КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття рівня магістра

За спеціальністю 121 Інженерія програмного забезпечення

На тему:

СИМЕТРИЧНА СИСТЕМА ОБМІНУ ІНФОРМАЦІЄЮ НА ОСНОВІ ІЗОМОРФІЗМУ КІЛЕЦЬ

Виконала студентка 2-го курсу магістратури
Юлія НОРТМАН _____
(підпис)

Науковий керівник:
професор, доктор фіз.-мат. наук
Сергій КРИВИЙ _____
(підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____
(підпис)

Роботу розглянуто й допущено до захисту
На засіданні кафедри інтелектуальних програмних систем
«10» травня 2023р.,
протокол № 9
Завідувач кафедри
О. ПРОВОТАР _____
(підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 52 сторінки, 8 ілюстрацій, 5 таблиць, 11 джерел посилань.

JAVA, TSS, АЛГОРИТМ, ГРУПА, ІЗОМОРФІЗМ, КІЛЬЦЕ, КРИПТОСИСТЕМА, СИСТЕМА ДІОФАНТОВИХ РІВНЯНЬ

Об’єктом роботи є побудова криптосистеми з використанням скінченних асоціативно-комутативних кілець з одиницею та розробка алгоритму побудови таких кілець і обміну інформацією між абонентами на базі сюр’єктивного ізоморфізму. Предметом роботи є протокол обміну повідомленнями на основі використання лінійних однорідних діофантових рівнянь в асоціативно-комутативному кільці з одиницею.

Метою роботи є обґрунтування та реалізація протоколу обміну повідомленнями на основі розв’язання систем лінійних діофантових рівнянь в асоціативно-комутативному кільці з одиницею.

Інструментами розробки є безкоштовне вільно поширюване інтегроване середовище розробки Intellij Idea для корпоративної та веб розробки Java застосунків, мова Java 11, технології Java Swing та Java AWT, брокер повідомлень RabbitMQ 3.9 та засіб для автоматизації роботи з проектами Apache Maven 3.8.

Результати роботи: досліджені способи розв’язання лінійних діофантових рівнянь в асоціативно-комутативному кільці та можливість їх застосування для побудови симетричної системи обміну інформацією. Реалізовано алгоритм побудови кільця ізоморфного кільцю лишків за модулем m, де m – довільне ціле число, протокол обміну повідомленнями на основі розв’язання лінійних діофантових рівнянь в асоціативно-комутативному кільці з одиницею та створено простий настільний додаток для демонстрації роботи протоколу.

Створений програмний продукт може використовуватись у різних галузях, де потрібен безпечний та надійний обмін повідомленнями, як наприклад фінансовий сектор, електронна комунікація, захист від дизасемблювання програмного забезпечення тощо.

В подальшому реалізація може бути удосконалена з метою покращення часової ефективності її роботи а також підвищення надійності роботи алгоритму.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ

АКК – Асоціативно-комутативне кільце

ЛНДР – Лінійне неоднорідне діофантове рівняння

ЛОДР – Лінійне однорідне діофантове рівннння

НСД – Найбільший спільний дільник

СЛНДР – Система лінійних неоднорідних діофантових рівнянь

СЛОДР – Система лінійних однорідних діофантових рівнянь

ВСТУП

Оцінка сучасного стану об’єкта дослідження або розробки. У сучасному світі, з постійним зростанням обсягів цифрової інформації та розвитком Інтернету, питання захисту даних набуває великого значення. На сьогоднішній день вже існує цілий ряд методів та протоколів, які використовуються в галузі криптографічного захисту інформації, таких як симетричне та асиметричне шифрування, протоколи керування ключами, цифрові підписи та технології блокчейн. Отже, зростання кількості обміну інформацією та вимоги до її захисту зумовлюють пошук нових підходів до розробки криптографічних систем.

Симетричні криптосистеми, такі як AES (Advanced Encryption Standard), стали основою сучасних криптографічних стандартів та забезпечують високий рівень безпеки інформації у багатьох сферах, включаючи фінансові послуги, електронний документообіг, телекомунікації та Інтернет речей. В останні роки відбулася активна робота над вивченням стійкості симетричних криптосистем до атак з використанням квантових комп'ютерів, що може призвести до розробки нових стандартів криптографічної безпеки.

Актуальність роботи та підстави для її виконання. Симетричні системи обміну інформацією використовуються в ряді криптографічних застосувань, проте, вони часто стикаються з проблемами, пов'язаними з безпекою ключів та складністю обчислень. Традиційні методи обміну інформацією можуть стати неефективними внаслідок зміни обчислювальних можливостей та появи квантових комп'ютерів. У зв’язку з цим виникає потреба у пошуку нових методів та підходів, які б забезпечили надійний обмін інформацією та водночас знизили ризики. Одним з важливих напрямів дослідження є розробка стійких та ефективних алгоритмів обміну ключами,

зокрема з використанням теоретико-кільцевих підходів, таких як лінійні діофантові рівняння та ізоморфізм кілець.

Протоколи обміну повідомленнями на основі кільцевої криптографії, зокрема з використанням асоціативно-комутативних кілець з одиницею, можуть забезпечити альтернативні способи побудови безпечних криптографічних протоколів, які можуть бути стійкими до різних видів атак та мати високу ефективність.

Мета й завдання роботи. Метою роботи є реалізація протоколу обміну повідомленнями на основі розв'язання лінійних діофантових рівнянь в асоціативно-комутативному кільці з одиницею та імплементація простого застосунку для демонстрації роботи. Для досягнення цієї мети було поставлено наступні цілі:

- Дослідити властивості асоціативно-комутативних кілець з одиницею та їх зв'язок з кільцями лишків;
- Дослідити й описати властивості та способи розв'язання систем лінійних діофантових рівнянь в кільці лишків та в асоціативно-комутативному кільці з одиницею;
- Проаналізувати протокол обміну повідомленнями в асоціативно-комутативному кільці з одиницею;
- Спроектувати та розробити програмну реалізацію протоколу обміну повідомленнями і дослідити її ефективність.

Об'єкт, методи й засоби дослідження або розроблення. Об'єктом дослідження є протокол обміну повідомленнями, який використовує властивості асоціативно-комутативних кілець з одиницею, адитивні групи яких є повноциклічними, та систем лінійних діофантових рівнянь над такими кільцями. При такому підході кільце має відносно невеликий порядок і тому не вимагає значного об'єму пам'яті. Крім того асоціативно-комутативне

кільце будується так, що воно є ізоморфним відповідному кільцю лишків і ця властивість дозволяє опустити побудову таблиць додавання та множення.

Надійність запропонованого протоколу повністю ґрунтується на ізоморфізмі кілець. Повідомлення для обміну між двома абонентами шифруються у АКК$_m$, де m - порядок кільця, а розшифрування та всі пов'язані розрахунки виконуються у відповідному кільці лишків. Для приховування цього ізоморфізму кілець використовується сюр'єктивне відображення іншого кільця, порядок якого строго більший за порядок кільця.

Можливі сфери застосування. Створений програмний продукт може використовуватися у будь-якій сфері, де цілісність і безпека даних є ключовим фактором як наприклад сектор фінансових послуг, електронний документообіг, телекомунікації, захист програмного забезпечення тощо. Також ще однією з можливих сфер застосування може бути протидія незаконному поширенню програмних продуктів.

Апробація роботи та публікації з теми роботи. Результати цієї роботи були представлені на 12-тій Міжнародній конференції з надійних систем, послуг та технологій, DESSERT 2022 10-го грудня 2022 року [11].

РОЗДІЛ 1 ТЕОРЕТИЧНА ЧАСТИНА

1.1 Базові математичні поняття. Кільця. Поля

1.1.1 Визначення та характеристики кільця лишків

Кільце лишків Z_m за модулем числа m – це скінченна алгебра $Z_m = (A = \{0, 1, \dots m-1\}, \Omega = \{+, *, -, {}^{-1}, 0, 1\})$, де $+$ та $*$ називаються бінарними операціями додавання і множення за модулем m відповідно. Для кільця лишків виконуються наступні властивості [1, 2]:

1. $(Z_m, +)$ є абелевою[*)] групою;
2. Операція множення $(*)$ є асоціативною, тобто $(a * b) * c = a * (b * c)$ для всіх $a, b, c \in Z_m$;
3. Операція множення є комутативною, тобто $a * b = b * a$ для всіх $a \in Z_m$;
4. Для кільця Z_m виконується закон дистрибутивності, тобто $(a + b) * c = (a * c) + (b * c)$ і $a * (b + c) = (a * b) + (a * c)$ для всіх $a, b, c \in Z_m$;
5. Кільце $\mathrm{Z_m}$ містить мультиплікативну одиницю (1), що означає $1 * a = a * 1 = a, 1 \in \mathrm{Z_m}$ для будь-якого $a \in \mathrm{Z_m}$;
6. Кільце $\mathrm{Z_m}$ містить адитивний нуль (0), що означає $0 + a = a + 0 = a, 0 \in \mathrm{Z_m}$ для будь-якого $a \in \mathrm{Z_m}$;
7. В кільці $\mathrm{Z_m}$ визначена операція $-$, яка дає протилежний елемент до заданого відносно операції додавання;
8. В кільці $\mathrm{Z_m}$ визначена операція ${}^{-1}$, яка дає обернений елемент до заданого відносно операції множення.

[*)] Абелева група – група з комутативною операцією.

В загальному випадку операція взяття оберненого відносно операції множення у кільці Z_m може бути визначена не завжди, оскільки у випадку, коли m не є простим числом, то серед елементів кільця Z_m будуть присутні дільники нуля (дільниками нуля називаються такі $a, b \in Z_m$, що $a * b = 0, a \neq 0, b \neq 0$), а для дільників нуля ця операція не визначена.

Кільце лишків називається примарним, якщо модуль m є степенем простого числа p, тобто $m = p^t$, де $t > 1, t \in N$. У випадку, коли $t = 1$ кільце Z_m є полем, оскільки p – просте число. Таке поле позначається як F_p.

1.1.2 Означення та властивості поля F_p

Поле F_p – це комутативне кільце з одиницею за модулем p, (де p – просте число) в якому кожен елемент $a \in F_p$ має обернений елемент $a^{-1} \in F_p$.

Поле F_p має наступні характеристики:

1. $(F_p, +)$ і $(F_p - \{0\}, *)$ – абелеві групи;
2. Для поля F_p виконується закон дистрибутивності $a * (b + c) = (a * b) + (a * c), a, b, c \in F_p$.

Оскільки, як було зазначено, модуль поля p є простим числом, то в F_p відсутні дільники нуля і тому порівняння $a * x \equiv b$ завжди буде мати розв'язок, при чому він буде єдиний [4].

1.2.3 Поняття ізоморфізму кілець

Два кільця R та $R`$ називаються ізоморфними, якщо можна встановити таку відповідність $f$ між їх елементами, що для будь-яких $a, b \in R$ та $a`, b` \in R`$, така що

$f(a + b) = f(a) + f(b) = a` + b`$ і $f(a * b) = f(a) * f(b) = a` * b`$,

причому $f(0) = 0`, \ f(1) = 1`, 0, 1 \in R, 0`, 1` \in R`$ [3].

Іншими словами, якщо кільця R та $R`$ ізоморфні, то нульовому елементу з кільця $R$ відповідає нульовий елемент кільця $R`$. Для перевірки цього

твердження потрібно взяти довільний елемент $a \in R$ та відповідний до нього елемент $a` \in R`$. Можна припустити що елементу $0 \in R$ співставний елемент $c` \in R`$. Тоді за визначенням елементу $a + 0 \in R$ має співставлятися елемент $a` + c` \in R`$, але оскільки $a + 0 = a$, тоді з цього випливає що $a` + c` = a`$, тобто $c`$ також є нульовим елементом в кільці R`.

Крім того, елементу $-a \in R$ співставляється елемент $-a` \in R`$. Доведення цього факту є аналогічним. Нехай елементу $-a \in R$ співставляється елемент $d` \in R`$. Тоді згідно з визначенням поняття кільця елементу $a + (-a) = 0$ повинен відповідати елемент $a` + d` = 0$, з чого випливає, що $d` = -a`$. Таким чином показано, що різниці елементів з кільця R співставна різниця елементів з кільця R` Аналогічними діями можна довести, що одиничний елемент з кільця R відповідає одиничному елементу з ізоморфного йому кільця R`, і така ж сама логіка може бути застосована для обернених елементів.

З цього можна зробити висновок, що елементи кілець можуть відрізнятися один від одного за структурою але бути співставними за своїми алгебраїчними властивостями. А отже будь-яка теорема, що була доведена в одному кільці буде виконуватися і в іншому ізоморфному йому кільці.

1.2.4 Розробка алгоритму побудови асоціативно-комутативного кільця з одиницею

У цьому пункті описано процес побудови асоціативно-комутативного кільця з одиницею, яке ізоморфне кільцю Z_m.

Для побудови такого кільця достатньо лише ввести операцію додаванн з одиницею. Розглянемо наступний приклад, у якому проілюстрований процес побудови кільця. Нехай дана множина цілих чисел $Z_6 = \{0, 1, 2, 3, 4, 5\}$, на якій потрібно побудувати асоціативно-комутативне кільце шостого порядку. Нехай введена операція додавання з одиницею наступним чином:

$0 + 1 = 1;\ 1 + 1 = 3; 1 + 2 = 0; 1 + 3 = 5; 1 + 4 = 2; 1 + 5 = 4.$

Наступним кроком буде послідовно знайти результати додавання з числом 3 оскільки $3 = 1 + 1$.

$3 + 2 = (1 + 1) + 2 = 1 + (1 + 2) = 1 + 0 = 1;$

$3 + 3 = (1 + 1) + 3 = 1 + (1 + 3) = 1 + 5 = 4;$

$3 + 4 = (1 + 1) + 4 = 1 + (1 + 4) = 1 + 2 = 0;$

$3 + 5 = (1 + 1) + 5 = 1 + (1 + 5) = 1 + 4 = 2.$

Після цього можна знайти результат додавання з числом 5, тому що $5 = 1 + 3$, а на цей момент вже відомі результати додавання елементів множини з числами один і три.

$5 + 2 = (1 + 3) + 2 = 1 + (3 + 2) = 1 + 1 = 3;$

$5 + 3 = (1 + 3) + 3 = 1 + (3 + 3) = 1 + 4 = 2;$

$5 + 4 = (1 + 3) + 4 = 1 + (3 + 4) = 1 + 0 = 1;$

$5 + 5 = (1 + 3) + 5 = 1 + (3 + 5) = 1 + 2 = 0.$

Для елемента $4 = 5 + 1$ результат операції додавання з іншими елементами множини наведені нижче:

$4 + 2 = (1 + 5) + 2 = 1 + (5 + 2) = 1 + 3 = 5;$

$4 + 3 = (1 + 5) + 3 = 1 + (5 + 3) = 1 + 2 = 0;$

$4 + 4 = (1 + 5) + 4 = 1 + (5 + 4) = 1 + 1 = 3;$

$4 + 5 = (1 + 5) + 5 = 1 + (5 + 5) = 1 + 0 = 1.$

І для елемента $2 = 4 + 1$ результат буде наступним:

$2 + 2 = (1 + 4) + 2 = 1 + (4 + 2) = 1 + 5 = 4;$

$2 + 3 = (1 + 4) + 3 = 1 + (4 + 3) = 1 + 0 = 1;$

$2 + 4 = (1 + 4) + 4 = 1 + (4 + 4) = 1 + 3 = 5;$

$2 + 5 = (1 + 4) + 5 = 1 + (4 + 5) = 1 + 1 = 2.$

На цьому побудову асоціативно-комутативного кільця з одиницею можна вважати завершеною, оскільки відомі всі результати операції додавання. Ці результати зручно зберігати в таблиці (Таблиця 1).

+	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	3	0	5	2	4
2	2	0	4	1	5	3
3	3	5	1	4	0	2
4	4	2	5	0	3	1
5	5	4	3	2	1	0

Таблиця 1 - Таблиця додавання $АКК_6$

Після того, як визначена таблиця додавання можна визначити операцію добутку. Для цього спочатку потрібно взяти елемент множини, обидва доданки якого є одиницями. В даному прикладі таким елементом виступає число 3. Після цього, використовуючи закон дистрибутивності, потрібно виконати наступні дії:

$$3 * 2 = (1 + 1) * 2 = (1 * 2) + (1 * 2) = 2 + 2 = 4;$$

$$3 * 3 = (1 + 1) * 3 = (1 * 3) + (1 * 3) = 3 + 3 = 4;$$

$$3 * 4 = (1 + 1) * 4 = (1 * 4) + (1 * 4) = 4 + 4 = 3;$$

$$3 * 5 = (1 + 1) * 5 = (1 * 5) + (1 * 5) = 5 + 5 = 0.$$

Наступним кроком потрібно взяти елемент, який є сумою одиниці і числа, для якого була визначена операція множення на попередньому кроці. У прикладі це буде елемент $3 + 1 = 5$. Результати операцій наведені нижче.

$$5 * 2 = (1 + 3) * 2 = (1 * 2) + (3 * 2) = 2 + 4 = 5;$$

$$5 * 3 = (1 + 3) * 3 = (1 * 3) + (3 * 3) = 3 + 4 = 0;$$

$$5 * 4 = (1 + 3) * 4 = (1 * 4) + (3 * 4) = 4 + 3 = 0;$$

$$5 * 5 = (1 + 3) * 5 = (1 * 5) + (3 * 5) = 5 + 0 = 5.$$

Аналогічним чином знаходиться результат множення з іншими елементами, які лишились. З отриманих результатів була побудована наступна Таблиця 2:

*	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	1	4	3	5
3	0	3	4	4	3	0
4	0	4	3	3	4	0
5	0	5	5	0	0	5

Таблиця 2 - Таблиця множення для $АКК_6$

З наведеного прикладу видно, що для визначення операцій додавання і множення в асоціативно-комутативному кільці з одиницею необхідно мати лише рядочок результатів додавання з одиницею, який будемо називати визначальним рядком. Отже постає питання, якими властивостями має володіти ця операція, щоб можна було коректно визначити операції кільця.

Для того, щоб дати відповідь на це питання операції додавання з одиницею поставимо у відповідність наступну підстановку:

$$f_1(0) = 0 + 1 = 1;$$
$$f_1(1) = 1 + 1 = a_1;$$
$$f_1(2) = 1 + 2 = a_2;$$
$$f_1(a_2) = 1 + a_2;$$
$$\dots$$
$$f_1(a_{k-1}) = 1 + a_{k-1};$$

Скінченна адитивна група k-го порядку називається повноциклічною, тоді і тільки тоді коли підстановка f_1 є повним циклом довжини k.

Має місце просте твердження.

Твердження 1. Повноциклічні групи однакових порядків ізоморфні між собою.

Доведення випливає з того факту, що повноциклічні групи є циклічними, а циклічні групи однакових порядків ізоморфні між собою [5].

Отже, зі сказаного вище можна переконатися що група, задана Таблиця 2 є повноциклічною.

1.2.5 Алгоритм побудови визначального рядка

Як було зазначено в попередньому пункті для побудови АКК з одиницею необхідно мати визначальний рядок. Нижче наведений алгоритм у форматі псевдокоду, за допомогою якого можна згенерувати визначальний рядок.

Нехай у кільці G_k порядку k задана функція

$$f(i) = a * i + c, \text{де } a, c \in G_k, \text{НСД}(a, k) = 1,$$

тоді алгоритм можна записати наступним чином:

$\text{GenerateDefiningRow}(a, c, k)$

1. for $i = 0$ to $k - 1$
2. $\quad b[i] := f(i)$
3. for $i = 0$ to $k - 1$
4. $\quad$ if $b[i] = 0 \wedge i \neq 0$
5. $\quad\quad$ Поміняти місцями елементи $b[i]$та $b[0]$
6. $\quad$ if $b[i] = 1 \wedge i \neq 1$
7. $\quad\quad$ Поміняти місцями елементи $b[i]$ та $b[1]$
8. $P[0] := b[0]$
9. for $i = 0$ to $k - 1$
10. $\quad P\big[b[i]\big] := b[i + 1]$
11. return P

Коректність цього алгоритму випливає з того, що коли НСД$(a,k)=1$ і i пробігає повну систему лишків по модулю k, то $a*i+c,\ i=\overline{1,k-1}$ також пробігає повну систему лишків [8].

Цей факт можна довести від супротивного. Нехай існують такі $i,j,\ i\neq j$ такі що $a*i+c\equiv a*j+c\ (mod\ k)$, що еквівалентне $a*i\equiv a*j\ (mod\ k)$. Оскільки НСД$(a,k)=1$ то $i\equiv j\ (mod\ k)$ тобто отримали суперечність.

Часова складність цього алгоритму складає $O(k*(\log k)^2)$. Алгоритм GenerateDefiningRow генерує не більше ніж $(k-2)\varphi(k)$ рядків, де φ – функція Ейлера[*)]. Крім того, для того, щоб ускладнити отриману перестановку абоненти, за домовленості, можуть виконати певні трансформації рядка, отриманого кроками 1-2. Прикладами таких трансформацій можуть бути декілька циклічних зсувів рядка тощо.

Легко побачити, що будь-яке кільце k-го порядку з адитивною повноциклічною групою ізоморфно кільцю лишків за модулем k. Дійсно, їх адитивні групи є повноциклічними однакового порядку, а ізоморфізм цих кілець є продовженням ізоморфізму їх адитивних груп. Такий ізоморфізм будується відразу після рядка, який генерується алгоритмом (кроки 3-7).

Тобто, генерація визначального рядка також визначає ізоморфізм між кільцем лишків Z_k і асоціативно-комутативним кільцем з одиницею G_k. Наприклад, маємо такі відповідники, де ізоморфне відображення буде наступним:

0	1	2	3	4	…	k-1
0	1	$b[2]$	$b[3]$	$b[4]$	…	$b[k-1]$

Таблиця 3 - Ізоморфізм між кільцем лишків і АКК з одиницею

[*)] Функція Ейлера, позначена як φ(n) або ϕ(n), є важливою функцією в теорії чисел, яка показує кількість чисел, менших за n, які є взаємно простими з n.

1.2 Лінійні діофантові рівняння

Діофантові рівняння – це рівняння з цілими коефіцієнтами та цілими розв'язками [6]. Ці рівняння були названі на честь Діофанта Александрійського, давньогрецького математика, який вивчав цей вид рівнянь. Діофантові рівняння можуть бути легко сформульовані, але їх розв'язок може бути дуже складним. У деяких випадках розв'язки можуть бути знайдені шляхом перебору всіх можливих значень, а в інших випадках – за допомогою більш складних алгоритмів.

Діофантові рівняння можуть мати різні форми. Одна з найбільш відомих – це рівняння Пелля, яке має вигляд $x^2 - dy^2 = 1$, де d - ціле число, яке не є квадратом іншого цілого числа. Це рівняння було вивчене вже в давні часи і було досліджене багатьма математиками. Наприклад, Леонард Ейлер використовував рівняння Пелля у своїх роботах з теорії чисел.

Існують різні методи розв'язування діофантових рівнянь. Один з найбільш відомих методів - це метод Ферма розв’язання такого типу рівнянь [7]. Цей метод використовується для розв'язання рівнянь вигляду $n = x^2 - y^2 = (x - y) * (x + y)$, де $x, y,$ та n - цілі числа. Тоді $u = x + y, v = x - y -$ дільники числа n. Звідси можна визначити $x = \frac{u+v}{2}$, $y = \frac{v-u}{2}$. Метод Ферма дозволяє визначати дільники числа n, яке не обов’язково є простим.

Другим методом є алгоритм Баха-Штрассена, який базується на ідеях методу Ферма та використовує ряд арифметичних перетворень для знаходження розв'язків діофантових рівнянь. Однак, цей алгоритм є досить складним, тому він використовується тільки для розв'язування дуже складних діофантових рівнянь.

Інший метод, який використовується – це метод Мінковського. Він ґрунтується на геометричних ідеях та використовує поняття ортогональної сітки у просторі. Цей метод дозволяє знаходити розв'язки діофантових рівнянь

шляхом визначення точок на цій сітці, що задовольняють деяким властивостям. Однак, він також може бути дуже складним, тому застосовується тільки для розв'язування дуже специфічних типів рівнянь.

Однією з основних областей де використовуються діофантові рівняння є криптографія. Зокрема, тут використовуються так звані «діофантові шифри», які шифрують повідомлення за допомогою цілих чисел. Вони засновані на складності розв'язування діофантових рівнянь, тому є досить надійними та безпечними.

1.3 Криптологія, криптографія та криптоаналіз

Криптологія – це галузь науки, яка вивчає основні закономірності, протиріччя, принципи, механізми, методи, протоколи, моделі, системи та засоби криптографічного захисту даних, здійснення криптоаналізу та певною мірою й приховування фактів передачі, оброблення, змісту даних. Криптологія складається з двох частин – криптографії та криптоаналізу.

Криптографія є наукою про методи й техніки забезпечення конфіденційності, цілісності, автентичності та невідтворюваності інформації. Вона має важливе значення для сучасного суспільства, оскільки гарантує безпеку комунікацій, фінансових транзакцій та інших важливих процесів.

У свою чергу криптоаналіз займається оцінкою сильних і слабких сторін шифрування, а також розробкою методів, які дозволяють зламувати криптосистеми.

1.3.1 Історія криптографії

Історія криптографії сягає глибокої давнини. Ще в античний час люди використовували шифри для передачі таємних повідомлень. Один з найвідоміших прикладів – шифр Цезаря, який полягав у заміні кожної літери алфавіту на букву, що була зсунута на певну кількість позицій.

У середньовіччі криптографія використовувалась для передачі державних таємниць, а також у військових цілях. У Новий час з'явилися нові методи шифрування, такі як шифр Віженера, а з розвитком математики – і більш складні системи, засновані на алгебрі та аналізі.

У XX столітті криптографія досягла нових вершин, зокрема завдяки розвитку комп'ютерних технологій. Під час Другої світової війни криптографічні відділи військ різних країн змагалися у розробці нових шифрів і методів їх злому. Наприклад, німецька машина «Енігма» вважалася непереможною, поки англійські криптоаналітики, очолювані Аланом Тюрінгом, не розробили методи її злому.

Після війни криптографія продовжувала розвиватися швидкими темпами. Завдяки роботі Клода Шеннона, який започаткував теорію інформації, було сформульовано математичні основи криптографії, а також сформовано поняття про ідеальний секретний шифр. У 70-х роках XX століття з'явилися сучасні методи криптографії, зокрема асиметричні криптосистеми, що дозволили вирішити проблему передачі секретного ключа.

1.3.2 Основні задачі криптографії

Основними задачами криптографії можна назвати наступні:

1. Забезпечення конфіденційності інформації – захист даних від ознайомлення з ними третіми, небажаними особами (зловмисниками). Поняття конфіденційної інформації означає, що дані є секретними, з обмеженим доступом.
2. Автентифікація – це підтвердження оригінальності сторін (тобто ідентифікація абонентів) і самих даних в процесі обміну інформацією. Тобто отримувач хоче упевнитись в тому, що інформація прийшла саме від легального абонента, а не від когось

стороннього навіть у тому випадку, коли сам абонент буде це заперечувати.

3. Забезпечення цілісності даних – надання гарантій того, що інформація при зберіганні та передачі не була змінена.
4. Забезпечення неможливості відмовитися від авторства – забезпечення того, що абонент не може відмовитись від факту передачі інформації і інших дій, які він здійснював. Після того, як повідомлення було відправлено отримувач може визначити, що воно було відправлено легальним абонентом. В свою чергу відправник може пересвідчитися, що дані були отримані тою особою, якій вони мали бути відправлені.

1.3.3 Основні визначення та поняття

1. Алфавіт A – це набір попарно різних символів, які використовуються для представлення відкритого тексту або шифротексту. Ці символи можуть бути буквами, цифрами, знаками пунктуації, пробілами або будь-якими іншими елементами, що використовуються в тексті. Алфавіт є основою для створення кодування та декодування повідомлень у криптографічних системах. Кількість літер алфавіту називається його потужністю, позначається |A|.
2. Шифротекст – це результат застосування криптографічного алгоритму (шифру) до відкритого тексту з використанням відповідного ключа. Шифротекст представляє собою зашифроване повідомлення, яке має незрозумілий та випадковий вигляд для стороннього спостерігача.
3. Шифрування – процеси криптографічного перетворення даних, що складаються із відповідних процесів зашифровування, в результаті виконання якого відкритий текст відображається в шифротекст, та

розшифрування, в результаті виконання якого шифротекст відображається у відкритий текст. Обидва процеси виконуються з використанням таємних ключів.

4. Кодування даних – процес перетворення сигналу з форми, зручної для безпосереднього використання даних, у форму, зручну для передачі, зберігання або обробки та навпаки.

1.3.4 Поняття криптосистеми

Криптосистема або шифр - це множина обернених функцій відображення множини відкритих текстів M на множину шифротекстів C, що залежать від ключа. Іншими словами, це алгоритм перетворення обернених перетворень відкритого тексту в шифротекст і навпаки. Кожне конкретне відображення відповідає зашифруванню з одним конкретним ключем. Для задання криптосистеми (шифру) необхідно вказати:

- Множину M відкритих текстів;
- Множину C шифротекстів;
- Множину K_1 та K_2 ключів для зашифрування і розшифрування відповідно;
- Алгоритм генерації ключів;
- Криптоалгоритм зашифрування $\mathrm{C} = \mathrm{E}_{\mathrm{K}_1}(\mathrm{M})$;
- Криптоалгоритм розшифрування $\mathrm{M} = \mathrm{D}_{\mathrm{K}_2}(\mathrm{C})$.

Для кожного ключа шифрування k_1 функція $\mathrm{C} = \mathrm{E}_{\mathrm{K}_1}(\mathrm{m})$ має бути оберненою:

$$\mathrm{D}_{\mathrm{K}_2}\left(\mathrm{E}_{\mathrm{K}_1}(\mathrm{m})\right) = m, \text{тобто } D = E^{-1}.$$

Оберненість – основна умова шифрування, за якої кожному зашифрованому повідомленню с відповідає одне початкове повідомлення m.

Узагальнена схема криптографічної системи, що забезпечує шифрування інформації що передається, показана на Рисунок 1. Відправник генерує відкритий текст початкового повідомлення M, яке повинно бути передано законному отримувачу по незахищеному каналу. За каналом слідкує зловмисник з метою перехопити та розкрити повідомлення. Для того, щоб він не зміг дізнатися зміст повідомлення M, відправник шифрує його за допомогою оберненого перетворення $\mathrm{E}_{\mathrm{K}_1}(\mathrm{M})$ та отримує шифротекст $\mathrm{C} = \mathrm{E}_{\mathrm{K}_1}(\mathrm{M})$, який відправляє отримувачу. Законний отримувач, прийнявши шифротекст C, розшифровує його за допомогою зворотного перетворення $D = E^{-1}$ та отримує початкове повідомлення у вигляді відкритого тексту M.

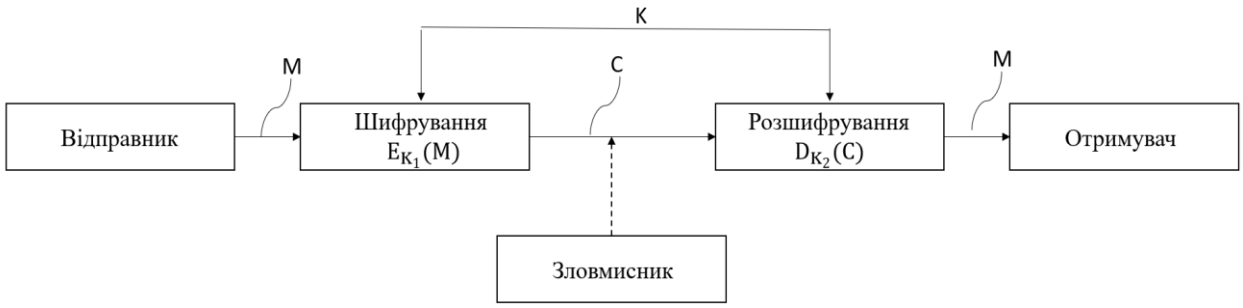

Рисунок 1 - Схема криптографічної системи

1.3.5 Типи криптоаналітичних атак

Дії криптоаналітика, які полягають у відновленні відкритого тексту з шифротексту без знання ключа чи шифру, вважаються атакою. У разі успішного виконання криптоаналітичної атаки на шифр, таку атаку можна назвати зломом шифру. Атаки бувають двох видів – пасивні та активні. Пасивна атака – це коли противник не може змінювати повідомлення, які передаються. При такій атаці можливе лише прослуховування повідомлень, їх розшифрування та аналіз трафіку. У разі активної атаки противник може змінювати повідомлення та навіть додавати власні.

1. Атака на основі лише відомого шифротексту. Криптоаналітик має доступ лише до шифротекстів $C_1, C_2, \dots C_i$ зашифрованих одним і тим же ключем та алгоритмом E_k. Метою криптоаналітика є розкриття відповідних відкритих текстів $M_1, M_2, \dots M_i$, або ще краще - визначення ключа шифрування для подальшого розшифрування інших повідомлень. Цей варіант відповідає моделі зовнішнього порушника, який має фізичний доступ до лінії зв'язку, але не має доступу до шифрувального та розшифрувального обладнання.
2. Атака на основі відомого відкритого тексту. Криптоаналітик має доступ не тільки до шифротекстів $C_1, C_2, \dots C_i$ та відповідних відкритих текстів $M_1, M_2, \dots M_i$ цих повідомлень, але також може мати доступ до стандартних документів з повторюваними блоками даних, що спрощує завдання по знаходженню ключа шифрування або способу розшифрування нових шифротекстів, отриманих за тим самим ключем. Також, цей вид атаки можливий при використанні глобального шифрування, коли вся інформація на магнітному носії записується у вигляді шифротексту, включаючи головний запис, загрузочний сектор, системні програми та інше. При викраденні такого носія (або комп'ютера) можна легко встановити, яка частина криптограми відповідає системній інформації, і отримати значний обсяг відомого початкового тексту для проведення криптоаналізу.
3. Атака з вибором відомого відкритого тексту. Криптоаналітик має доступ до шифротекстів $C_1, C_2, \dots C_i$ та відповідних відкритих текстів $M_1, M_2, \dots M_i$ цих повідомлень, але також може обирати відкриті тексти за власним бажанням, які потім отримує у зашифрованому вигляді. Такий криптоаналіз є більш потужним порівняно з криптоаналізом з відомим відкритим текстом, оскільки

криптоаналітик може вибирати для шифрування такі блоки відкритого тексту, які нададуть більше інформації про ключ. Метою криптоаналітика є пошук ключа шифрування або способу розшифрування нових шифротекстів, отриманих за тим самим ключем. Цей вид атаки відповідає моделі внутрішнього порушника. На практиці така ситуація може виникнути при залученні до криптоаналізу осіб, які не знають секретного ключа, але мають доступ до шифрувального обладнання через свої службові повноваження.

4. Атака з вибором відомого шифротексту. Криптоаналітик може вибирати різні шифротексти $C_1, C_2, \dots C_i$ для розшифрування та отримувати відповідні відкриті тексти $M_1, M_2, \dots M_i$. Завдання криптоаналітика полягає у пошуку ключа k, який використовувався для шифрування цих повідомлень. Цей тип криптоатак особливо цікавий для алгоритмів з відкритим ключем.

Також існують атаки по побічних каналах, які ґрунтуються на інформації, яку можна отримати з пристрою шифрування і яка не є відкритим текстом або шифротекстом (наприклад, час виконання операцій шифрування, енергоспоживання, електромагнітне випромінювання, аналіз кеш-пам'яті тощо). Завдання криптоаналітика полягає у визначенні ключа шифрування та відновленні відкритих текстів з шифротекстів.

1.3.6 Класифікація криптоалгоритмів

Існує декілька видів класифікації криптоалгоритмів. Перший клас це класифікація за секретністю алгоритмів. У ньому виділяють наступні підкласи:

- Обмежені криптосистеми, коли потрібно зберігати в таємниці сам алгоритм шифрування.

– Криптосистеми загального використання, засновані на правилі Керкгоффса. Правило Керкгоффса - це фундаментальне припущення криптоаналізу, про те, що секретність повідомлення цілком залежить від ключа, тобто весь механізм шифрування, окрім значення ключа, відомий зловмиснику. Як би то ні було, секретність алгоритму не є великою перешкодою: наприклад, для визначення типу програмно реалізованого криптографічного алгоритму потрібно лише декілька днів аналізу коду. Таким чином, далі приймаємо, що секретність шифру забезпечена секретністю ключа шифрування, а не секретністю алгоритму.

Другим є класифікація по типу організації секретного зв’язку:

– Симетричні криптосистеми – в яких ключі для шифрування та розшифрування збігаються або можуть легко обчислюватися один з іншого. Тут захист інформації забезпечується секретністю ключів (потрібен таємний канал обміну ключами). Ключі шифрування та розшифрування повинні бути відомі легальним користувачам і зберігатися в секреті від зловмисника (криптоаналітика). Якщо є група з n людей, в якій кожен повинен мати зв'язок один з одним, то для організації зв'язку потрібно $\frac{n(n-1)}{2}$ ключів, так як кожному потрібно $n-1$ ключ, щоб зв'язатися з рештою групи, але ключ між A та B може використовуватися в обох напрямках. Тому для симетричних криптосистем виникає завдання розподілу секретних ключів між легальними користувачами – надійне створення та доставка унікальних секретних ключів для всіх пар користувачів.
– Асиметричні криптосистеми або криптосистеми з відкритим ключем дозволяють організувати секретний зв'язок без

попереднього секретного обміну ключами. Криптосистема називається криптосистемою з відкритим ключем (асиметричною криптосистемою), якщо один з ключів (ключ шифрування), називаний відкритим ключем, відомий всім користувачам, включаючи криптоаналітика, а інший ключ відомий тільки передавальній або приймальній стороні. Цей ключ (для розшифрування) називається секретним (закритим). Вважають, що такі криптосистеми народилися у 1976 році, коли була опублікована стаття американських математиків Вайтфілда Діффі та Мартіна Хеллмана "Нові напрямки в криптографії", де вони ввели поняття односторонньої функції з лазівкою. На сьогоднішній день симетричні криптосистеми мають високу швидкість шифрування даних (десятки мегабайт в секунду на ПК), а відомі криптосистеми з відкритим ключем - низькою швидкістю (кілобайти в секунду на ПК). Водночас загальна доступність відкритого ключа для всіх користувачів в асиметричній криптосистемі дозволяє будувати зручні системи автентифікації користувачів та наступного розподілу ключів, оскільки немає необхідності тримати відкритий ключ в секреті. Тому симетричні криптосистеми через високу швидкість використовують для шифрування даних, в той час як асиметричні - для автентифікації та створення секретних сеансових ключів для симетричного шифрування даних.

1.4 Протокол обміну повідомленнями

Протокол обміну повідомленнями – це система правил і процедур, які визначають, як повідомлення мають бути зашифровані, передані, отримані та оброблені різними вузлами у мережі. Протоколи обміну повідомленнями

спрощують комунікацію між різними комп'ютерними системами, вирішують сумісність та забезпечують надійний обмін інформацією.

Ці протоколи використовуються в різних галузях, таких як комп'ютерні мережі, криптографія, телекомунікації та програмування. У криптографії, наприклад, протокол обміну повідомленнями може забезпечувати конфіденційність, цілісність та автентичність передачі даних, використовуючи криптографічні методи, такі як шифрування, гешування та цифрові підписи.

1.4.1 Алгоритм протоколу обміну повідомленнями

Перед початком передачі даних два абоненти, що хочуть обмінятися повідомленнями, за допомогою секретного каналу обмінялися трійкою чисел (a, c, k), де $k -$ порядок поля. Назвемо цих абонентів Аліса та Боб.

Ця трійка елементів є параметрами алгоритму GenerateDefiningRow, що був описаний вище. Використовуючи функцію $f(i) = a * i + c,$ було згенеровано визначний рядок, який за домовленостями між Алісою і Бобом трансформується однаковим чином (циклічними перестановками елементів, використанням шифрів підстановки та перестановки тощо).

Після цього Аліса і Боб виконують наступні дії:

1. Аліса будує систему лінійних виразів:

$$l(x)\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1q}x_q \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2q}x_q \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mq}x_q \end{cases},$$

де $a_{ij} \in G_k$. Аліса перетворює цю систему наступним чином:

$$L(x) = B_r(B_{r-1} \dots B_2(B_1(l(x) + a_1) \dots + a_{r-1}) + a_r) + a_{r+1},$$

де B_i – невироджені матриці розмірності $m \times m$ в асоціативно-комутативному кільці G_k, a_j – довільні вектори зсуву розмірності $1 \times m,\ i = 1,2, \dots, r,\ j = 1, \dots, r + 1.$

Після цього Аліса надсилає Бобу будь-яким відкритим каналом вирази $l(x)$ та $L(x)$.

2. Боб обирає довільний вектор $\bar{a}$, розмірності $1 \times q$. Боб розв'язує систему рівнянь $l(\bar{x}) = v,$ де v – повідомлення, яким Боб хоче обмінятися з Алісою. Далі Боб обчислює значення $l(\bar{a}) = d, L(\bar{x} + \bar{a}) = d_1$ в кільці G_k. Після цього Боб надсилає значення d та d_1 відкритим каналом Алісі, а значення вектора v, тобто початкового повідомлення, яке він хоче відправити, зберігає у секреті.

   Аліса обраховує обернені матриці до матриць B_i в кільці G_k. Аліса знаходить значення вектора v, оскільки має для цього всі необхідні дані.

1.4.2 Коректність наведеного алгоритму

Нижче представлено доведення того, що наведений протокол обміну працює коректно. Воно базується на властивостях лінійних операторів і функцій в асоціативно-комутативному кільці з одиницею. Якщо

$$d_1 = L(\bar{x} + \bar{a}) = B_r(B_{r-1} \dots B_2(B_1(l(\bar{x} + \bar{a}) + a_1) \dots + a_{r-1}) + a_r) + a_{r+1}$$

То

$$B_1^{-1}(B_2^{-1}(\dots B_{r-1}^{-1}(B_r^{-1}(d_1 - a_{r+1}) - a_r) \dots - a_2) - a_1 =$$

$$B_1^{-1}(B_2^{-1}(\dots B_{r-1}^{-1}(B_r^{-1}(B_r(B_{r-1} \dots B_2(B_1(l(\bar{x} + \bar{a}) + a_1) \dots + a_{r-1}) + a_r) + a_{r+1}) - a_{r+1}) - a_r) \dots - a_2) - a_1 = l(\bar{x} + \bar{a}).$$

З цього випливає, що $l(\bar{x}) = l(\bar{x} + \bar{a}) - l(\bar{a}) = l(\bar{x} + \bar{a}) - d.$

1.4.3 Криптоаналіз протоколу

Першим очевидним кроком при спробі злому алгоритму є спроба розв'язати систему лінійних рівнянь

$$l(\bar{a}) = d(mod\ k) \qquad (1)$$

для того, щоб знайти $l(\bar{a})$. Для цього потрібно знайти довільний розв'язок в асоціативно-комутативному кільці G_k, який задовольняє це рівняння, тобто розв'язок не обов'язково має дорівнювати початковому вектору $\bar{a}$.

Другим кроком є знаходження оберненої матриці D^{-1} для матриці що відповідає виразу $L(x)$ та систему рівнянь $l(x)$ і значення виразу $L(\bar{a})$. Для цього потрібно розв'язати систему рівнянь

$$D^{-1}\left(B_r\left(B_{r-1}\dots B_2(B_1(l(x)+a)\dots)\right)\right)=(a_{11},\ a_{21},\dots,a_{m1})^T. \qquad (2)$$

Обчислити різницю $L(\bar{x}+\bar{a})-L(\bar{a})$ і знайти значення $l(\bar{x})$.

$$L(\bar{x}+\bar{a})-L(\bar{a})=d_1\ (mod\ k) \qquad (3)$$

Описані кроки, виконані криптоаналітиком завершаться успіхом якщо останній знає ізоморфізм між кільцем лишків і асоціативно-комутативним кільцем з одиницею. Однак якщо ізоморфізм йому невідомий, то він не зможе знайти розв'язки (1), (2) і (3). Отже стійкість протоколу базується на ізоморфізмі.

Виходячи з побудови ізоморфізму, кількість кілець, ізоморфних кільцю лишків порядку k буде дорівнювати $(k-2)!$. Отже якщо змінювати визначальний рядок з кожною сесією передачі інформації, то протокол буде мати необхідну стійкість.

Залишається сформулювати умови сумісності системи лінійних діофантових рівнянь $l(\bar{x})=v$ в кільці Z_k. При чому ця система рівнянь повинна бути сумісною для довільного v. Сумісність такої системи випливає із теореми Фредгольма.

Теорема 2. Система $l(\bar{x})=v$ сумісна тоді і тільки тоді, коли кожний розв'язок системи лінійних однорідних рівнянь $A^T*u=0$ ортогональний вектору v, де A^T – матриця транспонована до матриці системи $l(\bar{x})=v$ [4].

З цієї теореми випливає, що коли серед рівнянь системи $l(\bar{x}) = v$ немає лінійно залежних рівнянь, то система однорідних рівнянь $A^T * u = 0$ має лише тривіальний нульовий розв'язок, який ортогональний до довільного вектора v.

РОЗДІЛ 2 ПРАКТИЧНА ЧАСТИНА

2.1 Використані технології

2.1.1 Мова програмування Java

Java є однією з найбільш популярних мов програмування в світі, що використовується для розробки великої кількості різних програмних продуктів, від мобільних додатків до великих корпоративних систем. Вона відома своєю кросплатформенністю, тобто програми, написані на цій мові, можуть запускатися на різних операційних системах без внесення додаткових змін. Java також має багату стандартну бібліотеку, яка містить велику кількість корисних класів і інтерфейсів, що спрощує розробку програм і дозволяє програмістам швидко створювати робочі прототипи.

Одна з ключових особливостей мови Java є її механізм автоматичного управління пам'яттю. Це означає, що програмістам не потрібно вручну виконувати операції з вивільнення та виділення пам'яті для об'єктів. Замість цього, Java використовує так званий "смітник" (garbage collector), який автоматично вивільняє пам'ять, яка більше не використовується програмою. Цей механізм зменшує кількість помилок, пов'язаних з управлінням пам'яттю, та полегшує життя програмістам.

Java є також мовою з високим рівнем абстракції, тобто програмісти можуть працювати зі складними структурами даних і алгоритмами на більш високому рівні абстракції, що дозволяє їм ефективніше вирішувати завдання і зменшує ризик помилок. Мова Java також має підтримку об'єктно-орієнтованого програмування, що дає можливість програмістам легко створювати і використовувати класи.

Ще однією особливістю Java є її підтримка мультипоточності. Java має вбудовану підтримку мультипотокового програмування, що дозволяє

програмістам створювати програми, які виконують багато завдань одночасно. Це дозволяє програмам працювати швидше та ефективніше і розподіляти завдання між різними потоками. Java має вбудовані механізми синхронізації та координації між потоками, що забезпечує безпеку виконання потоків та уникнення конфліктів.

Узагалі, Java є потужною та зручною мовою програмування, яка використовується для розробки великої кількості різноманітних програмних продуктів. Її безпека, кросплатформенність та автоматичне управління пам'яттю роблять її дуже привабливою для програмістів, які шукають ефективний спосіб розробки програм. Крім того, вона має велику кількість функцій, бібліотек та інструментів, що сприяє ефективній та швидкій розробці програм.

Отже враховуючи всі вищеописані фактори було вирішено обрати мову програмування Java версії 11 для реалізації протоколу обміну повідомленнями. Java 11 є версією з тривалою підтримкою (LTS), що забезпечує її підтримку як мінімум до вересня 2026 року.

2.1.2 Інструмент управління залежностями Maven

Maven є одним з найпопулярніших інструментів у світі Java-розробки. Це система управління проектами та залежностями, що дозволяє автоматизувати процеси збірки, тестування та розгортання програмного забезпечення. Цей інструмент дозволяє зменшити зусилля, необхідні для розробки та підтримки проектів, та сприяє швидкій розробці високоякісного програмного забезпечення.

Основною перевагою Maven є його система управління залежностями. Він дозволяє визначити залежності проекту та автоматично завантажує їх з централізованого репозиторію, що дозволяє забезпечити легке та швидке управління залежностями та забезпечити їх відповідність за допомогою

коректної версії. Maven також дозволяє додавати власні залежності до проекту, що дає змогу програмістам працювати з різними бібліотеками та інструментами для розробки.

За допомогою інструменту Maven розробники можуть автоматизувати процес збірки та розгортання програмного забезпечення, що дає можливість більш ефективного керування програмними продуктами. Крім того, Maven дозволяє програмістам створювати свої власні плагіни та додатки, що полегшують процес вирішення різноманітних завдань розробки.

2.1.3 Брокер повідомлень RabbitMQ

У процесі передачі повідомлень використовується RabbitMQ, брокер повідомлень, що функціонує на основі протоколу AMQP (Advanced Message Queuing Protocol), який дозволяє йому працювати з різними мовами програмування та платформами. [9]. Брокер повідомлень - це програмний інструмент для обміну повідомленнями між додатками, від відправника до отримувача. Серед широко застосовуваних брокерів можна виділити RabbitMQ, Apache Kafka, Redis, Amazon SQS та інші.

Однією з основних переваг RabbitMQ є його система повідомлень на базі черг. Ця система дозволяє зберігати повідомлення в черзі, поки вони не будуть готові для обробки. Коли повідомлення готове для обробки, воно вилучається з черги та передається до відповідного компонента програмного забезпечення. Це дозволяє підвищити надійність та масштабованість системи, забезпечуючи повну обробку повідомлень та запобігаючи втраті даних.

Іншою важливою перевагою RabbitMQ є його підтримка різноманітних механізмів обміну повідомленнями. Брокер повідомлень може використовувати різні механізми для передачі повідомлень, такі як прямий обмін, тематичний обмін, фанаут-обмін та інші. Це дозволяє програмістам вибрати найбільш підходящий механізм для їх потреб та забезпечити

ефективну комунікацію між різними компонентами програмного забезпечення.

RabbitMQ також надає можливість зберігати дані в пам'яті кешу та диску, що забезпечує швидкий доступ до даних та зменшує навантаження на базу даних. Також, RabbitMQ може використовувати різні інструменти для забезпечення масштабованості та відмовостійкості, такі як кластеризація та реплікація. Ці інструменти дозволяють розподіляти трафік між різними вузлами та забезпечувати неперервну роботу системи при виникненні збоїв.

Ще однією важливою функцією RabbitMQ є його підтримка SSL-шифрування. Це дає можливість забезпечити безпеку та конфіденційність обміну повідомленнями між різними компонентами програмного забезпечення. RabbitMQ також має підтримку для різних форматів повідомлень, таких як JSON, XML та інші, що дозволяє програмістам вибрати найбільш підходящий формат для їх потреб.

Крім того, RabbitMQ має велику кількість різноманітних інтеграцій з іншими інструментами та системами. Він може бути інтегрований з різними мовами програмування, такими як Java, Python, Ruby та інші, а також з різними базами даних та іншими сервісами, такими як Amazon Web Services, Microsoft Azure та інші.

В даному проекті обрано RabbitMQ через простоту встановлення та налаштування, а також гарантію збереження послідовності повідомлень [10]. Відправник передає повідомлення до компонента брокера, званого точкою обміну, до якого прив'язані одна або кілька черг. Обмінник розподіляє повідомлення відповідно до черг, звідки їх отримує споживач. Таким чином, обмінник відіграє роль відправної точки, а черга - точки виходу брокера. Для підключення до RabbitMQ сервера, потрібно задати параметри з'єднання, такі як назва хоста, номер порту, ім'я користувача, пароль тощо. У цій реалізації параметри вказуються у файлі application.properties, як показано на Рисунок 2.

```
rabbitmq.host=localhost
rabbitmq.port=5672
rabbitmq.username=username
rabbitmq.password=password
```

Рисунок 2 - Параметри для підключення до брокера повідомлень RabbitMQ

2.1.4 Технології Java Swing та Java AWT

Java AWT (Abstract Window Toolkit) та Java Swing є двома основними бібліотеками графічного інтерфейсу користувача для мови програмування Java.

Java AWT є низькорівневою бібліотекою для розробки графічного інтерфейсу. Вона забезпечує доступ до базових компонентів, таких як кнопки, поля введення, меню та інші. Java AWT використовує віджети, щоб створювати графічний інтерфейс користувача, та менеджери розміщення для їх вирівнювання. Недоліком Java AWT є те, що вона може виглядати неоднаково на різних операційних системах та може не мати підтримки для новіших функцій графічного інтерфейсу.

У свою чергу Java Swing є високорівневою бібліотекою для розробки графічного інтерфейсу користувача на мові Java. Вона дозволяє створювати більш складні інтерфейси, ніж Java AWT, та має більш розширену функціональність. Ця бібліотека використовує моделі компонентів, які дозволяють розробникам створювати свої власні компоненти та управляти ними. Недоліком Java Swing є те, що вона може вимагати більш великої кількості ресурсів комп'ютера, ніж Java AWT, та через може працювати більш повільно.

2.2 Реалізація протоколу

2.2.1 Архітектура системи

Оскільки всі операції над числами проводяться в кільці лишків за модулем k, то для зручності реалізації було вирішено ввести клас RingInteger, який інкапсулює в собі всю логіку роботи з полями лишків і видає зручні API для користування. Зокрема в класі RingInteger реалізовані наступні операції, описані в Таблиця 4:

Сигнатура метода	Опис операції	Значення, що повертається
public RingInteger add(RingInteger other)	Додавання двох чисел в полі лишків	$a + b \ (mod\ k)$
public RingInteger substract(RingInteger other)	Віднімання двох чисел в полі лишків	$a - b \ (mod\ k)$
public RingInteger complement()	Знаходження оберненого елемента (доповнення до числа)	$k - a \ (mod\ k)$
public RingInteger multiply(RingInteger other)	Множення двох чисел в полі лишків	$a * b \ (mod\ k)$
public RingInteger divide(RingInteger other)	Ділення двох чисел в полі лишків	$a \div b \ (mod\ k)$

Таблиця 4 - операції, які надає клас RingInteger

За допомогою вищеописаних операцій була реалізована вся логіка протоколу, зокрема алгоритм побудови таблиць додавання і множення псевдокод яких наведено нижче.

$additionTable(a, k)$

1. Визначити масив $T[k \times k]$
2. Присвоїти першому рядку результат
додавання з нулем
3. Присвоїти другому рядку результат
додавання з одиницею (визначний рядок)
4, Присвоїти $c := 1$
5. Вибрати елемент $c' = c + 1$
6. Для всіх x в T в рядку під номером c'просумувати $c' + x = (c + 1) + x$
$= c + (1 + x)$
7. $c := c', c' = c + 1$, якщо $c' = 0$ то зупинитися, інакше перейти на крок 6.

Реалізація побудови таблиці додавання наведена на Рисунок 3.

```
public static RingInteger[][] additionTable(int k, RingInteger[] unityRow) {
    RingInteger[][] table = new RingInteger[k][k];
    for(int i = 0; i < k; ++i) {
        Arrays.fill(table[i], RingInteger.zero(k));
        table[0][i] = RingInteger.valueOf(i, k);
        table[i][0] = RingInteger.valueOf(i, k);
        table[1][i] = unityRow[i];
        table[i][1] = unityRow[i];
    }
    int prevElement = 1;
    while (!table[prevElement][1].equals(RingInteger.zero(k))) {
        RingInteger element = table[prevElement][1];
        for(int i = 2; i < k; ++i) {
            table[element.getNumber()][i] = table[1][table[prevElement][i].getNumber()];
            table[i][element.getNumber()] = table[element.getNumber()][i];
        }
        prevElement = element.getNumber();
    }
    log.debug(Arrays.deepToString(table));
    return table;
}
```

Рисунок 3 – Реалізація побудови таблиці додавання

Процес побудови таблиці множення можна описати за допомогою наступного псевдокоду:

$multiplicationTable(a, k)$

1. Визначити масив $T[k \times k]$
2. Присвоїти першому рядку результат
множення з нулем, а другому – з одиницею
3. Присвоїти $c := 1$
4. Вибрати елемент $c' = c + 1$
5. Для всіх x в T в рядку під номером c'виконати $c' * x = (c + 1) * x$
$= c * x + x$
6. $c := c', c' = c + 1$, якщо $c' = 0$ то зупинитися, інакше перейти на крок 5.

Реалізація цього алгоритму наведена на Рисунок *4*.

```
public static RingInteger[][] multiplicationTable(int k, RingInteger[][] additionTable) {
    RingInteger[][] table = new RingInteger[k][k];
    for(int i = 0; i < k; ++i) {
        Arrays.fill(table[i], RingInteger.zero(k));
        table[0][i] = RingInteger.zero(k);
        table[i][0] = RingInteger.zero(k);
        table[1][i] = RingInteger.valueOf(i, k);
        table[i][1] = RingInteger.valueOf(i, k);
    }
    int prevElement = 1;
    while (!additionTable[prevElement][1].equals(RingInteger.zero(k))) {
        RingInteger element = additionTable[prevElement][1];
        for(int i = 2; i < k; ++i) {
            table[element.getNumber()][i] = additionTable[i][table[prevElement][i].getNumber()];
            table[i][element.getNumber()] = table[element.getNumber()][i];
        }
        prevElement = element.getNumber();
    }
    log.debug(Arrays.deepToString(table));
    return table;
}
```

Рисунок 4 - Реалізація алгоритму побудови таблиці множення

Нехай заданий наступний визначальний рядок: $g(0) = 1$, $g(1) = 4$, $g(2) = 3, g(3) = 0, g(4) = 2$. Тоді таблиці додавання і множення будуть мати наступний вигляд (Рисунок 5, Рисунок 6):

	0	1	2	3	4
0	0	1	2	3	4
1	1	4	3	0	2
2	2	3	1	4	0
3	3	0	4	2	1
4	4	2	0	1	3

Рисунок 5 - Таблиця додавання

	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	3	4	1
3	0	3	4	1	2
4	0	4	1	2	3

Рисунок 6 - Таблиця множення

2.2.2 Реалізація TSS алгоритму для розв’язку системи лінійних діофантових рівнянь

Основою вищеописаного алгоритму є розв’язання системи лінійних неоднорідних діофантових рівнянь в кільці ликів. Для знаходження базису множини всіх розв’язків СЛНДР використовується модифікований TSS алгоритм.

Нехай задана така система рівнянь:

$$S = \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n = b_q \end{cases} (mod\ m),$$

де $a_{ij}, b_i \in Z_m$. І нехай модуль має розклад на прості множники $m = p_1^{k_1} * p_2^{k_2} * \ldots * p_r^{k_r}$, тоді система S має еквівалентну їй систему, яка записується у вигляді

$$S' = \begin{cases} \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n = b_q \end{cases} (mod\ p_1^{k_1}) \\ \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n = b_q \end{cases} (mod\ p_2^{k_2}) \\ \cdots \\ \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n = b_q \end{cases} (mod\ p_r^{k_r}) \end{cases},$$

Еквівалентність цих двох систем випливає з того факту, що коли x є розв'язком системи S, то він буде ж і розв'язком кожної з підсистем S' за модулем $p_i^{k_i}$, тому що модуль m ділиться на кожне з чисел $p_i^{k_i}, i = 1, \ldots, r$. З іншого боку коли x є розв'язком системи S', то він також буде задовольняти розв'язкам будь-якої з її підсистем, а значить він буде розв'язком системи S, тому що модулі $p_i^{k_i}$ є взаємнопростими і, як вказано вище, $p_1^{k_1} * p_2^{k_2} * \ldots * p_r^{k_r} = m$.

Якщо ж перетворити систему лінійних неоднорідних рівнянь S' на систему лінійних однорідних рівнянь S'', то буде отримано систему наступного вигляду:

$$S'' = \begin{cases} S_1 = \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n - b_1x_0 = 0 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n - b_2x_0 = 0 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n - b_qx_0 = 0 \end{cases} (mod\ p_1^{k_1}) \\ S_2 = \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n - b_1x_0 = 0 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n - b_2x_0 = 0 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n - b_qx_0 = 0 \end{cases} (mod\ p_2^{k_2}) \\ \cdots \\ S_r = \begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n - b_1x_0 = 0 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n - b_2x_0 = 0 \\ \cdots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n - b_qx_0 = 0 \end{cases} (mod\ p_r^{k_r}) \end{cases},$$

Нехай вектор x_1 є розв'язком підсистеми S_1, тоді $x_1 * p_1^{\frac{m}{k_1}}$ буде розв'язком системи S''. Тобто для другої підсистеми S_2, яка є аналогічна першій лише за модулем $p_2^{k_2}$, для будь-якого рівняння $L_i\left(x_1 * p_1^{\frac{m}{k_1}}\right) = p_1^{\frac{m}{k_1}} * L_i(x_1) = p_1^{\frac{m}{k_1}} * a_i \equiv 0\ (mod\ p_2^{k_2})$, оскільки вираз $p_1^{\frac{m}{k_1}}$ кратний m, а a_i кратне $p_2^{k_2}$.

Аналогічні дії можна проробити і для розв'язку підсистеми $S_2, S_3, \dots, S_r$. Отже загальний розв'язок системи можна представити у вигляді $x = \sum_{i=1}^{r} p_i^{\frac{m}{k_i}} * x_i$.

З вищеописаних кроків можна зробити висновок, що для того щоб розв’язати систему лінійних діофантових рівнянь S в кільці лишків потрібно розв’язати супутні системи або в примарному кільці (у випадку якщо $k_i > 1$) або в полі лишків за модулем p_i. Алгоритми розв’язку таких систем можна знайти у [4].

2.2.3 Знаходження обернених матриць

Як зазначено вище, для роботи протоколу потрібне існування невироджених квадратних матриць B_i. Далі описаний спосіб побудови невиродженої квадратної матриці:

1. Вводиться верхньотрикутна або нижньотрикутна матриця, де всі діагональні елементи відмінні від нуля.
2. Застосовуються елементарні перетворення над рядками матриці, такі як перестановка рядків, множення рядка на ненульову константу та додавання одного рядка до іншого, помноженого на ненульову константу.

Отримана таким чином матриця буде невиродженою, оскільки її детермінант, який визначається як добуток елементів головної діагоналі, не дорівнює нулю, а ранг матриці співпадає з її розмірності [3].

Оскільки в алгоритмі потрібно обчислювати обернені матриці для матриць B_i, для прискорення роботи алгоритму було вирішено задати їх у вигляді

$$B_i = \begin{pmatrix} \boldsymbol{I_m} & \boldsymbol{A} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{I_m} & \boldsymbol{B} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{I_m} \end{pmatrix},$$

де I_m - одинична квадратна матриця розмірності $m \times m, m = \frac{n}{3}$, матриці A та B - невироджені квадратні матриці розмірності $m \times m$, побудовані за вищеописаним алгоритмом.

Існує ще один спосіб побудови обернених матриць. Нехай задана матриця $B = \begin{pmatrix} \boldsymbol{a_{11}} & \boldsymbol{a_{12}} & \dots & \boldsymbol{a_{1n}} \\ \boldsymbol{a_{21}} & \boldsymbol{a_{22}} & \dots & \boldsymbol{a_{2n}} \\ \dots & \dots & \dots & \dots \\ \boldsymbol{a_{n1}} & \boldsymbol{a_{n2}} & \dots & \boldsymbol{a_{nn}} \end{pmatrix}$ в кільці лишків по модулю k, для якої потрібно знайти обернену.

Описаним вище TSS методом розв'язується система рівнянь вигляду

$$\begin{cases} L_1(x) = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = 1 \\ L_2(x) = a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = 0 \\ \dots \\ L_q(x) = a_{q1}x_1 + a_{q2}x_2 + \cdots + a_{qn}x_n = 0 \end{cases} (mod\ k).$$

Координати отриманого в результаті розв'язку вектору будуть елементами першого стовпчика оберненої матриці. Аналогічним чином знаходимо другий стовпчик, з тою лише відмінністю, що стовпчик вільних членів системи буде складатися з нулів, окрім другого, який буде одиницею. Таким чином після n подібних операцій буде знайдено шукану матрицю, обернену до початкової.

Цей запропонований спосіб на відміну від попереднього не накладає жодних обмежень на матрицю, однак є більш затратний з часової точки зору.

2.2.4 Підготовка повідомлення

Наступним етапом в реалізації є процес співвідношення символів вхідного тексту до певних елементів кільця. З метою спрощення цією задачі, кожному символу алфавіту було присвоєно додатне число, починаючи з нуля.

Описаний підхід реалізований в класі Encoder, а відображення між символами алфавіту та відповідним числом з кільця зберігається у двонаправленному словнику, що представлений об'єктом класу

com.google.common.collect.HashBiMap. Двонаправлений словник - це словник, який забезпечує унікальність як ключів, так і значень, що в ньому містяться. Така особливість дозволяє створити відповідний зворотний словник, що складається з тих самих пар елементів, але зі значеннями початкового словника в якості ключів та ключами як значеннями. Вхідний алфавіт, а також нумерація його символів, подані у Додатку А.

Оскільки довжина повідомлення не завжди збігається з розмірами систем рівнянь $L(x)$ та $l(x)$, повідомлення передається у вигляді блоків довжиною m, де m - кількість рівнянь у системі $L(x)$. Якщо розмір блоку менший від m, блок доповнюється порожніми символами для досягнення потрібної довжини.

2.2.5 Приклад роботи протоколу

У цьому пункті наведений покроковий опис виконаних дій алгоритму з проміжними результатами. Нехай два абоненти – відправник (Боб) та отримувач (Аліса) хочуть обмінятися повідомленням «Hello, world!». Для зручності порядок поля був обраний рівний 125, що дозволяє абонентам використовувати всі великі і маленькі літери латинського алфавіту, цифри, а також інші символи, доступні на стандартній клавіатурі. За необхідністю порядок поля може бути збільшено або зменшено.

Нехай перед початком процесу обміну абоненти за допомогою секретного каналу обмінялися значеннями a і c, необхідними для побудови ізоморфізму і вибрали ці значення рівними $a = 7$ та $c = 7$. $\text{НСД}(7, 125) = 1$, а отже ці значення задовольняють вимогам протоколу.

Потрібно зазначити важливу особливість, яку дає наявність ізоморфізму між кільцем лишків Z_k та асоціативно-комутативним кільцем з одиницею G_k. Маючи такий ізоморфізм для виконання операцій над числами абоненту не обов’язково будувати таблиці додавання і множення, достатньо лише отримані

через зовнішній канал елементи з АКК за допомогою ізоморфізму відобразити у елементи кільця лишків Z_k, виконати операції над елементами відомим способом і перед відправкою абоненту зовнішнім каналом відобразити ці елементи назад у АКК. Таким чином це значно економить пам'ять і час, необхідну для роботи протоколу, оскільки не потребується збереження таблиць великої розмірності і не проводиться обчислення значень цих таблиць.

Аліса згенерувала таку систему рівнянь:

$$l_{Z_{125}}(x) = \begin{cases} 5x_1 + 6x_2 + 9x_3 + 21x_4 \\ 1x_2 + 11x_3 + 14x_4 \end{cases},$$

яка може бути представлена у вигляді матриці наступним чином:

$$\boldsymbol{A} = \begin{pmatrix} \mathbf{5} & \mathbf{6} & \mathbf{9} & \mathbf{21} \\ \mathbf{0} & \mathbf{1} & \mathbf{11} & \mathbf{14} \end{pmatrix}.$$

Матриця B має вигляд $\begin{pmatrix} \mathbf{2} & \mathbf{1} \\ \mathbf{24} & \mathbf{24} \end{pmatrix}$, вектори зсуву $a_0 = (83, 86)^T, a_1 = (28, 95)^T$.

Аліса обчислює вираз $L_{Z_{125}}(x) = B(l_{Z_{125}}(x) + a_0) + a_1$

$$\begin{pmatrix} \mathbf{2} & \mathbf{1} \\ \mathbf{24} & \mathbf{24} \end{pmatrix} \begin{pmatrix} \mathbf{5} & \mathbf{6} & \mathbf{9} & \mathbf{21} & \mathbf{83} \\ \mathbf{0} & \mathbf{1} & \mathbf{11} & \mathbf{14} & \mathbf{86} \end{pmatrix} + \begin{pmatrix} \mathbf{28} \\ \mathbf{95} \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{10} & \mathbf{13} & \mathbf{29} & \mathbf{56} & \mathbf{2} \\ \mathbf{120} & \mathbf{43} & \mathbf{105} & \mathbf{90} & \mathbf{56} \end{pmatrix} + \begin{pmatrix} \mathbf{28} \\ \mathbf{95} \end{pmatrix} =$$

$$\begin{pmatrix} \mathbf{10} & \mathbf{13} & \mathbf{29} & \mathbf{56} & \mathbf{30} \\ \mathbf{120} & \mathbf{43} & \mathbf{105} & \mathbf{90} & \mathbf{26} \end{pmatrix}$$

Перед відправкою Бобу матриць, що відповідають виразам $l_{Z_{125}}(x)$ та $L_{Z_{125}}(x)$ Аліса відображає елементи матриць у АКК G_{125} і відповідна матриця приймає вигляд

$$D_{G_{125}} = \begin{pmatrix} \mathbf{70} & \mathbf{91} & \mathbf{78} & \mathbf{17} & \mathbf{85} \\ \mathbf{90} & \mathbf{51} & \mathbf{110} & \mathbf{5} & \mathbf{57} \end{pmatrix},$$

А елементи матриці А у АКК приймають наступний вигляд:

$$A_{G_{125}} = \begin{pmatrix} \mathbf{35} & \mathbf{42} & \mathbf{63} & \mathbf{22} \\ \mathbf{0} & \mathbf{1} & \mathbf{77} & \mathbf{98} \end{pmatrix}.$$

Після цього Аліса надсилає матриці $A_{G_{125}}$ та $D_{G_{125}}$ відкритим каналом.

Після отримання повідомлення Боб бачить, що системи складаються з двох рівнянь, а отже він має розбити своє повідомлення на блоки по два елементи. Він бере перші дві літери повідомлення – «Не», замінює літери числами з кільця, та отримує вектор $v = (44, 4)$, який буде передавати Алісі.

Боб трансформує отримані коефіцієнти систем назад у кільце лишків і за допомогою TSS метода знаходить розв'язок системи

$$l_{Z_{125}}(x) = \begin{cases} 5x_1 + 6x_2 + 9x_3 + 21x_4 = 44 \\ 1x_2 + 11x_3 + 14x_4 = 4 \end{cases}.$$

Отримує вектор $\bar{x} = (34,44,69,59)$.

Боб вибирає випадковий вектор $\bar{a} = (92,119,74,43)$ та обчислює значення виразів

$$d = l_{Z_{125}}(\bar{a}) = (118,35), d_1 = L_{Z_{125}}(\bar{x} + \bar{a}) = (18,100).$$

Переводить отримані вектори в АКК і отримує $d_{G_{125}} = (76,120), d_{1_{G125}} = (7,75)$ і передає їх відкритим каналом Алісі.

Після отримання Аліса знов переводить вектори в зручне для обрахунків кільце лишків Z_{125} та виконує наступні обчислення – $B^{-1}(d_1 - a_1) - a_0 - d = \begin{pmatrix} \mathbf{1} & \mathbf{26} \\ \mathbf{124} & \mathbf{73} \end{pmatrix}\left(\begin{pmatrix} \mathbf{18} \\ \mathbf{100} \end{pmatrix} - \begin{pmatrix} \mathbf{28} \\ \mathbf{95} \end{pmatrix}\right) - \begin{pmatrix} \mathbf{83} \\ \mathbf{86} \end{pmatrix} - \begin{pmatrix} \mathbf{118} \\ \mathbf{35} \end{pmatrix} = \begin{pmatrix} \mathbf{44} \\ \mathbf{4} \end{pmatrix} = v$, що відповідає двом літерами «Не». Проміжні кроки для аналогічних дій для передачі решти повідомлення наведені у Таблиця *5*.

Вихідні символи	Закодовані символи	$\bar{x}$	$\bar{a}$	d	d_1	Отримане повідомлення
ll	$(11, 11)$	(54,116, 98,103)	$(88, 102, 66, 73)$	$(54, 100$	$(21, 0)$	$(11, 11)$

o,	(14,64)	(32,19,77,32)	(110,47,5,76)	(98,41)	(109,109)	(14,64)
<пробіл>w	(26,22)	(19,24,28,85)	(0,93,66,50)	(77,19)	(77,19)	(26,22)
or	(14,17)	(29,63,17,28)	(79,33,65,38)	(101,30)	(57,39)	(14,17)
ld	(11,3)	(10,107,120,59)	(86,43,102,114)	(0,11)	(66,1)	(11,3)
!<порожній символ>	(67,87)	(45,27,38,28)	(13,58,28,93)	(118,43)	(30,86)	(67,87)

Таблиця 5 - Проміжні кроки

2.2.6 Демонстрація роботи програми

Для того, щоб наочно продемонструвати роботу програми був створений десктопний застосунок з використанням технологій Java Swing та Java AWT, а також брокера повідомлень RabbitMQ.

Перед запуском програми її необхідно зібрати за допомогою утиліти Maven використовуючи команду mvn clean install. Після цього можна запускати програму. Для старту застосунку використовується наступна команда java -jar message-exchange-protocol.jar <режим> <порядок кільця>. Наприклад для запуску у режимі користувача команда буде мати такий вигляд: java -jar message-exchange-protocol.jar sender 125, а у режимі отримувача – java -jar message-exchange-protocol.jar receiver 125.

Інтерфейс програми дуже простий і інтуїтивно зрозумілий. Для вводу повідомлення використовується текстове поле у нижній частині вікна програми, процес відправки починається після натискання кнопки Send поруч

із полем вводу. Як тільки все повідомлення буде отримано і розшифровано отримувачем, то воно з’явиться на екрані.

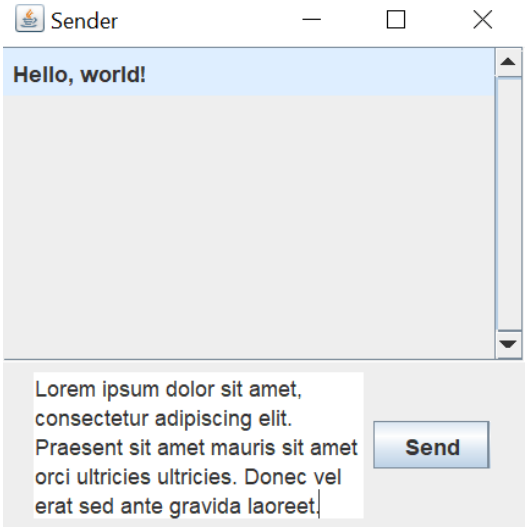

Рисунок 7 - Вигляд вікна для відправки повідомлення

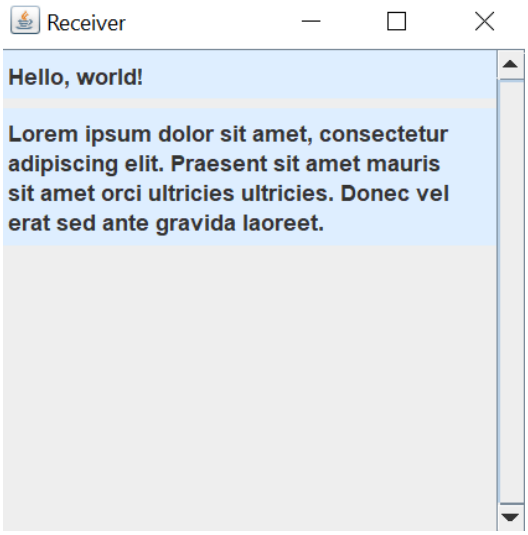

Рисунок 8 - Вікно отримувача

ВИСНОВКИ

В даній роботі було досліджено протокол обміну повідомленнями на основі ізоморфізму кілець з використанням лінійних функцій і операторів. В порівнянні з іншими існуючими системами шифрування даний перевага даного протоколу полягає у тому, що для нього незастосовний метод частотного аналізу. Друга перевага полягає у тому, що наведені операції є досить простими, але в той же час область над якою вони проводяться є закритою для зловмисника. До того ж використання ізоморфізму кілець дає можливість уникнути побудови таблиць додавання і множення, а самі кільця можуть бути невеликого порядку.

Надійність описаного протоколу базується на кількості ізоморфних кілець. Наприклад якщо порядок кільця лишків складає 125, то кількість ізоморфних йому асоціативно-комутативних кілець складає $123!$.

Недоліком системи можна вважати той факт, що довжина зашифрованого повідомлення вдвічі більша за довжину вихідного тексту.

Створений в результаті програмний продукт може використовуватися в будь-якій системі, де цілісність і надійність даних є ключовим пріоритетом.

Підхід до шифрування, описаний у даному протоколі може бути вдосконалений використанням прямого добутку кілець $G = G_1 \times G_2 \times \ldots \times G_s$. Перше кільце G_1 використовується для обміну ключами, друге – для шифрування, третє для обміну додатковою інформацією, тощо.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dummit D. Abstract Algebra / D. Dummit, R. Foote., 2004. – 408 с. – (3).
2. Косторикин А. И. Введение в алгебру / А. И. Косторикин., 2004. – 272 с. – (3).
3. Курош А. Г. Курс высшей алгебры / А. Г. Курош.. – 431 с. – (9).
4. Кривий С. Л. Лінійні діафантові обмеження та їх застосування / С. Л. Кривий., 2021. – 260 с.
5. Ганюшкін О. Г. Теорія груп / О. Г. Ганюшкін, О. О. Безущак., 2005. – 123 с.
6. Andreescu T. An Introduction to Diophantine equations / T. Andreescu, D. Andrica, I. Cucurezeanu.. – 358 с.
7. Coleman R. Fermat’s Last Theorem: an Introduction / R. Coleman, L. Zwald., 2022.
8. Shoup V. A Computational Introduction to Number Theoryand Algebra / Victor Shoup., 2008. – 588 с.
9. AMQP Advanced Message Queuing Protocol Specification [Електронний ресурс] // 0-9-1. – 2008. – Режим доступу до ресурса: https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf.
10. Videla A. RabbitMQ in Action / A. Videla, J. Williams. – Shelter Island: Manning Publications Co., 2012. – 288 с.
11. Symmetric System for Exchange Information on the Base of Surjective Isomorphism of Rings / С. Л.Кривий, О. О. Гріненко, В. Опанасенко, Ю. О. Нортман. // The 12th International Conference Dependable Systems, Services and Technologies, DESSERT’2022. – 2022.

ДОДАТОК А

Символ	Номер
a	1
b	2
c	3
d	4
e	5
f	6
g	7
h	8
i	9
j	10
k	11
l	12
m	13
n	14
o	15
p	16
q	17
r	18
s	19
t	20
u	21
v	22
w	23
x	24

y	25
z	26
<пробіл>	27
0	28
1	29
2	30
3	31
4	32
5	33
6	34
7	35
8	36
9	37
A	38
B	39
C	40
D	41
E	42
F	43
G	44
H	45
I	46
J	47
K	48
L	49

M	50
N	51
O	52
P	53
R	54
S	55
T	56
U	57
V	58
W	59
X	60
Y	61
Z	62
.	63
,	64
?	65
/	66
!	67
@	68
#	69
$	70
;	71
{	72
}	73
[	74