МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

Вершигора Максим Олександрович

Допускається до захисту:
в.о. завідувача кафедри
прикладної математики та
кібербезпеки,
д-р філософії з математики
_____Луценко А.В.
«___» _____ 20__ р.

Криптосистема на основі тернарних самоортогональних квазігруп

Спеціальність 113 Прикладна математика

Кваліфікаційна (магістерська) робота

Науковий керівник:
Сохацький Ф.М. доцент кафедри
прикладної математики та кібербезпеки
д-р фіз.-мат. наук, доцент

Оцінка: _____ / _____ / _____
(Бали/за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця — 2024

Анотація

Вершигора М.О. Криптосистема на основі тернарних самоортогональних квазігруп. Спеціальність 113 “Прикладна математика”, Освітня програма “Прикладна математика”. Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній роботі описано алгоритм побудови класу лівосиметричних квазігруп 32-го порядку. Встановлено, що за допомогою описаного алгоритму можна побудувати $2*6^{33}$ квазігруп 32-го порядку. Розроблено алгоритм симетричного шифрування інформації за допомогою тернарної квазігрупи 32-го порядку, яка будується як безповторна композиція двох бінарних квазігруп.

Ключові слова: квазігрупи, тернарні квазігрупи, самоортогональні квазігрупи, схрещений добуток, таблиця Келі, криптографія, шифр, секретний ключ, зашифрування, розшифрування, симетричне шифрування, кодування, алгоритм, криптоалгоритми, кібербезпека.

Документ має 58 сторінок та 40 джерел.

Annotation

Vershyhora M.O. Cryptosystems based on ternary self-orthogonal quasigroups. Specialty 113 “Applied Mathematics”, Programme “Applied Mathematics”. Vasyl’ Stus Donetsk National University, Vinnytsia, 2024.

In this qualifying paper we describe an algorithm for constructing a class of left-symmetric quasigroups of order 32. It is established that the described algorithm can be used to construct $2 * 6^{33}$ quasigroups of order 32. An algorithm for symmetric encryption of information using a ternary quasigroup of order 32, which is constructed as a unique composition of two binary quasigroups, is developed.

Keywords: quasigroups, ternary quasigroups, self-orthogonal quasigroups, cross product, Cayley table, cryptography, cipher, secret key, encryption, decryption, symmetric encryption, coding, algorithm, cryptographic algorithms, cybersecurity.

The document has 58 pages and 40 sources.

ЗМІСТ

ВСТУП

Актуальність теми дослідження. Тема "Криптосистеми на основі тернарних самоортогональних квазігруп" є досить актуальною. Оскільки обчислювальна потужність зростає з кожним роком, виникає необхідність створювати надійніші системи шифрування. Квазігрупи дуже добре підходять для застосування в криптографії, завдяки своїй структурі, особливостям і великій кількості. Використання квазігруп може відкрити нові перспективи для створення надійних шифрів та методів захисту інформації. Криптологія, як наука, розвивається з величезною швидкістю, адже постійно з'являються нові криптографічні схеми та алгоритми, нові стратегії побудови, нові області застосування, нові вимоги та нові атаки. Дослідження симетричних квазігруп відкриє нові можливості в криптографії.

Із зростанням порядку, кількість квазігруп збільшується експоненційно. На даний момент вдалося знайти точну кількість квазігруп лише до 11-го порядку включно. Їх кількість становить понад 10^{33}. У випадку, коли ключ шифру алгоритму є квазігрупою порядку $n > 11$, то такий ключ неможливо знайти повним перебором. Отже, завдання шифрування за допомогою квазігрупи полягає в тому, щоб знайти метод побудови такої великої кількості квазігруп, який гарантуватиме надійність криптосистеми для вирішення певного класу завдань. У цьому дослідженні пропонується як метод побудови схрещений добуток квазігрупи на систему квазігруп. За допомогою цього методу вдалося знайти метод побудови $2*6^{33} = 955039333193568106127032 32$ квазігруп 32-го порядку. Причому, встановлено взаємно однозначну відповідність між побудованими квазігрупами та цілими невід'ємними числами з проміжку $0 \leqslant n < 2 * 6^{33}$.

Мета дослідження. *Метою дослідження* є вивчення симетричного криптоалгоритму на основі тернарної квазігрупи 32-го порядку, яка будується як безповторна композиція двох бінарних квазігруп.

Завдання дослідження. Основними *завданнями дослідження* є:

- розроблено алгоритм побудови тернарних квазігруп 32-го порядку;
- розроблено алгоритм симетричного шифрування інформації за допомогою тернарної квазігрупи 32-го порядку;
- надати оцінку алгоритму одним із існуючих методів.

Об'єктом дослідження є квазігрупи та методи їх побудови.

Предметом дослідження є схрещені добутки квазігруп, з огляду на їх застосування для побудови симетричних криптосистем.

Наукова новизна отриманих результатів. На основі розробленого науковим керівником теоретичного матеріалу, отримано такі результати: створено алгоритм побудови тернарних квазігруп 32-го порядку за допомогою схрещеного добутку квазігруп 2-го порядку на три лівосиметричні квазігрупи 16-го порядку, які в свою чергу побудовані за допомогою схрещеного добутку квазігрупи 4-го порядку на квазігрупову алгебру також 4-го порядку; створений алгоритм симетричного шифрування інформації за допомогою тернарної квазігрупи 32-го порядку. Використання схрещеного добутку в криптографії здійснюється вперше.

Практичне значення отриманих результатів. Створений алгоритм симетричного шифрування може мати практичне значення у багатьох сферах. Побудовану криптосистему можна використати для захисту конфіденційності даних у мережах, електронних платежах та інших сферах, де важлива надійність та швидкість обробки інформації. Зокрема,

цей алгоритм можна використати для захисту інформації, яка передається в пакетах мережі Інтернет.

Структура кваліфікаційної роботи. Перший розділ роботи присвячений огляду літератури, який є основою отриманих результатів.

У першому параграфі другого розділу викладені необхідні теоретичні відомості, які розроблені науковим керівником Сохацьким Ф.М. [**37**]. У другому параграфі розроблено практичне застосування побудови лівосиметричних квазігруп 16-го та 32-го порядків та детальний алгоритм їх побудови. У третьому параграфі побудовано алгоритм зашифрування та розшифрування інформації за допомогою симетричного шифрування.

Третій розділ включає в себе реалізацію алгоритмів з другого розділу.

РОЗДІЛ 1
ОГЛЯД ЛІТЕРАТУРИ

Наведемо огляд основної літератури, яка лежить в основі даної розробки.

Криптологія - це наука про таємний зв'язок, яка складається із двох взаємодоповнюючих галузей: криптографії та криптоаналізу. Криптографія займається створенням нових методів, алгоритмів та схем для зашифрування та розшифрування інформації. Протягом багатьох століть криптографічні методи використовувалися для захисту таємності в дипломатичних, політичних та військових областях. Криптоаналіз займається різними атаками на криптографічні схеми з метою отримання прихованої інформації для подальшого використання. Між цими двома галузями існує глибокий зв'язок. Криптограф, який розробляє новий алгоритм, повинен перевірити його захищеність від різних криптоаналітичних атак, якщо він хоче, щоб його алгоритм був практичним і корисним.

Квазігрупи дуже добре підходять для застосування в криптографії, завдяки своїй структурі, особливостям і великій кількості. Однією з проблем є те, яку квазігрупу вибрати для використання та яким умовам квазігрупа повинна відповідати. Криптологія швидко розвивається, оскільки постійно з'являються нові схеми та алгоритми, стратегії побудови та області застосування, а також нові вимоги та атаки. Виявлення слабких місць у стандартах та необхідність покращення безпеки призводять до впровадження нових підходів та модифікацій існуючих алгоритмів.

Криптологія як наука розвивається з величезною швидкістю, адже постійно з'являються нові криптографічні схеми та алгоритми, нові стратегії побудови, нові області застосування, нові вимоги та нові атаки. Поява нових успішних атак і виявлення слабких місць у заявлених стандартах, а також вимоги до збільшення довжини ключів і блоків викликають необхідність нових підходів у проектуванні та забезпеченні безпеки оцінки, розгортання нових елементів побудови, модифікації існуючих алгоритмів та схем тощо.

У своїй роботі [**23**], авторка представляє математичні основи, термінологію та позначення n-арних квазігруп і квазігрупових перетворень. Авторка вводить нові типи квазігрупових перетворень, які згодом будуть використані для побудови криптографічних примітивів. Авторка пропонує новий метод обчислення кількості n-арних квазігруп малого порядку, проводить аналіз таблиць співвідношень і кореляційних матриць квазігруп 4-го порядку, а також деяких їх квазігрупових перетворень над рядками довжини 2.

Цитата [**23**, ст. 73]. "Більшість відомих конструкцій криптографічних примітивів, кодів, що виявляють та виправляють помилки, використовують структури з асоціативної алгебри, такі як групи, кільця та поля. Два видатні фахівці з квазігруп Денс Ж. і Кідвелл А. Д. [**13**] свого часу проголосили настання нової ери в криптології, яка полягає в застосуванні неасоціативних алгебраїчних систем у вигляді квазігруп і неогруп. Квазігрупи та їх комбінаторний еквівалент Латинські квадрати дуже добре підходять для цієї мети завдяки своїй структурі, своїм особливостей, великої кількості, а також через те, що вони приводять до

дуже простих і але ефективних примітивів. Тим не менш, в даний час дуже мало дослідників використовують ці інструменти, а криптографічна спільнота все ще вагається щодо них.

Перше застосування квазігрупи - латинського квадрата в криптографії датується 16 століттям. Йоганн Тритемій (1462-1516) винайшов поліалфавітний шифр з прогресивним ключем, названий шифром Тритемія, який перемикає алфавіт для кожної літери в повідомленні. Це можна зобразити, наприклад, для англійського алфавіту латинським квадратом 26×26. Кожен наступний рядок - це новий алфавіт, зсунутий на одну літеру вліво від попереднього. Ще одне раннє застосування наведено в докторській дисертації Шауфлера [**27**] від 1948 року, де він звів проблему розкриття шифру Вігтнера до мінімальної кількості входжень певного латинського квадрата, які б повністю визначали квадрат. Більшість результатів застосування квазігруп у криптології до кінця вісімдесятих років 20 століття описано в [**11**], [**12**]. Деякі новіші результати та теми не розглядаються в цій роботі, наприклад, схеми обміну секретами на основі квазігруп та протоколи з нульовим знанням, генерування NLPN-послідовностей, застосування критичних множин та степеневих множин латинських квадратів та рядів латинських квадратів у криптографії. ”

У розділі 3, авторка наводить огляд основних криптографічних примітивів, таких як хеш-функції, блокові та потокові шифри, генератори псевдовипадкових чисел та алгоритми з відкритим ключем, побудовані саме на основі квазігруп та квазігрупових перетворень. У більш ранніх розробках безпека базувалася на секретних квазігрупових операціях, великій кількості квазігруп одного порядку, великій кількості ізотопів для заданого носія тощо. Новіші розробки будують свою безпеку в основному

на складності розв'язання систем квазігрупових рівнянь, але також можна знайти захист, заснований на секретному порядку елементів в квазігрупових операціях, секретних лідерах та/або порядку використаних елементарних квазігрупових перетворень, секретного порядку використаних квазігруп використаних квазігруп з деякої наперед заданої множини квазігруп, розв'язування системи багатовимірних квадратичних функцій тощо.

Ось короткий опис однієї з найпопулярніших книг для вивчення основ криптографії та мережевої безпеки: "Криптографія та мережева безпека: принципи та практика" Вільяма Сталлінгса [**33**]. Книга детально розглядає широкий спектр тем, починаючи від основних концепцій криптографії і закінчуючи сучасними викликами безпеки мережі.

Один із головних аспектів книги – розгляд симетричних та асиметричних криптоалгоритмів, включаючи DES, AES, RSA та інші. Автор докладно пояснює принципи роботи цих алгоритмів, їх стійкість та застосування у різних сценаріях. Автор розглядає різні методи передачі секретних ключів та протоколи обміну ключами. Ось деякі з найважливіших та широко використовуваних методів:

- Протокол Діффі-Геллмана (Diffie-Hellman): Цей протокол дозволяє двом сторонам взаємно обмінюватися секретним ключем через відкриті канали зв'язку. Алгоритм Діффі-Геллмана є основоположним для безпечного обміну ключами.
- Протокол Кербероса: Він використовується для автентифікації та обміну ключами в розподілених системах. Керберос дозволяє сторонам встановлювати безпечні сеанси і обмінюватися ключами без передачі їх через мережу.

- Протоколи обміну ключами з використанням сертифікатів (наприклад, SSL/TLS): У сучасних мережевих системах часто використовуються протоколи, які використовують сертифікати для підтвердження і обміну ключами, такі як протоколи шифрування HTTPS.
- Одноразові коди (One-Time Codes): Використання одноразових кодів для обміну ключами також може бути розглянуте в контексті забезпечення безпеки при передачі ключів.
- Протоколи, що базуються на хеш-функціях (наприклад, HMAC): Ці протоколи можуть використовувати хеш-функції для створення аутентифікованих інформаційних дайджестів, які використовуються в процесі обміну ключами.

Це лише кілька прикладів методів та протоколів, які можуть бути розглянуті в книзі, щоб забезпечити безпечний обмін секретними ключами у симетричних криптосистемах.

Крім того, Сталлінгс звертає увагу на практичний аспект безпеки, обговорюючи протоколи і методи захисту інформації під час передачі через мережі. Книга також враховує сучасні виклики та тренди у галузі криптографії, зокрема, в контексті Інтернету речей (IoT) та хмарових технологій.

Загальний підхід книги є зрозумілим і доступним, що робить її корисним ресурсом як для студентів, так і для фахівців у галузі кібербезпеки та криптографії. "Криптографія та мережева безпека: принципи та практика"визнана за свій внесок у розвиток області і залишається важливим джерелом знань для тих, хто цікавиться сферою інформаційної безпеки.

“Криптографія з відкритим ключем та цифрові конверти, різновидності алгоритмів, порівняння алгоритмів”, цитата з [**39**]. “Найбільшим

класом є асиметричні криптологічні системи або системи з відкритим ключем. Головною ідеєю при створенні цього класу шифрів є генерація двох ключів. Один відкритий ключ поширюється по відкритих каналах зв'язку й використовується при шифруванні повідомлень. На прийомній стороні за допомогою секретного ключа проводиться розшифрування повідомлення. Основою при створенні таких шифрів, як сказано вище, є задачі з важким розв'язком. У якості таких задач у цей час використовуються задачі факторизації, дискретного логарифмування й методи теорії завадостійкого кодування.

Симетричні та асиметричні криптоалгоритми, їх порівняння:

Асиметричні алгоритми шифрування – алгоритми шифрування, які використовують різні ключі для шифрування та розшифрування даних. Головне досягнення асиметричного шифрування в тому , що воно дозволяє людям, що не мають існуючої домовленості про безпеку, обмінюватися секретними повідомленнями. Необхідність відправникові й одержувачеві погоджувати таємний ключ по спеціальному захищеному каналі цілком відпала. Процедура шифрування обрана так, що вона необоротна навіть по відомому ключу шифрування. Тобто, знаючи ключ шифрування й зашифрований текст, неможливо відновити вихідне повідомлення – прочитати його можна тільки за допомогою другого ключа – ключа дешифрування. А раз так, то ключ шифрування для відправлення листів якій-небудь особі можна взагалі не приховувати – знаючи його однаково неможливо прочитати зашифроване повідомлення. Тому, ключ шифрування називають в асиметричних системах "відкритим ключем", а от ключ дешифрування одержувачеві повідомлень необхідно тримати в секреті – він називається "закритим ключем". Алгоритми шифрування

й дешифрування створюються так, щоб знаючи відкритий ключ, неможливо було обчислити закритий ключ.

Симетричні алгоритми шифрування – спосіб шифрування, в якому для шифрування і дешифрування застосовується один і той же криптографічний ключ. До винаходу схеми асиметричного шифрування єдиним існуючим способом було симетричне шифрування. Ключ алгоритму повинен зберігатися в секреті обома сторонами. Алгоритми шифрування і дешифрування даних широко застосовуються в комп'ютерній техніці в системах приховування конфіденційної і комерційної інформації від не коректного використання сторонніми особами. Головним принципом у них є умова, що та приймає заздалегідь знають алгоритм шифрування, а також ключ до повідомлення, без яких інформація є всього лише набір символів, що не мають сенсу. Симетричні криптоалгоритми виконують перетворення невеликого (1 біт або 32-128 біт) блоку даних в залежності від ключа таким чином, що прочитати оригінал повідомлення можна тільки знаючи цей секретний ключ.

Симетричні криптоалгоритми діляться на:

- Скремблери.
- Блокові шифри.

Скремблерами називаються програмні або апаратні реалізації алгоритму, що дозволяють шифрувати побітно безперервні потоки інформації. Сам скремблер представляє із себе набір бітів, що змінюються на кожному кроці по певному алгоритму. Після виконання кожного чергового кроку на його виході з'являється біт, що шифрує, – або 0, або 1, що накладається на поточний біт інформаційного потоку операцією XOR ("побітне виключаюче АБО"). Основним недоліком алгоритмів скремблювання є їхня нестійкість до фальсифікації.

Блокові шифри в ході своєї роботи роблять перетворення блоку вхідної інформації фіксованої довжини і одержують результуючий блок того ж обсягу, але недоступний для прочитання стороннім особам, що не володіють ключем. Таким чином, схему роботи блокового шифру можна описати функціями $Z = EnCrypt(X, Key)$ і $X = DeCrypt(Z, Key)$. Ключ Key є параметром блокового алгоритму і являє собою деякий блок двійкової інформації фіксованого розміру. Вихідний (X) і зашифрований (Z) блоки даних також мають фіксовану розрядність рівну між собою, але необов'язково рівну довжині ключа.

Порівняння асиметричних і симетричних криптоалгоритмів:

- Асиметричні криптоалгоритми:
  - Криптосистема з відкритим ключем;
    - Для шифрування повідомлення використовується відкритий ключ, а при дешифруванні – закритий. Тобто, знаючи ключ шифрування й зашифрований текст, неможливо відновити вихідне повідомлення;
    - При порушенні конфіденційності k-ої робочої станції зловмисник довідається тільки "закритий" ключ k : це дозволяє йому читати всі повідомлення, що приходять абонентові k, але не дозволяє видавати себе за нього при відправленні листів;
    - В асиметричних системах кількість існуючих ключів пов'язане з кількістю абонентів лінійно (у системі з N користувачів використовуються $2 * N$ ключів).
- Симетричні криптоалгоритми:
  - Криптосистема з секретним ключем;

* Секретний ключ використовується і для шифрування, і для дешифрування. Тобто, знаючи ключ шифрування й зашифрований текст, ви зможете дешифрувати повідомлення;
* При порушенні конфіденційності якої-небудь робочої станції зловмисник одержує доступ до всіх ключів цього користувача й може відправляти, нібито від його імені, повідомлення всім абонентам, з якими "жертва" вела переписку;
* В симетричних системах число ключів зростає квадратично із збільшенням числа користувачів."

Книга "Introduction to Cryptography with Coding Theory" [**32**] авторства Wade Trappe та Lawrence C. Washington, базується на курсі криптографії для студентів старших курсів та аспірантів, який викладається в Університеті Меріленда з 1997 року, а також на курсі, який викладається в Університеті Ратгерса з 2003 року. При розробці курсів автори керувалися наступними вимогами:

- Курси повинні бути сучасними і охоплювати широкий спектр тем з математичної точки зору.
- Матеріал повинен бути доступним для математично зрілих студентів, які мають невеликий досвід у теорії чисел та комп'ютерному програмуванні.
- Приклади мають бути достатньо великими, щоб продемонструвати, як насправді працюють алгоритми.

Цитата з [**32**, ст.1] “Люди завжди прагнули тримати інформацію подалі від інших. У дитинстві багато хто з нас мав чарівні персні-дешифратори, щоб обмінюватися закодованими повідомленнями з друзями і, можливо, зберігати секрети від батьків, братів, сестер чи вчителів. Історія сповнена прикладів, коли люди намагалися зберегти інформацію в таємниці від супротивників. Царі та генерали спілкувалися зі своїми військами, використовуючи базові криптографічні методи, щоб не дати ворогу дізнатися важливу військову інформацію. Насправді, Юлій Цезар використовував простий шифр, який був названий на його честь.

З розвитком суспільства зростала потреба у більш складних методах захисту даних. Зараз, коли настала інформаційна ера, ця потреба є більш вираженою, ніж будь-коли. Оскільки світ стає все більш пов’язаним, попит на інформацію та електронні послуги зростає, а зі збільшенням попиту зростає і залежність від електронних систем. Вже зараз обмін конфіденційною інформацією, наприклад, номерами кредитних карток, через Інтернет є звичайною практикою. Захист даних та електронних систем має вирішальне значення для нашого способу життя.

Методи, необхідні для захисту даних, належать до галузі криптографії. Насправді цей предмет має три назви: криптографія, криптологія та криптоаналіз, які часто використовуються як взаємозамінні. Технічно, однак, криптологія - це всеохоплюючий термін для вивчення комунікації по незахищених каналах і пов’язаних з цим проблем. Процес розробки систем для цього називається криптографією. Криптоаналіз займається зламом таких систем. Звичайно, неможливо займатися ні криптографією, ні криптоаналізом, не маючи хорошого розуміння методів обох областей.

Часто для опису криптографії використовують термін "теорія кодування однак це може призвести до плутанини. Теорія кодування займається представленням символів вхідної інформації вихідними символами, які називаються кодовими символами. Теорія кодування охоплює три основні застосування: стиснення, секретність і виправлення помилок. За останні кілька десятиліть термін "теорія кодування"став асоціюватися переважно з кодами, що виправляють помилки. Таким чином, теорія кодування вивчає передачу інформації по зашумлених каналах і те, як гарантувати, що що отримане повідомлення є правильним повідомленням, на відміну від криптографії, яка захищає комунікацію через незахищені канали.

Хоча коди, що виправляють помилки, є лише другорядною темою цієї книги, автори підкреслюють, що в будь-якій реальній системі коди, що виправляють помилки, “використовуються разом з шифруванням, оскільки зміни одного біта достатньо, щоб повністю знищити повідомлення в добре розробленій криптосистемі”.

Сучасна криптографія - це галузь, яка значною мірою спирається на математику, комп’ютерні науки та кмітливість. Ця книга містить вступ до математики та протоколів, необхідних для забезпечення безпеки передачі даних та електронних систем, а також до таких методів, як електронні підписи та обмін секретною інформацією.”

Цитата з [**30**, ст. 194] “Зараз теорія застосування квазігруп у криптології переживає період досить бурхливого розвитку. Тому будь-який огляд результатів у цій галузі досліджень досить швидко застаріває. Тут ми наводимо переписану та доповнену форму більш ранніх версій [**28**],[**29**] такого роду оглядів.

Практично всі результати, отримані в галузі застосування квазігруп у криптології та теорії кодування до кінця вісімдесятих років XX століття, описано в [**11**],[**12**],[**13**].

Основні факти з теорії квазігруп можна знайти в [**4**], [**3**], [**5**], [**26**], [**8**], [**28**]. Інформацію про основні факти з криптології можна знайти в багатьох книгах, наприклад, [**1**],[**7**][**24**],[**25**].

Криптологія - це наука, яка складається з двох частин: криптографії та криптоаналізу. *Криптографія* - це наука про методи перетворення (шифрування) інформації з метою захисту цієї інформації від незаконного користувача. *Криптоаналіз* - наука про методи і способи розкриття шифрів [**18**].

У певному сенсі криптографія - це "захист тобто це наука про побудову нових шифрів, а криптоаналіз - це "атака тобто це наука і свого роду "мистецтво сукупність методів зламу шифрів. Ця ситуація схожа на ситуацію з розвідкою і контррозвідкою. Ці два об'єкти (криптографія та криптоаналіз) дуже близькі і не існує хорошого криптографа, який би не знав методів криптоаналізу. Зрозуміло, що криптологія залежить від рівня розвитку суспільства, науки та рівня розвитку технологій. Нагадаємо, шифр - це спосіб (метод, алгоритм) перетворення інформації з метою її захисту. перетворення інформації з метою її захисту. Ключ - це якась прихована частина (зазвичай, невелика) або параметр шифру.

Стеганографія - це сукупність засобів і методів приховування факту факту відправлення (або передачі) інформації, наприклад, повідомлення або листа. Зараз існують методи приховування факту відправлення інформації звичайною поштою, електронною поштою тощо.

В даному огляді під теорією кодування (Code Theory) будемо розуміти науку про захист інформації від випадкових помилок, що виникають

при перетворенні та пересиланні (передачі) цієї інформації. При пересиланні важливої та конфіденційної інформації, як нам здається, є сенс використовувати методи теорії кодування, криптології та стеганографії в комплексі [**21**].

У криптології часто використовують наступне правило Керкгофа (1835 - 1903) правило: опонент (незаконний користувач) знає всю процедуру шифрування (іноді частину відкритого або зашифрованого тексту), за винятком ключа. Багато авторів книг, присвячених криптології, ділять цю науку (іноді не звертаючи уваги на цей факт) на дві частини: до статті Діффі і Хеллмана [**16**] (так звана криптологія з непублічним (симетричним) ключем) і після цієї роботи (криптологія з публічним або несиметричним ключем). Практично саме стаття Діффі та Хеллмана відкрила нову еру в криптології. Більше того, з'явилася можливість застосувати ці нові підходи на практиці.

Особливо швидкий розвиток другої частини криптології пов'язаний з дуже швидким розвитком персональних комп'ютерів і мереж персональних комп'ютерів, інших електронних технічних пристроїв наприкінці XX століття. У цьому напрямку з'явилося багато нових математичних, криптографічних проблем з'явилося багато нових математичних, криптографічних проблем, деякі з них не вирішені. Розв'язання цих проблем має велике практичне значення. цих проблем має велике значення для практики.

Майже всі відомі конструкції кодів, що виявляють та виправляють помилки, криптографічні алгоритми та системи шифрування використовують асоціативні алгебраїчні структури, такі як групи та поля, див., наприклад, [**10**],[**22**]. Існує можливість використання таких неасоціативних

структур, як квазігрупи та неополя майже у всіх розділах теорії кодування, а особливо у криптології. Часто коди та шифри, побудовані на основі неасоціативних систем, показують кращі можливості, ніж відомі коди та шифри на основі асоціативних систем [**13**],[**20**].

Зауважимо, що в останні роки інтенсивно розвивається квантова теорія коду та квантова криптологія [**9**],[**19**],[**34**],[**31**]. Квантова криптологія також використовує теоретичні досягнення "звичайної" криптології [**6**].

Ефективність застосування квазігруп у криптології ґрунтується на тому, що квазігрупи є деякими "узагальненими перестановками" і кількість квазігруп порядку n більша за $n! - (n-1)! - \ldots - 2! - 1!$ [**11**].

Варто зазначити, що деякі з ранніх професійних криптологів, зокрема, А.А. Альберт, А. Дріско, М.М. Глухов, Дж.В. Россер, Е. Шенхардт, Д.І. Мендельсон, Р. Шауфлер були пов'язані з розвитком теорії квазігруп. Основними відомими "заявникам" квазігруп у криптології були (і є) Дж. Кідвелл [**14**], [**15**], [**11**], [**12**], [**13**].

У багатьох випадках в криптографії можлива заміна асоціативних систем на неасоціативні, і практично в будь-якому випадку ця заміна дає в деякому сенсі кращі результати, ніж використання асоціативних систем. Квазігрупи, незважаючи на свою простоту, мають різноманітні застосування в криптології. Багато нових криптографічних алгоритмів можуть бути сформовані на основі квазігруп."

РОЗДІЛ 2

ПОБУДОВА СИМЕТРИЧНИХ КРИПТОАЛГОРИТМІВ

У першій частині цього розділу ми детально розглянемо теоретичні аспекти, що становлять основу для подальшого розуміння та використання у дослідженні. У другій частині розділу ми звернемося до практичної реалізації здобутих теоретичних знань. Розробимо алгоритми побудови квазігруп 16-го і 32-го порядку та проілюструємо деталі на прикладах. У третій частині побудуємо симетричний алгоритм зашифрування та розшифрування даних за допомогою тернарної квазігрупи 32-го порядку. Практична частина цього розділу служитиме не лише застосуванням теорії в реальних умовах, але й важливим кроком у напрямку розвитку нових ідей та вирішення конкретних завдань.

2.1. Теоретичні основи

Розглянемо основні поняття з алгебри та теорії квазігруп, які будемо використовувати у роботі.

2.1.1. *Операції, групоїди, моноїди, групи.* Нехай Q — довільна множина, яку називають *носієм* для операцій та перетворень, які визначені на ній. Операцію f називатимемо:

- *унарною*, або *одномісною*, або *перетворенням* носія, якщо f є відображенням носія в себе, тобто $f : Q \to Q$.

  Образ елемента $a \in Q$ познається $f(a)$ (*лівий запис* образа) або $(a)f$ (*правий запис* образа). Іноді дужки будемо опускати, якщо це не приведе до непорозуміння: fa, af. Здебільшого перетворення позначатимемо α, β, γ, φ, ...;

- *бінарною* або *двомісною*, якщо вона відображає декартів квадрат носія в носій, тобто $f : Q^2 \to Q$.

  Образ пари $(a, b) \in Q^2$ позначається $f(a, b)$ (*лівий запис* позначення *після*), $(a, b)f$ (*правий запис* позначення *перед*) або afb (*середній запис* позначення *між*). При позначенні здебільшого вживають бінарні символи $+$, $-$, $\cdot$, $*$, $\circ$, $\cap$, $\otimes$, $\odot$, ...

Нехай Q — *носій*, бінарна операція f або $*$ називається:

- *асоціативною*, якщо x, y, z із носія, то виконується рівність

$$f(f(x, y), z) = f(x, f(y, z)), \qquad (x * y) * z = x * (y * z);$$

- *комутативною*, якщо x, y із носія, то виконується рівність

$$f(x, y) = f(y, x), \qquad x * y = y * x;$$

- *ідемпотентною*, якщо для всіх x із носія, то виконується рівність $f(x, x) = x$, $x * x = x$;
- *маґмовою*, якщо в носієві вона має нейтральний елемент, тобто елемент $e \in Q$ такий, що для всіх x із носія Q маємо $f(e, x) = f(x, e) = x$, а при позначенні між — $e * x = x * e = x$.

Пару $(Q; \cdot)$, яка складається з непорожньої множини Q та бінарної операції $(\cdot)$, що визначена на носієві Q, називають *групоїдом*.

Групоїд $(Q; \cdot)$ називається:

- *маґмою*, якщо він має нейтральний елемент e, позначення $(Q; \cdot, e)$;
- *напівгрупою*, якщо операція $(\cdot)$ асоціативна.
- *моноїдом*, якщо операція $(\cdot)$ асоціативна і має нейтральний елемент.
- *групою*, якщо він є моноїдом, в якому кожний елемент оборотний.

В моноїді $(Q;\cdot,e)$ елемент a називається *оборотним*, якщо він має *лівий* обернений та *правий* обернений елементи, тобто елементи ${}^{-1}a$, a^{-1} такі, що ${}^{-1}a\cdot a=e$, $a\cdot a^{-1}=e$.

2.1.2. *Означення квазігрупи.*

Означення 1. (*Комбінаторне означення*) Групоїд $(Q;f)$ називається *квазігрупою*, якщо операція f оборотна.

Означення 2. (*Алгебричне означення*) Алгебра $(Q;f,{}^{\ell}f,{}^{r}f)$ називається *квазігрупою*, якщо операція f оборотна, а операції ${}^{\ell}f$, ${}^{r}f$ є лівим і правим оберненими до f.

Нагадаємо, що підалгеброю алгебри називається підмножина носія, яка замкнена відносно операцій сигнатури. А підквазігрупою квазігрупи називають підмножину носія, яка разом з операціями сигнатури є квазігрупою.

При комбінаторному означенні квазігрупи не кожна підалгебра є підквазігрупою. Наприклад, множина цілих чисел стосовно з операції додавання утворюють квазігрупу (навіть групу). Підмножина натуральних чисел є її підалгеброю, але не є підквазігрупою, позаяк рівняння $x+2=1$ не має розв'язку в натуральних числах.

При алгебричному означенні кожна підалгебра містить значення обернених операцій, які є розв'язками рівнянь, тому кожна підалгебра є її підквазігрупою. В цьому випадку клас всіх квазігруп визначається тотожностями. Такі класи називаються многовидами і вони мають розвинений інструментарій вивчення: прямі добутки, підалгебри, гомоморфізми. До того ж, многовид квазігруп парастрофно та ізотопно замкнений, тобто містить парастрофи та ізотопи кожної квазігрупи. Це надає додаткових можливостей для їх вивчення.

2.1.3. *Схрещений добуток.* Схрещений добуток [**2**, Білоусов, 1967] є узагальненим поняття прямого добутку групоїдів.

Нехай $(P;\cdot)$ довільний групоїд, а $(Q;\Upsilon)$ — бінарна алгебра, тобто Υ множина бінарних операцій, які визначені на носієві Q. Співставимо кожній парі елементів із P деяку операцію із Υ. Дане відображення позначимо через α, тобто $\alpha : P \times P \to \Upsilon$. Образ пари $(p;q)$ позначимо $f_{p,q}$.

Означення 3. *Схрещеним добутком* групоїда $(P;\circ)$ на бінарну алгебру $(Q;\Upsilon)$ називається групоїд $(P \times Q; \underset{\alpha}{*})$, який визначається рівністю

$$(p,a) \underset{\alpha}{*} (q,b) := (p \circ q; f_{p,q}(a,b)), \qquad \alpha : (p,q) \mapsto f_{p,q} \tag{2.1}$$

для будь-яких p і q з множини P та будь-яких a і b з множини Q, при цьому операцію $(\circ)$, назвемо початковою операцією схрещеного добутку.

2.1.4. *Комутативність.*

Теорема 2.1. *Схрещений добуток групоїда $(P;\cdot)$ на бінарну алгебру $(Q;\Upsilon)$ є комутативним тоді і тільки тоді, коли групоїд $(P;\cdot)$ є комутативним та між операціями системи Υ існує залежність $f_{p,q} = {}^{s}f_{q,p}$ для довільних p,q із P.*

2.1.5. *Права симетрія, ліва симетрія та напівсиметрія.*

Означення 4. Операція $(\cdot)$ називається *право симетричною*, якщо для всіх x,y із носія Q та виконується рівність:

$$x \cdot xy = y. \tag{2.2}$$

Критерій 1. *Операція $(\cdot)$ право симетрична, тоді і тільки тоді, коли операція $(\cdot)$ збігається з правим діленням:*

$$(\cdot) = (\overset{r}{\cdot})$$

Теорема 2.2. *Схрещений добуток групоїда* $(P;\cdot)$ *на бінарну алгебру* $(Q;\Upsilon)$ *є право симетричним тоді і тільки тоді, коли групоїд* $(P;\cdot)$ *також право симетричний та для всіх* p,q *із* P *виконується рівність* ${}^{r}f_{p,q}=f_{p,pq}$.

Означення 5. Операція $(\cdot)$ називається ліво *симетричною*, якщо x,y із носія Q та виконується тотожність:

$$xy\cdot y=x. \tag{2.3}$$

Критерій 2. *Операція* $(\cdot)$ *лівосиметрична, тоді і тільки тоді, коли вона* $(\cdot)$ *збігається з своїм лівим діленням:*

$$(\cdot)=({}^{\ell}_{\cdot}).$$

Теорема 2.3. *Схрещений добуток групоїда* $(P;\cdot)$ *на бінарну алгебру* $(Q;\Upsilon)$ *є лівосиметричним тоді і тільки тоді, коли групоїд* $(P;\cdot)$ *ліво симетричний та для всіх* p,q *із* P *виконується рівність* ${}^{\ell}f_{p,q}=f_{pq,q}$.

Напівсиметричність можна визначити тотожністю $xy\cdot x=y$ або тотожністю $x\cdot yx=y$.

Теорема 2.4. *Для довільного групоїда* $(P;\cdot)$ *і бінарної алгебри* $(Q;\Upsilon)$ *рівносильні такі умови:*

1. Схрещений добуток $(P\times Q;*)$ *напівсиметрична квазігрупа;*

2. $(P;\cdot)$ *напівсиметрична квазігрупа і* $(Q;\Upsilon)$ *квазігрупова алгебра, така що* ${}^{s\ell}f_{p,a}=f_{p,qp}$*;*

3. $(P;\cdot)$ *напівсиметрична квазігрупа і* $(Q;\Upsilon)$ *квазігрупова алгебра, така що* ${}^{sr}f_{p,q}=f_{pq,p}$.

Твердження 2.5. *Кількість відображеннь множини з* m *елементів в множину з* k *елементів, буде* k^m.

Теоретичні відомості структуровані та взяті з дипломної роботи [**35**, ст. 5].

2.1.6. *Означення тернарної квазігрупи.*

Означення 6. Пара $(Q; f)$ називається *тернарною* квазігрупою, якщо кожне з рівнянь

$$f(x, a, b) = c,$$

$$f(a, y, b) = c,$$

$$f(a, b, z) = c,$$

має єдиний розв'язок для будь-яких a, b, c.

Тернарну квазігрупу можна побудувати за допомогою безповторної композиції двох бінарних квазігрупових операцій таким чином:

$$f(x_1, x_2, x_3) = (x_{1\sigma} \circ x_{2\sigma}) * x_{3\sigma},$$

$$f(x_1, x_2, x_3) = x_{1\tau} \circ (x_{2\tau} * x_{3\tau}),$$

де $(\circ)$ та $(*)$ – бінарні квазігрупові операції, $\sigma, \tau \in S_3$ – довільна перестановка чисел $1, 2, 3$.

В даній роботі зашифровувати інформацію будемо за допомогою тернарної функції:

$$f(x, y, z) = (x \circ y) * z, \tag{2.4}$$

а розшифровувати будемо іншою тернарною функцією:

$$g(x, y, z) = (x * z) \circ y, \tag{2.5}$$

яка відрізняється від $f(x, y, z)$ порядком змінних, саме тому вона називається парастрофом $f(x, y, z)$.

2.2. Алгоритми побудови квазігруп

В даній роботі, будемо зашифровувати і розшифровувати інформацію за допомогою однієї і тієї ж лівосиметричної квазігрупи 32-го порядку, таким чином матимемо симетричний криптоалгоритм. За допомогою схрещеного добутку лівосиметричної квазігрупи 2-го порядку на систему лівосиметричних квазігруп 16-го порядку будуємо квазігрупу 32-го порядку. З огляду на це, знайдемо деяку множину Υ лівосиметричних квазігруп 4-го порядку.

2.2.1. *Побудова лівосиметричних квазігруп 4-го порядку.* Побудуємо 6 лівосиметричних квазігруп 4-го порядку за формулою:

$$g(\bar{x}, \bar{y}) = \bar{x} \oplus \bar{y} \cdot A, \tag{2.6}$$

де A одна з матриць:

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}, \tag{2.7}$$

операція $(\cdot)$ є множенням вектора на матрицю, а $(\oplus)$ є операцією групи Клейна. Результатом є такі таблиці Келі:

g_0	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

g_1	00	01	10	11
00	00	10	01	11
01	01	11	00	10
10	10	00	11	01
11	11	01	10	00

g_2	00	01	10	11
00	00	11	10	01
01	01	10	11	00
10	10	01	00	11
11	11	00	01	10

g_3	00	01	10	11
00	00	01	11	10
01	01	00	10	11
10	10	11	01	00
11	11	10	00	01

g_4	00	01	10	11
00	00	11	01	10
01	01	10	00	11
10	10	01	11	00
11	11	00	10	01

g_5	00	01	10	11
00	00	10	11	01
01	01	11	10	00
10	10	00	01	11
11	11	01	00	10

(2.8)

Отриману множину лівосиметричних операцій позначимо через Υ:

$$\Upsilon := \{g_0, g_1, g_2, g_3, g_4, g_5\}. \tag{2.9}$$

2.2.2. *Схрещений добуток.* Розглянемо алгоритм побудови квазігрупи 16-го порядку за допомогою схрещеного добутку. В означенні 3 схрещеного добутку, групоїд $(P;\circ)$, відповідно до нашого завдання, буде $(Z_2^2;g_i)$, де операція $g_i \in \Upsilon$, а Υ є множиною вищезгаданих лівосиметричних оборотних операцій. Множина Z_2^2 має такий вигляд:

$$Z_2^2 = \{00, 01, 10, 11\}.$$

Квазігрупова алгебра $(Q;\Upsilon)$ є алгеброю $(Z_2^2;\Upsilon)$.

Отже, квазігрупу 16-го порядку будуємо на множині

$$P \times Q = Z_2^2 \times Z_2^2 = Z_2^4 :$$

$$Z_2^4 = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,$$
$$1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}.$$

Загальна формула побудови лівосиметричних квазігруп 16-го порядку за допомогою схрещеного добутку з отриманих лівосиметричних квазігруп має такий вигляд:

$$(j_1j_2j_3j_4) \underset{\alpha}{*} (k_1k_2k_3k_4) = (g_i(j_1j_2, k_1k_2); \alpha(j_1j_2k_1k_2)(j_3j_4, k_3k_4)), \qquad (2.10)$$

де α відображення $Z_2^4 \to \Upsilon$, а $\alpha(j_1j_2k_1k_2)$ – деяка операція із Υ, яка є образом четвірки $j_1j_2k_1k_2$.

Щоб отримати лівосиметричну операцію, необхідно скористатись теоремою 2.3. Конкретизуємо дану теорему для лівосиметричних квазігруп.

Наслідок 2.6. *Нехай множина Υ – визначена рівністю (2.9). Схрещений добуток лівосиметричної квазігрупи $(Z_2^2;g_i)$ на бінарну квазігрупову алгебру $(Z_2^2;\Upsilon)$, є лівосиметричним тоді і тільки тоді, коли*

для всіх $j_1j_2k_1k_2$ *із* Z_2^2 *виконується рівність:*

$$\alpha(j_1j_2k_1k_2) = \alpha(g_i(j_1j_2k_1k_2), k_1k_2). \tag{2.11}$$

Доведення. Оскільки початкова операція $g_i \in \Upsilon$, то квазігрупа $(Z_2^2; g_i)$ лівосиметрична, тобто $g_i = {}^{\ell}g_i$, для всіх $i = \{0, 1, 2, 3, 4, 5\}$. □

Початкова операція g_i, за допомогою схрещеного добутку, визначає деяку множину квазігруп 16-го порядку, кожна з яких визначається відображенням α, яке задовольняє формулі (2.11), тобто будова α залежить від операції g_i. Для того, щоб підкреслити цю залежність, позначимо відображення, яке відповідає g_i, через $\alpha^{(i)}$. Тому формулу (2.10) можна уточнити так:

$$(j_1j_2j_3j_4) \underset{\alpha^{(i)}}{*} (k_1k_2k_3k_4) = (g_i(j_1j_2, k_1k_2); \alpha^{(i)}(j_1j_2k_1k_2)(j_3j_4, k_3k_4)), \tag{2.12}$$

де операція $\alpha^{(i)}(j_1j_2k_1k_2) \in \Upsilon$ є образом четвірки $j_1j_2k_1k_2$ відносно відображення $\alpha^{(i)}$.

Взагалі кажучи, $\alpha^{(i)}$ має такий вигляд:

$$\alpha^{(i)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_5} & g_{i_6} & g_{i_7} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_8} & g_{i_9} & g_{i_{10}} & g_{i_{11}} & g_{i_{12}} & g_{i_{13}} & g_{i_{14}} & g_{i_{15}} \end{pmatrix}, \tag{2.13}$$

де $g_{i_0}, \ldots, g_{i_{15}}$ — операції із Υ.

Отож, потрібно знайти всі можливі значення $\alpha^{(i)}$, для описання всіх схрещених добутків. Розглянемо $g_{i_0}, \ldots, g_{i_{15}}$ як змінні, що набувають значення в множині Υ, та знайдемо, які з них незалежні. В нашому випадку залежність виражається через властивості операції g_i. Із співвідношення

(2.11) випливає, що четвірки

$$j_1j_2k_1k_2 \quad \text{та} \quad g_i(j_1j_2k_1k_2)k_1k_2 \tag{2.14}$$

мають однаковий образ. Якщо $g_i(j_1j_2k_1k_2) = j_1j_2$, то четвірки однакові, а в інших випадках різні. Позаяк множина $\mathbb{Z}_2^2$ чотириелементна, то в чотирьох випадках четвірки однакові. Тому є 10 четвірок, в яких образи різні. Позначимо ці образи через g_{i_0}, g_{i_1}, g_{i_2}, g_{i_3}, g_{i_4}, g_{i_5}, g_{i_6}, g_{i_7}, g_{i_8}, g_{i_9}. В наступному підрозділі знайдемо будову відображення $\alpha^{(i)}$ для кожного $i = 0, 1, 2, 3, 4, 5$.

2.2.3. *Параметри побудови схрещеного добутку.* Визначимо будову відображення $\alpha^{(i)}$ для кожної початкової операції із Υ. Покажемо це на прикладі $\alpha^{(3)}$, тобто відображення, яке відповідає операції g_3. Для зручності обчислень відтворимо таблицю Келі операції g_3:

g_3	00	01	10	11
00	00	01	11	10
01	01	00	10	11
10	10	11	01	00
11	11	10	00	01

Користуючись цією таблицею, обчислимо значення $g_3(j_1j_2k_1k_2)$ для першої четвірки 0000. Маємо $g_3(0000) = 00$, тобто четвірки (2.14) збігаються. Обчислимо наступну четвірку 0001: $g_3(0001) = 01$. Отримали, що операції із Υ, які відповідають четвіркам 0001 та 0101, однакові. Таким способом обчислимо всі четвірки в яких образи збігаються та заповнимо таблицю відповідними операціями:

$j_1j_2k_1k_2$	0000	0001	0010	0011	0100	0110	0111	1000	1001	1100
$g_i(j_1j_2k_1k_2)k_1k_2$	0000	0101	1110	1011	0100	1010	1111	1000	1101	1100
$\alpha^{(3)}(j_1j_2k_1k_2)$	g_{i_0}	g_{i_1}	g_{i_2}	g_{i_3}	g_{i_4}	g_{i_5}	g_{i_6}	g_{i_7}	g_{i_8}	g_{i_9}

На основі отриманої таблиці заповнюємо матрицю для $\alpha^{(3)}$:

$$\alpha^{(3)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_1} & g_{i_5} & g_{i_6} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_8} & g_{i_5} & g_{i_3} & g_{i_9} & g_{i_8} & g_{i_2} & g_{i_6} \end{pmatrix} \tag{2.15}$$

Аналогічно обчислюємо операції $\alpha^{(0)}, \alpha^{(1)}, \alpha^{(2)}, \alpha^{(4)}, \alpha^{(5)}$. Отримаємо:

$$\alpha^{(0)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_1} & g_{i_5} & g_{i_6} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_8} & g_{i_2} & g_{i_6} & g_{i_9} & g_{i_8} & g_{i_5} & g_{i_3} \end{pmatrix} \tag{2.16}$$

$$\alpha^{(1)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_5} & g_{i_2} & g_{i_6} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_1} & g_{i_8} & g_{i_6} & g_{i_9} & g_{i_5} & g_{i_8} & g_{i_3} \end{pmatrix} \tag{2.17}$$

$$\alpha^{(2)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_5} & g_{i_6} & g_{i_3} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_5} & g_{i_2} & g_{i_8} & g_{i_9} & g_{i_1} & g_{i_6} & g_{i_8} \end{pmatrix} \tag{2.18}$$

$$\alpha^{(4)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_5} & g_{i_2} & g_{i_6} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_5} & g_{i_8} & g_{i_3} & g_{i_9} & g_{i_1} & g_{i_8} & g_{i_6} \end{pmatrix} \tag{2.19}$$

$$\alpha^{(5)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_5} & g_{i_6} & g_{i_3} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_1} & g_{i_6} & g_{i_8} & g_{i_9} & g_{i_5} & g_{i_2} & g_{i_8} \end{pmatrix} \tag{2.20}$$

Як зазначено вище, кожне $\alpha^{(i)}$ визначається 10-ма незалежними змінними

$$g_{i_0}, g_{i_1}, g_{i_2}, g_{i_3}, g_{i_4}, g_{i_5}, g_{i_6}, g_{i_7}, g_{i_8}, g_{i_9},$$

які набувають значення в множині Υ. Позаяк, кожна змінна набуває 6 значень, то кожен тип $\alpha^{(i)}$ має 6^{10} різних відображень. Оскільки, маємо 6 різних типів $\alpha^{(i)}$, то маємо $6 \cdot 6^{10} = 6^{11} = 362\,797\,056$ квазігруп 16-го порядку, які можна побудувати за допомогою схрещеного добутку з операцій із Υ.

В наступному пункті детально розглянемо загальний принцип побудови лівосиметричної квазігрупи 16-го порядку за допомогою схрещеного добутку.

2.2.4. *Алгоритм побудови квазігруп 16-го порядку.* Розробимо алгоритм для побудови лівосиметричного схрещеного добутку.

Крок 1. Довільно обираємо число $0 \leqslant n < 6^{11} = 362\,797\,056$.

Крок 2. Представимо задане число n у шістковій системі числення довжиною у 11 цифр: $ii_0i_1 \dots i_9$. Для кращого розуміння заповнимо таку таблицю, де в другому рядку запишемо отримане число n:

i	i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9

(2.21)

Крок 3. Змінна i визначає тип схрещеного добутку, тому вибираємо матрицю $\alpha^{(i)}$.

Крок 4. В матриці $\alpha^{(i)}$ замінюємо індекс i_u операції g_{i_u} на відповідне число з таблиці (2.21) та отримуємо готове відображення $\alpha^{(i)}$.

Крок 5. Побудуємо порожню матрицю 16×16 для схрещеного добутку.

$\underset{\alpha^{(5)}}{*}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000																
0001																
0010																
0011																
0100																
0101																
0110																
0111																
1000																
1001																
1010																
1011																
1100																
1101																
1110																
1111																

Крок 6. Використовуючи формулу:

$$(j_1j_2j_3j_4) \underset{\alpha^{(i)}}{*} (k_1k_2k_3k_4) = (g_i(j_1j_2, k_1k_2); \alpha^{(i)}(j_1j_2k_1k_2)(j_3j_4, k_3k_4)),$$

обчислимо кожне значення матриці.

Крок 7. Для зручного та швидкого обчислення формули, визначимо зовнішній цикл алгоритму. Для цього надамо початкове значення 0000 четвірці $j_1j_2k_1k_2$ та замінимо $\alpha^{(i)}(0000)$ на значення g_i, яке ми беремо з таблиці (2.25):

$$(00j_3j_4) \underset{\alpha^{(i)}}{*} (00k_3k_4) = (g_i(00, 00); g_i(j_3j_4, k_3k_4)), \tag{2.22}$$

Крок 7.1 Визначимо внутрішній цикл. Для цього переберемо всі можливі значення четвірки $j_3j_4k_3k_4$ від 0000 до 1111 у формулі (2.27) та заповнимо відповідні комірки таблиці.

Крок 8. Повертаємось до крок 7 та надаємо наступне значення 0001 четвірці $j_1j_2k_1k_2$ та замінимо $\alpha^{(i)}(0001)$ на значення g_i з відображення $\alpha^{(i)}$.

Крок 9. Після того, як значення четвірки $j_3j_4k_3k_4$ у внутрішньому циклі алгоритму набуде значення 1111, виконуємо крок 7 знову, надавши наступне значення 0001 четвірці $j_1j_2k_1k_2$ та виконуємо внутрішній цикл, перебираючи всі значення четвірки $j_3j_4k_3k_4$ від 0000 до 1111.

Крок 10. Повторюємо крок 7 доки значення четвірки $j_1j_2k_1k_2$ не набуде значення 1111.

Крок 11. Виводимо готову матрицю 16×16.

2.2.5. *Ілюстрація алгоритму побудови квазігруп 16-го порядку на прикладі.*

Крок 1. Довільно обираємо число $0 \leqslant n < 6^{11} = 362\,797\,056$. Для прикладу, нехай $n := 360\,000\,000$.

Крок 2. Подамо вибране число n у шістковій системі числення довжиною у 11 цифр: $ii_0i_1 \dots i_9$. В нашому випадку, $n = 55420014400_6$. Для кращого розуміння заповнимо таблицю, в якій в другому рядку запишемо отримане число n:

i	i_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9
5	5	4	2	0	0	1	4	4	0	0

(2.23)

Крок 3. Змінна i визначає тип схрещеного добутку, тому вибираємо матрицю $\alpha^{(i)}$. У нашому випадку $i = 5$, вибираємо матрицю $\alpha^{(5)}$:

$$\alpha^{(5)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_{i_0} & g_{i_1} & g_{i_2} & g_{i_3} & g_{i_4} & g_{i_5} & g_{i_6} & g_{i_3} \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_{i_7} & g_{i_1} & g_{i_6} & g_{i_8} & g_{i_9} & g_{i_5} & g_{i_2} & g_{i_8} \end{pmatrix} \tag{2.24}$$

Крок 4. В матриці $\alpha^{(5)}$ замінюємо індекс i_u операції g_{i_u}. Замість індексів i_0, i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, i_9 підставляємо число, яке відповідає

i_u у таблиці (2.23):

g_{i_0}	g_{i_1}	g_{i_2}	g_{i_3}	g_{i_4}	g_{i_5}	g_{i_6}	g_{i_3}	g_{i_7}	g_{i_1}	g_{i_6}	g_{i_8}	g_{i_9}	g_{i_5}	g_{i_2}	g_{i_8}
g_5	g_4	g_2	g_0	g_0	g_1	g_4	g_0	g_4	g_4	g_4	g_0	g_0	g_1	g_2	g_0

В результаті отримаємо відображення $\alpha^{(5)}$:

$$\alpha^{(5)} = \begin{pmatrix} 0000 & 0001 & 0010 & 0011 & 0100 & 0101 & 0110 & 0111 \\ g_5 & g_4 & g_2 & g_0 & g_0 & g_1 & g_4 & g_0 \\ 1000 & 1001 & 1010 & 1011 & 1100 & 1101 & 1110 & 1111 \\ g_4 & g_4 & g_4 & g_0 & g_0 & g_1 & g_2 & g_0 \end{pmatrix} \tag{2.25}$$

Обидва параметри схрещеного добутку, тобто g_5 та $\alpha^{(5)}$, однозначно визначені, тому формула для обчислення операції $*_{\alpha^{(5)}}$ має такий вигляд:

$$(j_1j_2j_3j_4) *_{\alpha^{(5)}} (k_1k_2k_3k_4) = (g_5(j_1j_2, k_1k_2); \alpha^{(5)}(j_1j_2k_1k_2)(j_3j_4, k_3k_4)), \tag{2.26}$$

Крок 5. Побудуємо порожню матрицю 16×16 для схрещеного добутку.

$*_{\alpha^{(5)}}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000																
0001																
0010																
0011																
0100																
0101																
0110																
0111																
1000																
1001																
1010																
1011																
1100																
1101																
1110																
1111																

Крок 6. Використовуючи формулу (2.26), обчислимо кожне значення матриці.

Крок 7. Для зручного та швидкого обчислення формули, визначимо зовнішній цикл алгоритму. Для цього надамо початкове значення 0000 четвірці $j_1j_2k_1k_2$ та замінимо $\alpha^{(5)}(0000)$ на значення g_5, яке ми беремо з таблиці (2.25):

$$(00j_3j_4) *_{\alpha^{(5)}} (00k_3k_4) = (g_5(00, 00); g_5(j_3j_4, k_3k_4)), \tag{2.27}$$

Крок 7.1 Визначимо внутрішній цикл. Для цього переберемо всі можливі значення четвірки $j_3j_4k_3k_4$ від 0000 до 1111 у формулі (2.27) та заповнимо відповідні комірки таблиці.

Крок 8. Після того, як значення четвірки $j_3j_4k_3k_4$ у внутрішньому циклі алгоритму набуде значення 1111, виконуємо крок 7 знову, надавши наступне значення 0001 четвірці $j_1j_2k_1k_2$ та виконуємо внутрішній цикл, перебираючи всі значення четвірки $j_3j_4k_3k_4$ від 0000 до 1111.

Крок 9. Повторюємо крок 7 доки значення четвірки $j_1j_2k_1k_2$ не набуде значення 1111.

Крок 10. Виводимо готову матрицю 16×16.

$*_{\alpha^{(5)}}$	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0001	0001	0000	0011	0010	0101	0100	0111	0110	1001	1000	1011	1010	1101	1100	1111	1110
0010	0010	0011	0000	0001	0110	0111	0100	0101	1010	1011	1000	1001	1110	1111	1100	1101
0011	0011	0010	0001	0000	0111	0110	0101	0100	1011	1010	1001	1000	1111	1110	1101	1100
0100	0100	0101	0110	0111	0000	0001	0010	0011	1100	1101	1110	1111	1000	1001	1010	1011
0101	0101	0100	0111	0110	0001	0000	0011	0010	1101	1100	1111	1110	1001	1000	1011	1010
0110	0110	0111	0100	0101	0010	0011	0000	0001	1110	1111	1100	1101	1010	1011	1000	1001
0111	0111	0110	0101	0100	0011	0010	0001	0000	1111	1110	1101	1100	1011	1010	1001	1000
1000	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111
1001	1001	1000	1011	1010	1101	1100	1111	1110	0001	0000	0011	0010	0101	0100	0111	0110
1010	1010	1011	1000	1001	1110	1111	1100	1101	0010	0011	0000	0001	0110	0111	0100	0101
1011	1011	1010	1001	1000	1111	1110	1101	1100	0011	0010	0001	0000	0111	0110	0101	0100
1100	1100	1101	1110	1111	1000	1010	1011	1001	0100	0101	0110	0111	0000	0001	0011	0010
1101	1101	1100	1111	1110	1001	1011	1010	1000	0101	0100	0111	0110	0001	0000	0010	0011
1110	1110	1111	1100	1101	1010	1000	1001	1011	0110	0111	0100	0101	0010	0011	0001	0000
1111	1111	1110	1101	1100	1011	1001	1000	1010	0111	0110	0101	0100	0011	0010	0000	0001

(2.28)

2.2.6. *Алгоритм побудови квазігруп 32-порядку.* Розглянемо алгоритм побудови квазігруп 32-го порядку, які є основою побудови криптоалгоритмів. Квазігрупи 32-го порядку будуємо за допомогою схрещеного добутку лівосиметричної квазігрупи 2-го порядку на систему лівосиметричних квазігруп 16-го порядку, які належать Ψ. Де Ψ – це множина квазігруп 16-го порядку, які можна побудувати за допомогою схрещеного добутку (2.2.4). Множину лівосиметричних операцій 2-го порядку

позначимо таким чином:

$$P := \{p_0, p_1\},$$

де p_0 та p_1 мають вигляд:

$$\begin{array}{c|cc} p_0 & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \end{array} \qquad \begin{array}{c|cc} p_1 & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array} \tag{2.29}$$

Формулу побудови записуємо з формули (2.1):

$$(x_1x_2x_3x_4x_5) \underset{\alpha^{(i)}}{\circ} (y_1y_2y_3y_4y_5) = (p_i(x_1y_1), \alpha^{(i)}_{x_1y_1}(x_2x_3x_4x_5, y_2y_3y_4y_5)), \tag{2.30}$$

де $p_i \in P$, $\alpha^{(i)}$ – відображення $Z_2^2 \to \Psi$, а $\alpha^{(i)}_{x_1y_1}$ – деяка операція із Υ, яка є образом пари x_1y_1, а четвірки $x_2x_3x_4x_5$, $y_2y_3y_4y_5$ належать Z_2^4. Множини Z_2^2 та Z_2^4 мають такий вигляд:

$$Z_2^2 = \{00, 01, 10, 11\}.$$

$$\begin{aligned} Z_2^4 = \{&0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, \\ &1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\}. \end{aligned}$$

Оскільки будова $\alpha^{(i)}$ залежить від вибору операції $p_i \in P$, то маємо такі $\alpha^{(0)}$ та $\alpha^{(1)}$, які будуємо аналогічно до наслідку 2.6:

$$\alpha^{(0)} = \begin{pmatrix} 00 & 01 & 10 & 11 \\ f_{i_0} & f_{i_1} & f_{i_2} & f_{i_1} \end{pmatrix} \tag{2.31}$$

$$\alpha^{(1)} = \begin{pmatrix} 00 & 01 & 10 & 11 \\ f_{i_0} & f_{i_1} & f_{i_0} & f_{i_2} \end{pmatrix} \tag{2.32}$$

де $f_{i_0}, f_{i_1}, f_{i_2}$ – функційні змінні, які набувають значення в множині Ψ. Множину, яка містить необхідні квазігрупи 16-го порядку, для побудови

квазігрупи 32-го порядку, позначимо $\Phi \in \Psi$. Множина Z_2^2 чотирьохелементна, а оскільки $\alpha^{(i)}$ має бути лівосиметричною, то множина Φ може містити не більше трьох квазігруп 16-го порядку.

Зазначимо, що деякі пари у матрицях $\alpha^{(i)}$ мають однаковий образ, відповідно до формули (2.11). Конкретизуємо дану формулу для нашого випадку:

$$\alpha(x_1 y_1) = \alpha(p_i(x_1 y_1), y_1) \tag{2.33}$$

Розглянемо детальніше $\alpha^{(1)}$. Обчислимо значення $p_i(x_1 y_1)$ для першої пари 00. Маємо $p_1(00) = 1$, згідно формулі (2.33), отримуємо $\alpha(00) = \alpha(10)$, тобто операції, що відповідають парам 00 та 10 однакові. Аналогічно обчислимо для наступної пари 01. Маємо: $p_1(01) = 0$, згідно формулі (2.33): $\alpha(01) = \alpha(01)$.

Для побудови квазігрупи 32-го порядку, нам необхідні, як випливає з формул (2.31) та (2.32), три квазігрупи 16-го порядку. Оскільки для побудови квазігруп 16-го порядку використовуємо $\alpha^{(i)}$, які визначаються 10-ма змінними, що набувають 6 значень, то кожен тип $\alpha^{(i)}$ має 6^{10} відображень. А для побудови трьох квазігруп 16-го порядку, нам необхідні 11 параметрів у шістковій системі числення для кожної, тобто 33 параметри всього. Позначимо їх:

$$\underbrace{i_0 \dots i_{10}}_{\bar{a}}, \underbrace{i_{11} \dots i_{21}}_{\bar{b}}, \underbrace{i_{22} \dots i_{32}}_{\bar{c}}.$$

Деякі послідовності із $\bar{a}, \bar{b}, \bar{c}$ можуть збігатись, а можуть бути різними. Якщо дві послідовності збігаються, то множина Φ – двоелеметна, а якщо всі послідовності однакові – множина Φ одноелементна. Якщо всі послідовності $\bar{a}, \bar{b}, \bar{c}$ різні, то множина Φ – трьохелементна.

Початкове число i може містити лише 0 або 1. Отже, існує всього $2 \cdot 6^{33}$ квазігруп 32-порядку, які ми побудуємо користуючись формулою (2.30).

Розробимо алгоритм побудови схрещеного добутку 32-го порядку.

Крок 1. Довільно обираємо число $0 \leqslant n < 2 \cdot 6^{33}$.

Крок 2. Подамо вибране число n у шістковій системі числення довжиною у 34 цифри: $i, i_0, \ldots, i_{32}$.

Крок 3. Початкове значення i може набувати значення лише $\{0, 1\}$ та вказує нам тип схрещеного добутку, тому вибираємо матрицю $\alpha^{(i)}$, відповідно до вказаного i.

Крок 4. Щоб побудувати три квазігрупи 16-го порядку, розіб'ємо послідовність $i_0, \ldots, i_{32}$, на підпослідовності по 11 цифр:

$$\underbrace{i_0, \ldots, i_{10}}_{\bar{a}}, \underbrace{i_{11}, \ldots, i_{21}}_{\bar{b}}, \underbrace{i_{22}, \ldots, i_{32}}_{\bar{c}}.$$

Кожну підпослідовність розглядаємо як параметри для побудови квазігруп 16-го порядку за алгоритмом (2.2.4). Відповідні операції позначимо через f_0, f_1, f_2.

Крок 5. В матриці $\alpha^{(i)}$ замість змінної f_{i_j} ставимо операцію f_i. В результаті отримаємо відображення $\alpha^{(i)}$.

Крок 6. Побудуємо порожню матрицю 32×32 для схрещеного добутку.

Крок 7. Використовуючи формулу (2.30), обчислимо кожне значення матриці.

Крок 8. Визначимо зовнішній цикл алгоритму. Надамо початкове значення 00 для x_1y_1. Замінимо $\alpha^{(i)}(x_1y_1)$ на значення f_i з відображення $\alpha^{(i)}$.

Крок 8.1 Визначимо внутрішній цикл. Для цього переберемо всі можливі значення $x_2x_3x_4x_5y_2y_3y_4y_5$, починаючи з 00000000,

додаючи 1 у двійковій системі числення при кожній ітерації, та заповнимо відповідні комірки таблиці.

Зовнішній цикл буде виконуватись, доки пара x_1y_1 не набуде значення 11.

Крок 9. Після того, як значення $x_2x_3x_4x_5y_2y_3y_4y_5$ у внутрішньому циклі алгоритму набуде 11111111, повторюємо знову крок 8, надаємо наступне значення 01 для x_1y_1 та виконуємо внутрішній цикл, перебираючи всі можливі значення $x_2x_3x_4x_5y_2y_3y_4y_5$ від 00000000 до 11111111.

Крок 10. Повторюємо крок 8 доки значення x_1y_1 не набуде 11.

Крок 11. Виводимо готову матрицю 32×32.

Було оцінено методом "грубої сили" можливий час на перебір всіх можливих квазігруп даного алгоритму. Найсучасніший комп'ютер Frontier, який знаходиться в США, та має потужність 1194 PFlop/s, виконує менше ніж 10^{20} операцій за секунду [**40**]. Припустимо, що для побудови однієї квазігрупи потрібно виконати одну операцію, тоді такому комп'ютеру знадобиться 130 мільйонів років для побудови всіх можливих квазігруп. Отже, навіть при ефективній оптимізації, зловмиснику знадобиться мільйони років, тому даний алгоритм неможливо взламати методом повного перебору.

2.2.7. *Ілюстрація алгоритму побудови квазігруп 32-порядку на прикладі.* Продемонструємо даний алгоритм на прикладі:

Крок 1. Довільно обираємо число $0 \leqslant n < 2 \cdot 6^{33}$. Для прикладу, нехай:

$$n = 12345678_{10}.$$

Крок 2. Подамо вибране число n у шістковій системі числення довжиною у 34 цифри:

$$n = 0000000000000000000000001120335530_6$$

Крок 3. Початкове значення $i = 0$, тому вибираємо матрицю $\alpha^{(0)}$.

Крок 4. Щоб побудувати три квазігрупи 16-го порядку, розіб'ємо послідовність $i_0, \ldots, i_{32}$, на підпослідовності по 11 цифр:

$$\underbrace{00000000000}_{\bar{a}}, \underbrace{00000000000}_{\bar{b}}, \underbrace{01120335530}_{\bar{c}}$$

Кожну підпослідовність розглядаємо як параметри для побудови квазігруп 16-го порядку за алгоритмом (2.2.4). Відповідні операції позначимо через f_0, f_1, f_2.

Крок 5. В матриці $\alpha^{(0)}$ замість змінної f_{i_j} ставимо операцію f_i. В результаті отримаємо відображення $\alpha^{(0)}$:

$$\alpha^{(0)} = \begin{pmatrix} 00 & 01 & 10 & 11 \\ f_0 & f_1 & f_2 & f_1 \end{pmatrix} \tag{2.34}$$

Оскільки перші дві підпослідовності $\bar{a}$ та $\bar{b}$ збігаються, то і квазігрупи 16-го порядку будуть однаковими. Тобто, множина Φ складається лише з двох квазігруп 16-го порядку: f_0 та f_1. Перепишемо $\alpha^{(0)}$:

$$\alpha^{(0)} = \begin{pmatrix} 00 & 01 & 10 & 11 \\ f_0 & f_0 & f_1 & f_0 \end{pmatrix} \tag{2.35}$$

Крок 6. Побудуємо порожню матрицю 32×32 для схрещеного добутку.

Крок 7. Використовуючи формулу (2.30), обчислимо кожне значення матриці.

Крок 8. Визначимо зовнішній цикл алгоритму. Надамо початкове значення 00 для x_1y_1. Замінимо $\alpha^{(0)}(00)$ на значення f_0 з відображення $\alpha^{(0)}$.

Крок 8.1 Визначимо внутрішній цикл. Для цього переберемо всі можливі значення $x_2x_3x_4x_5y_2y_3y_4y_5$, починаючи з 00000000, додаючи 1 у двійковій системі числення при кожній ітерації, та заповнимо відповідні комірки таблиці.

$$(00000) \underset{\alpha^{(0)}}{\circ} (00000) = (p_0(00), f_0(0000, 0000)) = (0, 0000) = 00000.$$

Зовнішній цикл буде виконуватись, доки пара x_1y_1 не набуде значення 11.

Крок 9. Після того, як значення $x_2x_3x_4x_5y_2y_3y_4y_5$ у внутрішньому циклі алгоритму набуде 11111111, повторюємо знову крок 8, надаємо наступне значення 01 для x_1y_1 та виконуємо внутрішній цикл, перебираючи всі можливі значення $x_2x_3x_4x_5y_2y_3y_4y_5$ від 00000000 до 11111111.

Крок 10. Повторюємо крок 8 доки значення x_1y_1 не набуде 11.

Крок 11. Виводимо готову матрицю 32×32. Результат наведений у додатку Б.

2.3. Симетричні алгоритми

2.3.1. *Алгоритми шифрування та розшифрування за допомогою тернарної квазігрупи.* Для шифрування та розшифрування інформації використаємо симетричний алгоритм на основі тернарної квазігрупи 32-го порядку, яку будуємо як безповторну композицію двох бінарних квазігруп. Використовуючи попередньо розроблений алгоритм 2.2.7, будуємо дві бінарні квазігрупи $(Z_2^4; \circ)$ та $(Z_2^4; *)$, де операції $(\circ)$ та $(*)$ будемо називати множенням, а результат їх дій – добутком. Побудовані квазігрупи можуть бути як лівосиметричними, так і правосиметричними, тому для

них можуть виконуватись такі тотожності відповідно:

$$(x \circ y) \circ y = x, \qquad x \circ (x \circ y) = y. \tag{2.36}$$

Шифрувати символи будемо за допомогою міжнародного стандарту кодування символів Unicode. Українські символи кодуються в діапазоні від U+0410 до U+0451 для великих літер і від U+0430 до U+0451 для малих літер. Наприклад, літера "А"кодується як U+0410, а "а"– як U+0430. Шифрувати будемо у двійковій системі числення, де кожен символ поданий у вигляді 16-бітного кодування. Таблиця кодування наведена в додатку А.

Ось загальний алгоритм для шифрування та розшифрування інформації за допомогою симетричного шифрування:

1. Аліса, тобто відправник, готує інформацію, яку вона буде передавати адресату (Бобу).

2. Аліса визначає пару чисел n_1 та n_2, як загальний секретний ключ, які є в межах $0 \leqslant n < 2 \cdot 6^{33}$. При цьому, за допомогою лівих верхніх індексів ℓ або r, Аліса визначає відповідно лівосиметричність чи правосиметричнсть квазігруп.

3. Аліса таємним способом передає секретний ключ Бобу.

4. Аліса використовує ключ для побудови тернарної квазігрупи 32-го порядку. Для зручності, позначимо квазігрупу n_1 операцією $(\circ)$, а квазігрупу n_2 операцією $(*)$.

5. Використовуючи побудовану тернарну квазігрупу, Аліса зашифровує інформаційний рядок за допомогою функції (2.4):

5.1 Аліса подає кожен символ у вигляді 16-бітного кодування. Отримаємо бінарну послідовність, яка складається лише з 0 та 1.

5.2 Цю бінарну послідовність розбиваємо по 5-ть бітів. Щоб інформаційний рядок був кратний 5-ти, необхідно додати нулі попереду. Отримаємо інформаційний рядок такого вигляду:

$$a_1, a_2, a_3, a_4, a_5, \ldots, a_{m-2}, a_{m-1}, a_m,$$

де кожне a_i складається з п'ятьох бітів.

5.3 Починаємо зашифровувати інформаційний рядок: два перших елементи a_1 та a_2 залишаються незмінними. Переписуємо їх у зашифрований рядок з позначенням a'_1 та a'_2.

5.4 Наступним зашифровуємо елемент a_3. Для цього знаходимо добуток:

$$a'_3 = (a_3 \circ a_1) * a_2.$$

Отримаємо:

$a_1, a_2, a_3, \ldots, a_{m-3}, a_{m-2}, a_{m-1}, a_m,$ – інформаційний рядок

a'_1, a'_2, a'_3 – зашифрований рядок

5.5 Наступним зашифровуємо елемент a_4. Для цього знаходимо добуток:

$$a'_4 = (a_4 \circ a_2) * a_3.$$

Отримаємо:

$a_1, a_2, a_3, \ldots, a_{m-3}, a_{m-2}, a_{m-1}, a_m,$ – інформаційний рядок

a'_1, a'_2, a'_3, a'_4 – зашифрований рядок

5.6 Таким чином обчислюємо кожне значення інформаційного рядка та отримуємо зашифрований рядок:

$$a'_1, a'_2, a'_3, a'_4, \ldots, a'_{m-2}, a'_{m-1}, a'_m, \tag{2.37}$$

який готовий до передавання Бобу.

6. Аліса відправляє зашифрований інформаційний рядок (2.37) Бобу через відкритий канал.

7. Використовуючи той самий ключ, Боб розшифровує зашифрований інформаційний рядок за допомогою функції (2.5):

7.1 Будує цю саму тернарну квазігрупу 32-го порядку за n_1 та n_2.

7.2 Боб починає розшифровувати рядок (2.37): перші елементи a'_1 та a'_2 зашифрованого рядка залишаються незмінними. Переписуємо їх у розшифрований рядок: a_1 та a_2.

7.3 Щоб розшифрувати наступний елемент a'_3, застосуємо операції $(\circ)$ та $(*)$ у зворотньому порядку:

$$a_3 = (a'_3 * a'_2) \circ a'_1.$$

7.4 Розшифруємо наступний елемент a'_4:

$$a_4 = (a'_4 * a'_3) \circ a'_2.$$

Отримаємо:

$a'_1, a'_2, a'_3, a'_4, \ldots a'_{m-2}, a'_{m-1}, a'_m,$ – зашифрований рядок

a_1, a_2, a_3, a_4– розшифрований рядок

7.5 Таким чином розшифровуємо кожний елемент зашифрованого рядка. Отримаємо розшифрований рядок, який складається з 0 та 1.

8. Щоб отримати оригінальний текст, Боб розділяє рядок по 16 бітів, починаючи з кінця послідовності, де кожних 16 бітів означатимуть певний символ. Залишковими нулями на початку можна знехтувати.

2.3.2. *Ілюстрація алгоритму зашифрування та розшифрування даних на прикладі.* Продемонструємо алгоритм зашифрування та розшифрування на прикладі слова "небо". Ключем буде число $n_1 = 12345678_{10}$ та $n_2 = 25916923854202103_{10}$. У пункті (2.2.7), ми побудували лівосиметричну квазігрупу 32-го порядку для числа n_1. Так само будуємо лівосиметричну квазігрупу для числа n_2.

Отож, першим кроком випишемо кодування кожної букви та запишемо у вигляді двійкової системи числення.

Буква	Код	Двійкова сис.числ.
н	$U + 043D$	0000010000111101
е	$U + 0435$	0000010000110101
б	$U + 0431$	0000010000110001
о	$U + 043E$	0000010000111110

(2.38)

Випишемо отримані значення у послідовність, яку будемо шифрувати:

0000010000111101000001000011010100000100001100010000010000111110

Оскільки маємо 64 символів, маємо додати один 0 вкінці послідовності, щоб кількість символів в рядку була кратна 5. Для зручності розіб'ємо послідовність по 5 бітів:

00000 10000 11110 10000 01000 01101 01000 00100 00110 00100 00010 00011 11100

Зашифровувати будемо за допомогою функції (2.4). Дві перші п'ятірки 00000 та 10000 залишаємо незмінними та записуємо у зашифрований рядок. Зашифровані елементи будемо позначати з штрихом, тобто $00000'$ та $10000'$. Зашифровуємо наступний елемент 11110. Для

цього знаходимо добуток:

$$(11110 \circ 00000) * 10000 = 01110'.$$

Записуємо отримане значення $01110'$ третім елементом зашифрованого рядка:

$$00000\ 10000\ 11110\ 10000\ 01000\ 01101\ 01000\ 00100\ 00110\ 00100\ 00010\ 00011\ 11100$$
$$00000', 10000', 01110'$$

Зашифровуємо наступний елемент 10000:

$$(10000 \circ 10000) * 11110 = 11110'.$$

Записуємо отримане значення $11110'$ у зашифрований рядок:

$$00000\ 10000\ 11110\ 10000\ 01000\ 01101\ 01000\ 00100\ 00110\ 00100\ 00010\ 00011\ 11100$$
$$00000', 10000', 01110', 11110'$$

Таким чином зашифровуємо кожне значення інформаційного рядка. В результаті отримаємо зашифрований рядок:

$$00000', 10000', 01110', 11110', 00110', 10101', 01101', 00001', 01010', 00110', 00000', 00101'. \tag{2.39}$$

Отриманий рядок готовий до передавання адресату.

Розшифруємо отриманий рядок за допомогою функції (2.5). Будемо використовувати ті ж самі операції $(\circ)$ та $(*)$ у зворотньому порядку. Якщо для зашифрування елемента ми використовували операцію $(\circ)$, а до отриманого результату – операцію $(*)$, то тепер для розшифрування будемо використовувати операцію $(*)$ до зашифрованого елемента, а до отриманого результату – операцію $(\circ)$. Отож, для розшифрування рядка (2.39) перші два елементи $00000'$ та $10000'$ залишаємо незмінними та записуємо у розшифрований рядок. Для розшифрування наступного

елемента $01110'$ знайдемо добуток:

$$(01110' * 10000) \circ 00000 = 11110.$$

Отримане значення 11110 записуємо у розшифрований рядок:

$00000', 10000', 01110', 11110', 00110', 10101', 01101', 00001', 01010', 00110', 00000', 00101'.$
$00000, 10000, 11110$

Розшифруємо наступне значення $11110'$:

$$(11110' * 11110) \circ 10000 = 10000,$$

та запишемо у розшифрований рядок:

$00000', 10000', 01110', 11110', 00110', 10101', 01101', 00001', 01010', 00110', 00000', 00101'.$
$00000, 10000, 11110, 10000$

Таким чином розшифровуємо кожний елемент та записуємо у розшифрований рядок:

00000 10000 11110 10000 01000 01101 01000 00100 00110 00100 00010 00011 11100

Щоб перетворити розшифрований рядок у символи, необхідно розділити його по 16 бітів. Кожних 16 бітів означатимуть певний символ. Залишковими нулями вкінці можна знехтувати.

РОЗДІЛ 3

АНАЛІЗ АЛГОРИТМУ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

В даному розділі розглянуто переваги

3.1. Загальний опис обраної мови програмування

Python – потужна та універсальна мова програмування, яка здобула широку популярність завдяки своїй простоті, ефективності та розширюваності. Її читабельний синтаксис дозволяє розробникам писати код швидше та з меншею кількістю стрічок, що сприяє підвищенню продуктивності та зручності управління проектами.

Однією з ключових переваг Python є його універсальність. Він підтримує різні парадигми програмування, такі як процедурне, об'єктно-орієнтоване та функціональне програмування, що робить його ідеальним вибором для різних типів завдань.

Python також славиться своєю великою активною спільнотою та розгалуженим екосистемою бібліотек і фреймворків. Велика кількість готових рішень полегшує розробку програм та прискорює їх впровадження. Завдяки цьому, Python використовується в різних галузях, включаючи веб-розробку, аналіз даних, машинне навчання, штучний інтелект та інші.

Однією з важливих рис Python є його переносимість. Код, написаний на Python, може працювати на різних операційних системах без необхідності внесення значних змін. Це робить мову привабливою для розробників, які хочуть створювати кросплатформенні додатки.

Завдяки своїй високій ефективності та розширюваності, Python використовується великою кількістю великих корпорацій і стартапів. Він

не тільки простий для вивчення для новачків, але й надійний для досвідчених розробників. Усі ці фактори роблять Python однією з найпопулярніших мов програмування, що визначає його значення у сучасному світі програмування.

3.2. Реалізація програми

На основі криптоалгоритму було створено консольний додаток для зашифрування та розшифрування тексту на мові програмування python. Лістинг коду у додатку Г.

На блок-схемі в додатку В наведено алгоритм зашифрування тексту. Розшифрування тексту працює так само. Розглянемо детально код програми. Було створено декілька функцій:

```
def decimal_to_senary(decimal_num):
    length=34
    if decimal_num == 0:
        return '0' * length
    senary = ''
    while decimal_num > 0:
        remainder = decimal_num % 6
        senary = str(remainder) + senary
        decimal_num = decimal_num // 6
    senary = senary.rjust(length, '0')  # Pad with leading zeros to reach the desired length
    return senary

def find_element(matrix, main_row_value, main_column_value):
    main_row = 0
    if len(main_row_value) == len(main_column_value):
        if len(main_row_value) == 1:
            main_row = [format(i, '01b') for i in range(2)]
        elif len(main_row_value) == 2:
            main_row = [format(i, '02b') for i in range(4)]
        elif len(main_row_value) == 4:
            main_row = [format(i, '04b') for i in range(16)]
        main_column = main_row
        if main_row_value in main_row and main_column_value in main_column:
            row_index = main_row.index(main_row_value)
            col_index = main_column.index(main_column_value)
            return matrix[row_index][col_index]
```

```
27          else:
28              return None
29      else:
30          print("Lenght of main row and column element is different")
```

Перша функція переводить число з десяткової в шісткову систему числнення з додаванням нуляв на початок, щоб отримати необхідну довжину числа. Друга функція - фунція для пошуку елементу в матриці, вхідними даними є матриця(список списків) та номер головного стовбця та рядка. В результаті виконання функція повертає значення в матриці заданій матриці.

```
1  def create_alpha(alpha_i, number):
2      result = []
3      number_str = str(number)[1:] # exclude first character that is 'i' from
       number
4      for index in alpha_i:
5          if 0 <= index < len(number_str):
6              result.append(int(number_str[index]))
7          else:
8              result.append(0)
9      return result
10
11 def create_matrix(h_i,alpha_new):
12     result = []
13     main_row_16 = [format(i, '04b') for i in range(16)]
14     for a in main_row_16:
15         row = []
16         for b in main_row_16:
17             j1j2 = a[:2]
18             k1k2 = b[:2]
19             j3j4 = a[2:]
20             k3k4 = b[2:]
21             res = find_element(h_i, j1j2, k1k2)
22             index = main_row_16.index(j1j2+k1k2)
23             res2 = find_element(hlists.get(alpha_new[index]),j3j4,k3k4)
24             resres = res+res2
25             row.append(resres)
26         result.append(row)
27     return result
```

`create_alpha` фунція будує альфа для заданого числа, а `creat_matrix` будує лівосиметричну квазігрупу 16 порядку.

```
hlists = {
    0: [['00','01','10','11'],
    ['01','00','11','10'],
    ['10','11','00','01'],
    ['11','10','01','00']],
    1: [['00','10','01','11'],
    ['01','11','00','10'],
    ['10','00','11','01'],
    ['11','01','10','00']],
    2: [['00','11','10','01'],
    ['01','10','11','00'],
    ['10','01','00','11'],
    ['11','00','01','10']],
    3: [['00','11','10','01'],
    ['01','10','11','00'],
    ['10','01','00','11'],
    ['11','00','01','10']],
    4: [['00','11','01','10'],
    ['01','10','00','11'],
    ['10','01','11','00'],
    ['11','00','10','01']],
    5: [['00','10','11','01'],
    ['01','11','10','00'],
    ['10','00','01','11'],
    ['11','01','00','10']]
}

alphas = {
    0: [0,1,2,3,4,1,5,6,7,8,2,6,9,8,5,3],
    1: [0,1,2,3,4,5,2,6,7,1,8,6,9,5,8,3],
    2: [0,1,2,3,4,5,6,3,7,5,2,8,9,1,6,8],
    3: [0,1,2,3,4,1,5,6,7,8,5,3,9,8,2,6],
    4: [0,1,2,3,4,5,2,6,7,5,8,3,9,1,8,6],
    5: [0,1,2,3,4,5,6,3,7,1,6,8,9,5,2,8]
}

p_lists = {
    0: [['0','1'],['1','0']],
    1: [['1','0'],['0','1']]
}

alphas_32 = {
    0: [0,1,2,1],
    1: [0,1,0,2]
```

```
}
```

Також є набір констант, які використовуються в побудові лівосиметричних квазігруп 16-го порядку.

```
def code(a,b,p_list_i,alpha_i,matrix_16_list):
    x1 = a[:1]
    x2x3x4x5 = a[1:]
    y1 = b[:1]
    y2y3y4y5 = b[1:]
    main_row_4 = [format(j, '02b') for j in range(4)]
    res = find_element(p_list_i, x1, y1) # p_i(x1y1)
    index = main_row_4.index(x1+y1)
    res2_index_of_matrix = alpha_i[index] # alpha(^i)(_x1y1)
    res2 = find_element(matrix_16_list[res2_index_of_matrix], x2x3x4x5,
    y2y3y4y5) # alpha(^i)(_x1y1)(x3x4x5x6y3y4y5y6)
    resres = str(res)+str(res2)
    return resres
```

Ця функція повертає значення для зашифрованого елементу з квазігрупи 32-го порядку. Це зроблено для того, щоб не будувати цілу квазігрупу 32-го порядку і зберігати її в пам'яті комп'ютера.

```
def encode(binary_data, n1, n2, matrix_16_list, matrix_16_list2):
    result = ""
    result += binary_data[:5*2]
    ii = int(str(n1)[:1])
    ii2 = int(str(n2)[:1])
    for i in range(0, len(binary_data) - 5 * 2, 5):
        first_three = binary_data[i:i + 5 * 3]
        variable1 = first_three[0:5]
        variable2 = first_three[5:10]
        variable3 = first_three[10:15]
        encoded_element = code(variable3, variable1, p_lists.get(ii),
    alphas_32.get(ii), matrix_16_list)
        encoded_element2 = code(encoded_element, variable2, p_lists.get(ii2
    ), alphas_32.get(ii2), matrix_16_list2)
        result += encoded_element2
    return result

def decode(binary_data, n1, n2, matrix_16_list, matrix_16_list2):
    result = ""
    result += binary_data[:5*2]
    ii = int(str(n1)[:1])
```

```
    ii2 = int(str(n2)[:1])
    for i in range(0, len(binary_data) - 5 * 2, 5):
        first_three = binary_data[i:i + 5 * 3]
        three_result = result[i:i + 5 * 3]
        variable1 = three_result[0:5]
        variable2 = three_result[5:10]
        variable3 = first_three[10:15]
        encoded_element = code(variable3, variable2, p_lists.get(ii2),
    alphas_32.get(ii2), matrix_16_list2)
        encoded_element2 = code(encoded_element, variable1, p_lists.get(ii)
    , alphas_32.get(ii), matrix_16_list)
        result += encoded_element2
    return result
```

Дві функції encode та decode, які використовують розроблений алгортм для зашифрування та розшифрування тексту.

Наступним чином виглядає довідка програми:

```
user» python3 main.py -h
usage: main.py [-h] [-r] [-e] [-d] [number1] [number2] text

CLI Tool for text encoding and decoding using quasigroups

positional arguments:
  number1        Specify the first number
  number2        Specify the second number
  text           Specify some text

options:
  -h, --help     show this help message and exit
  -r, --random   Generate random numbers
  -e, --encode   Encode the specified text
  -d, --decode   Decode the specified text
```

Для зручності користування було додано можливість випадкового вибору чисел:

```
user» python3 matrix32.py -e 12345 9081831293471431 text
00000000011101111011110101100110011111010011101111011110011101111
```

Або можна власноруч додати числа:

```
user» python3 main.py -e 29951222932379622616960739
    468337462370314740713933 48 text
0000000001110111101010111110011001111110001110110101100111011 10
```

Ось так виглядає розшифрування:

```
user» python3 matrix32.py -d 12345 9081831293471431
    0000000001110111101111010110011001111101001110111101111001110 1111
text
```

Коли введений ключ буде невірним, розшифрований текст буде не тим, який зашифрували:

```
user» python3 matrix32.py -d 468337462370314740713933 48
    644592655373410085712042 6
    0000000001110111101111010110011001111101001110111101111001110 1111
p???
```

ВИСНОВОК

В ході виконання кваліфікаційної роботи на тему “Криптосистеми на основі тернарних самоортогональних квазігруп”, виконано всі поставленні завдання, а саме:

- розроблено алгоритм побудови тернарних квазігруп 32-го порядку;
- розроблено алгоритм симетричного шифрування інформації за допомогою тернарної квазігрупи 32-го порядку;
- надано оцінку алгоритму методом “грубої сили”.

Була створена програма на основі побудованого алгоритму, яка не будує та не зберігає всі квазігрупи 32-го порядку, тому що їхній розмір великий і для цього необхідно багато пам’яті. Програма під час виконання обраховує значення певних елементів для заданих квазігруп, щоб уникнути зайвих обчислень.

Отже, створений алгоритм демонструє можливість використання тернарних квазігруп в зашифруванні та розшифруванні інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Baker H.J. and Piper F. Cipher Systems: the Protection of Communications. Northwood, London, 1982.
2. Belousov V.D., Foundations of the theory of quasigroups and loops. M.: Nauka, 222 (Russian),(1967).
3. Belousov. V.D. Elements of Quasigroup Theory: a special course. Kishinev State University Printing House, Kishinev, 1981.
4. Belousov. V.D. Foundations of the Theory of Quasigroups and Loops. Nauka, Moscow, 1967.
5. Belousov. V.D. n-Ary Quasigroups. Stiintsa, Kishinev, 1971.
6. Bennet C. H. and Brassard G. Quantum cryptography: Public key distribution and coin tossing. In Proceedings of the IEEE International Conference on Computers, Systems, and Signal Processing, page 175, Bangalore, 1984.
7. Beutelspacher A. Cryptology: An introduction to the science of encoding, concealing and hiding. Vieweg, Wiesbaden, 2002. (in German).
8. Charles F. Laywine and Gary L. Mullen. Discrete Mathematics Using Latin Squares. John Wiley and Sons, Inc., New York, 1998
9. Cyril Branciard, Nicolas Gisin, Barbara Kraus, and Valerio Scarani. Security of two quantum cryptography protocols using the same four qubit states. Phys. Rev. A, 72:032301, 2005.
10. Dehornoy P. Braid-based cryptography. Contemp. Math., Group theory, statistics, and cryptography, 360:5–33, 2004.
11. D'enes J. and Keedwell. A. D. Latin Squares and their Applications. Acad?emiai Kiad?o, Budapest, 1974.
12. D'enes J. and Keedwell. A. D. Latin Squares. New Development in the Theory and Applications, volume 46 of Annals of Discrete Mathematics. North-Holland, 1991.
13. D'enes J. and Keedwell. A. D. Some applications of nonassociative algebraic systems in cryptology. P.U.M.A., 12(2):147–195, 2002.
14. D'enes J. Latin squares and non-binary encoding. In Proc. conf. information theory, CNRS, pages 215–221, Paris, 1979.
15. D'enes J. On latin squares and a digital encrypting communication system. P.U.M.A., Pure Math. Appl., 11(4):559–563, 2000.

16. Diffie W. and Hellman M.F.. New directions in cryptography. IEEE, Transactions of Information Theory, IT-22:644–654, 1976.
17. Dimitrova V., Bakeva V., Popovska-Mitrovik A., Krapez A. Cryptographic Properties of Parastrophic Quasigroup Transformation // 2012.
18. Dorichenko S.A. and Yashchenko V.V.. 25 sketches on ciphers. Teis, Moscow, 1994.
19. Ekert A. From quantum, code-making to quantum codebreaking. In Proceedings of the symposium on geometric issues in the foundations of science, Oxford, UK, June 1996 in honour of Roger Penrose in his 65th year, pages 195–214. Oxford University Press, 1998.
20. Koscielny C. NLPN Sequences over GF(q). Quasigroups Relat. Syst., 4:89–102, 1997.
21. Koscielny C. Stegano crypto graphy with maple 8. Technical report, Institute of Control and Computation Engineering, University of Zielona Gora, 2003.
22. Magliveras S.S., Stinson D.R., and Tran van Trung. New approach to designing public key cryptosystems using one-way function and trapdoors in finite groups. J. Cryptology, 15:285–297, 2002.
23. Mileva Aleksandra. Cryptographic Primitives with Quasigroup Transformations, 2010.
24. Moldovyan N.A. Problems and methods of cryptology. S.-Petersburg University Press, S.-Petersburg, 1998.
25. Moldovyan N.A., Moldovyan A.A., and Eremeev M.E.. Cryptology. From primitives to syntez of algorithms. S.-Petersburg University Press, S.-Petersburg, 2004.
26. Pflugfelder H.O.. Quasigroups and Loops: Introduction. Heldermann Verlag, Berlin, 1990.
27. SchaufHler R.. Eine Anwendung zyklischer Permutationen und ihre Theorie. PhD thesis, Philipps-Universitat Marburg, 1948.
28. Shcherbacov V.A. Elements of quasigroup theory and some its applications in code theory, 2003.
29. Shcherbacov V.A. On some known possible applications of quasigroups in cryptology, 2003.
30. Shcherbacov V.A. Quasigroups in cryptology. 2009.

31. Shor P.W. Quantum computing. In Proc. Intern. Congress of Mathematicians, pages 467–486, Berlin, 1998.
32. Wade Trappe, Lawrence C. Washington. "Introduction to Cryptography with Coding Theory 2002.
33. William Stallings. Cryptography and Network Security: Principles and Practice, 7th Edition, ISBN 978-0-13-444428-4, published by Pearson Education © 2017
34. Zbingen H., Gisin N., Huttner B., Muller A., and Tittel W.. Practical aspects of quantum cryptographical key distributions. J. Cryptology, 13:207–220, 2000.
35. Бондар А.О., Вершигора М.О., Сохацький Ф.М. “Методи побудови квазігруп та їх застосування”, Вінниця – 2022.
36. Сохацький Ф.М. Invertible binary functions and quasigroups of the order four // XIII Міжн. алгебр. конф. в Україні: 6-9 лип. 2021 р. / Київ — КНУ ім. Т. Шевченка, 2021 — 77-78 с.
37. Сохацький Ф.М., Рукопис “Схрещені добутки квазігруп та їх застосування у криптографії”, 2023.
38. Сохацький Ф.М., Луценко А.В., Фриз І.В .Побудова квазігруп з властивістю оборотності // Мат. методи та фіз.-мех. поля. 64, 2021.
39. `https://sites.google.com/view/blog-ua/`
40. `https://www.top500.org/system/180047/`

ДОДАТОК А

КОДУВАННЯ UNICODE

Подамо українські літери у кодуванні Unicode та їх представлення у двійковій системі числення.

Буква	Код	Двійкова сис.числ.
А	$U+0410$	0000010000010000
Б	$U+0411$	0000010000010001
В	$U+0412$	0000010000010010
Г	$U+0413$	0000010000010011
Ґ	$U+0490$	0000010010010000
Д	$U+0414$	0000010000010100
Е	$U+0415$	0000010000010101
Є	$U+0404$	0000010000000100
Ж	$U+0416$	0000010000010110
З	$U+0417$	0000010000010111
И	$U+0418$	0000010000011000
І	$U+0406$	0000010000000110
Ї	$U+0407$	0000010000000111
Й	$U+0419$	0000010000011001
К	$U+041A$	0000010000011010
Л	$U+041B$	0000010000011011
М	$U+041C$	0000010000011100
Н	$U+041D$	0000010000011101
О	$U+041E$	0000010000011110
П	$U+041F$	0000010000011111
Р	$U+0420$	0000010000100000
С	$U+0421$	0000010000100001
Т	$U+0422$	0000010000100010
У	$U+0423$	0000010000100011
Ф	$U+0424$	0000010000100100
Х	$U+0425$	0000010000100101

(А.1)

Буква	Код	Двійкова сис.числ.
Ц	$U+0426$	0000010000100110
Ч	$U+0427$	0000010000100111
Ш	$U+0428$	0000010000101000
Щ	$U+0429$	0000010000101001
Ь	$U+042C$	0000010000101100
Ю	$U+042E$	0000010000101110
Я	$U+042F$	0000010000101111

(А.2)

Буква	Код	Двійкова сис.числ.
а	$U+0430$	0000010000110000
б	$U+0431$	0000010000110001
в	$U+0432$	0000010000110010
г	$U+0433$	0000010000110011
ґ	$U+0491$	0000010010010001
д	$U+0434$	0000010000110100
е	$U+0435$	0000010000110101
є	$U+0454$	0000010001010100
ж	$U+0436$	0000010000110110
з	$U+0437$	0000010000110111
и	$U+0438$	0000010000111000
і	$U+0456$	0000010001010110
ї	$U+0457$	0000010001010111
й	$U+0439$	0000010000111001
к	$U+043A$	0000010000111010
л	$U+043B$	0000010000111011
м	$U+043C$	0000010000111100
н	$U+043D$	0000010000111101
о	$U+043E$	0000010000111110
п	$U+043F$	0000010000111111
р	$U+0440$	0000010001000000
с	$U+0441$	0000010001000001
т	$U+0442$	0000010001000010
у	$U+0443$	0000010001000011
ф	$U+0444$	0000010001000100
х	$U+0445$	0000010001000101
ц	$U+0446$	0000010001000110
ч	$U+0447$	0000010001000111

(А.3)

Буква	Код	Двійкова сис.числ.
ш	$U+0448$	0000010001001000
щ	$U+0449$	0000010001001001
ь	$U+044C$	0000010001001100
ю	$U+044E$	0000010001001110
я	$U+044F$	0000010001001111

(А.4)

Представимо цифри та деякі символи у кодуванні Unicode та їх представлення у двійковій системі числення.

Цифра	Код	Двійкова сис.числ.
0	$U + 0030$	0000000000110000
1	$U + 0031$	0000000000110001
2	$U + 0032$	0000000000110010
3	$U + 0033$	0000000000110011
4	$U + 0034$	0000000000110100
5	$U + 0035$	0000000000110101
6	$U + 0036$	0000000000110110
7	$U + 0037$	0000000000110111
8	$U + 0038$	0000000000111000
9	$U + 0039$	0000000000111001

(A.5)

Символ	Код	Двійкова сис.числ.
Пробіл	$U + 0030$	0000000000110000
Крапка	$U + 0031$	0000000000110001
Кома	$U + 0032$	0000000000110010
Двокрапка	$U + 0033$	0000000000110011
Коса риска	$U + 0034$	0000000000110100
Апостроф	$U + 0035$	0000000000110101
Знак оклику	$U + 0036$	0000000000110110
Знак питання	$U + 0037$	0000000000110111

(A.6)

ДОДАТОК Б

ТАБЛИЦЯ 32-ГО ПОРЯДКУ

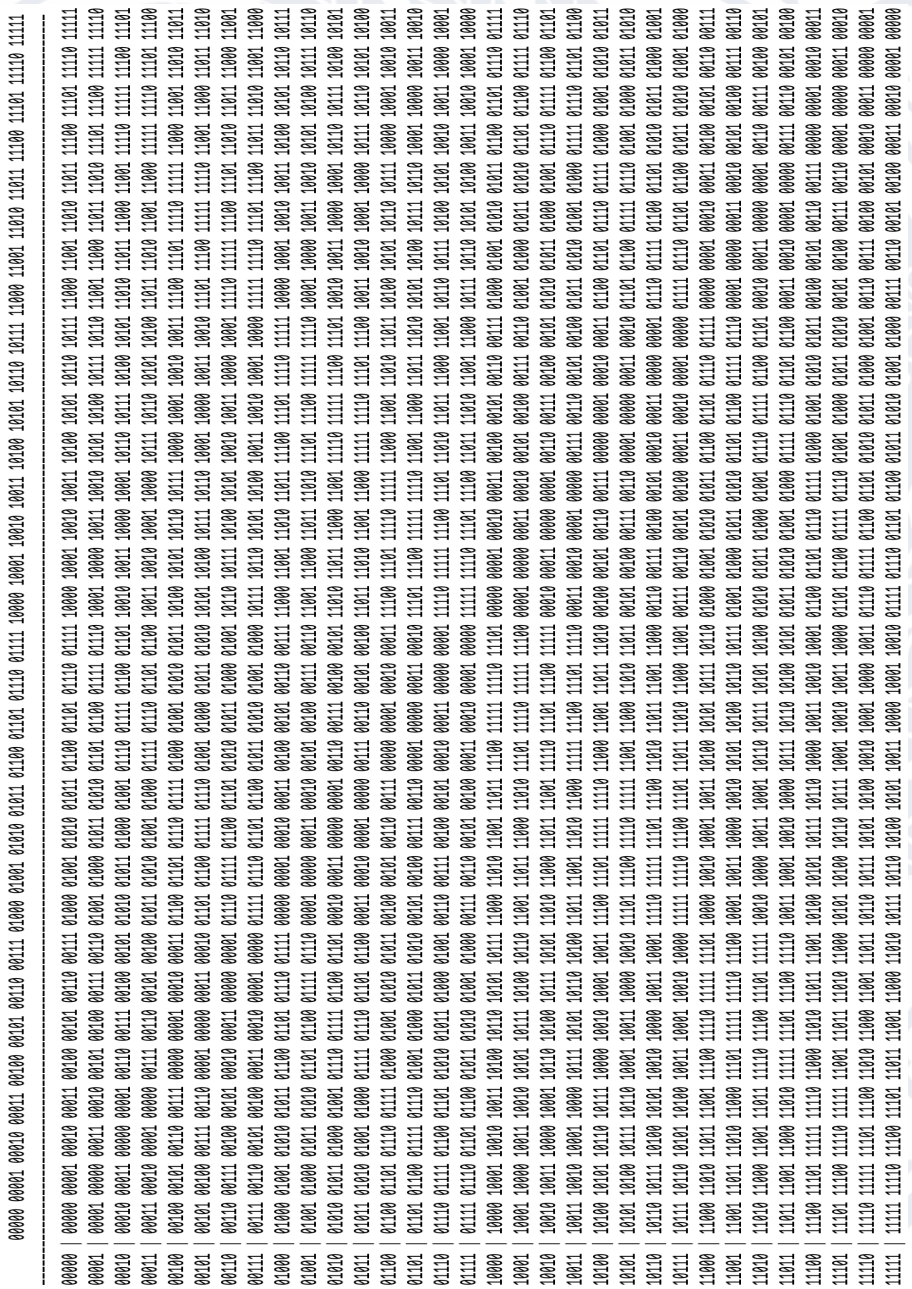

Рис. 1. Таблиця 32-го порядку для n=12345678

ДОДАТОК В

БЛОК-СХЕМА ЗАШИФРУВАННЯ

Рис. 1. Блок-схема зашифрування

ДОДАТОК Г

ЛІСТИНГ ПРОГРАМИ

```
import argparse
import random

def decimal_to_senary(decimal_num):
    length=34
    if decimal_num == 0:
        return '0' * length
    senary = ''
    while decimal_num > 0:
        remainder = decimal_num % 6
        senary = str(remainder) + senary
        decimal_num = decimal_num // 6
    senary = senary.rjust(length, '0')  # Pad with leading zeros to reach the desired length
    return senary

def find_element(matrix, main_row_value, main_column_value):
    main_row = 0
    if len(main_row_value) == len(main_column_value):
        if len(main_row_value) == 1:
            main_row = [format(i, '01b') for i in range(2)]
        elif len(main_row_value) == 2:
            main_row = [format(i, '02b') for i in range(4)]
        elif len(main_row_value) == 4:
            main_row = [format(i, '04b') for i in range(16)]
        main_column = main_row
        if main_row_value in main_row and main_column_value in main_column:
            row_index = main_row.index(main_row_value)
            col_index = main_column.index(main_column_value)
            return matrix[row_index][col_index]
        else:
            return None
    else:
        print("Lenght of main row and column element is different")

def create_matrix(h_i,alpha_new):
    result = []
    main_row_16 = [format(i, '04b') for i in range(16)]
    for a in main_row_16:
        row = []
        for b in main_row_16:
            j1j2 = a[:2]
            k1k2 = b[:2]
```

```
43              j3j4 = a[2:]
44              k3k4 = b[2:]
45              res = find_element(h_i, j1j2, k1k2)
46              index = main_row_16.index(j1j2+k1k2)
47              res2 = find_element(hlists.get(alpha_new[index]),j3j4,k3k4)
48              resres = res+res2
49              row.append(resres)
50          result.append(row)
51      return result
52
53  def create_alpha(alpha_i, number):
54      result = []
55      number_str = str(number)[1:] # exclude first character that is 'i' from
        number
56      for index in alpha_i:
57          if 0 <= index < len(number_str):
58              result.append(int(number_str[index]))
59          else:
60              result.append(0)
61      return result
62
63  def process_pair(pair):
64      alpha_new = create_alpha(alphas.get(int(pair[0])), pair)
65      matrix_res = create_matrix(hlists.get(int(pair[0])),alpha_new)
66      return matrix_res
67
68  hlists = {
69      0: [['00','01','10','11'],['01','00','11','10'],['10','11','00','01'],[
        '11','10','01','00']],
70      1: [['00','10','01','11'],['01','11','00','10'],['10','00','11','01'],[
        '11','01','10','00']],
71      2: [['00','11','10','01'],['01','10','11','00'],['10','01','00','11'],[
        '11','00','01','10']],
72      3: [['00','11','10','01'],['01','10','11','00'],['10','01','00','11'],[
        '11','00','01','10']],
73      4: [['00','11','01','10'],['01','10','00','11'],['10','01','11','00'],[
        '11','00','10','01']],
74      5: [['00','10','11','01'],['01','11','10','00'],['10','00','01','11'],[
        '11','01','00','10']]
75  }
76
77  alphas = {
78      0: [0,1,2,3,4,1,5,6,7,8,2,6,9,8,5,3],
79      1: [0,1,2,3,4,5,2,6,7,1,8,6,9,5,8,3],
80      2: [0,1,2,3,4,5,6,3,7,5,2,8,9,1,6,8],
81      3: [0,1,2,3,4,1,5,6,7,8,5,3,9,8,2,6],
82      4: [0,1,2,3,4,5,2,6,7,5,8,3,9,1,8,6],
83      5: [0,1,2,3,4,5,6,3,7,1,6,8,9,5,2,8]
```

```
}

def code(a,b,p_list_i,alpha_i,matrix_16_list):
    x1 = a[:1]
    x2x3x4x5 = a[1:]
    y1 = b[:1]
    y2y3y4y5 = b[1:]
    main_row_4 = [format(j, '02b') for j in range(4)]
    res = find_element(p_list_i, x1, y1) # p_i(x1y1)
    index = main_row_4.index(x1+y1)
    res2_index_of_matrix = alpha_i[index] # alpha(^i)(_x1y1)
    res2 = find_element(matrix_16_list[res2_index_of_matrix], x2x3x4x5,
    y2y3y4y5) # alpha(^i)(_x1y1)(x3x4x5x6y3y4y5y6)
    resres = str(res)+str(res2)
    return resres

def encode(binary_data, n1, n2, matrix_16_list, matrix_16_list2):
    result = ""
    result += binary_data[:5*2]
    ii = int(str(n1)[:1])
    ii2 = int(str(n2)[:1])
    for i in range(0, len(binary_data) - 5 * 2, 5):
        first_three = binary_data[i:i + 5 * 3]
        variable1 = first_three[0:5]
        variable2 = first_three[5:10]
        variable3 = first_three[10:15]
        encoded_element = code(variable3, variable1, p_lists.get(ii),
    alphas_32.get(ii), matrix_16_list)
        encoded_element2 = code(encoded_element, variable2, p_lists.get(ii2
    ), alphas_32.get(ii2), matrix_16_list2)
        result += encoded_element2
    return result

def decode(binary_data, n1, n2, matrix_16_list, matrix_16_list2):
    result = ""
    result += binary_data[:5*2]
    ii = int(str(n1)[:1])
    ii2 = int(str(n2)[:1])
    for i in range(0, len(binary_data) - 5 * 2, 5):
        first_three = binary_data[i:i + 5 * 3]
        three_result = result[i:i + 5 * 3]
        variable1 = three_result[0:5]
        variable2 = three_result[5:10]
        variable3 = first_three[10:15]
        encoded_element = code(variable3, variable2, p_lists.get(ii2),
    alphas_32.get(ii2), matrix_16_list2)
        encoded_element2 = code(encoded_element, variable1, p_lists.get(ii)
    , alphas_32.get(ii), matrix_16_list)
```

```
127             result += encoded_element2
128     return result
129
130 p_lists = {
131     0: [['0','1'],['1','0']],
132     1: [['1','0'],['0','1']]
133 }
134
135 alphas_32 = {
136     0: [0,1,2,1],
137     1: [0,1,0,2]
138 }
139
140 def main():
141     parser = argparse.ArgumentParser(description='CLI Tool for text
    encoding and decoding using quasigroups')
142
143     parser.add_argument('-r', '--random', action='store_true', help='
    Generate random numbers')
144     parser.add_argument('-e', '--encode', action='store_true', help='Encode
     the specified text')
145     parser.add_argument('-d', '--decode', action='store_true', help='Decode
     the specified text')
146     parser.add_argument('number1', type=int, nargs='?', default=None, help=
    'Specify the first number')
147     parser.add_argument('number2', type=int, nargs='?', default=None, help=
    'Specify the second number')
148     parser.add_argument('text', type=str, help='Specify some text')
149     args = parser.parse_args()
150
151     if args.random:
152         args.number1 = random.randint(1, 2 * 6**33)
153         args.number2 = random.randint(1, 2 * 6**33)
154         print(f"Keys are {args.number1} {args.number2}")
155     if args.number1 is None or args.number2 is None:
156         parser.error('You need to specify 2 numbers')
157     if args.number1 < 0 or args.number2 < 0:
158         parser.error('Number can not be negative')
159     if args.encode and args.decode:
160         parser.error('Choose either -e/--encode or -d/--decode, not both')
161     text = args.text
162     number1 = decimal_to_senary(args.number1)
163     number2 = decimal_to_senary(args.number2)
164     matrix_16_0 = process_pair(number1[0:11])
165     matrix_16_1 = process_pair(number1[11:22])
166     matrix_16_2 = process_pair(number1[22:33])
167     matrix_16_list = [matrix_16_0, matrix_16_1, matrix_16_2]
168     matrix_16_02 = process_pair(number2[0:11])
```

```
169     matrix_16_12 = process_pair(number2[11:22])
170     matrix_16_22 = process_pair(number2[22:33])
171     matrix_16_list2 = [matrix_16_02, matrix_16_12, matrix_16_22]
172 
173     text = ''.join(format(ord(char), '016b') for char in text)
174     zeros_needed = (5 - len(text) % 5) % 5 # Calculate the number of zeros
    needed to make the length a multiple of 5
175     text += '0' * zeros_needed # Append the necessary zeros
176 
177     if args.encode:
178         encoded = encode(text, number1, number2, matrix_16_list,
    matrix_16_list2)
179         print(encoded)
180     elif args.decode:
181         decoded = decode(args.text, number1, number2, matrix_16_list,
    matrix_16_list2)
182         binary_chunks = [decoded[i:i+16] for i in range(0, len(decoded),
    16)]
183         result_text = ''.join(chr(int(chunk, 2)) for chunk in binary_chunks
    )
184         print(result_text)
185 
186 if __name__ == '__main__':
187     main()
```

main.py