких на порядки перевищує розрядність термінальних комп'ютерних платформ систем комп'ютерного управління. Ці платформи, виходячи із специфіки їх використання, мають низьку продуктивність, зумовленою вимогам до низького споживання потужності, компактності, малої вартості. Відповідно, об'єктивно виникають труднощі реалізації складних обчислень модулярного експоненціювання на таких платформах в режимі реального часу.

2. Проведений аналіз існуючих алгоритмів модулярного експоненціювання засвідчив, що обидва їх варіанти мають строго послідовний характер і не можуть бути розпаралелені. Огляд існуючих підходів до прискорення виконання базової операції модулярного експоненціювання показав, що вони базуються на використанні передобчислень, які потребують значних за обсягом об'ємів пам'яті, що недопустимо для компактних термінальних мікроконтролерів. За

цієї ж причини не можна використовувати спеціалізовані потужні криптопроцесори, які на апаратному рівні реалізують операцію модулярного експоненціювання. Виходячи з цього, найбільш доцільне рішення полягає в залученні віддалених обчислювальних потужностей в рамках хмарних технологій.

3. Огляд існуючих методів обчислення модулярної експоненти з залученням віддалених обчислювальних потужностей показав, що в свої більшості вони організовані у вигляді двох обчислювальних процесів, один із яких реалізується на термінальній обчислювальній платформі, а інший на віддаленій. Для виключення можливості відновлення секретного коду експоненти на віддалених системах застосовується адитивне її розкладення на дві компоненти, одна з яких оброблюється на термінальному мікроконтролері, а інша – на віддаленій системі. Рівень захищеності, при цьому визначається кількістю розрядів коду експоненти, які оброблюються на термінальній обчислювальній платформі. Цим чинником обмежується часова ефективність використання хмарних технологій.

Відповідно, для підвищення ефективності залучення віддалених комп’ютерних систем для прискорення виконання на термінальній обчислювальній платформі модулярного експоненціювання потрібні нові організації розподілення обчислень, які дозволяють зменшити обчислювальне навантаження на термінальний мікроконтролер.

РОЗДІЛ 2
РОЗРОБКА МЕТОДУ РОЗПОДІЛЕНОГО ОБЧИСЛЕННЯ МОДУЛЯРНОЇ ЕКСПОНЕНТИ З БЕЗПЕЧНИМ ЗАЛУЧЕННЯМ ВІДДАЛЕНИХ СИСТЕМ

Здійснений в попередньому розділі аналіз відомих підходів до розподіленого обчислення базової операцій несиметричної криптографії – модулярного експоненціювання з використанням віддалених комп'ютерних потужностей показав, що вони орієнтовані на адитивне розкладення коду експоненти, який в реальних системах являє собою закритий, тобто секретний, ключ термінальних пристроїв IoT. Використання адитивного розкладення коду експоненти дозволяє значною мірою спростити обчислювальні процедури при розподіленні обчислень. Разом з тим, їх використання надає більше інформації потенційному зловмиснику для порушення захисту, в силу того, що частина розрядів коду експоненти надаються в його розпорядження при здійсненні віддалених обчислень. Крім того, характерною рисою відомих методів організації розподіленого обчислення модулярної експоненти є те, що обмін даними між термінальною обчислювальною платформою та віддаленими комп'ютерними системами здійснюється на початку та в кінці процесу, тобто обмін даними в процесі обчислювального процесу між вказаними платформами практично відсутній. Це спрощує зловмиснику незаконне відновлення коду експоненти за даними, що надіслані на віддалену комп'ютерну систему. Відповідно, подальше підвищення ефективності може бути досягнуто на шляху ускладнення перетворень над кодом експоненти при формуванні часткових обчислювальних процесів.

Відповідно, мета дослідження полягає в підвищенні ефективності використання хмарних обчислень для прискорення реалізації криптографічних механізмів з відкритим ключем на термінальних комп'ютерних платформ систем віддаленого моніторингу та управління на базі технологій IoT.

Для досягнення поставленої мети потрібно проаналізувати різні можливості організації розподіленого обчислення модулярної експоненти на термінальній та віддалених комп'ютерних платформах, які б забезпечували мінімальний час обпилення та виключали можливість відновити секретний код експоненти на віддалених системах, за даними що їм надаються.

На основі такого аналізу потрібно вибрати найбільш ефективний варіант організації розподілених обчислень модулярної експоненти, детально розробити його в вигляді методу, придатного для практичного використання, а також здійснити теоретичне та експериментальне дослідження його ефективності по критеріям прискорення обчислень та рівню захищеності від спроб відновлення секретного коду експоненти на віддалених і потенційно непідконтрольних комп'ютерних системах, які залучаються для прискорення обчислень.

2.1. Аналіз можливостей організації розподіленого обчисленнямодулярної експоненти з використанням хмарних технологій.

Відомо [12], що обчислення модулярної експоненти $A^E \bmod M$ над числами, розрядність n яких на порядки перевищує розрядність r процесору може здійснюватися за одним з двох класичних алгоритмів. Обидва алгоритми передбачають послідовний аналіз розрядів коду експоненти E і відрізняються порядком, в якому цей аналіз здійснюється. В алгоритмі з аналізом розрядів коду експоненти починаючи зі старших, на кожному із n циклів виконується модулярне піднесення до квадрату поточного результату R, початкове значення якого дорівнює одиниці: R=1. Крім того, в залежності від значення поточного розряду коду експоненти E здійснюється модулярне множення поточного результату R на число A: $R = R \cdot A \bmod M$

Теоретично можливо здійснити змішаний варіант алгоритму модулярного експоненціювання, при якому аналіз проводиться одночасно зі старших і з молодших розрядів коду експоненти. Наприклад, обчислюється $5^{13} \bmod 19=17$. Код експоненти, в яку модулярно підноситься число A=5 дорівнює: E =

$13=1101_2$ На першому кроці реалізації змішаного алгоритму виконується операція: R=A=5, D=5, P=5. На другому кроці здійснюються операції $R=R^2 \bmod M = 25 \bmod 19 = 6$ та $D=D^2 \bmod M = 6$.

Згідно з викладеним вище, чисельними компонентами операції модулярного експоненціювання $A^E \bmod M$ виступають: інформаційна складова *A,* закритий ключ *E* та значення модуля *M*, який є частиною відкритого ключа криптосистеми. Цілком ясно, що обробка ключа *E* на віддалених непідконтрольних і тому потенційно небезпечних обчислювальних потужностях вимагає на порядки більш високого рівня захисту в порівнянні з числовою компонентою *A*. Це зумовлено тим, що приватний ключ термінального пристрою *E* використовується багатократно протягом тривалого часу, тобто його незаконне відновлення при здійсненні хмарних обчислень надає можливості для його подальшого несанкціонованого використання з метою підробки даних. Натомість інформаційна компонента *A* змінюється в кожному сеансі обміну даними між термінальним мікроконтролером і ядром системи. Крім того, для багатьох застосувань інформаційна компонента *A* використовується у відкритому вигляді, зокрема в механізмах цифрового підпису пакетів даних з термінальних мікроконтролерів.

Ефективність реалізації модулярного експоненціювання з використанням хмарних обчислень визначається двома чинниками:

- прискоренням обчислення модулярної експоненти на термінальному мікроконтролері за рахунок залучення віддалених обчислювальних потужностей;

-рівнем захищеності від незаконної реконструкції секретного ключа за даними, що надаються на віддалену комп’ютерну систему.

Найбільший ефект в плані прискорення обчислення модулярної експоненти із залученням хмарних технологій теоретично може бути досягнений за рахунок:

- найбільшого суміщення в часі обчислювальних процесів на віддалених комп’ютерних системах та термінальному мікроконтролері;
- ефективного використання можливостей організації декількох паралельних процесів на віддалених комп’ютерних системах.

Як було показано в оглядовому розділі, найбільш часто для захисту від несанкціонованого відновлення секретного коду експоненти використовується адитивне розкладення коду експоненти. При такому розкладенні частина більша адитивних компонентів оброблюється на віддалених обчислювальних потужностях. При традиційних способах розкладення коду експоненти на адитивні компоненти алгоритм виконання експоненціювання не відіграє суттєвої ролі в силу того, що кожен із вузлів, які приймають участь в розподіленому обчисленні виконує повний набір операцій обробки одного розряду коду експоненти.

Проведений аналіз показав, що рівень захищеності обчислень модулярної експоненти з використанням можливостей віддалених обчислювальних процесів визначається кількістю розрядів секретного коду експоненти, обробка яких здійснюється на термінальному мікроконтролері. Іншими словами, для підвищення рівня захищеності потрібно, щоб якомога більше розрядів секретного коду експоненти не надсилалися в хмару, а оброблювалися на термінальному мікроконтролері. Природним обмеженням кількості розрядів коду експоненти, які оброблюються на термінальному мікроконтролері виступає його низька продуктивність, зумовлена тим що він має бути компактним, дешевим та мати низьке споживання потужності.

Один із найбільш перспективних напрямків підвищення ефективності залучення хмарних технологій для обчислення модулярної експоненти на термінальних мікроконтролерах систем комп’ютерного управління вирішення проблеми підвищення рівня захищеності за рахунок збільшення кількості розрядів коду експоненти, що оброблюються на термінальному мікроконтролері, без втрати швидкодії, полягає в зменшенні на нього обчислювального навантаження. Це може бути досягнуто шляхом розділення

обчислювального процесу на дві частини, одна з яких, прямо пов'язана з секретними розрядами коду експоненти, оброблюється на термінальному мікроконтролері, а інша – в хмарі. В методах розподіл обчислювального процесу модулярного експоненціювання багаторозрядних чисел здійснюється на рівні циклів обробки розрядів коду експоненти. Це означає, що частина із n циклів обчислення модулярної експоненти реалізується на термінальному мікроконтролері, а інша – на віддалених комп'ютерних системах. В відомих рішеннях здебільшого застосовується таке розподілення обчислень в яким молодші (або старші, в залежності від типу алгоритму модулярного експоненціювання) розряди коду експоненти оброблюються на термінальному мікроконтролері, а старші – на віддалених обчислювальних потужностях. Значення h –кількості розрядів коду експоненти, яки оброблюються на термінальному мікроконтролері вибирається при цьому таким чином, щоб досягти найбільшого суміщення в часі роботи термінальних та віддалених комп'ютерних платформ і забезпечити при цьому рівень захищеності, визначений специфікою застосування системи віддаленого управління на базі технології Інтернету речей. Якщо позначити через q кількість розрядів коду експоненти, які оброблюються на термінальній обчислювальній платформі, і значення яких не передається на віддалені комп'ютерні системи, яка визначається потрібним для конкретного застосування рівнем захищеності, то $h<q$.

Одним із простих способів реалізації безпечного обрахування модулярної експоненти є застосування розбиття коду експоненти на випадкові доданки. Для цього код експоненти E перетворюється на T=E- ξ, де ξ - випадкове мале число. Після цього код T розбивається на h випадкових доданків δ0, δ1,…,δh-1 так, що $T = \delta 0, + \delta 1 +\ldots + \delta_{h-1}$. Якщо код A не є таємним (для прикладу, в протоколах ідентифікації віддалених користувачів, описаних у першому розділі) то процес віддаленого обчислення складається з таких кроків:

1. Користувач вигадує випадкове число ξ та перетворює код експоненти E за формулою T=E- ξ.

2. Користувач розбиває код T на h випадкових доданків $\delta_0, \delta_1,\ldots,\delta_{h-1}$.
3. Користувач надсилає в систему код A та масив $\delta_0, \delta_1,\ldots,\delta_{h-1}$.
4. Система виконує:

$$r_0 = A^{\delta_0} \bmod N\,,\ r_1 = A^{\delta_1} \bmod N$$

5. Користувач паралельно виконує $D = A^{\xi} \bmod N$
6. Користувач отримує результат.

Існує спеціальний випадок розбиття коду експоненти на компоненти, який полягає в тому, що код поділяється на розряди, які складаються з $n_0,n_1,\ldots,n_{h-1}$ бітів так, що $n_0+ n_1+\ldots+n_{h-1}= n$. Група δ0 складається з перших n0 бітів коду степеня , група δ1 складається з наступних n1 розрядів: і так далі. Тоді, розряди групи δ0 формують значення g0, розряди підгрупи δ1 формують значення g1, розряди підгрупи δ_{h-1} формують значення g_{h-1}:

$$\forall l \in \{0,1,\ldots,n-1\}: g_l. \qquad (2.1)$$

Отже, код степеню *E* перетворюється на доданки [20]:

$$E = g_0 + g_1 \cdot 2^{h_0} + g_2 \cdot 2^{h_0+h_1} = \sum_{l=0}^{m-1} g_l \cdot 2^{\sum_{j=0}^{l-1} n_j}$$

Перезначивши змінні в формулі на вигляд w_0=1, $w_1 = 2^{n_0}$, $w_2 = 2^{n_0+n_1}$,..., $w_{h-1} = 2^{n_0+n_1+n_2+\ldots+n_{h-2}}$, тоді степінь *E* можливо записати у значно коротшому та інтуітивнішому записі:

Операцію модулярного експоненціювання $A^E \bmod M$ можна представити у вигляді множників, використовуючи розкладання коду експоненти на адитивні складові та випадкові трансформації:

$$A^E \bmod M = ((A^{g_0} \bmod M)^{w_0} \cdot (A^{g_1} \bmod M)^{w_1} = (\prod_{l=0}^{h-1} (A^{g_l} \bmod M)^{h_l}) \bmod M$$

де δ_0 ,δ_1, адитивні складові, які отримані випадковим чином під час розкладання коду експоненти, (A) - код, що не є секретним, (M) - модуль.

Ці добутки можна рахувати на відстані, без залежностей один від одного на віддалених компютерних ресурсах без надсилання їм прихованих ключів A та

E. З цією метою запропонована така логіка обчислення модулярної експонентації A^E mod M.

1. Користувач рахує R_0, R_1,..., R_{h-1}
2. Розраховані числа R1,…,Rh-1 і w1,…,wh-1 користувач відправляє в віддалену систему.
3. Система одночасно рахує $D_1 = R_1^{h_1} \, mod \, M$, $D_2 = R_2^{h_2} \, mod \, M$..., $D_{h-1} = R_{k-1}^{h_{k-1}} \, mod \, M$ після цього розраховані значення повертаються назад користувачеві.
4. Користувач рахує кінцевий результат у такому формулюванні

$$A^E \, mod \, M = (R_1 \cdot \prod_{j=1}^{k-1} D_j \, mod \, M) \, mod \, M. \qquad (2.2)$$

Пропонована логіка обрахування модулярного степеню на віддалених серверних ресурсах може бути продемонстровано таким чином для A, E і M невеликої розрядності. Нехай модуль M=143, A=96. E= 9310 = 10111012. A^E mod M = 9693mod 143 = 83. Розраховані числа R_1=138, R_2=64 а також числа w1= 8, w2= 32 відправляються на сторонні сервери. Там одночасно рахуються D_1 та D_2, які надсилаються відправнику. Користувач рахує кінцевий результат.

Для обчислення модулярної експоненти A^E mod M за традиційною логікою [12] потрібно зробити n кроків, при кожному з яких виконується модулярне піднесення до другого степеню та модулярне мультиплікування на A. При умові, що нинішній біт коду степеню рівний 1. Таким чином, середнє число N_0 виконання модулярного множення дорівнює N_0= 1.5·n.

Для обчислення модулярної експоненти за такою логікою, середня арифметична розрядність кожної з h груп складає n/h. Якщо розрядність усіх груп рівнозначна, то користувач повинен зробити лише n/h операцій модулярного мультиплікування. Однак, це значно спростить потенційному хакеру відтворення коду експоненти. При рандомному обранні кількості елементів в групі, число операцій модулярного множення для розрахунку R_0, R_1,…,R_{h-1} в середньому дорівнює n *1/2

Пропонована організація обчислення модулярної експоненти на віддалених серверних ресурсах має ту перевагу, що дуже важко відтворити код експоненти E, який є насправді секретним ключем криптосистеми RSA. Хакер, який знає відкритий секрет D криптосистеми RSA. Теоретично, він може відновити код E, якщо він дізнається невідомі йому компоненти розбиття коду експоненти.

Враховуючи той факт, що сума розрядностей тільки чисел p і q рівна розрядності n модуля M, тож легко побачити, що потрібна кількість спроб перебору дорівнює кількості проб для злому криптосистеми RSA, що реально застосовуваних в даний час значеннях n=2 * 1024 і більше недосяжний технічно. Інакше кажучи, кількість обчислювальних ресурсів для перемоги і викриття пропонованої схеми забезпесеної імплементації порядку дій RSA на віддалених обчислювальних ресурсах майже рівний, необхідним для викриття самої криптосистеми RSA.

Реальне значення коефіцієнта r_a в кілька порядків менше, ніж n, що дає можливість проігнорувати чисельні значення перших доданків у знаменнику. З іншого боку, у практичних застосуваннях швидкість обчислення модулярної експоненти визначається величиною, оберненою до альфа-співвідношення між времям виконання операцій мультиплікування та додавання, оціненим, згідно з експериментальними даними, як 60. Використання параметрів p і q модуля M прискорює роботу приблизно на 20-30%, порівняно з розкладенням на доданки коду експоненти. Важливо відзначити, що такий підхід досягає вищого рівня захисту і вимагає менших ресурсів для віддаленого виконання модулярного експоненціювання та відновлення секретних елементів операції.

Експериментальне дослідження вражає своєю безапелляційністю: значення коефіцієнта α, що корелюють із якістю Інтернет-підключення, безжально розкидані від 0.01 до 0.004, здебільшого стагніруючи на мізерних 0.015. А вот і вражаючий ефект - застосування цієї технологічної стратегії для захищеного обчислення модулярної експоненти в хмарному середовищі не просто прискорює, але фактично вибухово трансформує виконання

криптографічних протоколів захисту даних на вбудованих термінальних контролерах Інтернету речей, даруючи неймовірний приріст швидкості в середньому на цілих 40 разів.

Але важко уявити, що ця ідеальна концепція завжди працює на практиці. Ключева проблема полягає в тому, що ключі для несиметричних криптосистем, як правило, створюються зовнішніми організаціями, які не розкривають користувачам і замовникам складові розкладу модуля M на прості множники. Така невіддільність від контролю може становити серйозний обмежувальний фактор для широкого впровадження цієї ідеї.

Давайте обчислимо розкладення експоненти за розглянутим методом для наданого прикладу:

1. Задане число: $5^{11} \bmod 19 = 6$

2. Експонента E = 11, d = 1 (вибираємо для прикладу)

3. Розкладення на множники: E = 11 - d = 10. Відповідно, множники a = 5 та b

Обчислення на обчислювальній платформі:

- Визначимо B = 6.

- Визначимо U = A^d mod M = 511^1 mod 19 = 511 mod 19 = 6.

Потім відправляється до хмари:

Результат множиться на U:

Результат = U * (B^b mod M) = 6 * 17 mod 19 = 102 mod 19 = 6.

Отже, обчислення показує, що зазначений метод дозволяє отримати бажаний результат. Однак, як вже зазначалося, ефективність та безпека цього підходу може суттєво залежати від конкретних умов та розмірів використовуваних чисел в реальних криптографічних застосуваннях.

Розглянемо тактику зловмисника. Він знає лише значення a та володіє завданням відновити значення двох параметрів: дільника b і адитивної поправки d. Відомо, що зловмисник може визначити порядок E за кодом a і оцінити його близькість до відомого йому модулю M.

Для визначення порядку E використовуємо властивість

$$E\phi(M) \equiv 1(\mathrm{mod}M)$$- функція Ейлера. [29]

Знаючи близьке значення порядку E, зловмисник може спробувати різні можливі значення b, поки не знайде таке, що вираз відновить порядок E з відомим йому модулем M.

Ця атака показує, що важливо вибирати експоненту E та інші параметри з урахуванням високого рівня складності атак такого типу, які можуть використовувати інформацію про порядок. Також важливо враховувати заходи безпеки при використанні криптографічних алгоритмів, такі як вибір великих простих чисел та інші методи для запобігання підбору ключа.

Обсяг можливого перебору для відновлення параметрів b і d визначається довжиною їхнього представлення у бітовому вигляді, позначеною як l. Обсяг додаткових обчислень також зростає пропорційно збільшенню довжини удвічі — 2^k. Вибір різних значень експоненти E для процесу шифрування дозволяє зловмиснику вірно оцінювати порядок параметра b, що в свою чергу збільшує складність завдання перебору при можливому атакуванні системи.

Для збільшення безпеки можна впроваджувати зміни у значенні адитивної поправки d, змінюючи його знак. Цей підхід може ефективно подвоїти обсяг перебору без витрат додаткових ресурсів часу на термінальному мікроконтролері.

Зміна знаку d додасть додаткову ступінь неспроможності зловмисника в аналізі та атаках на систему. Змінюючи значення d, система стає більш стійкою до атак, що базуються на аналізі та переборі значень параметрів. В такий спосіб, забезпечуючи постійні зміни у знаку d, можна підвищити рівень безпеки криптографічної системи.

Ще однією альтернативою для підвищення рівня безпеки є використання багаторівневих схем розкладення коду експоненти. Значення мультиплікативного множника a подається як $a = g \times h + s$. Процес включає обчислення B на віддаленій системі. Далі, на термінальному мікроконтролері обчислюється C. Після цього на термінальному мікроконтролері обчислюється R. Такий підхід, що використовує багаторівневе мультиплікативно-адитивне розкладення коду експоненти, має великий потенціал підвищити рівень

захищеності системи від спроб підбору на віддаленій комп’ютерній системі. Змінюючи кількість рівнів розкладення, можна гнучко адаптувати систему, ускладнюючи завдання зловмисникам. Зауважимо, що використання багаторівневого мультиплікативно-адитивного розкладення коду експоненти може збільшити обчислювальні витрати на термінальному мікроконтролері, але при цьому значно підвищить стійкість системи до атак.

Захист інформаційної складової, зокрема коду A операції модулярного експоненціювання, є ключовою задачею для забезпечення безпеки термінального мікроконтролера. Один із традиційних методів вирішення цієї задачі, який може ефективно поєднуватися з різними стратегіями захисту коду експоненти, включає множення коду A на спеціально підібране число U. Значення U обирається так, щоб виконувалася умова $U \cdot a \cdot b \bmod M = 1$

Наприклад, якщо $M = 19$ і $E = 11$, тоді a·b=10, і відповідно обирається U=18, щоб забезпечити 18·11mod 19=118·11mod19=1.

Цей метод сприяє додатковій захисту важливого коду від небажаних вторгнень та забезпечує надійність операцій модулярного експоненціювання на термінальному мікроконтролері. Загалом, комбінування цього підходу з іншими заходами безпеки дозволяє створити комплексний механізм захисту конфіденційної інформації в системі.

З використанням прикладу, де $7^{11} \bmod 19 = 11$, процес обчислень на термінальному мікроконтролері та віддаленій комп'ютерній системі можна описати наступним чином:

1. На термінальному мікроконтролері A·Umod19=7·18mod19=12.
2. Значення D=12 та a=5 відправляються на віддалену комп'ютерну систему.
3. Віддалена система обчислює D·amod19=12·^{5}mod19=8 і повертає це значення на термінальний мікроконтролер.
4. На термінальному мікроконтролері обчислюється B·Bmod19=8·2mod19=7
5. Здійснюється останнє множення 7·7mod19=11 і таким чином отримується правильний результат.

Цей процес ілюструє етапи використання спеціально обраного множника U, який дозволяє безпечно виконувати обчислення на термінальному мікроконтролері та передавати результати на віддалену систему, забезпечуючи при цьому надійний захист конфіденційної інформації.

В цілому, в якості мультиплікативної маски U може бути обране будь-яке число. В такому випадку потрібно виконувати всі обчислення як на віддалених комп'ютерних системах, так і на мікроконтролері, використовуючи замасковане значення $D=A \cdot U \bmod M$. Після отримання результату $R'=D^{E} \bmod M$, його необхідно помножити на мультиплікативну інверсію числа $U^{E} \bmod M$.

Наприклад, якщо U = 15, потрібно обчислити $u=U^{a \cdot b} \bmod M$ і знайти мультиплікативну інверсію u^{-1} по модулю M. У даному прикладі, при U = 15 та u = 4, мультиплікативна інверсія останнього по модулю 19 дорівнює 5, тобто u^{-1} = 5.

Отже, на термінальному мікроконтролері попередньо обчислюється $D=A \cdot U \bmod M$, яке надсилається на віддалену комп'ютерну систему. Після повернення обчисленого в хмарі значення на термінальний мікроконтролер, ним обчислюється результат R за формулою $R=B \cdot b \cdot u-1 \cdot A \cdot d \bmod M$

Для даного прикладу, де $D=7 \cdot 15 \bmod 19=10$ і $B=D^{a} \bmod M=105 \bmod 19=3$, термінальним мікроконтролером обчислюється результат $R=32 \cdot 5 \cdot 7 \cdot 11 \bmod 19=7$ $R=32 \cdot 5 \cdot 7 \cdot 11 \bmod 19=7$

Задача перевірки автентичності отриманих керуючих посилок від центральної системи управління для термінальних пристроїв систем комп’ютерного управління віддаленими об’єктами є важливою. У даному сценарії віддаленому мікроконтролеру надсилається повідомлення, підтверджене цифровим підписом. Для перевірки автентичності мікроконтролер повинен виконати операцію модулярного експонентування.

В цьому контексті мікроконтролер відправляє системі хмари підписаний хеш-сигнатуру *H* разом із відкритим ключем системи *E* та модулем *M*, і отримує відповідь $h = H^{E} \bmod M$. Щоб унеможливити можливу заміну значення h з боку

хакера, рекомендується обчислювати h розподілено: частково на мікроконтролері та частково у хмарі.

Цей підхід забезпечує безпеку процесу перевірки автентичності, оскільки важливий етап виконується на самому мікроконтролері, а не повністю у хмарі. Такий розподіл обчислень зменшує ризик можливих атак і забезпечує більш надійний механізм перевірки підпису.

Іншим можливим варіантом організації захисту коду експоненти є його перетворення шляхом конкатенації. Суть цієї ідеї полягає в розділенні коду експоненти на декілька фрагментів, представлених як $E = Ek - 1 \cdot 2k + Ek - 2\ 2k - g + \cdots + E \cdot 2e + E0$У найпростішому варіанті кількість фрагментів дорівнює двом, тобто Такий підхід реалізується шляхом конкатенації фрагментів, тобто $E = W \mid\mid L$

Цей підхід дозволяє розділити обчислення експоненти між віддаленою системою та термінальним мікроконтролером, що може сприяти підвищенню безпеки системи та зменшенню обчислювального навантаження на термінальному пристрої.

Зростання рівня захищеності у такій схемі пояснюється тим, що зловмисник не має інформації про конкретні значення менших розрядів коду експоненти та навіть їх кількість. У цьому контексті більш перспективним може виглядати варіант, коли віддалена комп'ютерна система передає не більші, а менші розряди коду експоненти.

Цей підхід додатково ускладнює завдання зловмисника, оскільки він не має повної інформації про експоненту, зокрема про більш молодші біти. Такий варіант передачі може сприяти збільшенню непередбачуваності та безпеки схеми, ускладнюючи можливі атаки на систему.

Ще один спосіб організації безпечних обчислень модулярної експоненти використовує віддалені обчислювальні потужності. На цих системах виконується обчислення модулярної експоненти, використовуючи схему з молодших розрядів коду експоненти. Основна ідея полягає у розрахунку значень D та R на віддалених комп’ютерних системах. Під час кожного кроку

обчислень значення D підноситься до модулярного квадрату ($D\hat{}2\mathrm{mod}M$), а значення R множиться на обчислене значення D з подальшим модулярним множенням $R = D \cdot R\mathrm{mod}M$, якщо і-тий розряд коду експоненти E дорівнює одиниці ($e\hat{}i = 1$)

На термінальному мікроконтролері випадковим чином формується множина h номерів для обрахунку модулярної експоненти на віддалених обчислювальних потужностях. Ці обчислення виконуються для бітів E з множиною $\Theta = \{n1, n2, \ldots, n\}$ Після цього на основі випадково обраної множини Θ формується множина значень бітів коду експоненти на обраних позиціях $n1, n2, \ldots, nh$: $\Xi = \{\epsilon 1, \epsilon 2, \ldots, \epsilon h\}\Xi = \{\epsilon 1, \epsilon 2, \ldots, \epsilon\ h\}$ Для кожного $l \in \{1,2,\ldots,h\}, \epsilon l \in \{0,1\}$, випадковим чином обирається значення, що відповідає бітам експоненти E. Тут визначається випадково для кожного l і використовується для подальших обчислень модулярної експоненти на віддалених обчислювальних потужностях.

Код експоненти W для обчислень на віддаленій системі формується зі вихідного коду E, але обнуляються розряди, номери яких входять до множини Θ.

Код A перемножується на числове значення U таке, що U^W mod M=1, створюючи шифрований код B. Цей шифрований код B передається в хмару як результат подальшого модулярного множення: B=A·UmodM.

Отримані коди B і W на термінальному мікроконтролері, що були сформовані цим методом, відправляються на віддалену комп'ютерну систему, разом із списком номерів бітів експоненти, які входять до множини Θ. Значення Q на термінальному мікроконтролері встановлюється в нуль: Q=1.

Віддалена обчислювальна система обраховує модулярну експоненту за алгоритмом з менших номерів коду експоненти W. На кожному кроці обрахунку результату D підноситься до модулярного квадрату, тобто D стає рівним D^2modM, і вираз R множиться на це обраховане число D, тобто рахується модулярне множення R=D·RmodM, тоді наступний розряд коду експоненти W

дорівнює 1 (w_i=1). Якщо теперішній крок і є елементом множини Θ, система відправляє на термінальний контролер новий результат обрахованого коду. Термінальний контролер, отримавши від обчислювальної системи код Di при ϵ_i=1, виконує модулярне множення Q=Q·DimodM.

Після завершення обчислень модулярного експоненціювання на віддаленій комп'ютерній системі, іншими словами обрахування $R=B^W modM$, програма висилає термінальному мікроконтролеру кінцевий результат - R.

Перевага даної концепції полягає в гнучкості вибору рівня захисту та швидкодії шляхом регулювання параметра *h* у широких межах. Однак недоліком є складність у відповідності виконання обчислень між віддаленою комп'ютерною системою та термінальними мікроконтролерами, що може ускладнити процес реалізації.

Працездатність і визначається декількома важливими факторами. По-перше, пришвидшення імплементації криптографічних алгоритмів з відкритим ключем грає вирішальну роль у продуктивності системи, впливаючи на швидкодію та ефективність операцій. Оцінка рівня захищеності секретних даних під час їх віддаленої обробки є ще одним ключовим аспектом. Вона визначає, наскільки надійно система забезпечує конфіденційність та інтегрітет даних у процесі передачі та обчислень.

Такий метод дозволяє гнучко регулювати рівень захисту та швидкість операцій, змінюючи параметр h. Однак, існує складність у забезпеченні синхронізації роботи між віддаленою комп'ютерною системою та термінальними мікроконтролерами, що може ускладнити реалізацію даної ідеї.

Такий підхід може мати значний потенціал у використанні, але потребує докладних вивчень та оптимізації для забезпечення найкращої ефективності та безпеки при використанні в реальних умовах.

Під час оцінки зменшення часу обрахунків модулярного експонентування у криптографії з відкритим ключем запропоновано використовувати коефіцієнт прискорення β. Цей коефіцієнт визначається як співвідношення часу T_0

виконання операції на термінальному мікроконтролері до часу Tr її реалізації з використанням віддалених комп'ютерних систем.

Очевидно, що коефіцієнт прискорення β значною мірою залежить від кількості бітів h коду експоненти, які обробляються на термінальному мікроконтролері.

Аналіз показав, що для підвищення ефективності віддаленого обчислення модулярної експоненти в хмарі найбільше доцільно використовувати змішану мультиплікативно-адитивну форму розкладення коду експоненти. Використання виключно мультиплікативних форм виявилося менш оптимальним, оскільки це не дозволяє ефективно суміщати обчислення в хмарі та на термінальних мікроконтролерах. У підсумку, рекомендується застосовувати зазначений підхід для забезпечення оптимальної ефективності та безпеки віддалених обчислень.

2.2. Розробка методу розподіленого обчислення модулярної експоненти з адивно-мультиплікативним розкладенням коду експоненти.

Для реалізації запропонованого підходу пропонується адитивно-мультиплікативне розкладення секретного коду експоненти у вигляді:

$$E = g \cdot (q \cdot U + b) + w. \qquad (2.3)$$

Наприклад, U=15, обчислити $u = U^{a \cdot b} \bmod M$, а потім знайти множинну інверсію $u \text{ — } u^{-1}$ за модулем M. У рамках розглянутого прикладу, тобто при U=15, u=4, а його множинна інверсія за модулем 19 дорівнює 5: $u^{-1} = 5$. Відповідно, користувачем попередньо обчислюється $D = A \cdot U \bmod M$, яке передається в віддалену комп'ютерну систему, де обчислюється $B = D^a \bmod M$.

Формально запропонований метод швидкого захищеного обчислення модулярної експоненти $A^E \bmod M$ при реалізації з використанням віддалених комп'ютерних систем сводиться до виконання наступної послідовності дій:

1. Випадковим чином генеруються два числа h і a, сумарна розрядність r яких значно менша розрядності n коду експоненти E : $r = r_h + r_a << n$. Обирається випадкове ціле число v так, щоб різниця E-v націло ділилась на a:

(E-v) mod a = 0. Обирається довільне ціле число b таке, щоб (E-v) /a – b націло ділилась на h.

2. За формулою обчислюється значення W – коду експоненти, переданого на віддалену комп'ютерну систему.

3. На мікроконтролері обчислюється значення $G = A^h \bmod M$. Обчислене значення G разом з кодом експоненти W і модулем M відсилається в хмару.

4.На мікроконтролері обчислюються значення $Y = A^v \bmod M$ і $L=A^b \bmod M$.

5. На віддаленій комп'ютерній системі обчислюється $Q = G^W \bmod M$. Обчислене значення Q повертається на мікроконтролер.

6. Після отримання коду Q на мікроконтролері обчислюється модульне добуток $U = Q*L \bmod M$.

7. На мікроконтролері обчислюється значення $X = U^a \bmod M$.

8. На мікроконтролері обчислюється результат у вигляді $R = X*Y \bmod M$.

Наприклад, U=15, обчислити u =Ua-b mod M, а потім знайти мультиплікативну інверсію u - u-1 за модулем M. У рамках розглянутого прикладу, тобто при U=15, u=4, а його мультиплікативна інверсія за модулем 19 дорівнює 5: u-1 =5. Відповідно, користувач попередньо обчислює D=A-U mod M, яке передається у віддалену комп'ютерну систему.

Формально пропонований метод швидкого захищеного обчислення модулярної експоненти AE mod M під час реалізації з використанням віддалених комп'ютерних систем зводиться до виконання такої послідовності дій:

Випадковим чином генеруються два числа h і a, сумарна розрядність r яких значно менша за розрядність n коду експоненти E : $r = rh + ra << n$. Вибирається випадкове ціле число v так, щоб різниця E-v націло ділилася на a: (E-v) mod a =0.

Вибирається довільно ціле число b таке, щоб (E-v)/a - b націло ділилася на h.

2. За формулою (2.4) обчислюють значення W - коду експоненти, що передається на віддалену комп'ютерну систему.

3 На мікроконтролері обчислюється значення G = Ah mod M. Обчислене значення G разом із кодом експоненти W і модулем M відсилається в хмару.

4. На мікроконтролері обчислюються значення Y = Av mod M і L=Ab mod M.

5. На віддаленій комп'ютерній системі обчислюється $Q = G^E \bmod M$.. Обчислене значення Q повертається на мікроконтролер.

6. Після отримання коду Q на мікроконтролері обчислюється модулярний добуток $U = G \cdot Q \bmod M$.

7. На мікроконтролері обчислюється значення $X = U^b \bmod M$.

8. На мікроконтролері обчислюється результат у вигляді R = X·U mod M.

Запропонований підхід розкладає процес обрахування модулярної експоненти на 3 фази. Перша та третя фази виконуються на процесорних засобах користувача, в той час як друга фаза відбувається на віддалених обчислювальних потужностях. Цей метод сприяє оптимізації ресурсів та підвищенню ефективності віддалених обчислень.

Оцінка ефективності методу ґрунтується на двох ключових критеріях

1) Максимальне Підвищення Продуктивності: Визначаємо, наскільки метод сприяє ефективнішому виконанню операції модулярного експоненціювання користувачем. Мета полягає в значущому підвищенні швидкодії цієї операції.

2)Збереження Високого Рівня Захищеності: Ретельно вивчаємо, як метод впливає на захищеність передачі операндів B, V, M у відкриту систему. Мета - забезпечити, щоб використання методу не компрометувало безпеку передачі інформації.

Таким чином, пропонований метод дозволяє виконати операцію модулярного експоненціювання в області хмарних обчислень більш ефективно, зменшуючи середню кількість операцій модулярного множення з 1.5·n до більш економічних значень. Це вказує на збільшення продуктивності виконання операції модулярного експоненціювання користувачем у порівнянні з класичним методом.

Кількість операцій модулярного множення, виконаних користувачем, у загальному вигляді можна визначити як 1.5·(z+y), що в контексті практичного застосування при значеннях n=4096, U=32, та z≤32 може суттєво скоротити кількість операцій порівняно з класичним методом, забезпечуючи тим самим високий рівень продуктивності.

Отже, у випадку використання запропонованого методу, кількість операцій модулярного множення, які виконує користувач, суттєво скорочується. При повному виконанні користувачем модулярного експоненціювання, загальна кількість операцій дорівнює 4096·1.54096·1.5, а за запропонованим методом користувачу потрібно виконати тільки 64·1.564·1.5 операцій модулярного множення. Тож, кількість операцій зменшується у 64 рази, що вказує на значний приріст у продуктивності.

Експериментальні дослідження підтверджують теоретичні оцінки ефективності запропонованого методу. Час виконання операцій модулярного експоненціювання виявився зменшеним у 60 разів при розрядності операндів 4096 і у 28 разів при розрядності 2048. Щодо рівня захищеності, зловмисник, маючи доступ до кодів B і V, стикається з труднощами. Він не може оцінити правильність вибраного варіанту, оскільки не знає коду A. Зловмиснику може доводитися перебирати величезну кількість варіантів, що технічно неможливо здійснити.

У порівнянні із класичною задачею злому RSA, де потрібно аналізувати 2^4096 варіантів, запропонований метод вимагає аналізу 2^4127 варіантів. Це призводить до значного ускладнення завдання злому, роблячи його технічно неможливим у практичному плані і, таким чином, ще сильніше збільшує рівень захищеності.

Отримані результати досліджень вказують на те, що запропонований метод захищеного модулярного експоненціювання є ефективним для застосування на віддалених обчислювальних системах, особливо на малопотужних термінальних мікроконтролерах і мобільних пристроях. Метод дозволяє значно скоротити час виконання операцій модулярного

експоненціювання, а також підвищити рівень захищеності компонент цих операцій завдяки спеціальному маскуванню та модифікації коду експоненти. Призначений в першу чергу для використання в мережах Інтернету та для мобільних пристроїв, метод може забезпечити ефективний криптографічний захист при роботі з великими розрядностями чисел.

Так, головна перевага запропонованого методу полягає у високому співвідношенні між об'ємом обчислень, які виконуються на віддалених обчислювальних потужностях, і обчисленнями, які здійснюються безпосередньо на процесорі користувача. Це сприяє ефективнішому використанню ресурсів і дозволяє значно скоротити час виконання операцій модулярного експоненціювання.

Застосування наведеного методу має велике практичне значення, оскільки воно сприяє прискоренню впровадження криптографічних протоколів для захисту даних на мікроконтролерах з обмеженою розрядністю та термінальних застосунках у обчислювальних мережах. Це, в свою чергу, дозволяє забезпечити ефективне функціонування широкого спектру систем управління та моніторингу в режимі реального часу у захищеному виконанні.

Механізм захисту елементів ґрунтується на використанні спеціального маскування для елементів, що підносяться до ступеня, та змінення коду степеня. Новаторськість новоутвореного підходу полягає в ефективному використанні повторюваних елементів дій модулярного експоненціювання, що випливають з теореми Ферма. Це призводить до збільшення вагомості операцій, які виконуються на серверних обчислювальних ресурсах, і відповідно прискорює процес модулярного експоненціювання. Теоретичні розрахунки та експериментальні дослідження підтверджують, що запропонований підхід допомагає значно зменшити час виконання операцій експоненціювання за рахунок ефективного використання віддалених обчислювальних ресурсів.

Застосування маскування в розробленому методі значно підвищує непроникність компонент модулярного експоненціювання. Ця стратегія спрямована передусім на слабкі термінальні процесори та портативні пристрої,

які використовують протоколи захисту інформації у мережах. Вона суттєво зменшує час виконання великомасштабних операцій модулярного експоненціювання. Дослідження виявили, що для подібних обчислювачів, якщо з розрядністю 8192, виконання алгоритмів захисту даних не тільки вписується в установлені обмеження часу, але й суттєво їх перевищує.

Можливість одночасного засекречення експоненти A та величини D по коду L та відомому значенню величини N є важливою перевагою такого підходу. Його основою є мультиплікативно-адитивне розкладання експоненти A, тобто її представленні як $A = L \cdot y - w$.

При розкладанні експоненти відбувається підбір двох випадкових чисел y та w. При цьому має задовольнятись умова $(A - w) \bmod y = 0$, а розрядність b_y параметра y мусить не перевищувати 5.

Розроблений метод визначається такою послідовністю дій:

1. Термінальний комп'ютер реалізує операцію модулярного піднесеннч степеню.
2. У хмару надсилаються результат H та величини L і N.
3. На віддаленій комп'ютерній системі здійснюється обрахунок модулярної експоненти $M = H^L \bmod N$ з подальшим надсиланням результату M на термінальний мікроконтролер.
4. Водночас на термінальному комп'ютері виконується обчислення.
5. З використанням отриманих результатів M та P на термінальному мікроконтролері проводиться розрахунок модулярного добутку.

Нехай згенеровано закритий ключ $A = 97$ та число $N = 239$. $A = L \cdot y - w$. Нехай параметр y буде рівним 4. Значення величини w може дорівнювати 5, відповідно до умови $(A - w) \bmod y = 0$, оскільки $(97 - 5) \bmod 4 = 0$.

Наступним кроком згідно пункту 2 є відправка результату H і чисел $L = 46$ та $N = 189$ у хмару з подальшим обрахунком модулярної експоненти $M = H^L \bmod N = 81^{46} \bmod 189 = 135$ та відправкою результату $M = 135$ на термінальний комп'ютер.

2.3. Організація розподіленого обчислення модулярної експоненти з використанням обміну проміжними даними.

Як зазначалося в оглядовому розділі дисертації одним із перспективних хляхів підвищення ефективності обчислення модулярної експоненти з залученням хмарних обчислень полягає в застосуванні передачі даних

Описана вище процедура розподіленого обчислення модулярної експоненти в хмарі та термінальній обчислювальній платформі може ілюстрована наступним прикладом.

Нехай секретний код експоненти $E= 101\ 011\ 111_2 = 351_{10}$, модуль M = 43, а число, що підноситься до ступені A=12. Відповідно, правильне значення $A^E \bmod M = 12^{351} \bmod 43 = 2$

Вибрана множина Θ номерів розрядів експоненти, які оброблюються на термінальному мікроконтролері: Θ= {3,5,6,9 }, значення відповідних бітів: ϑ = { 1,1,0,1}. Після обнуління вибраних розрядів в коді експоненти отримується код експоненти, який передається в хмару: $W = 001\ 001\ 011_2 = 75$.

На віддаленій системі обчислюється $R = A^W \bmod M = 12^{75} \bmod 43 = 32$. Крім того, з хмари повертаються по мірі готовності коди: $D_1 = 12^4 \bmod 43 = 10$, $D_2 = 12^{16} \bmod 43 = 24$, $D_3 = 12^{32} \bmod M = 17$ та $D_4 = 12^{256} \bmod M = 10$.

На термінальному мікроконтролері обчислюється $Q = D_1 \cdot 1 = 10$; $Q = D_2 \cdot Q = 10 \cdot 24 \bmod 43 = 25$; $Q = D_4 \cdot Q = 10 \cdot 25 \bmod 43 = 35$. По отриманню результату з хмари кінцевий результат обчислюється у вигляді: $Q = Q \cdot R \bmod M = 35 \cdot 32 \bmod 43 = 2$.

При віддаленому обчисленні модулярної експоненти ми можемо приховувати лише одиниці. Це, потенційно, спрощує задачу зловмиснику: мін може при підборі замість одиниць підставляти одиниці. Нехай модуль M = 43, A=12, а код експоненти $E=1011_2 =11$. Відповідно, $12^{11} \bmod 43 = 26$. В хмару передається код $W=1111_2$. Хмара повертає нам $12^{15} \bmod 43 = 2$. Теоретично нам потрібно результат розділити на $12^4 = 20736 : 2^{15} = 15407021574586368 /20736$

= 743008370688 mod 43 = 26. Проте операції ділення у нас нема. Модулярна інверсія A^{-4} mod 43 = A^{42-4} mod 43 = A^{38} mod 43. 12^{-4} mod 43 = 12^{38} mod 43 = 13.

Більшої ефективності можна добитися шляхом симетричної заміни бітів коду експоненти. нехай модуль M=19. Якщо взяти другий розряд коду експоненти ε=2, то циклічно інверсний для нього є розряд к = 18-2 = 16. Нехай код експоненти E = 00011=3. Ми формуємо код W=10101=21. Результат 12^3 mod 1 9 = 18 дорівнює результату 12^{21} mod 19 = 18, тобто гомоморфного шифрування виконувати не потрібно. Гомоморфне шифрування полягає в тому, що здійснено перенос одиниці на позицію, яка є модулярно інверсною до неї. Для цього прикладу одиниця з позиції ε=2 перенесена на модулярно інверсну позицію к=16.

Для модуля 19 можна визначити такі пари модулярно-інверсних позицій <1,17>,<2,16>,<4,14>,<8,10>. Що на практиці означає пара <8,10>. Це означає, що A^8 mod 19 інверсна A^{10} mod 19. Нехай код експоненти E = 00011=3. Реалізація гомоморфного шифрування пари <8,10> означає, що якщо до коду експоненти E = 00011 додати 18, то нічого не зміниться: 12^{3+18} mod 19 = 12^{21} mod 19 = 18.

Маємо постійний код експоненти E= 101 011 1101_2 = 701. Для шифрування A віднімається одиниця і ділиться на 2: E = U·2 + 1. Для прикладу, що розглядається: U= 101 011 110_2. Код U розділяється на три секції і дві компоненти: W = 100 010 110 та Q = 001 001 000. Експонента для компоненти W обчислюється в хмарі, а експонента для компоненти Q обчислюється на мікроконтролері. Через кожні три кроки хмара повертає значення D, а в кінці повертає значення R′. Хмарі в якості числа, яке підноситься до ступеня задається B = A^2 mod M. Якщо модуль M=743, а число A=529, то B = 529^2 mod 743 = 473.

На першому циклі в хмарі обчислюються значення: R_1=1, D_2 = B^2 mod M = 473^2 mod 743 = 86; R_2 = 86, D_3 = D_2^2 mod M = 86^2 mod 743 = 709; R_3 = R_2·D_3 mod M = 86·709 mod 743 = 48, D_4 = D_3^2 mod M = 709^2 mod 743 = 413. Хмара

повертає на мікроконтролер $D_4 = 413$. Сам мікроконтролер в першому циклі не виконує ніяких операцій.

В другому циклі в хмарі обчислюються значення: $R_4=48$, $D_5 = D_4^2 \bmod M = 413^2 \bmod 743 = 422$; $R_5 = R_4 \cdot D_5 \bmod M = 48 \cdot 422 \bmod 743 = 195$, $D_6 = D_5^2 \bmod M = 422^2 \bmod 743 = 507$; $R_6 = R_5 = = 195$, $D_7 = D_6^2 \bmod M = 507^2 \bmod 743 = 714$. Хмара повертає на мікроконтролер $D_7 = 714$. Сам мікроконтролер обчислює $Y=1 \cdot D_4 = 413$.

В третьому циклі в хмарі обчислюються значення: $R_7= R_6 = 195$, $D_8 = D_7^2 \bmod M = 714^2 \bmod 743 = 98$; $R_8 = R_7 = 195$, $D_9 = D_8^2 \bmod M = 98^2 \bmod 743 = 688$; $R_9 = R_8 \cdot D_9 \bmod M = 195 \cdot 688 \bmod 743 = 420$. Хмара повертає на мікроконтролер $R_9 = 420$.

Сам мікроконтролер обчислює $Y=Y \cdot D_7 \bmod M = 413 \cdot 714 \bmod 743 = 654$.

Після цього, в рамках фінальної стадії мікроконтролер обчислює модулярний добуток: $Y \cdot R_9 \cdot A \bmod M = 654 \cdot 420 \cdot 529 \bmod 743 = 182$. Легко переконатися, що отриманий результат співпадає з $A^E \bmod M = 529^{701} \bmod 743 = 182$.

Зловмисник має здогадатися про значення η бітів, з яких починається кожен фрагмент. Тобто йому потрібно проаналізувати значення 2^η варіантів. При цьому мікроконтролер, в середньому виконує $\eta/2$ множень. Співвідношення ξ, яке дорівнює кількості варіантів перебору до числа множень на мікроконтролері визначається формулою:

$$\xi_1 = \frac{2^{\eta+1}}{\eta}. \qquad (2.4)$$

Якщо аналізувати два молодших розряди кожного фрагмента, то кількість варіантів становить $4^\eta = 2^{2 \cdot \eta}$ При цьому середня кількість модулярних множень на мікроконтролері в одному фрагменті становить: $0.25 \cdot 1 + 0.25 \cdot 2 + 0.25 \cdot 3 = 2.25$. Відповідно, всього середня кількість модулярних множень дорівнює $2.25 \cdot \eta$. Для цього варіанту коефіцієнт ξ обчислюється у вигляді:

$$\xi_2 = \frac{2^{2 \cdot \eta}}{2.25 \cdot \eta} \approx \frac{2^{2 \cdot \eta - 1}}{\eta}. \qquad (2.5)$$

Співвідношення γ показників ефективності для двох і одного розрядів коду експоненти складає:

$$\gamma = \frac{\xi_2}{\xi_1} = \frac{2^{2\cdot\eta-1}}{2^{\eta+1}} = 2^{\eta-2}. \qquad (2.6)$$

З цього випливає, що вигідніше застосовувати два розряди ніж один при числі фрагментів більше за два. Наприклад, якщо кількість фрагментів 16, то при задіяні двох молодших розрядів кожного фрагменту кількість можливих варіантів $4^{16} = 4.295\cdot10^9$. Середня кількість множень на мікроконтролері $2.25\cdot16 = 36$. Відповідно, $\xi_2 = 1.19\cdot10^7$. При задіяні одного молодшого розряду кількість їх можливих значень 2^{16}, кількість модулярних множень – 8. $\xi_1 = 8192$. $\gamma = \xi_2/\xi_1 = 14564$. Теоретично обчислене за формулою (2.14) значення $\gamma = 2^{14} = 16384$.

Додатково можна здійснити задання з мікроконтролера плаваючих точок повергнення поточних значень D з хмари. Це дозволить зменшити середню кількість множень на мікроконтролері і таким чином підвищити показник ефективності ξ.

В порівнянні з адитивним розкладенням коду експоненти, джерелом невизначеності в запропонованому методі виступають дві компоненти:

- дії, які виконує термінальний мікроконтролер в процесі обчислення модулярної експоненти;

- локалізація дій, що виконуються термінальним мікроконтролером.

В узагальненому вигляді рівень захищеності адитивного шифрування коду експоненти визначається тим скільки розрядів оброблюються на мікроконтролері і як вони локалізовані по розрядному полі коду експоненти. Якщо не переривати обчислення, то вказані розряди можуть знаходитися або на початку, або в кінці розрядного поля.

Якщо розвивати ідеї використання декількох віддалених обчислювальних систем, то можна задіяти дві таких системи. Перша обчислює A в степені $\beta = 2^n$, тобто $A^\beta \bmod M$. Повертає на термінальний мікроконтролер значення $A^k \bmod M$, $A^{k\cdot2} \bmod M$, …, $A^\beta \bmod M$, де $\gamma = 2^\eta$, при тому, що $\eta < n$. В рамках цієї ідеї *n*–

розрядний код $E = e_0 + e_1 \cdot 2 + e_2 \cdot 2^2 + \ldots + e_{n-1} \cdot 2^{n-1}$ експоненти поділяється на наступні адитивні складові:

$$E = G + Y + L, \tag{2.7}$$

де L - k-розрядний код, який складається з молодших k розрядів коду експоненти E, G – n-розрядний код, q старших розрядів якого співпадає з відповідними розрядами коду експоненти E, а n-q молодших розрядів дорівнюють нулю, Y – (n-q)-розрядний код, старші n-k-q розрядів якого співпадають з однойменними розрядами експоненти E, а молодші k розрядів дорівнюють нулю. Формально, зазначені адитивні складові коду експоненти E можуть бути представлені у наступному вигляді:

$$L = \sum_{l=0}^{k-1} e_l \cdot 2^l, \quad Y = \sum_{i=k}^{n-q-2} e_i \cdot 2^i, \quad G = \sum_{j=n-q-1}^{n-1} e_j \cdot 2^j .$$

Першій віддаленій комп’ютерній системі задається код A і вона послідовно обчислює коди $A^k \bmod M$, $A^{2k} \bmod M$, $A^{3k} \bmod M$,…, $A^{2\cdot(k-1)} \bmod M$. Обчислення здійснюється шляхом послідовного виконання n-1 кроків, в кожному з яких отриманий на попередньому кроці результат підноситься до квадрату по модулю M. При цьому, перша віддалена система повертає на термінальний мікроконтролер коди перших k-1 результатів: $D_1=A^2 \bmod M$, $D_2=A^4 \bmod M$, $D_3 = A^8 \bmod M$,…, $D_{k-1}= A^{2\cdot(k-1)} \bmod M$, а також останніх q з отриманих результатів: $D_{n-q-1}=A^{2\cdot(n-q-1)} \bmod M$, $D_{n-q}=A^{2\cdot(n-q)} \bmod M$, …, $D_{n-1}= A^{2\cdot(k-1)} \bmod M$.

На термінальному мікроконтролері обчислюється значення $P_1=A^L \bmod M$, а також значення $P_2 = A^G \bmod M$ у вигляді:

$$P_1 = A^L \bmod M = \prod_{i=1}^{k-1} D_i^{e_i} \bmod M ,$$

$$P_2 = A^G \bmod M = \prod_{j=n-q-1}^{n-1} D_j^{e_j} \bmod M .$$

На другій віддаленій комп’ютерній системі обчислюється значення $P_3=A^Y \bmod M$. На термінальному мікроконтролері остаточний результат $A^E \bmod M$ обчислюється у вигляді добутку: $R=P_1 \cdot P_2 \cdot P_3$. Кількість γ операцій модулярного

множення над n-розрядними числами, які виконуються на термінальній обчислювальній платформі визначається при цьому у вигляді: $\gamma = 0.5 \cdot (k+q)$.

Викладене вище може бути ілюстровано таким прикладом: нехай я якості закритого ключа використовується код експоненти E= 57·61 = 3477 = 1101 1001 0101_2, тобто $n = 12$.

Роботу розробленого методу можна продемонструвати на наступному числовому прикладі. Нехай проводиться обчислення модулярної експоненти $A^E \text{mod} M$ за алгоритмом з молодших розрядів. Суть цього алгоритму полягає в обчисленні модулярної експоненти з використанням двох змінних *D* та *R*, де *D* – це модулярне піднесення до квадрату, а *R* – модулярне множення. В рамках поточного прикладу обчислюється модулярна експонента $5^{2517} \text{mod} 2347 = 919$. Таким чином $A=5$, $E=2517$, $M=2347$. Код експоненти E у двійковому представленні має вигляд $E=100111010101_2$.

Першим кроком є поділ коду експоненти на три сегменти: e_1, e_2, e_3. Сегменти e_1 та e_3 – невеликої розрядності для обчислення на терміальному мікроконтролері, а сегмент e_2 – значно більшої розрядності для обчислення на віддаленій комп'ютерній системі. Наступним етапом є відправка необхідних параметрів до двох хмарних систем.

Перша хмарна система (далі ХС1), отримує значення модуля *M* та даних *A* для обчислення *D* – модулярного піднесення до квадрату. Друга хмарна система (далі ХС2) для повноцінного обчислення модулярної експоненти за алгоритмом з молодших розрядів отримує значення модуля *M*, даних *A* та експоненти e_2h, де *h* – це сегмент з нулів розрядність якого рівна розрядності e_1.

Покрокові обчислення на термінальному мікроконтролері, ХС1 та ХС2 представлені у вигляді таблиці 1.

Організація обчислення $A^E \text{ mod } M$ при зазначеному розкладенні коду *E* експоненти зводиться до обчислення $P_Y = A^Y \text{ mod M}$ в хмарі, обчислення $P_L = A^L \text{ mod M}$, $P_G = A^G \text{ mod M}$, а також $P_U = P_Y{}^U \text{ mod M}$ на термінальному мікроконтролері. На цій же платформі обчислюється остаточний результат у вигляді модулярного добутку $A^E \text{ mod } M = P_L \cdot P_G \cdot P_U \text{ mod M}$.

Код експоненти E може бути розкладений на мультиплікативно-адитивні компоненти у наступному вигляді:

$$E = G + U \cdot Y + L,$$

де G – n-розрядний код, q старших розрядів якого співпадає з відповідними розрядами коду експоненти E, а n-q молодших розрядів дорівнюють нулю, $U \cdot Y + L$ – (n-q)-розрядний код, який складається з молодших n-q розрядів коду експоненти, L - k-розрядний код, U та Y – мультиплікативні складові. Розрядність коду $U = u_0 + u_1 \cdot 2 + u_2 \cdot 2^2 + .. + u_{l-1} \cdot 2^{l-1}$ дорівнює l.

Кінцевим результатом обчислень є код R = 919.

$$E = \sum_{j=\eta}^{n} e_j \cdot 2^j + \sum_{i=k}^{\eta-1} e_i \cdot 2^i + \sum_{l=1}^{k-1} e_l \cdot 2^l . \qquad (2.8)$$

Першій віддаленій комп'ютерній системі задається код A і вона послідовно обчислює коди $D_1 = A^2 \bmod M$, $D_2 = A^4 \bmod M$, $D_3 = A^8 \bmod M$,…, $D_{n-1} = A^{2^{n-1}} \bmod M$. Обчислення здійснюється шляхом послідовного виконання n-1 кроків, в кожному з яких отриманий на попередньому кроці результат підноситься до квадрату по модулю M. При цьому, перша віддалена система повертає на термінальний мікроконтролер коди перших k-1 результатів: $D_1 = A^2 \bmod M$, $D_2 = A^4 \bmod M$, $D_3 = A^8 \bmod M$,…, $D_{k-1} = A^{2\cdot(k-1)} \bmod M$, а також останніх q з отриманих результатів: $D_{n-q-1} = A^{2^{n-q-1}} \bmod M, ..., D_{n-1} = A^{2^{n-1}} \bmod M$.

Другій віддаленій комп'ютерній системі задаються коди A і Y, після чого вона обчислює код $P_Y = A^Y \bmod M$. По закінченні формування коду P_Y, вона послідовно обчислює $F_1 = P_Y{}^2 \bmod M$, $F_2 = P_Y{}^4 \bmod M$, …, $F_{l-1} = A^{2^{l-1}} \bmod M$.

При несиметричному шифруванні код A є секретним, а результат шифрування $A^E \bmod M$ не є секретним. При дешифруванні код A несекретний, а результат дешифрування $A^E \bmod M$ розглядається як секретний. Рівень захищеності вимірюється обсягом ресурсів, витрачених зловмисником для незаконного відновлення секретних операндів A і E при наявності у нього кодів B, L, h, k, а також модуля M.

Таблиця 2.1

Покрокові обчислення на термінальному мікроконтролері

ТМК R = 1, D = A e_1=101, e_3=1001		**ХС1** D = A		**ХС2** R = 1, D = A e_2h=11010000
Розрахунки				
R = R*DmodM R = 1*5mod2347 = 5 $D_1 = 5^2 \text{mod} 2347 = 25$		$D_1 = D^2 \text{mod} M$ $D_1 = 5^2 \text{mod} 2347 = 25$		$D_1 = 5^2 \text{mod} 2347 = 25$
	D_2,	$D_2 = 25^2 \text{mod} 2347 = 625$		$D_2 = 25^2 \text{mod} 2347 = 625$
	← *D_3*	$D_3 = 625^2 \text{mod} 2347 = 1023$		$D_3 = 625^2 \text{mod} 2347 = 1023$
		$D_4 = 1023^2 \text{mod} 2347 = 2114$		$D_4 = 1023^2 \text{mod} 2347 = 2114$
		$D_5 = 2114^2 \text{mod} 2347 = 308$		R=1*2114mod2347=2114
		$D_6 = 308^2 \text{mod} 2347 = 984$		
		$D_7 = 984^2 \text{mod} 2347 = 1292$		$D_5 = 2114^2 \text{mod} 2347 = 308$
R=5*625mod2347=778		$D_8 = 1292^2 \text{mod} 2347 = 547$		$D_6 = 308^2 \text{mod} 2347 = 984$
		$D_9 = 547^2 \text{mod} 2347 = 1140$		R=2114*984mod2347=734
	D_9,	$D_{10} = 1140^2 \text{mod} 2347 = 1709$		
	D_{10},	$D_{11} = 1709^2 \text{mod} 2347 = 1013$		$D_7 = 984^2 \text{mod} 2347 = 1292$
R=778*547mod2347=759	← *D_{11}*			R=734*1292mod2347=140
R=759*1013mod2347=1398	←		→	
R=1398*140mod2347=919			*R*	

Коди $F_1,F_2,\ldots,F_{l-1}$ надсилаються на термінальний мікроконтролер, на якому обчислюється значення $P_{\mathrm{L}} = A^{\mathrm{L}} \bmod M$, $P_{\mathrm{G}} = A^{\mathrm{G}} \bmod M$, а також $P_{\mathrm{U}} = P_{\mathrm{Y}}{}^{\mathrm{U}} \bmod M$ у вигляді:

$$P_L = A^L \bmod M = \prod_{i=1}^{k-1} D_i^{e_i} \bmod M .$$

$$P_G = A^G \bmod M = \prod_{j=n-q-1}^{n-1} D_j^{e_j} \bmod M .$$

$$P_U = P_Y{}^U \bmod M = \prod_{j=1}^{l-1} F_j^{u_j} \bmod M .$$

U·Y – (n-q)-розрядний код, старші n-k-q розрядів якого співпадають з однойменними розрядами експоненти E, а молодші k розрядів дорівнюють нулю. Цей код L - k-розрядний код, який складається з молодших k розрядів коду експоненти E. Формально, зазначені адитивні складові коду експоненти E можуть бути представлені у наступному вигляді:

$$L = \sum_{l=0}^{k-1} e_l \cdot 2^l, \quad Y = \sum_{i=k}^{n-q-2} e_i \cdot 2^i, \quad G = \sum_{j=n-q-1}^{n-1} e_j \cdot 2^j . \tag{2.9}$$

Першій віддаленій комп’ютерній системі задається код A і вона послідовно обчислює коди $A^2 \bmod M$, $\mathrm{A}^4 \bmod \mathrm{M}$, $\mathrm{A}^8 \bmod \mathrm{M},\ldots, \mathrm{A}^{2\cdot(\mathrm{n}-1)} \bmod \mathrm{M}$. Обчислення здійснюється шляхом послідовного виконання n-1 кроків, в кожному з яких отриманий на попередньому кроці результат підноситься до квадрату по модулю M. При цьому, перша віддалена система повертає на термінальний мікроконтролер коди перших k-1 результатів: $D_1=A^2 \bmod M$, $D_2=A^4 \bmod M$, $D_3 = A^8 \bmod M,\ldots, D_{k-1}= A^{2\cdot(k-1)} \bmod M$, а також останніх q з отриманих результатів: $D_{n-q-1}=A^{2\cdot(n-q-1)} \bmod M$, $D_{n-q}=A^{2\cdot(\mathrm{n}-q)} \bmod M$, …, $D_{n-1}= A^{2\cdot(k-1)} \bmod M$.

Виконання модулярного експоненціювання $A^E \bmod M$ на термінальному мікроконтролері з залученням хмарних обчислень за запропонованим методом починається з гомоморфного шифрування інформаційної компоненти A: вона підноситься до модулярного квадрату з фіксацією отриманого результату в B: нехай E= 101 **01**1 **1**11$_2$ = 351$_{10}$, модуль M = 43, а число, що підноситься до ступені A=12. Відповідно, правильне значення $\mathrm{A}^{\mathrm{E}} \bmod \mathrm{M} = 12^{351} \bmod 43 = 2$

$B = A^2 \bmod M$. $R_6 = R_5 =$ = 195, $D_7 = D_6^2 \bmod M = 507^2 \bmod 743 = 714$. Хмара повертає на мікроконтролер $D_7 = 714$. Сам мікроконтролер обчислює $Y=1 \cdot D_4 = 413$. $E_0 = 110_2 = 6$, $E_1 = 011_2 = 3$, $E_2 = 101_2 = 5$. $E = 5 \cdot 2^6 + 3 \cdot 2^3 + 7$. Попередньо рахується $2^6 \bmod 43 = 21$ $2^3 \bmod 43 = 8$. В рамках цього прикладу розрядності складових адитивного розкладення коду експоненти дорівнюють: $m_0 = 2$ $m_1 = 1$ $m_2 = 3$. Відповідно, в хмару передається $W = (E_2\text{-}m_3) \cdot 2^6 + (E_1\text{-}m_1) \cdot 2^3 + E_0\text{-}m_0 = (5\text{-}3) \cdot 2^6 + (3\text{-}1) \cdot 2^3 + (7\text{-}2) = 149$. $R = 12^{149} \bmod 43 = 28$.

Ефективність запропонованого методу може оцінюватися за двома критеріями:

- за ступенем захищеності від несанкціонованої реконструкції секретних операндів A і E на віддаленій комп'ютерній системі за заданими їй значеннями B, L, h, k, M;

– за прискоренням обчислення модулярної експоненти на термінальній обчислювальній платформі за рахунок використання можливостей хмарних систем.

Крім цих кодів, йому відоме обчислене в хмарі значення R, а також вірний результат модулярного експоненціювання $A^E \text{mod} M$, який можна отримати, здійснюючи контроль в Інтернеті обміну даними між елементами системи управління.

Принципово можливі два сценарії здійснення атаки на процес віддаленого обчислення. За першим із них, коди A і E зловмиснику не відомі. Для відновлення значення A зловмисник може здійснити перебір усіх можливих кодів A, модулярне піднесення до квадрата і порівняння з відомим йому кодом B, тобто підібраний код A' визначається досягненням рівності $A'^{\,2} \text{mod} M = B$. Вважаючи, що на практиці розрядність A у більшості випадків дорівнює n = 2048, середній обсяг перебору становить $2^{n-1} = 2^{2047} \approx 10^{616}$, що значно перевищує можливості сучасних комп'ютерних систем. Після підбору A' за аналогічною схемою слід виконати підбір E.

За другим сценарієм зловмисник певним незаконним чином отримує інформацію про значення A. У цьому випадку критерієм підбору є рівняння B^{L+Q} modM = A^{E} modM. Найдоцільнішою тактикою реконструкції коду E є перебір зловмисником значень коду Q. На практиці зловмисник виходить із того, що обсяг обчислень C, реалізованих на термінальному мікроконтролері, обмежений певною величиною. За наявності доступу до термінального мікроконтролера зловмисник може оцінити приблизне значення C. Крім того, зловмиснику відомо, що чисельне значення C, що вимірюється кількістю операцій модулярного множення, визначається за формулою(2.11):$C = 1{,}5 \cdot \eta + 1 + \beta + k$, (2)де η – сумарна кількість розрядів кодів $q_0,q_2,\ldots,q_{k-1}$.

Обсяг перебору всіх можливих значень ŋ-розрядного двійкового коду $q_0,q_1,\ldots,q_{k-1}$ оцінюється як $2^{ŋ}$. З іншого боку, число □ можливих варіантів □ розрядностей кодів $q_0,q_1,\ldots,q_{k-1}$, сумарна кількість яких дорівнює ŋ, може бути визначено як розв'язання відомої комбінаторної задачі розподілу ŋ однакових предметів за k групами (з можливістю порожніх груп). Число θ таких варіантів визначається формулою

З використанням відомої формули Стірлінга (Stirling) вираз (3) може бути перетворено до вигляду:

Таким чином, загальне число F варіантів перебору значень кодів $q_0,q_1,\ldots,q_{k-1}$ з урахуванням усіх варіантів їхніх розрядностей, які в сумі дорівнюють ŋ, визначається добутком $2^{\eta}\cdot\theta$:

$$F = 2^{ŋ} \cdot \frac{(ŋ+k-1)!}{ŋ!\cdot(k-1)!}. \qquad (2.10)$$

Це означає, що для незаконної реконструкції секретного коду E за умови, що відомі значення A, B, L, k, модуля M, а також істинне значення A^{E} mod M, зловмиснику буде потрібно виконати F операцій модулярного експоненціювання. Іншими словами, обчислене за формулою (5) значення F можна використовувати як міру ресурсів, які необхідно затратити для

незаконної реконструкції коду експоненти за умови використання запропонованого методу.

Проведені розрахунки свідчать про те, що виходячи зі специфіки конкретного використання, практично завжди може бути забезпечено заданий рівень захищеності відповідним вибором значень ŋ і k.

Виходячи з вимог до безпеки, що визначаються специфікою конкретних практичних застосувань, оцінюють граничний обсяг F' ресурсів, за якого злом секретного коду E економічно виправданий. З використанням формули (5) визначаються чисельні значення ŋ' і k', що відповідають граничному обсягу F' ресурсів, які зловмисник може затратити для атаки на запропонований метод.

Припустимо, на віддаленій комп'ютерній системі здійснюється незаконна реконструкція секретного коду E з використанням 1000 криптопроцесорів 7955 фірми Hi/fn. Обчислення модулярної експоненти над 2048-розрядними числами на такому криптопроцесорі займає 82.75 мс. Також відомо, що граничний час на несанкціоноване відновлення коду E не перевищує 1 року, що зумовлено періодом зміни цього коду. У такому разі, щоб гарантувати ймовірність підбору коду E менше ніж 1% за рік безперервної роботи, значення F' має становити не менше ніж $3{,}81099698 \cdot 10^{11}$.

Прискорення реалізації криптографічних протоколів з відкритим ключем на термінальних платформах Інтернету Речей за рахунок використання можливостей хмарних обчислень характеризується двома показниками:
коефіцієнтом α, що визначається співвідношенням обсягу обчислень C, реалізованих на термінальному мікроконтролері, до загального обсягу обчислень модулярної експоненти:

$$\alpha = \frac{C}{1{,}5 \cdot n} = \frac{1{,}5 \cdot ŋ + 1 + \beta + k}{1{,}5 \cdot n} \qquad (2.11)$$

Враховуючи, що $\beta \leq 1$, $k << n$, а реальне значення $n \geq 2048$, формулу (6) можна спростити:

$$\alpha \approx \frac{ŋ}{n} \qquad (2.12)$$

коефіцієнтом γ прискорення, що визначається співвідношенням часу T_0 обчислення експоненти A^E modM на термінальному мікроконтролері до часу T обчислення цієї експоненти із залученням хмарних технологій.

$$\gamma = \frac{T_0}{T} \tag{2.13}$$

де T_0 час модулярного множення n-розрядних чисел на термінальному мікроконтролері; T - час обчислення модулярної експоненти із залученням віддалених систем.

Обчислення $Q = B^{q_0} \, mod \, M \cdot D_1{}^{q_1} \, mod \, M \cdot D_2{}^{q_2} \, mod \, M \cdot \ldots \cdot D_{k-1}^{q_{k-1}} \, mod \, M$ на термінальному мікроконтролері може поєднуватися в часі з обчисленням R у хмарі. Необхідною умовою скорочення загального часу T обчислення модулярної експоненти із залученням віддалених комп'ютерних систем є якомога більше поєднання роботи термінального мікроконтролера і з обчисленням у хмарі. Це означає, що розрядності $q_0,q_1,\ldots,q_{k-1}$ повинні вибиратися таким чином, щоб обчислення Q відбувалося без пауз. Якщо позначити через T_c час пересилання вихідних даних у хмару, обчислення на віддалених комп'ютерних системах і повернення результату R, то за повного поєднання віддалених обчислень і тих, що виконуються на термінальній обчислювальній платформі, виконується:

$$(1.5 \cdot \eta + (k\text{-}1)) \cdot t_{mm} = T_c \, , \tag{2.14}$$

де t_{mm} – час модулярного множення на термінальному мікроконтролері.

Якщо $(1.5 \cdot \eta' + (k'\text{-}1)) \cdot t_{mm} < T_c$, тобто при визначеному з умов безпеки значенні ŋ' і k' час обчислень на мікроконтролері менший за T_c, то реальне значення ŋ може бути збільшене для підвищення рівня захисту без шкоди для часових характеристик до рівня, що визначається з рівняння :

$$\eta = 0.75 \cdot (\frac{T_c}{t_{mm}} - k + 1). \tag{2.15}$$

У цьому випадку, коефіцієнт прискорення обчислення модулярної експоненти при застосуванні запропонованого методу визначається формулою:

$$\gamma = \frac{T_0}{T_C + (\beta+2) \cdot t_{mm}} = \frac{1.5 \cdot n \cdot t_{mm}}{(1.5 \cdot \eta + k\prime + \beta + 1) \cdot t_{mm}} = \frac{1.5 \cdot n}{1.5 \cdot \eta + k\prime + \beta + 1} \tag{2.26}$$

Якщо $(1.5\cdot \eta' + (k\text{-}1))\cdot t_{mm} > T_c$, то реальне значення ŋ = ŋ'. У цьому випадку, коефіцієнт γ визначається формулою:

$$\gamma = \frac{1.5\cdot n\cdot t_{mm}}{(1.5\cdot \eta\prime + k\prime + \beta + 1)\cdot t_{mm}} = \frac{1.5\cdot n}{1.5\cdot \eta\prime + k\prime + \beta + 1}. \quad (2.17)$$

Висновки до розділу 2.

За результатами досліджень, які складають другий розділ і направлені на вибір найбільш перспективного напрямку підвищення ефективності розподіленого обчислення базової операції криптографічних алгоритмів з відкритим ключем - модулярного експоненціювання на термінальних мікроконтролерах систем моніторингу і управління віддаленими об'єктами з використанням технологій IoT, а також хмарних обчислень можуть бути зроблені наступні висновки:

1. Проведений аналіз показав, що в основі переважної більшості існуючих підходів до підвищення ефективності обчислювальної реалізації криптографії з відкритим ключем в системах комп'ютерного систем моніторингу і управління віддаленими об'єктами на базі технологій IoT лежить розкладення секретного коду експоненти на адитивні та мультиплікативні складові, яке дозволяє ефективно організувати розподілене обчислення між термінальними та віддаленими обчислювальними платформами. Показано, що рівень захищеності від спроб відтворення коду експоненти на віддалених обчислюваних потужностях за даними, що на них передаються, залежить від кількості операцій модулярного множення, які здійснюються на термінальному мікроконтролері.

2. Теоретично обґрунтовано, розроблено та досліджено захищену реалізацію модулярного експоненціювання на віддалених комп'ютерних системах, який відрізняється тим, що секретний код експоненти розкладається на мультиплікативно-адитивні компоненти, частина з яких використовується при віддалених обчисленнях, а частина – на термінальному мікроконтролері. Теоретично та експериментально показано, що запропонований метод дозволяє підвищити швидкість реалізації криптографічних алгоритмів з відкритим

ключем на термінальних мікроконтролерах систем комп'ютерного управління в середньому у 70 разів. При цьому надійно забезпечується захист від ренонструкції секретних компонентів криптосистеми за даними, що надаються на віддалені системи.

3. Розроблено метод розподіленого обчислення базової операції криптографії з відкритим ключем – модулярного експоненціювання з використанням хмарних технологій, відмінність якого полягає в організації передачі проміжних даних з віддалених систем на термінальний мікроконтролер для зменшення обчислювального навантаження на нього, що дозволяє збільшити кількість розрядів секретного коду експоненти, що обробляються на ньому і, відповідно, на 2-3 порядки підвищити рівень захищеності від спроб незаконної реконструкції коду експоненти на віддалених комп’ютерних системах, за даними, що надаються на них.

РОЗДІЛ 3
СТВОРЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИКОНАННЯ МЕТОДУ ВІДДАЛЕНОГО ЗАХИЩЕНОГО МОДУЛЯРНОГО ЕКСПОНЕНЦІЮВАННЯ.

3.1. Архітектура програмного забезпечення.

Перша мета розробленої програми - імплементація та створення моделі алгоритмів, описаних у минулому розділі. Програма має забезпечувати зручний інтерфейс для гнучкої настройки параметрів та структури системи, або відображати реальну систему, що включає в себе віддалений пристрій (клієнт), інтерфейс для взаємодії з віддаленим обчислювачем та сам хмарний обчислювач.

Основні компоненти програми:

1. Віддалений пристрій (клієнт):

-Надсилає дані до віддаленої системи обробки.

2. Інтерфейс взаємодії з віддаленим обчислювачем:

- Забезпечує взаємодію між віддаленим пристроєм та хмарним обчислювачем.

3. Хмарний обчислювач:

- Можливість налаштовувати структуру обчислювача, включаючи кількість ядер, швидкість зв’язку між вузлами та інші параметри.

Програмний продукт повинен забезпечувати зручний та інтуїтивно зрозумілий інтерфейс для користувача, що дозволяє легко виконувати моделювання та налаштовувати параметри системи

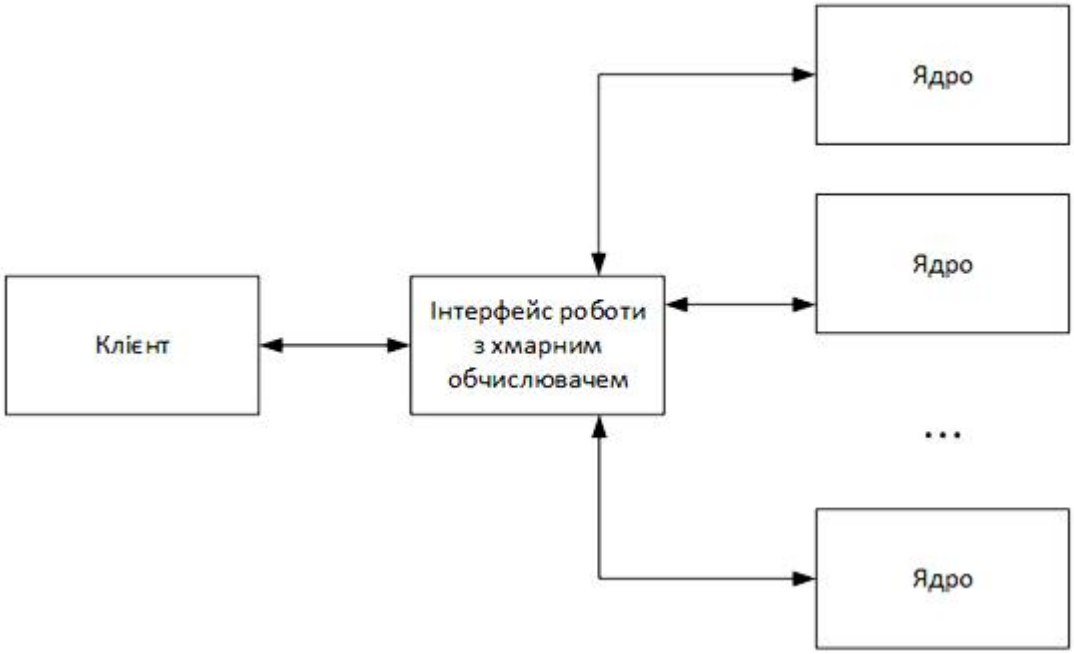

Рис 3.1. Архітектура програмної системи

Взаємодія між клієнтом та сервером в контексті комп'ютерних систем розглядається як імплементація концепції «ведучий-ведений», де клієнт виступає у ролі ведучого. Це означає, що клієнт виступає як автор взаємодії, надає інформацію для перетворення та очікує на результат від сервера. Функціональні обов'язки клієнтського модуля в розробленому програмному забезпеченні включають в себе ініціацію з'єднання, передачу даних для обробки, очікування результатів, обробку отриманих даних та управління параметрами взаємодії. Тобто клієнт займається такими завданнями:

1. Одержує повний набір даних щодо поточного режиму роботи, включаючи числові параметри, якщо це потрібно. У цьому контексті клієнт володіє можливістю здійснювати обчислення $A\hat{}E \; mod \; N$ на основі конкретно вказаних ведучим параметрів або користуватися випадково створеними числами. Важливим аспектом є можливість динамічного визначення розрядності чисел для ефективного проведення обчислень.
2. Клієнт проводить обов’язкові обрахунки, серед яких включені::
    a. У випадку вибору режиму за замовчуванням, клієнт випадковим чином розбиває показник степені на h частин;

b. Здійснюється аналіз показника степені для виявлення значної кількості повторюючихся бітів. Після чого відбувається розбиття показника степені з урахуванням цієї інформації;
c. Виконується частотний аналіз експоненти для знаходження груп бітів, які часто зустрічаються.;
d. Клієнт формує необхідні дані, які будуть відправлені віддаленому серверу. На цьому етапі може використовуватися алгоритм Бернарда Монтгомері для ефективної обробки обчислень..

3. Клієнт передає віддаленому обчислювачу підготовлену та зарані оброблену інформацію для подальших обчислень.
4. Після виконання обчислень на віддаленому обчислювачі, клієнт отримує результати роботи та завершує останні етапи обчислень.
5. У разі необхідності, клієнт взаємодіє з управлінням збору статистичних даних та висилає сигнал про вивід отриманих даних у файл для майбутньої обробки.

3.2. Управління взаємодією з віддаленими комп’ютерними системами.

Система управління взаємодією з віддаленими комп’ютерними системами може бути повно охарактеризована за допомогою структури, включаючи кількість підключених вузлів, їх взаємозв'язки та обчислювальну потужність. З метою ефективного моделювання функціонування хмарного обчислювача була розроблена структура класів, що включає:

1. Клас Node:

- Являє собою окремий вузол обчислювальної системи.

2. Клас Cloud_Controller:

- Має завдання визначення нової структури обчислювача, включаючи кількість вузлів та їх обчислювальну потужність.

- Здійснює передачу завдань вузлам та отримання результатів.

- Взаємодіє з моделлю мережі.

3. Клас Network_Controller:

- Відповідає за моделювання роботи системи між клієнтом та хмарним обчислювачем.

4. Клас Task:

- Реалізує інтерфейс та являється завданням обчислення.

3.3. Підсистема для завантаження статистичних даних.

Модуль збору статистики зосереджений на основній задачі підрахунку арифметичних операцій, таких як множення, ділення та додавання. Його функціональність тісно пов'язана з клієнтськими класами та інтерфейсом хмарного обчислювача. Основна мета модуля - забезпечення можливості виконання підрахунку числа операцій, необхідних для обчислення модулярного експоненціювання, але й розкриття окремої статистики для операцій, які відбуваються на клієнтській стороні.

Ключовим аспектом системи є здатність експорту зібраних даних в вигляді, який зручний для майбутньої роботи. З цією метою обрано формат CSV, що надає можливість легкої інтеграції з табличними редакторами, такими як Microsoft Excel або аналогічні. Це відкриває можливості для розширеного аналізу даних, статистичних операцій та побудови графіків в контексті подальших досліджень та оптимізації.

CSV (від англійського "comma-separated values", що перекладається як "значення, розділені комою", іноді вживається термін "character-separated values" - "значення, розділені символом") є форматом файлів, спеціально призначеним для представлення табличних даних. У цьому форматі кожне поле в рядках відокремлюється комою, а перехід на новий рядок позначається символом нового рядка. Поля, які включають коми, декілька рядків або лапки (позначені подвійними лапками), мають бути взяті в лапки з обох боків для коректного визначення меж полів. Цей формат забезпечує зручність інтеграції з

табличними редакторами, такими як Microsoft Excel або Google Sheets, сприяючи подальшому аналізу і обробці даних.

3.4. Компонент виконання процедури Монтгомері.

Впровадження алгоритму Монтгомері виявилося значним проривом у швидкості операцій піднесення до ступеня за модулем. Цей алгоритм використовує додаткові операції для переходу від складної операції ділення на вказане число N до більш ефективних бітових зсувів, змінюючи при цьому модуль на одну з ступенів двійки.

Додатково, у програмі реалізований окремий модуль для обчислення виконаних операцій. Це надає можливість порівнювати результати обчислень за різними методами та відслідковувати їх ефективність при змінних параметрах. Такий підхід дозволяє отримувати об'єктивні дані щодо продуктивності та вибору оптимального методу в залежності від конкретних вимог та умов використання.

У наступних частинах надано значно ретельнішу доповідь про функціонал цих систем.

Імплементована система застосовує 3 головні структури даних. По-перше, це структура для зберігання довгих цілих чисел, які використовуються для основ та експонент попереджаючи їх надсилання в хмарну систему (Blind_Payload). По-друге, це спосіб для зберігання результатів ділення (Division_Result). По-третє, це структура для представлення додатних довгих цілих чисел (Long_Integer), пояснення якої буде розглянуто в цій частині, оскільки ця архітектура має ключове значення для всього програмного забезпечення. Опис Blind_Payload та Division_Result буде представлено у відповідних пакетах для кращого зрозуміння їх функціоналу.

Внутрішній стан об'єкта Long_Integer зберігається у масиві цілих 32-бітних чисел. Для ефективної роботи з цим масивом реалізовано ряд методів, включаючи:

1. Метод get_Value(): Цей метод повертає цілий масив як єдну одиницю (get_).

2. Методи для роботи з окремими елементами масиву:

- get_Int(int): Повертає 32-бітне значення за вказаним індексом.

- set_Int(int, int): Встановлює 32-бітне значення за вказаним індексом.

- get_Long(int): Повертає 64-бітне беззнакове значення за вказаним індексом.

- setLong(int, long): Встановлює 64-бітне беззнакове значення за вказаним індексом.

3. Метод get_Size(): Для полегшення роботи з Long_Integer, цей метод повертає 0, якщо переданий індекс перевищує або дорівнює кількість елементів у масиві. Якщо встановлюється значення за індексом, що перевищує поточну кількість елементів, внутрішній масив автоматично розширюється для включення нового елементу за вказаним індексом.

Продовжуючи опис методів класу Long_Integer:

4. Метод stripLeadingZeroes(): Цей метод прибирає нульові 32-бітні блоки з початку числа. Після виклику цього методу розмір числа, який повертається методом get_Size(), може стати меншим, ніж був до виклику цього методу.

5. Методи зсуву числа ліворуч або праворуч на біт або вказану кількість біт:

- shiftL(): Здійснює зсув числа ліворуч на один біт.

- shiftR(): Здійснює зсув числа праворуч на один біт.

- shiftL(int): Здійснює зсув числа ліворуч на вказану кількість біт.

- shiftR(int): Здійснює зсув числа праворуч на вказану кількість біт. Крайні біти, що втрачаються під час зсуву, не впливають на розмір числа, який залишається незмінним.

6. Метод compare_2 (Long_Integer): Цей метод порівнює два числа Long_Integer.

Система програм включає чотири фундаментальні модулі: арифметика з довгими цілими числами (забезпечує виконання базових арифметичних операцій над LongInteger), модульні операції (містить методи обчислення залишку від ділення, модулярного множення та піднесення до ступеня),

безпечна експонентація (реалізована для застосування на віддаленому сервері), а також важливий допоміжний модуль.

Пакет longintegerarithmetics розроблено для виконання базових арифметичних операцій з об'єктами типу LongInteger. Цей пакет структурований у вигляді чотирьох ключових компонентів, які відображають різні аспекти виконання операцій: adder (додавання), subtrac2r (віднімання), Mul (множення) та divisor (ділення). Така архітектура сприяє ефективному управлінню та оптимізації арифметичних обчислень з урахуванням конкретних потреб користувача.

Зважаючи на конвенційний метод додавання чисел із багатьма розрядами в системі числення з основою2^{32}, можна виразити операцію додавання математично наступним чином:

$$Результат=(Доданок1+Доданок2)mod2^{32n}$$

Де Доданок1 та Доданок2 представляють числа типу LongInteger, а n - розмірність результату. Результат визначається як сума доданків, знята за модулем від 2^32n, забезпечуючи коректність арифметичних операцій в контексті конкретної системи числення.

Розмірність різниці повинна відповідати розміру зменшуваного числа. Важливим критерієм для вхідних даних є умова, що число, яке зменшується, повинне бути більшим або рівним від’ємнику — отже, різниця має бути нульовою або позитивною (згідно з визначенням класу LongInteger). Це обов'язкова умова. Також, аналогічно до реалізації інтерфейсу Adder, Subtrac2rImpl використовує класичний спосіб віднімання чисел, що складаються з багатьох цифр у системі числення з основою 232. Розмірність різниці відповідає розміру зменшуваного числа.

Для множення двох чисел типу LongInteger використовується пакет Mul. Він складається з інтерфейсу Mul і його реалізації MulImpl. Ця реалізація застосовує традиційний метод множення багатоцифрових чисел у 232-річній системі числення. Добуток має таку ж кількість цифр, як сума кількостей цифр множників.

Для ділення двох чисел типу LongInteger використовується пакет divisor. Він складається з інтерфейсу Divisor і його реалізації DivisorImpl. Ця реалізація застосовує алгоритм Д.Кнута для ділення багатоцифрових чисел у 232-річній системі числення. Результат ділення містить частку і залишок, які представлені у вигляді DivisionResult. Частка має таку ж кількість цифр, як ділене, залишок — як дільник.

Для виконання операцій за модулем, таких як обчислення залишку, множення і піднесення до степеня за модулем, використовується пакет Modoperations. Він складається з трьох підпакетів, які реалізують відповідні операції: Modcalcula2r, ModMul, ModExp.

Для обчислення залишку від ділення number на modulo використовується пакет modulocalcula2r. Він складається з інтерфейсу ModCalcula2r і його реалізації ModCalcula2rImpl. Ця реалізація використовує інтерфейс Divisor, який передається в конструктор класу, для ділення чисел і отримання об’єкту класу DivisionResult. З цього об’єкту повертається залишок від ділення. Результат має таку ж кількість цифр, як modulo.

Для множення двох чисел за модулем використовується пакет ModMul. Він складається з інтерфейсу ModMul і його реалізації ModMulImpl. Ця реалізація використовує той самий метод множення багатоцифрових чисел у 232-річній системі числення, що і MulImpl, але з однією відмінністю. На кожному етапі алгоритму результат зменшується за модулем. Для цього використовується інтерфейс ModuloCalcula2r, який передається в конструктор класу при створенні ModMulImpl. Можна використати існуючу реалізацію обчислення залишку, або створити і використати нову, більш оптимальну реалізацію інтерфейсу ModuloCalcula2r. Результат має таку ж кількість цифр, як модуль.

Для піднесення до степеня за модулем використовується пакет ModExp. Він складається з інтерфейсу ModExp і трьох його реалізацій: BinR2LModExp, MontgomeryExp і RemoteExp. Ці реалізації використовують різні методи для обчислення

BinR2LModExp і MontgomeryExp - це дві реалізації інтерфейсу ModExp для піднесення до степеня за модулем. Вони використовують різні двійкові схеми для обчислення

BinR2LModExp використовує схему “справа наліво”, а MontgomeryExp - схему Монтгомері.

RemoteExp - це клас, який використовує проксі-патерн для відправлення модулярного експонеціювання на сервер за допомогою REST-запитів. Він отримує відповідь від сервера, переводить її і повертає це значення. Цей клас приймає в конструкторі строку запиту і об’єкт RestTemplate, який належить до пакету org.framworkServer і знаходиться в Maven артефакті org.:-web. У цій програмі використано версію 4.7.5. Цей клас є частиною класу, який також імплементує інтерфейс ModExponena2r

Пакет secureexponentiation призначений для виконання безпечного віддаленого ескпонеціювання та містить три підпакети: blind, Server та Client, а також клас SecureExp.

Для “приховування” даних, які потрібно піднести до степеня за модулем, використовується пакет blind. Він генерує і зберігає BlindPayload - основи і експоненти, які використовуються для цього. Цей пакет складається з даних BlindPayload, інтерфейсу BlindPayloadProv і його реалізації BlindPayloadProvImpl.

Для генерації BlindPayload за експонентою і модулем використовується метод getBlindPayloadForExponent(LongInteger, LongInteger) інтерфейсу BlindPayloadProv. Він приймає ці два параметри і повертає BlindPayload, який складається з BlindExponent і BlindBase. Ці два компоненти задовольняють такі умови: • 0 < blindExpo – exponent < 232 • blindBase BlindExpo modul = 1

Для виконання алгоритму безпечного експонеціювання використовується клас SecureExp, який імплементує інтерфейс ModExp з пакету Modexponentiation. Пакет Server містить приклади використання і конфігурації цього класу: класи TestServer і ServerConfig. ServerConfig - це конфігурація фреймворку , яка базується на анотаціях. У цій конфігурації використано

стандартні реалізації інтерфейсів BlindPayloadProv, Subtrac2r, ModMul, а також вибрано MontgomeryExp для локального експонеціювання і RemoteExp для віддаленого експонеціювання.

Сервер отримує від клієнта REST-запит у шістнадцятковій системі числення, що містить основу, експоненту та модуль для модулярного експоненціювання. Прийняті дані перетворюються у формат LongInteger. Залежно від вибраного алгоритму ModExp, сервер обчислює модулярну експоненту у форматі LongInteger. Отримані результати перетворюються назад у шістнадцяткову систему числення та надсилаються клієнту у вигляді строки. Сервер базується на Maven артефактах фреймворку , Цей фреймворк включає в себе контейнер. У конфігурації сервера для реалізації використовується клас Exp.

Програма має графічний інтерфейс користувача, який складається з чотирьох вікон:

1. Вікно обрання режиму роботи, де користувач може вибрати одиночний експеримент, експериментів, налаштувань.
2. Вікно налаштувань одиночного експерименту, де користувач може ввести параметри (основа, показник, модуль) самостійно, з файлу або випадково. Розрядність параметрів дорівнює 2048. Також можна активувати збір статистики і збереження результату в файл.
3. Вікно налаштувань набору експериментів, де користувач може задати кількість наборів і експериментів в них (усереднена статистика кожного набору зберігається в файл), а також вибрати алгоритм для операції: алгоритм з пункту 2.2, або його модифіковані версії з попередньою обробкою показника - пошуком найбільшого підрядка або частотним аналізом.
4. Вікно налаштувань програми, в якому можна вказати шляхи до файлів виводу і статистики, а також налаштувати кількість вузлів на сервері і кількість ядер на клієнті.

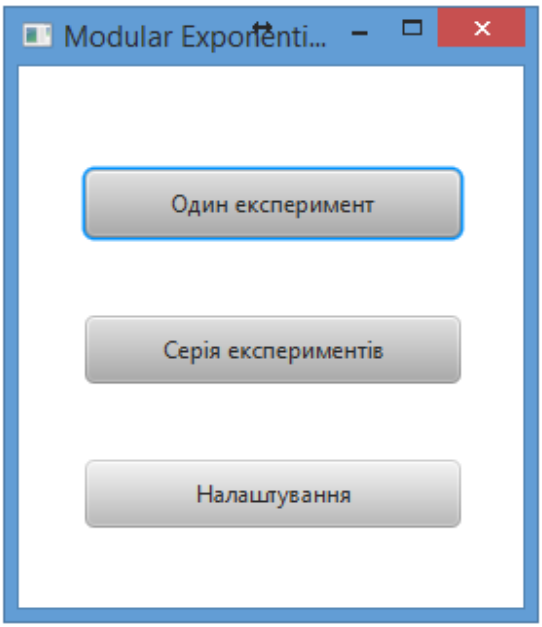

Рис 3.2. Головне вікно

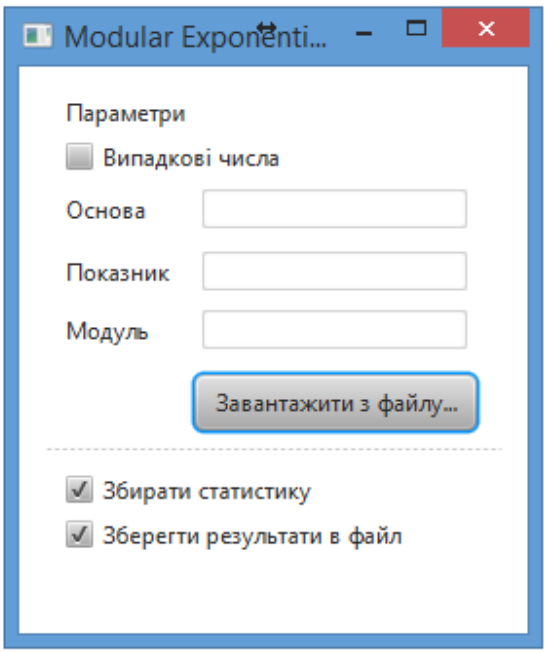

Рис 3.3. Вікно одиничного експерименту

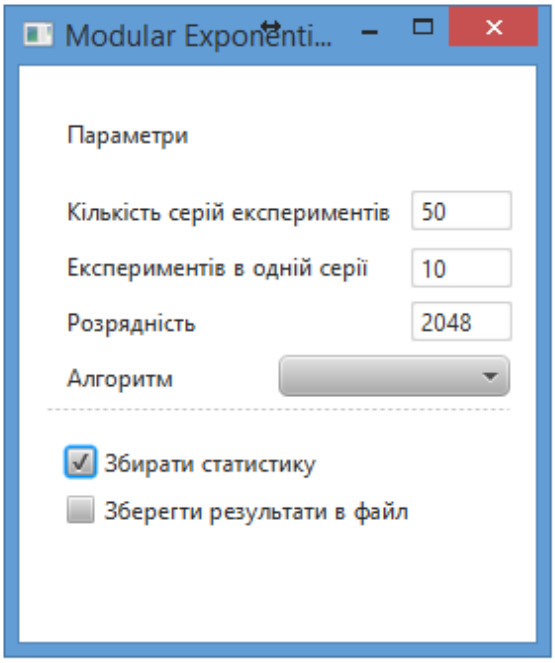

Рис 3.4. Вікно серії експериментів

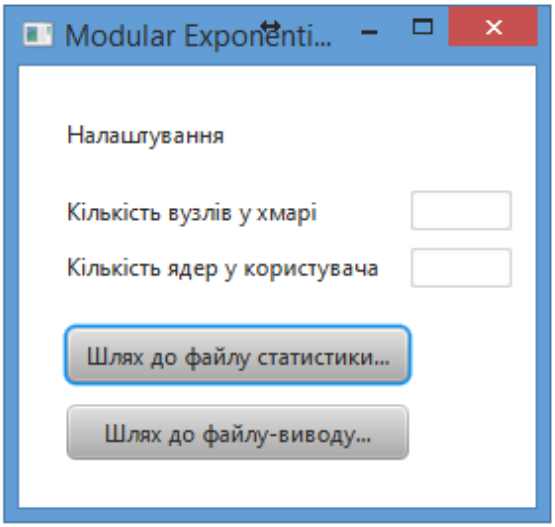

Рис 3.5. Вікно опцій програми

Для побудови і розгортання сервера потрібно виконати такі кроки:

• запустити консольну команду “mvn -clean -install” в головній директорії проекту.

• скопіювати файл secureexponentiation-Client-1.0.0.zip з директорії target модуля diploma-Client на сервер і розпакувати його. • перевірити, чи є шлях до JRE 7 в директорії classpath, і якщо ні, то додати його в classpath в файлі startClient.sh (для Unix- подібних систем) або (для Windows серверів);

• запустити файл startClient.sh (для Unix- подібних систем) або startClient.bat (для Windows серверів).

Щоб створити клієнта для безпечного експонеціювання, потрібно виконати такі дії:

• запустити і виконати консольну команду в головній директорії проекту для його збірки;

• перемістити файл з директорії target модуля diploma-base в директорію, де він може бути використаний в середовищі розробки;

• створити Java клієнта, використовуючи як зразок клієнта і конфігурації TestServer і ServerConfig; Середовище, в якому проводиться збірка проекту, повинно мати такі компоненти:

• система автоматизації збірки Apache Maven JDK8

Для розгортання сервера потрібна платформа, яка підтримує: Windows або Unix-подібна операційна система

3.5. Аналіз експерементального дослідження.

На рисунку 3.6 представлено результати дослідів, які були виконані в програмі, розробленій для цієї роботи. Для підвищення статистичної надійності дослідів, кожний експеримент повторювався 25 разів, а потім обчислювалося середнє значення результатів.

З графіка видно, що експеримент підтримує теоретичні міркування про те, що застосування запропонованого методу приблизно втричі знижує кількість операцій, які виконуються.

За допомогою експерименту було показано, що теоретичні висновки про те, що запропонований метод приблизно утричі скорочує кількість операцій, які виконуються на клієнті, мають підтвердження на графіку.

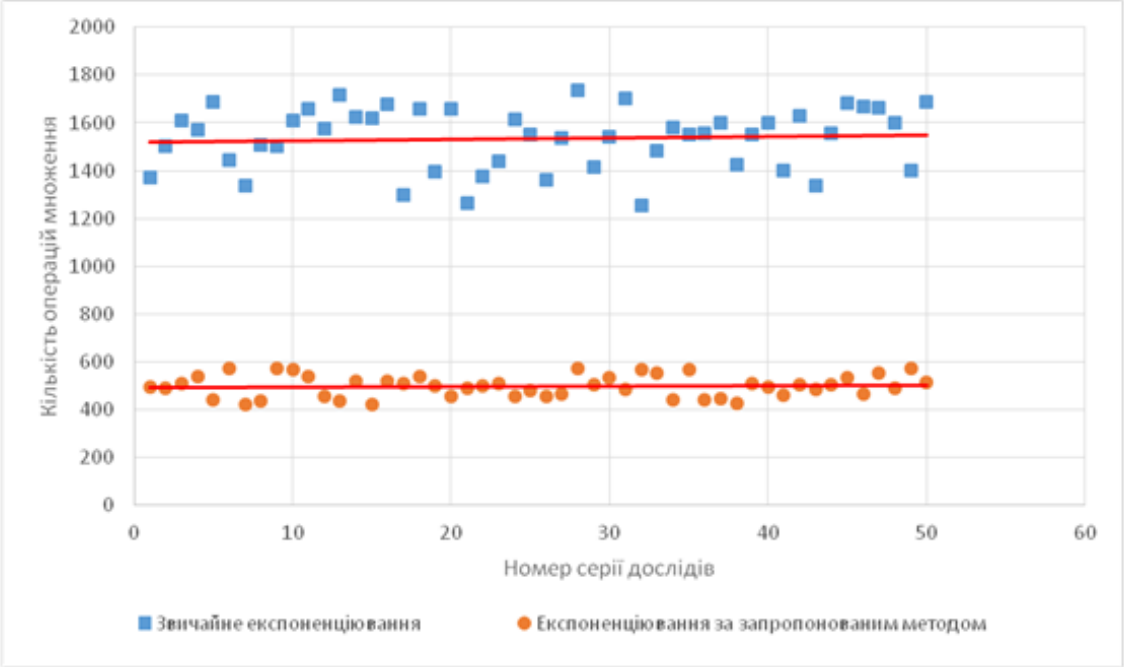

Рис. 3.6. Результати першої серії дослідів

Крім того, було проведено моделювання роботи запропонованого методу з попередньою обробкою показника. У цьому наборі експериментів оцінювалась загальна кількість операцій, які необхідно виконати. За результатами досліджень, використання запропонованих методів дозволяє зменшити від 10 до 20 відсотків операцій.

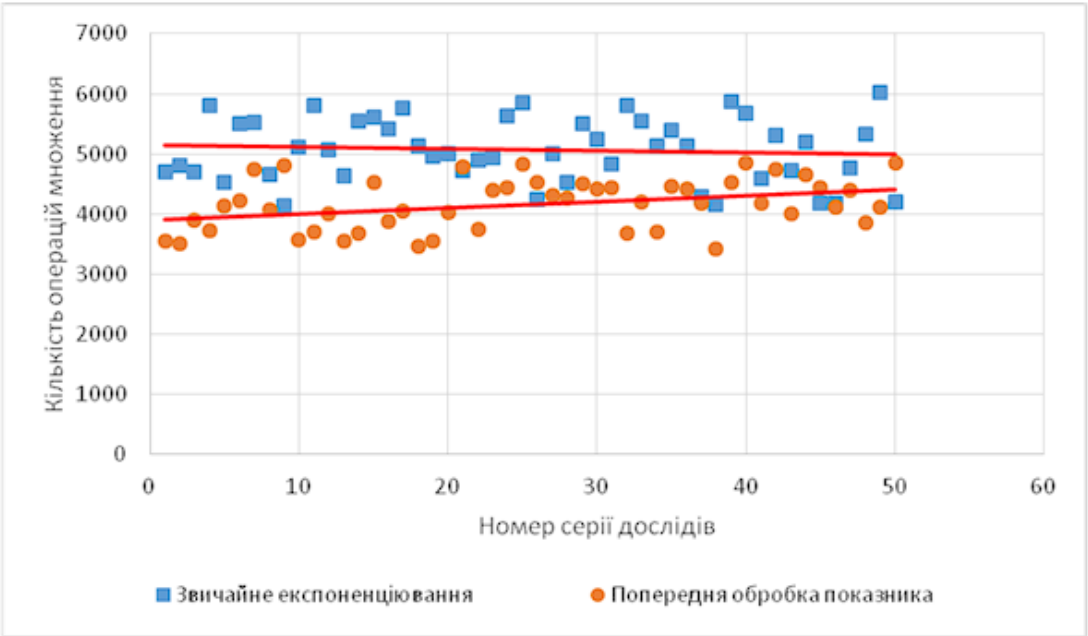

Рис. 3.7. Результати другої серії дослідів

Висновки до розділу 3.

Задля імплементації і дослідження розглянутого методу безпечного експонеціювання на сервері було розроблено програму. З розробки цієї програми можна зробити такі висновки:

1. Програма, яка дозволяє безпечно виконувати експонеціювання на віддаленому сервері, складається з двох компонентів: серверної частини, яка готова до встановлення і запуску, і базової бібліотеки, яка призначена для розробки клієнтських додатків. Бібліотека має гнучку архітектуру, засновану на використанні інтерфейсів, і дозволяє легко модифікувати клієнтські додатки для безпечного експонеціювання, додаючи різні реалізації цих інтерфейсів, наприклад, для модульного експонеціювання або модульної редукції.
2. Робота програми була перевірена за допомогою різних тестів. Крім того, програма була застосована для експериментальної оцінки ефективності запропонованого методу безпечного експонеціювання на сервері. Експерименти показали, що кількість операцій, які виконуються на термінальному пристрої, знижується в 65-75 разів, що сприяє прискоренню реалізації Інтернет способів збереження конфіденційності.
3. За допомогою програми було проведено моделювання, яке підтвердило теоретичні результати щодо характеристик роботи запропонованого методу. Зокрема, було встановлено, що кількість операцій, які виконуються на стороні клієнта, знижується приблизно в три рази.
4. Застосування способів попередньої обробки показника допомогає зменшити загальне число дій, які потрібні для завершення операції модульного експоненціювання на 15-25%.

РОЗДІЛ 4
РОЗРОБЛЕННЯ СТАРТАП–ПРОЕКТУ

4.1. Опис проблеми.

Провідною характеристикою сучасного розвитку усіх сфер людського життя є залучення інформаційних технологій для ефективного вирішення широких класів практичних задач. Зокрема існують такі типи задач, в галузях автоматизації та управління, ефективність вирішення яких може бути стрімко підвищенна за рахунок використання засобів інформаційних технологій. Саме інтеграція засобів інформаційних технологій в процеси вирішення практичних забезпечує більш високий рівень якості та точності отримуваних результатів.

Розширення інформаційної інтеграції виражається в стрімкому збільшенні використання мережевих технологій та постійній потребі збільшення потужності обчислювальних ресурсів, що застосовуються для вирішення прикладних задач. Таким чином, якскравим проявом інтеграції засобів інформаційних технологій в широкий спектр галузей стали хмарні технологій. Саме завдяки хмарним технологійям, широке коло користувачів отримало змогу користуватись практично необмеженими обчислювальними поужностями для вирішення своїх прикладних задач. Тепер у невеликих компаній та численних дослідницьких центрів зникла потреба купівлі власних потужних комп'ютерних систем та програмного забезпечення. Завдяки хмарним технологійям ці ресурси можуть братись в оренду на комерційній основі.

Окрім цього, хмарні технологій роблять рентабельним створення суперкомп'ютерів. В період, коли суперкомп'ютери не використовуються для вирішення безпосередньо тих стратегічних задач, задля яких вони були спроектовані, їхні обчислювальні потужності можуть бути розприділенні та ефективно використанні для вирішення інших задач. Тобто хмарні технології забезпечують більш ефективне використання обчислювальних потужностей в цілому, реалізуючи це в глобальному маштабі.

Попри всі описані переваги хмарних технологій, процес їхньої інтегреції, в напрямку хмарних обчислень, стримується. Основною причиною цього є

незахищенність даних, в процессі їхньої віддаленої обробки. В порівнянні з хмарними сховищами даних, для яких можливо виконати шифрування і цим забезпечити неможливість несанкціонованого доступу зловмисником до інформації, під час здійснення обчислень дані повинні бути в відкритому вигляді, або зашифрованні таким чином, щоб елемент криптографічного захисту не впливав на коректність отримуваного результату обробки. Тобто, основною проблемою при використанні хмарних обчислень є потреба в механізмах захисту даних, що можуть бути викоистаними під час виконання самих обчислень.

Існує кілька складностей реалізації захищенної обробки даних з використанням хмарних технологій. Перша з них полягає в тому що використовуваний інструмент криптографічного захисту не має впливати на отримуванний результат. Тобто повинна існувати процедура точного гарантованого відновлення оригінального результату з отриманих захищених обчислень. Це веде за собою потребу розробки індивідуальних засобів захисту даних під кожен тип їхньої обробки.

Іншою сладністю захисту даних під час їхньої обробки є віднаходження таких інструментів криптографічного захисту, використання яких потребувало менших ресурсів ніж безпосередня обробки цих даних на обчислювальних ресурсах користувача. Першочерговою задачею є досягнення збільшення ефективності розв’язку прикладних задач з використанням хмарних технологій, тобто процес захисту даних повинен використовувати таку кількість ресурсів, яка робитиме його ефективним в порівнянні з виконанням обробки цих даних на обчислювальних ресурсах користувача.

Окрім цього, елемент криптографічного захисту даних під під час їхньої віддаленої обробки повинен гарантувати високий рівеннь захищенності цих даних. Тобто ресурси, які потрібно затратити зловмиснику для доступу до інформації повинні мати більшу вартість ніж безпосередньо вартість інформації, що захищається.

Всі вищеназвані умови успішного застосування віддалелних захищенних обчислень є фактоами що суттєво стримують розповсюдження цього надрямкухмарних технологій. Особливістю цих умов є складність віднаходження достатньо надійних механізмів криптографічного захисту при одначасно низькій ресурсоємності шифрування і дешифрування. Проте вирішення ієї задачі відкриє шорокі можливості більш ефективного вирішення широкого класу прикладних задач. Зокрема, задач, рішення яких передбачає моніторинг об'єктів шляхом обробки та аналізу зображень отрианих з систем відеоспостереження.

Таким чином, наукова задача підвищення ефективності обробки зображень методом середньоарифметичної фільтрації, за рахунок залучення віддалених обчислювальних ресурсів в рамках хмарних технологій, являє важливу та актуальну задачу для сучасного етапу розвитку інформаційних технологій.

При виконанні віддаленої середньоарифметичної фільтрації зображень, користувач спершу надслає дані каналами Інтернет на віддалену багтопроцесорну систему, на якій виконується обробка цих даних. При цьому віддалена комп'ютерна система є непідконтрольною для користувача. Більше того, користувач може навіть не володіти інформацією про те, яка саме комп'ютерна система оброблює його дані. Таким чином, існує високий ризик виникнення наступних ситуацій:

1) Зловмисник може перехопити дані в процесі їхньої передачі на віддалену комп'ютерну систему. Оскільки, в рамках хмарних технологій для передачі даних використовуються канали Інтернет, то існує високий ризик перехоплення даних кваліфікованим спеціалістом з мережевих технологій, що має на меті отримання певної вигоди. При передачі даних каналами Інтернет використовуються протоколи, що гарантують певний рівень захищенності, проте цей рівень не є достатнім для захисту стратегічно важливих даних.

2) Зловмисник може здійснити спробу до даних, завдяки протникнення в власне багатопроцесорну систему, що їх. Віддалена комп'ютерна система, на якій здійснюються обчислення є повністю непідконтрольною користувачу,

тобто відповідальність за захист даних, що оброблюються повністю несе власник цієї системи. Таким чином, користувач не має ніяких гарантій щодо рівня захисту своїх даних, який забезпечується компанією, що надає свої пояслуги хмарних обчислень.

3) Оскільки обчислення відбуваються на непідконтроьній користувачу обчислювальній системі і користувач може навіть не володіти ніякою інформацією про цю систему, то існує викоий ризик доступу до даних власником цієї системи.

Особливо гостро дана проблема стоїть для систем моніторингу об'єктів з космосу, адже вартість знімків є надзвичайно високою. Окрім цього існують системи контролю, які використовуються правоохоронними структурами. Для них захищенність даних є стратегічно важливою умовою.

Для виключення можливостей несанкціонованого доступу зловмисником до зображень, що передаються каналами Інтернет та оброблюються на віддаленій комп'ютерній системі за допомогою середньоарифметичної фільтрації з метою підвищення їхньої якості, пропонується метод віддаленої захищенної середньоарифметичної фільтрації зображень з залученням хмарних технологій.

В рамках магістерської дисертації розроблена технологія захисту зображень, що передаються на віддаленні багатопроцесорні системи для виконання середньоарифметичної фільтрації з метою поліпшення їхньої якості. Метод, що пропонується, захищає зображення від спроб несанкціонованого доступу до них шляхом повної реконструкції секретних ключів та шляхом часткового відновленя зображень статистичними методами. При цьому запропонований метод використовує такі процедури криптографічного захисту, які є суттєво менш ресурсоємними в порівнянні з існуючими процедурами прискоренної незахищеної середньоарифметичної фільтраці. Тобто запропонований метод вирішує першочергову задачу прискорення середньоарифметичної фільтрації зображень.

Основні напрямки застосування розробленої ідеї та вигоди користувачам від її практичного застосування зведені в таблицю 4.1 для широкого круга користувачів для яких важливим є момент досягнення високої оперативності обробки зображень при збереженні їх конфеденційності.

Аналіз потенційних техніко-економічних переваг запропонованої ідеї (чим відрізняється від існуючих аналогів та замінників) порівняно із пропозиціями конкурентів передбачає:

1. визначення переліку техніко-економічних властивостей та характеристик ідеї ;
2. визначення попереднього кола конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку;
3. проводиться збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;
4. проводиться порівняльний аналіз показників: для власної ідеї визначаються показники, що мають а) гірші значення (W, слабкі); б) аналогічні (N, нейтральні) значення; в) кращі значення (S, сильні).
5. розробка фінансового плану є важливим етапом у впровадженні системи віддаленої реалізації криптографічних алгоритмів з відкритим ключем. Цей план повинен включати прогнозовані витрати на розробку та маркетинг продукту, а також очікувані прибутки та рентабельність. Фінансовий план допоможе забезпечити необхідні ресурси для успішного впровадження системи
6. вибір каналів продажу та розповсюдження продукту, таких як роздрібні магазини, онлайн-платформи або прямі продажі. Вибір правильних каналів може допомогти досягти більшої аудиторії та збільшити продажі
7. стратегія продажу та розповсюдження: Це включає вибір каналів продажу та розповсюдження продукту, таких як роздрібні магазини,

Таблиця 4.1.

Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Організація прискореної захищеної середньоарифметичної фільтрації зображень з залученням віддалених обчислювальних ресурсів в рамках хмарних технологій	1. В системах обробки аерокосмічних знімків з метою зондування земної поверхні, моніторингу її стану та контролю виникнення природних катаклізмів.	Підвищення якості контролю за рахунок вчасної обробки зображень. Це досягається за рахунок прискорення виконання базової процедури підвищення якості знімків, а саме їхньої середньоарифметичної фільтрації.
	2. В системах моніторингу переміщення людей шляхом їх розпізнавання	Підвищення якості спостереження шляхом збільшеня актуальності оброблюваних даних, що доясягається за рахунок прискорення попередньої обробки зображень.
	3. В системах аналізу зображень медичного спрямування, а саме флюрографічних знімків, УЗД знімків	Збільшення ефективності системи завдяки можливості обробки більшої кількості знімків за одиницю часу, що має наслідком збільшення аудиторії.

4.2. Аналіз ринкових можливостей запуску стартап-проекту.

Основними характеристиками розробленого проекту організації прискореної захищеної середньоарифметичної фільтрації зображень з залученням віддалених обчислювальних ресурсів в рамках хмархих технологій можна вважати:

- Рівень захищеності від спроб повної реконструкції зашифрованого зображення – вимірюється з витратах зловмисника на перебір секретних ключів, що можуть бути використані в рамках запропонованого методу.
- Рівень захищеності від спроб наближеного відновлення зображень, тобто реконструкції основних контурів – вимірюється з витратах зловмисника для здійснення статистичного аналізу великої кількості захищених зображенья також відновлення одного з двох використовуваних типів секретних ключів.
- Об'єм додаткових часових ресурсів на реалізацію криптографічних елементів захисту зображень - вимірюється в процентному збільшенні/зменшенні часу виконання програми.
- Об'єм додаткових ресурсів пам'яті, необхідних для реалізації запропонованих методів захисту зображень - вимірюється в об’ємі додаткових ресурсів пам'яті в Кбайтах.

В якості конкуруючих проектів можна розглядати проект SPECTR-4M обробки аерокосмічних зображень з залученням віддалених хмарних обчислень [64]. В ньому реалізується захищенна організація медіанної фільтрації зображень з залученням хмарних обчислень. Технічна реалізація методу передбачає впорядкування вибірки кольорів і співставлення їм у відповідність проміжки поступово розбитого інтервалу. За рахунок цього виконується основна вимога медіанної фільтрації – збереження порядку числа, що робить можливим порівняння шифрованих аналогічним до порівняння реальних значень. Це забезпечує коректність дешифрування.

Секретність перетворень, що виконуються в процесі інтервального методу шифрування забезпечується зарахунок того, що таблиця інтервалів значень, що відповідають одному конкретному реальному значення генерується самим користувачем і використовується ним в якості секретного ключа.

Представлений в розробці проект забезпечує більш високий рівень захищеності від спроб доступу до зображень за допомогою інструментів статистичного аналізу, в порівнянні з конкурентами.

Іншим конкуруючим проектом є MAGM, в якому реалізується захищенна обробка зображень методом середньоарифметичної фільтрації з залученням віддалених хмарних обчислень [28]. В якості захисту використовується додавання до кожного із операнду випадкового цілого, кратного певному модулю m. Дешифрування виконується шляхом знаходження залишку від ділення суми на модуль m, за умови, що модуль m більший за суму. Дешифрування значення елементу після операції ділення можливе за умови, що ділене буде кратним як подільнику, так і модулю m. Це вводить значні обмеження на секретні ключі методу, що має наслідком різке зменішення об'єму ресурсів потрібних для перебору всіх ключів.

Представлений в розробці проект забезпечує вищий рівень захисту від спробо повного відновдення зображень в порівнянні з методом реалізований системаю MAGM.

У таблиці 4.2 наведено порівняльні характеристики проекту з конкурент-тами технологіями.

Таким чином, представлений в розробці проект забезпечує суттєве прискорення виконання фільтрації зображень в порівнянні з конкурентами. Також, представлений проект гарантує високий рівень захищеності даних від повної реконструкції зображення та спроб наближеного відновлення зображень статистичними методами.

Рівень захищенності зображень в конукуруючих проектах реалізовано також на достатньо високому рівні, проте вони здатні забезпечити знечно менше ефективність обробки зображень з точки зору швидкодії. Тобто, представлений в розробці проект забезпечує значно вищий рівень прискорення фільтрації.

Проте з урахуванням сучасної тенденції до здешевлення апаратної пам'яті комп'ютерних систем, в розробленому проекті вдало вирішено компоміс між виграшем в часі та програшем в використуваних ресурсах пам'яті.

Таблиця 4.2

Визначення сильних, слабких характеристик ідеї проекту

№ пп	Техніко-економічні характеристики ідеї	Оцінка концепції конкурентів			W	N	S
		Мій проект	SPECTR-4M	MAGM			
1	Рівень захищеності від спроб повної реконструкції зашифрованого зображення	2 500 000	70 000	90 000			+
2	Рівень захищеності від спроб наближеного відновлення зображень, тобто реконструкції основних контурів	50 000	80 000	50 000		+	
3	Об'єм додаткових часових ресурсів на реалізацію функцій захисту	-100%	30%	70%			+
4	Об'єм додаткових ресурсів пам'яті на реалізацію функцій захисту	110	90	40	+		

Разом з тим, запропонований в проекті метод потребує дещо більших ресурсів пам’яті, адже передбачає використання великої кількості секретних ключів. Секретні ключі представляють собою повноцінні зображення та системи лінійних перетворень, представлених в матричному вигляді.

Важливою ланкою розробки стартапу є аудит технології, за допомогою якої можна реалізувати ідею проекту. Вихідним результатом проекту є програмне забезпечення. Визначення технологічної здійсненності ідеї проекту передбачає аналіз таких складових:

8. за якою технологією буде реалізовано програмний продукт відповідно до ідеї проекту ?
9. чи існують такі технології, чи їх потрібно розробити/доробити ?
10. чи доступні такі технології авторам проекту ?

Вихідним результатом проекту є програмне забезпечення, яке реалізує захищену середньоарифметичну фільтрацію зображень. Операція середньоарифметичної фільтрації є стандартизованою, достатньо повно дослідженою для різних типів зображень та їх кодування. Для практичної реалізації середньоарифметичної, як і медіанної фільтрації дотепер створені програмні засоби, орієнтовані на рівні типи обчислювальних платформ. Відповідно, розробка програмного забезпечення здійснюється за добре відомими технологіями програмної інженерії відповідними фахівцями. Тому технології для отримання результату проекту вже існують і ці технології є загальновідомими та загальнодоступними.

Наступним етапом стартапу є визначення ринкових можливостей програми, яка створена на основі проекту. Ці можливості можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Значну ринкову привабливість проекту додає динамічний розвиток робототехнічних комплексів на основі концепції технічного зору та штучного інтелекту. За даними [39] створення таких комплексів для масової заміни людьми роботизованими комплексами на основі технічного зору стане пріоритетним напрямком в найближче десятиліття.

Чільне місце в цих системах відіграють системи обробки зображень, яке для портативних вбудованих мікроконтролерів може відбуватися в режимі реального часу за рахунок залучення віддалених обчислювальних потужностей. Важливим етапом розробки стартапу є визначення потенційних клієнтів ринку программного забезпечення.

Таблиця 4.3

Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	100
2	Загальний обсяг продаж, (для сектора)	50000
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Значна конкуренція, велика кількість аналогів
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	28%

З даних таблиці 4.3 слідує, що, в цілому, за проведеним попереднім оцінюванням, ринок програм для захисту зображень в процесі їх віддаленої обробки і, зокрема їх фільтрації для видалення імпульсних завад, є привабливим для входження.

Іншою зацікавленою стороною є правоохоронні організації та приватні охоронні бюро. Це пов’язано з потребою надзвичайно швидкої обробки результатів моніторингу. В разі, якщо дані не оброблюватимкуться своєчасно, компанії та організації цієї сфери не зможуть якісно надавати свої послуги і будуть нести грошові збитки через недотриманння умов співпраці зі своїми клієнтами.

Проведений аналіз ринкової спроможності показав, що запропонований програмний продукт викликає зацікавленість у широкого кола компаній та організацій, що є учасниками ринку. Найбільшу зацікавленість у розроблених

Таблиця 4.4

Аналіз ринку збуту

№ пп	Потреба, що формує ринок	Цільові сегменти ринку	Відмінності у поведінці різних потенційних цільових груп	Вимоги споживачів до товару
1	Потреба швидкої обробки потоків аерокомічних зображень	Державні організації та установи, приватні компанії, які займаються картографією та моніторингом навколишнього середовища.	Жорсткі обмеження на часові параметри	Надійний захист зображень під час їхньої віддаленої обробки
2	Забезпечення високої оперативності моніторингу стану правопорядку	Правоохоронні органи, приватні охоронні компанії та бюро.	Жорсткі часові обмеження, наявність перешкод, активне втручання в роботу	Достовірність даних, що надтодять обробленими з віддаленої комп'ютерної системи
3	Забезпечення конфіденційності обробки великої кількості зображень медичного спрямування	Лікарні, приватні компанії, що займаються обстеженням та дослідженням захворювань	Активне втручання в роботу	Конфіденційність оброблюваних зображень в процесі їхньої віддаленої обробки та достовірність результатів

Це пов’язано з потребою своєчасної обробки потоків з мільйонів зображень в межах орстких часових рамок.

У разі неможливості забезпечення достатньої швидкості обробки зображень, можуть виникнути ситуації, при яких важливі зміни об’єкту чи події не були своєчасно виявленими та локалізованими. Тобто, в таких ситуаціях компанії нестимуть великі збитки через невиконання своїх основних задач.

Значну зацікавленість до прискорення фільтрації зображень проявляють лікарні та компанії, що займаються обстеженнями та дослідженнями в сфері медицини. Вони потребують обробки великої кількості знімків, конфіденційність яких має бути збереженою у зв’язку з етичним аспектом.

Окрім цього, існує велика кількість людей зацікавлених в доступі до інформації про стан здров’я певної людини, з метою отримання грошової вигоди. Тобто, лікарні та компанії, потребують обробки великої кількості власних знімків з використанням віддаленх ресурсів, задля можливості підвищення якості, точності та оперативності надання своїх послуг.

Таким чином, розроблений проект задовільняє інтереси кілької маштабних груп-учасників ринку і, при цьому є конкурентноспроможним.

В таблиці 4.5 представлено складання SWOT-аналізу (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities).

Окрім цього, для задач, при вирішення яких основна інформаційна складова міститься в деталях, запропонований метод забезпечує на порядки вищий рівень захисту. Це досягнуто за рахунок використання кількох типів секретних ключі в межах обробки одного зображення

Також, ринок збуту може бути розширено невеликими дослідницькими центрами та компаніями невеликого масштабу.

Розробка може бути ефективно використана в перспективних робото технічних комплексів з функціями технічного зору, що працюють в режимі реального часу і реалізують функції сприйняття та розпізнавання зображень.

Таблиця 4.5

SWOT- аналіз стартап-проекту

Сильні сторони: Метод реалізує концепцію єдиного власника секретної інформації і, таким чином, забезпечує захищенність зображень. Розроблений метод забезпечує більший рівень прискорення обробки зображень в порівнянні з конкуруючими проектами, тобто вирішує посталену задачу на порядок ліпше, в порівнянні з конкуруючими проектами.	Слабкі сторони: Слабкою стороною є потреба в додатковій пам'яті для збереження секретних ключів методу. Також відповідальність за захищенність зберігання секретних ключів методу несе користувач.
Можливості: Розширення ринку збуту та закріплення методу як стандарту в віддаленої обробки зображень спрямованої на підвищенння їхньої якості, шляхом видалення імпульсних завад.	Загрози: Компанія зобов'язана дбати про захщенність даних, що використовуються в якості секретних ключів та про дотримання рекомендацій, щодо використання розробленого продукту. В зворотньому випадку рівень захищенності даних може суттєво знизитись, що призведе до можливостей несанкціонованого доступу до них зловмисником.

Висновки до розділу 4.

Проведенні дослідження, спрямованні на виявлення і оцінку можливостей ринкового впровадження розробленого проекту, що складає магістерську дисертацію дозволяють зробити наступні висновки:

1. Розроблений програмний продукт та ідея, що покладена в його основу мають велику конкурентоздатність в силу того, що здатні забезпечити на порядок вищий рівнень ефективності з точки зору прискорення обробки зображень, що й лежить в основі поставленої мети досліджень. Окрім цього, розроблений метод конкурує з існуючими проектами в аспекті забезпечення захищенності оброблюваних даних. Зорема, він забезпечує достатньо високий рівень захисту від часткового відновлення зображення статистиними методами. Тобто, він на такому ж рівні захищає зображення від відновлення лише основних контурів, що для деяких класів задач є стратегічно важливою інформаціїною складовою знімку..

2. Існує широке коло потенційних покупців програмного забезпечення для обробки зображень з високим рівнем оперативності або навіть в режимі оперативного часу. До основних зацікавлених в запропонованій розробці учасників ринку відносяться фірми, що займаються моніторингом об’єктів з метою відстлідковування природних катаклізмів, правоохоронні організації та приватні охоронні структури, лікарні, дослідницькі центри та компанії що працють в сфері медицини.

3. Проведений аналіз показав відносно незначний рівень конкуренції в сфері віддаленної захищенної обробки зображень. Це пов’язано з тим, що існуючі на разі проекти не здатні забезпечити достатньої швидкості обробки даних, яка могла б задовільнити основних учасників ринку збуту, а саме компанії, що займаються моніторингом об’єктів, та організації, які зацікавленні в моніторингу правопорядку.

ВИСНОВКИ

В результаті виконання магістрської дисертації, направленої на вирішення задачі розробки ефективної організаціїрозподіленого обчислення модулярної експоненти на тремінальній платформі систем управління на базі IoT і хмарнихобчислень отримані результати, які можуть бути сформульовані наступним чином:

1. Використання технологій Інтернету речей, яке базується на застосування в якості середовища обміну даними Інтернет дозволяє значно підвищити ефективність широкого класу систем моніторингу стану об’єктів реального світу, а також управління ними. Разом з тим, використання потенційно вразливого середовища глобальних мереж створює загрози стороннього втручанні в роботу згаданих систем. Для її вирішення потрібно застосовувати повний спектр засобів криптографічного захисту даних, передбаченими протоколами мережевої безпеки. Частина цих засобів базується на складних в обчислювальному плані операціях модулярного експоненціювання, що виконуються над числами, розрядність яких на порядки перевищує розрядність термінальних комп’ютерних платформ систем комп’ютерного управління. Ці платформи, виходячи із специфіки їх використання, мають низьку продуктивність, зумовленою вимогам до низького споживання потужності, компактності, малої вартості. Відповідно, об’єктивно виникають труднощі реалізації складних обчислень модулярного експоненціювання на таких платформах в режимі реального часу.

2. Проведений аналіз існуючих алгоритмів модулярного експоненціювання засвідчив, що обидва їх варіанти мають строго послідовний характер і не можуть бути розпаралелені. Огляд існуючих підходів до прискорення виконання базової операції модулярного експоненціювання показав, що вони базуються на використанні передобчислень, які потребують значних за обсягом об’ємів пам'яті, що недопустимо для компактних термінальних мікроконтролерів. За цієї ж причини не можна використовувати спеціалізовані потужні криптопроцесори, які на апаратному рівні реалізують операцію модулярного

експоненціювання. Виходячи з цього, найбільш доцільне рішення полягає в залученні віддалених обчислювальних потужностей в рамках хмарних технологій.

3. Огляд існуючих методів обчислення модулярної експоненти з залученням віддалених обчислювальних потужностей показав, що в свої більшості вони організовані у вигляді двох обчислювальних процесів, один із яких реалізується на термінальній обчислювальній платформі, а інший на віддаленій. Для виключення можливості відновлення секретного коду експоненти на віддалених системах застосовується адитивне її розкладення на дві компоненти, одна з яких оброблюється на термінальному мікроконтролері, а інша – на віддаленій системі. Рівень захищеності, при цьому визначається кількістю розрядів коду експоненти, які оброблюються на термінальній обчислювальній платформі. Цим чинником обмежується часова ефективність використання хмарних технологій.

4. Проведений аналіз показав, що в основі переважної більшості існуючих підходів до підвищення ефективності обчислювальної реалізації криптографії з відкритим ключем в системах комп’ютерного систем моніторингу і управління віддаленими об’єктами на базі технологій IoT лежить розкладення секретного коду експоненти на адитивні та мультиплікативні складові, яке дозволяє ефективно організувати розподілене обчислення між термінальними та віддаленими обчислювальними платформами. Показано, що рівень захищеності від спроб відтворення коду експоненти на віддалених обчислюваних потужностях за даними, що на них передаються, залежить від кількості операцій модулярного множення, які здійснюються на термінальному мікроконтролері.

5. Теоретично обґрунтовано, розроблено та досліджено захищену реалізацію модулярного експоненціювання на віддалених комп'ютерних системах, який відрізняється тим, що секретний код експоненти розкладається на мультиплікативно-адитивні компоненти, частина з яких використовується при віддалених обчисленнях, а частина – на термінальному мікроконтролері. Теоретично та експериментально показано, що запропонований метод дозволяє

підвищити швидкість реалізації криптографічних алгоритмів з відкритим ключем на термінальних мікроконтролерах систем комп'ютерного управління в середньому у 70 разів. При цьому надійно забезпечується захист від ренонструкції секретних компонентів криптосистеми за даними, що надаються на віддалені системи.

6. Розроблено метод розподіленого обчислення базової операції криптографії з відкритим ключем – модулярного експоненціювання з використанням хмарних технологій, відмінність якого полягає в організації передачі проміжних даних з віддалених систем на термінальний мікроконтролер для зменшення обчислювального навантаження на нього, що дозволяє збільшити кількість розрядів секретного коду експоненти, що обробляються на ньому і, відповідно, на 2-3 порядки підвищити рівень захищеності від спроб незаконної реконструкції коду експоненти на віддалених комп’ютерних системах, за даними, що надаються на них.

7. Програма, яка дозволяє безпечно виконувати експонеціювання на віддаленому сервері, складається з двох компонентів: серверної частини, яка готова до встановлення і запуску, і базової бібліотеки, яка призначена для розробки клієнтських додатків. Бібліотека має гнучку архітектуру, засновану на використанні інтерфейсів, і дозволяє легко модифікувати клієнтські додатки для безпечного експонеціювання, додаючи різні реалізації цих інтерфейсів, наприклад, для модульного експонеціювання або модульної редукції.

8. Робота програми була перевірена за допомогою різних тестів. Крім того, програма була застосована для експериментальної оцінки ефективності запропонованого методу безпечного експонеціювання на сервері. Експерименти показали, що кількість операцій, які виконуються на термінальному пристрої, знижується в 65-75 разів, що сприяє прискоренню реалізації Інтернет способів збереження конфіденційності.

9. За допомогою програми було проведено моделювання, яке підтвердило теоретичні результати щодо характеристик роботи запропонованого методу.

Зокрема, було встановлено, що кількість операцій, які виконуються на стороні клієнта, знижується приблизно в три рази.

10. Застосування способів попередньої обробки показника допомогає зменшити загальне число дій, які потрібні для завершення операції модульного експоненціювання на 15-25%.

11. Розроблений програмний продукт та ідея, що покладена в його основу мають велику конкурентоздатність в силу того, що здатні забезпечити на порядок вищий рівнень ефективності з точки зору прискорення обробки зображень, що й лежить в основі поставленої мети досліджень. Окрім цього, розроблений метод конкурує з існуючими проектами в аспекті забезпечення захищенності оброблюваних даних. Зорема, він забезпечує достатньо високий рівень захисту від часткового відновлення зображення статистиними методами. Тобто, він на такому ж рівні захищає зображення від відновлення лише основних контурів, що для деяких класів задач є стратегічно важливою інформаціїною складовою знімку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Elgazzar Khalid. Revisiting the internet of things: New trends, opportunities and grand challenges /Khalid Elgazzar, Haytham Khalil, Taghreed Alghamdi, Ahmed Badr, Ghadeer Abdelkader Abdelrahman Elewah and Rajkumar Buyya // Frontiers in Internet of Things.-2022.-Vol.1.- P.1-18. doi.org/10.3389/friot2022.1073780.
2. Kumar Sachin. Internet of Things is a revolutionary approach for future technology enhancement: review/ Sachin Kumar, Prayag Tiwari, Mikhail Zymbler // Journal of Big Data. -2019.- № 6.- P.63-71.
3. Jurcut A.D., Xu R.R. Introduction to IoT Security/ In IoT Security: Advances in Authentication. – 2020.- pp. 1-64. DOI:10.1002.9781119527978.ch2.
4. Unal D., Ali-Ali A., Catak F.O. A secure and efficient Internet of Things cloud encryption scheme with forensics investigation compatibility based on identity-based encryption. Future Generation Computer Systems. Vol. 125. 2021.- pp. 433-445. DOI: 10.1016 /J.FUTURE.2021.06.050.
5. Bardis N. Secure, Green Implementation of Modular Arithmetic Operations for IoT and Cloud Applications. In book Green IT Engineering: Components, Networks and System Implementation. - 2017.- pp. 43-64. DOI: 10.1007/978-3-319-55595-9_3.
6. Meneghello F., Calore M., Zucchetto D., Polese M., Zalella A. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. IEEE Internet of Things Journal. Vol. 6.-No.5.- 2019.- pp.8182-8201. DOI: 11.1109/JIOT.2019.2935189.
7. Menezes Alfred, Handbook of Applied Cryptography. / Alfred Menezes, Paul C. van Oorschot, and Scott A. Vanstone // CRC Press. – 2001. – 780 p.
8. Boiarshyn A. Organization of parallel execution of modular multiplication to speed up the computational implementation of public-key cryptography/ A. Boiarshyn , O. Markovskyi, B. Ostrovska // Information, Computing and Intelligent systems. – 2022. – № 3.- P.26-32.
9. Markovskyi O.P. Secure Modular Exponentiation in Cloud Systems./ O.P. Markovskyi, N.Bardis, N.Doukas, S.Kirilenko. //Proceedings of The Congress on

Information Technology, Computational and Experimental Physics (CITCEP 2015), 18-20 December 2015, Krakow, Poland, C. 266-269.

10. Noot M.M. Current research on Internet of Things (IoT) / M.M. Noot, W.H. Hassan. // Compute Network. -Vol.148. -No.15. -2019. -pp.283-294.

11. Khan M.A. IOT security: Review, blockchain solution and open challenges / M.A. Khan, K. Salah // Future Generation Computer Systems. -Vol.82. -No.5. -2018. -pp.395-411.

12. Fox G. Using Clouds for Technical Computing / G. Fox, D. Gannon // Cloud Computing and Big Data. Amsterdam: IOS Press. -2018. -pp.81-102.

13. Boroujerdi N. Cloud Computing: Changing Cogitation about Computing / N. Boroujerdi, S. Nazem // IJCSI International Journal of Computer Science Issues, -Vol. 9, -Issue 4. -2012. -No.3. -pp.169-180.

14. Самофалов К.Г., Рамзи Анвар Салиба Сунна, Левчун Д.Ю. Ускоренная реализация модулярного экспоненцирования на малоразрядных микропроцессорах и встроенных микроконтроллерах.// Проблеми інформатизації та управління. Збірник наукових праць: Випуск 4(15).-К.,НАУ.- 2005.- С.144-153.

15. Bos J.N., Coster M. Additional chain heuristics.// Cryptographic Hardware and Embedded System- CHES'2014. LNCS-2116, Springer-Verlag.- 2014.- P.143-151.

16. Boyko V., Pienado M., Venkatesan R. Speeding up log and factorization based schemes via precomputations // Advances in cryptography –Proceeding of EUROCRYPT'98, LNCS-658, Springer-Verlag.- 1998.-P. 221-235.

17. Костенко Ю. В. Метод защищенного модулярного экспоненцирования на удаленных компьютерных системах / Ю. В. Костенко, А.П. Марковский, О.В. Русанова // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка. К.: ТОО „ВЕК+". -№ 64. -2016. -С. 51-54.

18. Буйбарова М.Ф. Метод захищеної реалізації перетворень Фур'є на віддалених розподілених комп'ютерних системах / Буйбарова М.Ф., Виноградов Ю.М., Приймак В.Ю. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка. К.: ТОО „ВЕК+".- № 64.- 2016.- С. 64-71.

19. Haches G., Quisquater J.J., Montgomery multiplication with no final subtraction.// Cryptographic Hardware and Embedded System- CHES'2013. LNCS-2465, Springer-Verlag.- 2013.- P.293-301.

20. Hong S.M., Oh S.Y., Yoon H. New modular multiplication algorithms for fast modular exponentiation // Proceeding of Advances in Cryptology Eurocrypt'96, LNCS-1070, Springer-Verlag.- 1996.- P. 166-177.

21. Задірака В.К., Олексик О.С. Комп'ютерна арифметика багато розрядних чисел: Наукове видання.-К.:Вища школа.-2003.-264 с.

22. Коутинхо С. Введение в теорию чисел. Алгоритм RSA. М.: Постмаркет.- 2001.- 323.

23. Кнут Д. Искусство программирования для ЭВМ. Т.2. Получисленные алгоритмы. М.:Мир.- 1977.- 843 с.

24. Макконелл Д.Х. Основы современных алгоритмов. М.: Вильямс.- 2004.-512 с

25. Mirataei A. Fast secure calculation of the open key cryptography procedures for iot in clouds/ A. Mirataei, M. Haidukevych, O. Markovskyi // Information, Computing and Intelligent systems. – 2022. – № 3.- P.56-62.

26. Марковський О.П. Захищена реалізація фільтрації зображень в GRID-системах / Марковський О.П., Невдащенко М.В., Білашевська А.М. // Вісник Національного технічного університету України "КПІ" Інформатика, управління та обчислювальна техніка, – Київ: ВЕК+ – 2014. – № 61. - С.105-109.

27. Молдовян А.А., Молдовян Н.А. Введение в криптосистемы с открытым ключом. С-Пб.: БХВ-Петербург,- 2004.-322 с.

28. Jose Alberto de Jesus Borges. A Secure Cloud Computing Method for Rapid Implementation of Cryptographic Data Protection in IoT / Jose Alberto de Jesus Borges, Haidukevych Oleksandra, Mirataei Alireza, Nikos Doukas. // Proceeding of The 13th IEEE International Conference on Dependable Systems, Services and Technologies, DESSERT'2023 13-15 October, 2023, Athens, Greece. – IEEE publication. – 2023. – pp. 50-53.

ДОДАТОК А. ЛІСТИНГ ПРОТОТИПУ СХЕМИ

```
1.  public void shiftRight() {
2.      boolean oneFromLeft = false;
3.      for (int i = this.getSize() - 1; i >= 0 ; i--) {
4.          long newValue = getLong(i) >> 1;
5.          if (oneFromLeft) {
6.              newValue += TWO_POWER_31;
7.          }
8.          oneFromLeft = Integer.lowestOneBit(this.getInt(i)) == 1;
9.          this.setInt(i, (int)newValue);
10.     }
11. }
12. public void shiftRight(int numberOfBit) {
13.    for (int i = 0; i < numberOfBit; i++) {
14.        shiftRight();
15.    }
16. }
17. public int[] getValue() {
18.     return Arrays.copyOf(value, value.length);
19. }
20. public int getInt(int index) {
21.     return getIntOrZero(index);
22. }
23. public void setInt(int index, int value) {
24.     checkIndex(index);
25.     this.value[index] = value;
26. }
27. private void checkIndex(int index) {
28.     if (index >= getSize()) {
29.         this.value = Arrays.copyOf(this.value, index + 1);
30.     }
31. }
32. public long getLong(int index) {
33.     return Integer.toUnsignedLong(getInt(index));
34. }
35. public void setLong(int index, long value) {
36.     setInt(index, (int)value);
37. }
38. public int getSize() {
39.     return value.length;
40. }
41. public void stripLeadingZeroes() {
42.     int realLength = value.length;
43.     for (int i = value.length - 1; i >=0; i--) {
44.         if (value[i] == 0) {
45.             realLength--;;
46.         } else {
47.             break;
48.         }
49.     }
50.     value = Arrays.copyOf(value, realLength);
51. }
```

```
52.    public void shiftLeft() {
53.        boolean oneFromRight = false;
54.        for (int i = 0; i < this.getSize() ; i++) {
55.            int newValue = this.getInt(i) << 1;
56.            if (oneFromRight) {
57.                newValue += 1;
58.            }
59.            oneFromRight = this.getInt(i) < 0;
60.            this.setInt(i, newValue);
61.        }
62.    }
63.    public void shiftLeft(int numberOfBit) {
64.        for (int i = 0; i < numberOfBit; i++) {
65.            shiftLeft();
66.        }
67.    }
68.    public int compareTo(LongInteger that) {
69.        int maxLength = Math.max(this.getSize(), that.getSize());
70.        int comparisionResult = 0;
71.        for (int i = maxLength - 1; i >= 0; i--) {
72.            comparisionResult = Integer.compareUnsigned(this.getIntOrZero(i),
    that.getIntOrZero(i));
73.            if (comparisionResult != 0) {
74.                return comparisionResult;
75.            }
76.        }
77.        return 0;
78.    }
79.    private int getIntOrZero(int index) {
80.        return index < value.length ? value[index] : 0;
81.    }
82.    @Override
83.    public boolean equals(Object o) {
84.        if (this == o) return true;
85.        if (o == null || getClass() != o.getClass()) return false;
86.        LongInteger that = (LongInteger) o;
87.        return this.compareTo(that) == 0;
88.    }
89.    @Override
90.    public String toString() {
91.        return "LongInteger{" +
92.                "value=" + HelperUtilities.longIntegerToHex(this) +
93.                '}';
94.    }
95. }
```

Інтерфейс Adder

```
1. package ua.kpi.voloshyna.longintegerarithmetics.adder;

2. import ua.kpi.voloshyna.longintegerarithmetics.LongInteger;
3. public interface Adder {
4.     LongInteger calculate(LongInteger number1, LongInteger number2);
```

```
}
```

Клас AdderImpl

```
package ua.kpi.voloshyna.longintegerarithmetics.adder;

import ua.kpi.voloshyna.longintegerarithmetics.LongInteger;
public class AdderImpl  implements Adder {
    public LongInteger calculate(LongInteger number1, LongInteger number2) {
        int maxLength = Math.max(number1.getSize(), number2.getSize());
        LongInteger result = new LongInteger(maxLength + 1);
        int carry = 0;
        for (int i = 0; i < result.getSize(); i++) {
            long tempResult = number1.getLong(i) + number2.getLong(i) + carry;
            if ((tempResult >>> 32) > 0) {
                tempResult = tempResult & 0x00000000FFFFFFFFL;
                carry = 1;
            } else {
                carry = 0;
            }
            result.setLong(i, tempResult);
        }
        return result;
    }
}
```

Клас DivisionResult

```
package ua.kpi.longintegerarithmetics.divisor;

import ua.kpi.longintegerarithmetics.LongInteger;
public class DivisionResult {


    private LongInteger quotient;
    private LongInteger remainder;
    public DivisionResult(LongInteger quotient, LongInteger remainder) {
        this.quotient = quotient;
        this.remainder = remainder;
    }
    public LongInteger getQuotient() {
        return quotient;
    }
    public LongInteger getRemainder() {
        return remainder;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        DivisionResult result = (DivisionResult) o;
```

```
21.       if (quotient != null ? !quotient.equals(result.quotient) : result.quotient != null)
    return false;
22.       return remainder != null ? remainder.equals(result.remainder) : result.remainder
    == null;
23.    }
24.    @Override
25.    public String toString() {
26.       return "DivisionResult{" +
27.             "quotient=" + quotient +
28.             ", remainder=" + remainder +
29.             '}';
30.    }
31. }
```

Інтерфейс Divisor

```
1.  package ua.kpi.longintegerarithmetics.divisor;

2.  import ua.kpi.longintegerarithmetics.LongInteger;
3.  public interface Divisor {
4.     DivisionResult calculate(LongInteger dividend, LongInteger divisor);
5.  }
```

Клас DivisorImpl

```
1.  package ua.voloshyna.longintegerarithmetics.divisor;

2.  import ua.kpi.longintegerarithmetics.LongInteger;
3.  import ua.kpi.longintegerarithmetics.adder.Adder;
4.  import ua.kpi.longintegerarithmetics.mulitplier.Multiplier;
5.  import ua.kpi.longintegerarithmetics.subtractor.Subtractor;
6.  import java.util.Arrays;
7.  import static ua.kpi.longintegerarithmetics.LongInteger.*;
8.  public class DivisorImpl implements Divisor {
9.     private Multiplier multiplier;
10.    private Subtractor subtractor;
11.    private Adder adder;
12.    public DivisorImpl(Multiplier multiplier, Subtractor subtractor, Adder adder) {
13.       this.multiplier = multiplier;
14.       this.subtractor = subtractor;
15.       this.adder = adder;
16.    }
17.    public DivisionResult calculate(LongInteger dividend, LongInteger divisor) {
18.       if (divisor.getSize() == 0 || divisor.compareTo(ZERO) == 0) {
19.          throw new IllegalArgumentException();
20.       }
21.       if (dividend.compareTo(divisor) < 0) {
22.          return new DivisionResult(ZERO, dividend);
23.       }
24.       LongInteger dividendCopy = new LongInteger(dividend.getValue());
25.             LongInteger divisorCopy = new LongInteger(divisor.getValue());
26.       dividendCopy.stripLeadingZeroes();
```

```
27.       divisorCopy.stripLeadingZeroes();
28.       LongInteger quotient = new LongInteger(dividendCopy.getSize());
29.       //Special case not covered in main algorithm - divisor is one-digit (digit has 32 bits)
30.       if (divisorCopy.getSize() == 1) {

31.          return divideByOneDigitNumber(dividendCopy, divisorCopy);
32.       }
33.       //Normalizing dividend and divisor
34.       LongInteger normDivisor = new LongInteger(divisorCopy.getValue());
35.       LongInteger normDividend = new
    LongInteger(Arrays.copyOf(dividendCopy.getValue(), dividendCopy.getSize() + 1));
36.       int leadingZeroBitsInModulo =
    calculateNumberOfLeadingZeros(divisorCopy.getInt(divisorCopy.getSize() - 1));
37.       for (int i = 0; i < leadingZeroBitsInModulo; i++) {
38.          normDivisor.shiftLeft();
39.          normDividend.shiftLeft();
40.       //Main algo
41.       int dividendLength = dividendCopy.getSize();
42.       int divisorLength = divisorCopy.getSize();
43.       for (int i = dividendLength - divisorLength; i >= 0; i--) {
44.          int currentDividendIndex = i + divisorLength;
45.          LongInteger dividendValue = longIntegerSubPart(normDividend,
    currentDividendIndex - 1, currentDividendIndex + 1);
46.          LongInteger divisorValue = longIntegerSubPart(normDivisor, divisorLength - 1,
    divisorLength);
47.          boolean firstDigitsAreEqual = normDividend.getLong(currentDividendIndex) ==
    divisorValue.getLong(0);
48.          DivisionResult divisionResult = divideByOneDigitNumber(dividendValue,
    divisorValue);
49.          LongInteger quotientValue = firstDigitsAreEqual ?
    TWO_POWER_32_MINUS_1 : divisionResult.getQuotient();
50.          LongInteger remainderValue = firstDigitsAreEqual
51.                ? subtractor.calculate(dividendValue, multiplier.calculate(quotientValue,
    divisorValue))
52.                : divisionResult.getRemainder();
53.          LongInteger temp1 = multiplier.calculate(quotientValue, new LongInteger(new
    int[]{normDivisor.getInt(divisorLength - 2)}));
54.          LongInteger temp2 = new LongInteger(new
    int[]{normDividend.getInt(currentDividendIndex - 2), remainderValue.getInt(0)});
55.          while (quotientValue.compareTo(TWO_POWER_32) >= 0
56.                || temp1.compareTo(temp2) > 0) {
57.             quotientValue = subtractor.calculate(quotientValue, ONE);
58.             remainderValue = adder.calculate(remainderValue, divisorValue);
59.             temp1 = multiplier.calculate(quotientValue, new LongInteger(new
    int[]{normDivisor.getInt(divisorLength - 2)}));
60.             temp2 = new LongInteger(new
    int[]{normDividend.getInt(currentDividendIndex - 2), remainderValue.getInt(0)});
61.          }
62.                LongInteger subpartOfDividend = longIntegerSubPart(normDividend,
    currentDividendIndex - divisorLength, currentDividendIndex + 1);
63.          LongInteger divisorMultilpiedByQuotient = multiplier.calculate(normDivisor,
    quotientValue);
```

```
64.             while (subpartOfDividend.compareTo(divisorMultilpiedByQuotient) < 0) {
65.                 quotientValue = subtractor.calculate(quotientValue, ONE);
66.                 divisorMultilpiedByQuotient =
    subtractor.calculate(divisorMultilpiedByQuotient, normDivisor);
67.             }
68.             LongInteger subtractResult = subtractor.calculate(subpartOfDividend,
    divisorMultilpiedByQuotient);
69.         }
70.             for (int j = 0; j < divisorLength + 1; j++) {
71.                 normDividend.setInt(j + currentDividendIndex - divisorLength,
    subtractResult.getInt(j));
72.             }
73.             quotient.setLong(i, quotientValue.getLong(0));
74.         }
75.         for (int i = 0; i < leadingZeroBitsInModulo; i++) {
76.             normDividend.shiftRight();
77.         }
78.         normDividend.stripLeadingZeroes();
79.         return new DivisionResult(quotient, normDividend);
80.     }
81.     private LongInteger longIntegerSubPart(LongInteger number, int fromIndexInclusive,
    int toIndexExclusive) {
82.         int[] subValue = Arrays.copyOfRange(number.getValue(), fromIndexInclusive,
    toIndexExclusive);
83.         return new LongInteger(subValue);
84.     }
85.     private int calculateNumberOfLeadingZeros(int number) {
86.         int result = 0;
87.         long numberUnsignedLong = Integer.toUnsignedLong(number);
88.         for (int i = 31; i > 0; i--) {
89.             if (((numberUnsignedLong >>> i) & 0x0000000000000001L) == 0) {
90.                 result++;
91.             } else {
92.                 return result;
93.             }
94.         }
95.         return result;
96.     }
97.     private DivisionResult divideByOneDigitNumber(LongInteger longNumber,
    LongInteger oneDigitNumber) {
98.         LongInteger quotient = new LongInteger(longNumber.getSize());
99.         LongInteger remainder = new LongInteger(1);
100.                long dividerValue = oneDigitNumber.getLong(0);
101.                long remainderValue = 0;
102.                long dividendValue = 0;
103.                long quotientValue = 0;
104.                long carry = 0;
105.                for (int j = longNumber.getSize() - 1; j >= 0; j--) {
106.                    dividendValue = (remainderValue  << 32) | longNumber.getLong(j);
107.                    boolean dividendValueOverFlow = dividendValue < 0;
108.                    if (dividendValueOverFlow) {
```

Інтерфейс Multiplier

```
1. package ua.kpi.longintegerarithmetics.mulitplier;
2. import ua.kpi.longintegerarithmetics.LongInteger;
3. public interface Multiplier {
4.     LongInteger calculate(LongInteger number1, LongInteger number2);
5. }
```

Клас MultiplierImpl

```
1. package ua.kpi.longintegerarithmetics.mulitplier;
2. import ua.kpi.longintegerarithmetics.LongInteger;
3. public class MultiplierImpl implements Multiplier {
4.     private static final long HEX_00000000FFFFFFFF =
   Long.valueOf("00000000FFFFFFFF", 16);
5.                 carry = Math.abs(dividendValue % 2);
6.                 dividendValue = dividendValue >>> 1;
7.             }
8.             quotientValue = dividendValue / dividerValue;
9.             remainderValue = dividendValue - quotientValue * dividerValue;
10.            if (dividendValueOverFlow) {
11.                quotientValue = quotientValue << 1;
12.                remainderValue = remainderValue << 1 + carry;
13.                while (remainderValue > dividerValue) {
14.                    remainderValue -= dividerValue;
15.                    quotientValue++;
16.                }
17.            }
18.            quotient.setLong(j, quotientValue);
19.        }
20.        remainder.setLong(0, remainderValue);
21.        return new DivisionResult(quotient, remainder);
22.    }
23.    public LongInteger calculate(LongInteger number1, LongInteger number2) {
24.        LongInteger result = new LongInteger(number1.getSize() + number2.getSize());
25.        for (int i = 0; i < number2.getSize(); i++) {
26.            for (int j = 0; j < number1.getSize(); j++) {
27.                long tempResult = number2.getLong(i) * number1.getLong(j);
28.                long r0 = tempResult & HEX_00000000FFFFFFFF;
29.                long r1 = tempResult >>> 32;
30.                result = addToLongInteger(result, new long[]{r0, r1}, i + j);
31.            }
32.        }
33.        return result;
34.    }
35.    private LongInteger addToLongInteger(LongInteger result, long[] adder, int
   indexToAdd) {
36.        int resultIndex = indexToAdd;
37.        int adderIndex = 0;
38.        long carry = 0;
39.        do {
40.            long resultInt = result.getLong(resultIndex) + adder[adderIndex] + carry;
```

```
            carry  = resultInt >>> 32;
            result.setLong(resultIndex, resultInt & HEX_00000000FFFFFFFF);
            resultIndex++;
            adderIndex++;
        } while (adderIndex < adder.length);
        while (carry > 0) {
            long resultInt = result.getLong(resultIndex) + carry;
            carry  = resultInt >>> 32;
            result.setLong(resultIndex, resultInt & HEX_00000000FFFFFFFF);
            resultIndex++;
        }
        return  result;
    }
}
```

```
} package ua.kpi.longintegerarithmetics.subtractor;
import ua.kpi.longintegerarithmetics.LongInteger;
public class SubtractorImpl implements Subtractor {
    private static final long TWO_POWER_32 = 0x100000000L;
    public LongInteger calculate(LongInteger number1, LongInteger number2) {
        if (number1.compareTo(number2) < 0) {
            throw new IllegalArgumentException();
        }
        int maxLength = Math.max(number1.getSize(), number2.getSize());
        LongInteger result = new LongInteger(maxLength);
        int borrow = 0;
        for (int i = 0; i < maxLength; i++) {
            long tempResult = number1.getLong(i) - number2.getLong(i) - borrow;
            if (tempResult < 0) {
                tempResult += TWO_POWER_32;
                borrow = 1;
            } else {
                borrow = 0;
            }
            result.setLong(i, tempResult);
        }
        return result;
    }
}
```

Клас BinaryRightToLeftModularExponentiator

```
package ua.kpi.modularoperations.modularexponentiator;
import ua.kpi.longintegerarithmetics.LongInteger;
import ua.kpi.modularoperations.modulocalculator.ModuloCalculator;
import ua.kpi.modularoperations.modulomultiplier.ModularMultiplier;
public class BinaryRightToLeftModularExponentiator implements ModularExponentiator
{
    private ModularMultiplier modularMultiplier;
    private ModuloCalculator moduloCalculator;
```

```
8.    public BinaryRightToLeftModularExponentiator(ModularMultiplier modularMultiplier,
      ModuloCalculator moduloCalculator) {
9.        this.modularMultiplier = modularMultiplier;
10.       this.moduloCalculator = moduloCalculator;
11.   }
12.   public LongInteger calculate(LongInteger base, LongInteger exponent, LongInteger
      modulo) {
13.       LongInteger result = new LongInteger(new int[]{1});
14.       if (base.compareTo(LongInteger.ZERO) == 0) {
15.           return LongInteger.ZERO;
16.       }
17.       if (exponent.compareTo(LongInteger.ZERO) == 0) {
18.           return result;
19.       }
20.       LongInteger exponentCopy = new LongInteger(exponent.getValue());
21.       base = moduloCalculator.calculate(base, modulo);
22.       for (int i = 0; i < exponentCopy.getSize(); i++) {
23.           long exponentCurrentInt = exponentCopy.getLong(i);
24.           for (int j = 0; j < 32; j++) {
25.               if (Long.lowestOneBit(exponentCurrentInt) == 1) {
26.                   result = modularMultiplier.calculate(result, base, modulo);
27.               }
28.               exponentCurrentInt = exponentCurrentInt >> 1;
29.               base = modularMultiplier.calculate(base, base, modulo);
30.           }
31.           exponentCopy.setInt(i, 0);
32.           if (exponentCopy.compareTo(LongInteger.ZERO) == 0) {
33.               break;
34.           }
35.       }
36.       return result;
37.   }
38.   public LongInteger calculate(LongInteger base, LongInteger exponent, LongInteger
      modulo) {
39.       if (base.compareTo(LongInteger.ZERO) == 0) {
40.           return LongInteger.ZERO;
41.       }
42.       if (base.compareTo(modulo) >=0) {
43.           base = moduloCalculator.calculate(base, modulo);
44.       }
45.       LongInteger r = new LongInteger(modulo.getSize() + 1);
46.       r.setInt(modulo.getSize(), 1);
47.       LongInteger r2 = new LongInteger(modulo.getSize() * modulo.getSize() + 1);
48.       r2.setInt(modulo.getSize() * 2, 1);
49.       LongInteger mStroke = calculateNumberStroke(modulo,
      LongInteger.TWO_POWER_32);
50.       LongInteger temp = moduloCalculator.calculate(r2, modulo);
51.       LongInteger xStroke = montgomeryMultiplication(base, temp, modulo, mStroke);
52.       LongInteger a = moduloCalculator.calculate(r, modulo);
53.       boolean firstOneMet = false;
54.       for (int i = exponent.getSize() - 1; i >=0; i--) {
55.           for (int j = 31; j >= 0; j--) {
```

```
56.             boolean isOneBit = ((exponent.getInt(i) >>> j) & 0x00000001) == 1;
57.             if (!firstOneMet && !isOneBit) {
58.                continue;
59.             }
60.             firstOneMet = true;
61.             a = montgomeryMultiplication(a, a, modulo, mStroke);
62.             if (isOneBit) {
63.                a = montgomeryMultiplication(a, xStroke, modulo, mStroke);
64.             }
65. }       a = montgomeryMultiplication(a, LongInteger.ONE, modulo, mStroke);
66.       return  a;
67.    }
68.    private LongInteger calculateNumberStroke(LongInteger number, LongInteger
    modulo) {
69.       LongInteger stroke = subtractor.calculate(modulo, getModularInverse(number,
    modulo));
70.       stroke.stripLeadingZeroes();
71.       return stroke;
72.    }
73.    protected LongInteger montgomeryMultiplication(LongInteger x, LongInteger y,
    LongInteger m, LongInteger mStroke) {
74.       if (x.compareTo(m) >=0) {
75. //          throw new IllegalArgumentException("x is >= m");
76.          x = moduloCalculator.calculate(x, m);
77.       }
78.       if (y.compareTo(m) >=0) {
79. //          throw new IllegalArgumentException("y is >= m");
80.          y = moduloCalculator.calculate(y, m);
81.       }
82.       // b = 2^32
83.       //n - getSize of modulo (number of integers)
84.       int n = m.getSize();
85.       LongInteger a = new LongInteger(n + 1);
86.       LongInteger y0 = new LongInteger(new int[]{y.getInt(0)});
87.       for (int i = 0; i < n; i++) {
88.          LongInteger xi = new LongInteger(new int[]{x.getInt(i)});
89.          LongInteger a0 = new LongInteger(new int[]{a.getInt(0)});
90.          LongInteger u = multiplier.calculate(adder.calculate(multiplier.calculate(xi, y0),
    a0), mStroke);
91.          LongInteger ui = new LongInteger(new int[]{u.getInt(0)});
92.          a = adder.calculate(a, adder.calculate(multiplier.calculate(xi, y),
    multiplier.calculate(ui, m)));
93.          a.shiftRight(32);
94.       }
95.       if (a.compareTo(m) >=0) {
96.          a = subtractor.calculate(a, m);
97.       }
98.       a.stripLeadingZeroes();
99.       return a;
100.          }
101.          private LongInteger getModularInverse(LongInteger number, LongInteger
    modulo) {
```

```
102.                BigInteger numberBigDec = new
    BigInteger(HelperUtilities.longIntegerToHex(number), 16);
103.                BigInteger moduloBigDec = new
    BigInteger(HelperUtilities.longIntegerToHex(modulo), 16);
104.                BigInteger numberModularInverseBigDec =
    numberBigDec.modInverse(moduloBigDec);
105.                return new
    LongInteger(numberModularInverseBigDec.toString(16).toCharArray());
106.            }
107.        }
```