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

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

Кваліфікаційна наукова
праця на правах рукопису

СЕВЕРІН АНДРІЙ ІВАНОВИЧ

УДК 004.056

ДИСЕРТАЦІЯ

АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ ПРИВАТНИХ НАБОРІВ ДАНИХ У ЗАДАЧАХ КЛАСИФІКАЦІЇ

121 Інженерія програмного забезпечення
12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ А. І. Северін
(підпис)

Науковий керівник: Онай Микола Володимирович, кандидат технічних наук, доцент

Київ – 2024

АНОТАЦІЯ

Северін А. І. Алгоритмічне та програмне забезпечення захисту приватних наборів даних у задачах класифікації. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії в галузі знань 12 Інформаційні технології за спеціальністю 121 Інженерія програмного забезпечення. – Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2024.

Впровадження систем аналізу даних і штучного інтелекту набуває все більшого поширення у різних аспектах людського життя. Окрім вже звичних випадків застосування таких систем у електронній комерції (наприклад, підбір рекомендацій користувачеві) та соціальній сфері (виявлення спаму, модерування коментарів), такі інструменти стрімко поширюються й для персонального використання (наприклад, чатботи ChatGPT, Google Bard, Microsoft Copilot, хоча вони з’явились лише впродовж останніх двох років).

В основі систем, що використовують методи машинного навчання, лежать дані. Вони є необхідним елементом як для навчання систем аналізу даних і штучного інтелекту, так і для їх тестування. Чим більше різнопланових даних, аналізується, тим точнішою є побудована програмна система. Найчастіше джерелом даних для програмних рішень з використанням машинного навчання є реальний світ. Іноді дані генерують програмним шляхом, намагаючись відтворити певні характеристики даних. Проте, незважаючи на те, що кількість створюваних та оброблюваних даних стрімко зростає, дані досить часто містять щонайменше частину приватної інформації, що обмежує їх використання для систем аналізу даних і штучного інтелекту.

Приватні дані – інформація, яка є конфіденційною, чутливою або секретною. Прикладами секретних даних є військові, фінансові та державні

дані. Конфіденційні дані – дані, що дозволяють ідентифікувати людину або компанію, їх прикладами є серія та номер паспорту, реєстраційний податковий номер та номер автомобіля. Прикладами чутливих даних є дані, що містять медичні діагнози пацієнтів. Збереження приватності даних є вкрай важливим, адже втрата приватності може призвести до дуже негативних наслідків (передусім різноманітних злочинів та недобросовісної конкуренції).

Таким чином, вище описані задачі визначають актуальну науково-технічну задачу вдосконалення алгоритмічного та програмного забезпечення захисту приватних наборів даних у системах з використанням штучного інтелекту, яка вирішується у даній дисертаційній роботі для задачі класифікації.

Метою дисертаційної роботи є удосконалення процесу оброблення приватних наборів даних для програмних систем інтелектуального аналізу даних.

У першому розділі дисертаційної роботи розглянуто основні етичні аспекти використання систем штучного інтелекту та проблеми до яких може призвести їх ігнорування. Проаналізовано загрози приватності у таких системах, зокрема атаки інверсії, отруєння та логічного висновку. Проведено комплексний порівняльний аналіз методів збереження приватності в машинному навчанні (методи генерації синтетичних даних, анонімізації даних, диференційної приватності, гомоморфного шифрування та федеративного навчання), що дозволило виявити основні проблеми існуючих методів, які потребують досліджень. Розроблено вимоги до програмного забезпечення захисту приватних наборів даних у задачах класифікації.

У другому розділі розроблено алгоритмічні методи міжбазисних перетворень елементів скінченних полів. Проаналізовано особливості використання полів Галуа в гомоморфних методах збереження приватності, а також визначено залежність часу виконання операцій над елементами

скінченних полів від базису (поліноміального чи нормального), в якому представлені елементи. Запропоновано метод пошуку поліномів, який відрізняється від існуючого використанням простих чисел у десятковому представленні замість поліномів й дозволяє зменшити обчислювальну складність процесу пошуку нормальних многочленів. Розроблено модифікований спосіб для переходу між базисами, який полягає у використанні рекурентної формули, що дозволяє зменшити як кількість пам'яті, що використовується, так і обчислювальну складність.

У третьому розділі розроблено алгоритмічно-програмний метод захисту приватних наборів даних. Проаналізовано математичне підґрунтя для побудови алгоритмічно-програмних методів з використанням нейронних мереж. Запропоновано метод функціонального шифрування даних, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах аналізу даних та штучного інтелекту шляхом зменшення їх розмірності й функціонального шифрування отриманих даних з використанням приватного ключа. Запропоновано модифікацію моделі шифрування даних, яка полягає у використанні двовимірних згорткових нейронних мереж і дозволяє застосовувати модель шифрування даних, що представлені набором пікселів, з яких складається зображення. Проаналізовано метрики для оцінки методів захисту наборів даних.

Четвертий розділ присвячено розробленню програмного забезпечення реалізації запропонованих методів для захисту приватних наборів даних та проведенню експериментальних досліджень. Запропоновано архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних. Розроблено програмну систему, яка дозволяє виконувати обчислення над елементами поля $GF(p^m)$, проводити експериментальні дослідження, використовуючи поліноміальне й нормальне представлення елементів поля $GF(p^m)$, задавати різні значення вхідних параметрів p та m, а також генерувати різні набори тестових даних

залежно від нормальних поліномів поля Галуа. Проведено експериментальні дослідження запропонованих методів міжбазисних перетворень скінченних полів. Розроблено програмну систему вирішення задачі класифікації на приватних наборах даних, що реалізує метод функціонального шифрування для захисту приватних наборів даних й дозволяє вирішувати задачу класифікації, використовуючи як оригінальні дані, так і зашифровані. Проведено експериментальні дослідження запропонованого методу функціонального шифрування. Проаналізовано шляхи інтеграції розроблених програмних систем.

У дисертаційній роботі отримано низку **нових наукових результатів**, зокрема **уперше** запропоновано архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних, характерною особливістю якої є захист приватних наборів даних, шляхом функціонального шифрування, що відбувається на стороні клієнта, і дозволяє збільшити кількість наборів даних для навчання загальнодоступних систем аналізу даних і штучного інтелекту.

Уперше запропоновано модифікацію програмної моделі шифрування даних, яка відрізняється від існуючої використанням двовимірних згорткових нейронних мереж, замість одновимірних, і дозволяє застосовувати модель шифрування з використанням нейронних мереж до даних, що представлені набором пікселів, з яких складається зображення.

Уперше розроблено алгоритмічно-програмний метод функціонального шифрування наборів даних, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах аналізу даних та штучного інтелекту шляхом зменшення їх розмірності й функціонального шифрування отриманих даних з використанням приватного ключа.

Уперше розроблено алгоритмічно-програмний метод пошуку нормальних поліномів серед незвідних, який відрізняється від існуючого використанням простих чисел у десятковому представленні замість

поліномів, що дозволяє зменшити обчислювальні витрати алгоритму пошуку незвідних многочленів з $O(n^3)$ до $O(n \log(\log n))$ і, як наслідок, спростити міжбазисні перетворення у бінарних скінченних полях з метою пришвидшення виконання операцій над елементами поля у методах гомоморфного шифрування даних.

Уперше розроблено модифікований спосіб побудови матриці переходу між поліноміальним та нормальним базисами скінченного поля, який полягає у використанні рекурентної формули $\alpha_{i+1} = t^{p^{i+1}} = t^{p^i \cdot p} = (\alpha_i)^p$ замість обчислення остачі від ділення елемента $t^{p^{i+1}}$ на незвідний поліном, що дозволяє зменшити кількість використовуваної пам'яті з n^{p^i} до $n \cdot p$, а також обчислювальну складність з $O\left(m^{p^i}\right)$ до $O\left(m^p\right)$.

Основні наукові результати дисертаційної роботи **опубліковано** у 7 наукових працях, зокрема у 4 наукових статтях, включаючи 1 статтю опубліковану у закордонному фаховому виданнях третього квартиля (Q3), яке проіндексоване в базі даних Scopus, 1 статтю опубліковану у виданні, яке проіндексоване в базі даних Web of science, і 2 статті опубліковані у фаховому виданні, включеному до переліку наукових фахових видань України з присвоєнням категорії «Б» та у 3 матеріалах науково-технічних конференцій.

Ключові слова: прикладне програмне забезпечення, архітектура програмного забезпечення, інтелектуальна система, машинне навчання, збереження конфіденційності, збір і аналіз даних, гомоморфне шифрування, скінченне поле, завадостійкі коди, функціональне шифрування, нейронна мережа, згорткова нейронна мережа, генеративна конкуруюча мережа, ущільнення даних, задача класифікації.

SUMMARY

Severin A. Algorithms and software for protecting private datasets in classification tasks. – Qualifying scientific work, manuscript.

PhD thesis in the field of knowledge 12 Information technologies in specialty 121 Software Engineering. – National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2024.

The implementation of data analysis and artificial intelligence systems is becoming increasingly widespread in various aspects of human life. In addition to the usual cases of application of such systems in e-commerce (for example, selecting recommendations for the user) and the social sphere (spam detection, comment moderation), such tools are also rapidly spreading for personal use (for example, chatbots ChatGPT, Google Bard, Microsoft Copilot, although they appeared only during the last two years).

Systems using machine learning methods are based on data. They are a necessary element both for training data analysis and artificial intelligence systems, and for their testing. The more diverse data is analyzed, the more accurate the built software system is. Most often, the data source for software solutions using machine learning is the real world. Sometimes the data is generated by software, trying to reproduce certain characteristics of the data. However, even though the amount of data being created and processed is growing rapidly, the data quite often contains at least some private information, which limits its use for data analysis and artificial intelligence systems.

Private data – information that is confidential, sensitive or secret. Examples of secret data are military, financial, and government data. Confidential data – data that allows the identification of a person or company, examples of which are passport series and number, tax registration number and car number. Examples of sensitive data are data containing medical diagnoses of patients. The privacy-preserving of data is extremely important because the loss of privacy can lead to very negative consequences (primarily various crimes and unfair competition).

Thus, the presence of these problems determines the actual scientific and technical problem of improving algorithmic and software protection of private datasets in systems using artificial intelligence, which is solved in this dissertation for the classification problem.

The purpose of the dissertation is to improve the process of processing private datasets for software systems of intelligent data analysis.

In the first section of the thesis, the main ethical aspects of using artificial intelligence systems and the problems that ignoring them can lead to were examined. Threats to privacy in such systems are analyzed, including inversion, poisoning, and logical inference attacks. A comprehensive comparative analysis of privacy preservation methods in machine learning (methods of synthetic data generation, data anonymization, differential privacy, homomorphic encryption, and federated learning) was conducted, which revealed the main problems of existing methods that require research. Software requirements for private datasets protection in classification tasks were developed.

In the second section, the algorithmic methods for change-of-basis conversion of finite field elements were developed. The peculiarities of using Galois fields in homomorphic privacy-preserving methods were analyzed, and the dependence of the execution time of operations on the elements of finite fields on the basis (polynomial or normal) in which the elements are represented was determined. A method for finding polynomials was proposed, which differs from the existing one by using simple numbers in decimal representation instead of polynomials and allows to reduce the computational complexity of finding normal polynomials process. A modified way for conversion between bases was developed, which is based on the use of a recurrent formula, which allows to reduce both the amount of memory used and the computational complexity.

In the third section, an algorithmic and software method for protecting private datasets were developed. The mathematical basis for building such methods using neural networks was analyzed. A method of functional data encryption was proposed, the feature of which is the possibility of using private

datasets in publicly available data analysis and artificial intelligence systems by reducing their size and functionally encrypting the received data using a private key. A modification of the data encryption model was proposed, which is based on the use of two-dimensional convolutional neural networks and allows the encryption model to be applied to data represented by a set of pixels that make up an image. Metrics for evaluating dataset protection methods were analyzed.

The fourth section is devoted to software development for the implementation of the proposed methods for the protection of private datasets and conducting experimental studies. The architecture of the software system for solving the classification problem based on private data was proposed. A software system was developed that allows to perform calculations on the field elements of $GF(p^m)$, to conduct experimental studies using the polynomial and normal representation of the field elements of $GF(p^m)$, to set different values of the input parameters p and m, and also to generate different sets of test data depending on the normal polynomials of the Galois field. Experimental studies of the proposed methods of change-of-basis conversion of finite fields were carried out. A software system for solving the classification problem on private datasets was developed that implements the method of functional encryption to protect private data sets and allows solving the problem of classification using both original and encrypted data. Experimental studies of the proposed method of functional encryption were carried out. Ways of integration of the developed software systems were analyzed.

A number of **new scientific results** were obtained in the dissertation, in particular, **for the first time** the architecture of the software system for solving the classification problem based on private data is proposed, the characteristic feature of which is the protection of private datasets by means of functional encryption that occurs on the client side, and allows to increase the number of datasets for training publicly available data analysis and artificial intelligence systems.

For the first time, a modification of the software data encryption model is proposed, which differs from the existing one by using two-dimensional convolutional neural networks instead of one-dimensional ones and allows applying the encryption model using neural networks to data represented by a set of pixels that make up an image.

For the first time, an algorithmic software method of functional encryption of datasets was developed, the feature of which is the possibility of using private datasets in publicly available data analysis and artificial intelligence systems by reducing their size and functionally encrypting the received data using a private key.

For the first time, an algorithmic software method for finding normal polynomials among irreducible polynomials was developed, which differs from the existing one by using simple numbers in decimal representation instead of polynomials, which allows reducing the computational cost of the algorithm for finding for irreducible polynomials from $O(n^3)$ to $O(n \log(\log n))$ and, as a result, to simplify change-of-basis conversions in binary finite fields in order to speed up operations on field elements in homomorphic data encryption methods.

For the first time, a modified way of constructing the conversion matrix between polynomial and normal bases of a finite field was developed, which is based on using a recurrent formula $\alpha_{i+1} = t^{p^{i+1}} = t^{p^i \cdot p} = (\alpha_i)^p$ instead of calculating the remainder from dividing an element $t^{p^{i+1}}$ by an irreducible polynomial, which allows reducing the amount of memory used from n^{p^i} to $n \cdot p$, as well as the computational complexity from $O\left(m^{p^i}\right)$ to $O\left(m^p\right)$.

The main scientific results of the dissertation **were published** in 7 scientific papers, in particular in 3 scientific articles, including 1 article published in a foreign scientific journal of the third quartile (Q3), which is indexed in the Scopus database, 1 article published in a scientific journal that is indexed in the Web of science database and 2 articles published in scientific journals included in the list

of scientific journals of Ukraine in category "Б", as well as in 3 materials of scientific and technical conferences.

Keywords: application software, software architecture, intelligent system, machine learning, privacy-preserving, data collection and analysis, homomorphic encryption, finite field, error control codes, functional encryption, neural network, convolutional neural network, generative adversarial network, data compression, classification problem.

Список публікацій здобувача / List of publications of the applicant:

Стаття у фаховому виданні, яке проіндексоване в базі даних Web of science / the article published in a scientific journal, which is indexed in the Web of science database:

1. Modified Change-of-Basis Conversion Method in $GF(2^m)$ / I. A. Dychka, V. P. Legeza, M. V. Onai, A. I. Severin. // Radio Electronics, Computer Science, Control. — 2020. — №2. — С. 117–128 — DOI: 10.15588/1607-3274-2020-2-12.

Стаття у закордонному фаховому виданні третього квартиля (Q3), яке проіндексоване в базі даних Scopus / the article published in a foreign scientific journal of the third quartile (Q3), which is indexed in the Scopus database:

2. Method of Performing Operations on the Elements of $GF(2^m)$ Using a Sparse Table / I. Dychka, M. Onai, A. Severin, C. Hu. // International Journal of Computer Network and Information Security (IJCNIS). — 2024. — Vol. 16, №1. — pp. 61-72 — DOI: 10.5815/ijcnis.2024.01.05.

Статті у фахових виданнях, включених до переліку наукових фахових видань України з присвоєнням категорії «Б» / the articles published in scientific journals included in the list of scientific journals of Ukraine in category «Б»:

3. Северін А.І. Методи збереження приватності в машинному навчанні. / М.В. Онай, А.І. Северін // Вісник Хмельницького національного університету Серія: «Технічні науки». — 2023. — №6. — С. 274-280 — DOI: 10.31891/2307-5732-2023-329-6-274-280.
4. A. Severin. Architecture of a software system for solving the classification problem based on private data. / M. Onai, A. Severin // Herald of Khmelnytskyi national university. Technical Sciences. — 2024. — №1. — pp. 244-247 — DOI: 10.31891/2307-5732-2024-331-36.

Публікація у матеріалах наукової конференції / the publication in the proceedings of the scientific conference:

5. Северін А.І. Метод захисту набору даних зображень для вирішення задачі класифікації. / М.В. Онай, А.І. Северін // Прикладна математика та комп'ютинг. ПМК-2020 : тринадцята наук. конф. магістрантів та аспірантів, Київ, 18-20 листопада 2020 р. : зб. тез доп. / [ред кол.: Дичка І.А. та ін.] . — К. : Просвіта, 2020. — С. 221-226.
6. Северін А.І. Комплексний порівняльний аналіз методів збереження приватності в машинному навчанні. / М.В. Онай, А.І. Северін // Актуальні задачі сучасних технологій : зб. тез доповідей XII міжнар. наук.-практ. конф. Молодих учених та студентів, (Тернопіль, 6-7 грудня 2023) / М-во освіти і науки України, Терн. націон. техн. ун-т ім. І. Пулюя [та ін.]. — Тернопіль: ФОП Паляниця В. А., 2023. — С. 406-407.
7. Северін А.І. Модифікований підхід для побудови матриці міжбазисних перетвореннь у $GF(p^m)$. / М.В. Онай, А.І. Северін // Матеріали XI науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 13-14 грудня 2023 р.). — Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2023 — С. 110.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Алгебраїчна структура (система) – непорожня множина з визначеним на ній набором операцій та відношень, що задовольняють певній системі аксіом.

Бінарне поле Галуа – це скінченне поле для якого параметр $p = 2$, його позначення – $GF(2^m)$.

Гомоморфне шифрування – це форма шифрування даних, при якій виконання обчислень на зашифрованих даних даватимуть результати, розшифрована версія яких є ідентичною результатам виконання обчислень на початкових даних. Згідно з архітектурою, одна мережа генерує нові дані, використовуючи генеративні методики, а інша їх оцінює їх справжність, використовуючи дискримінаційні алгоритми.

Замкнена множина – множина, в якій результатом виконання над елементами множини, визначених на ній операцій, є елемент цієї ж множини.

Згорткові нейронні мережі (CNNs – convolutional neural networks) – клас глибоких нейронних мереж, що складаються хоча б з одного згорткового шару й реалізують метод прямого поширення інформації.

Конкуруючі нейронні мережі (GANs – Generative adversarial networks) – архітектура глибокого навчання без вчителя, основна ідея якої полягає в тому, щоб поставити одну нейронну мережу проти іншої у грі з нульовою сумою.

Метрика – критерій, що використовується для кількісної оцінки властивостей програмного забезпечення.

Набір даних для затвердження (validation set) – підмножина даних, яка використовується на етапі навчання й дозволяє об’єктивно оцінити навчену модель в певний момент часу. Ця підмножина дозволяє регулювати гіперпараметри моделі під час навчання; однак, не використовується для

навчання моделі, її використовують для перевірки моделі в кінці кожної епохи навчання.

Навчальний набір даних – підмножина даних, на якій здійснюється навчання моделі.

Незвідний многочлен – поліном, який не дорівнює константі та який не можна розкласти на множники у заданому полі.

Нормальний поліном – незвідний поліном, за допомогою якого можна побудувати матрицю переходу між поліноміальним і нормальним базисах.

Нотація Ландау (O) – математична нотація для формального запису асимптотичної поведінки функцій, яку використовують для оцінки обчислювальної складності алгоритмів.

Операція Фробеніуса – операція піднесення до степеня p^k, де p – характеристика поля, а k – будь-яке натуральне число.

ПЗ – програмне забезпечення.

Поле – це множина, у якій визначено дві бінарні операції (наприклад, додавання та множення) та обернені їм (віднімання та ділення, крім ділення на нульовий елемент), а також операції задовольняють законам асоціативності, комутативності й дистрибутивності.

Поле Галуа – це поле, яке складається зі скінченної кількості елементів, позначається $GF(p^m)$.

Приватні дані – інформація, яка є конфіденційною, чутливою або секретною.

Тестовий набір даних – підмножина даних, на якій відбувається тестування навченої моделі. Процес тестування відбувається, після проходження моделлю первинної перевірки набором для затвердження (validation set).

Точність моделі до певного значення (Accuracy) для задачі класифікації – відношення кількості випадків правильної класифікації екземплярів даних ($N_{correct}$) до загальної кількості екземплярів даних (N_{total}).

Федеративне навчання – архітектурний підхід для навчання систем штучного інтелекту, ідея якого полягає в навчанні моделі на розподілених локальних пристроях (наприклад, телефонах, планшетах, персональних комп’ютерах чи серверах), при якому немає необхідності збору даних на одному централізованому сервері.

Функціональне шифрування – це тип шифрування, який забезпечує більш точний контроль над доступом до зашифрованих даних порівняно з традиційними методами шифрування. Його метою є вибіркове розкриття певної інформації або функції, що містяться в зашифрованих даних, авторизованим сторонам, зберігаючи при цьому конфіденційність решти даних.

Шаблони проєктування – архітектурні конструкції, які часто використовуються та призначені для ефективного вирішення типових задач, що виникають в процесі розроблення архітектури програмного забезпечення.

ВСТУП

Актуальність теми. Впровадження систем аналізу даних і штучного інтелекту набуває все більшого поширення у різних аспектах людського життя. Окрім вже звичних випадків застосування таких систем у електронній комерції (наприклад, підбір рекомендацій користувачеві) та соціальній сфері (виявлення спаму, модерування коментарів), такі інструменти стрімко поширюються й для персонального використання (наприклад, чатботи ChatGPT, Google Bard, Microsoft Copilot, хоча вони з’явились лише впродовж останніх двох років).

В основі систем, що використовують методи машинного навчання, лежать дані. Вони є необхідним елементом як для навчання систем аналізу даних і штучного інтелекту, так і для їх тестування. Чим більше різнопланових даних, аналізується, тим точнішою є побудована програмна система. Найчастіше джерелом даних для програмних рішень з використанням машинного навчання є реальний світ. Іноді дані генерують програмним шляхом, намагаючись відтворити певні характеристики даних. Проте, незважаючи на те, що кількість створюваних та оброблюваних даних стрімко зростає, дані досить часто містять щонайменше частину приватної інформації, що обмежує їх використання для систем аналізу даних і штучного інтелекту.

Приватні дані – інформація, яка є конфіденційною, чутливою або секретною. Прикладами секретних даних є військові, фінансові та державні дані. Конфіденційні дані – дані, що дозволяють ідентифікувати людину або компанію, їх прикладами є серія та номер паспорту, реєстраційний податковий номер та номер автомобіля. Прикладами чутливих даних є дані, що містять медичні діагнози пацієнтів. Збереження приватності даних є вкрай важливим, адже втрата приватності може призвести до дуже негативних наслідків (передусім різноманітних злочинів та недобросовісної конкуренції).

Таким чином, вище описані задачі визначають актуальну науково-технічну задачу вдосконалення алгоритмічного та програмного забезпечення захисту приватних наборів даних у системах з використанням штучного інтелекту, яка вирішується у даній дисертаційній роботі для задачі класифікації.

Зв'язок роботи з науковими програмами, планами, темами. Дослідження за темою дисертаційної роботи провадились у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» в рамках виконання ініціативної науково-дослідної роботи «Розроблення та дослідження засобів зберігання секретності приватних наборів даних при побудові систем аналізу даних і штучного інтелекту» (номер державної реєстрації 0121U109925).

Мета і задачі дослідження. Метою дисертаційної роботи є удосконалення процесу оброблення приватних наборів даних для програмних систем інтелектуального аналізу даних.

Відповідно до поставленої мети основними задачами дослідження є:

- аналіз загроз приватності даних у системах аналізу даних та штучного інтелекту;
- аналіз алгоритмічно-програмних методів збереження приватності в машинному навчанні;
- розроблення та дослідження алгоритмічних методів виконання міжбазисних перетворень у полях Галуа, арифметика яких лежить в основі методу гомоморфного шифрування;
- розроблення програмного забезпечення виконання операцій над елементами скінченних полів у поліноміальному та нормальному базисах, а також виконання міжбазисних перетворень;
- розроблення архітектури програмної системи для вирішення задачі класифікації на основі приватних даних;

- розроблення та дослідження методів функціонального шифрування даних, з метою їх подальшого застосування при побудові систем штучного інтелекту;
- розроблення програмного забезпечення вирішення задачі класифікації з використанням розробленого методу функціонального шифрування;
- аналіз розробленого програмного забезпечення захисту приватних наборів даних.

Об'єкт дослідження – процеси оброблення приватних даних у програмних системах інтелектуального аналізу даних.

Предмет дослідження – методи розроблення алгоритмічного та програмного забезпечення систем захисту приватних наборів даних у задачах класифікації.

Методи дослідження: теорія програмування, теорія програмних систем, теорія алгоритмів, теорія скінченних полів, методи оптимізації, методи штучного інтелекту.

Наукова новизна одержаних результатів полягає у наступному:

1. **Уперше** запропоновано архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних, характерною особливістю якої є захист приватних наборів даних, шляхом функціонального шифрування, що відбувається на стороні клієнта, і дозволяє збільшити кількість наборів даних для навчання загальнодоступних систем аналізу даних і штучного інтелекту.
2. **Уперше** запропоновано модифікацію програмної моделі шифрування даних, яка відрізняється від існуючої використанням двовимірних згорткових нейронних мереж, замість одновимірних, і дозволяє застосовувати модель шифрування з використанням нейронних мереж до даних, що представлені набором пікселів, з яких складається зображення.

3. **Уперше** розроблено алгоритмічно-програмний метод функціонального шифрування наборів даних, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах аналізу даних та штучного інтелекту шляхом зменшення їх розмірності й функціонального шифрування отриманих даних з використанням приватного ключа.
4. **Уперше** розроблено алгоритмічно-програмний метод пошуку нормальних поліномів серед незвідних, який відрізняється від існуючого використанням простих чисел у десятковому представленні замість поліномів, що дозволяє зменшити обчислювальні витрати алгоритму пошуку незвідних многочленів з $O(n^3)$ до $O(n \log(\log\ n))$ і, як наслідок, спростити міжбазисні перетворення у бінарних скінченних полях з метою пришвидшення виконання операцій над елементами поля у методах гомоморфного шифрування даних.
5. **Уперше** розроблено модифікований спосіб побудови матриці переходу між поліноміальним та нормальним базисами скінченного поля, який полягає у використанні рекурентної формули $\alpha_{i+1} = t^{p^{i+1}} = t^{p^i \cdot p} = (\alpha_i)^p$ замість обчислення остачі від ділення елемента $t^{p^{i+1}}$ на незвідний поліном, що дозволяє зменшити кількість використовуваної пам'яті з n^{p^i} до $n \cdot p$, а також обчислювальну складність з $O\left(m^{p^i}\right)$ до $O\left(m^p\right)$.

Практичне значення одержаних результатів полягає у спрощенні процесу розроблення загальнодоступних систем аналізу даних і штучного інтелекту на основі приватних даних, яке передбачає застосування розроблених методів захисту приватних наборів даних.

Розроблене алгоритмічне та програмне забезпечення для захисту приватних наборів даних, яке застосовано при виконанні ініціативної науково-дослідної роботи «Розроблення та дослідження засобів зберігання

секретності приватних наборів даних при побудові систем аналізу даних і штучного інтелекту» (номер державної реєстрації 0121U109925) для шифрування даних, з метою їх подальшого використання в загальнодоступних системах штучного інтелекту.

Особистий внесок здобувача. Усі основні результати дисертаційного дослідження, які представлені до захисту, одержані автором особисто. У публікаціях, написаних у співавторстві, здобувачеві належать наступні результати. У роботі [1] здобувачем розроблено метод пошуку нормальних поліномів для міжбазисних перетворень бінарних скінченних полів, незвідні поліноми у якому пропонується шукати як прості числа представлені у системі числення з основою 2. У роботі [2] здобувачем показано, як час виконання операцій над елементами розширеного поля Галуа залежить від базису, в якому представлені елементи. У роботі [3] здобувачем проведено комплексний порівняльний аналіз методів збереження приватності в машинному навчанні. У роботі [4] здобувачем запропоновано архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних, особливістю якої є захист приватних наборів даних шляхом функціонального шифрування, що відбувається на стороні клієнта. У роботі [5] здобувачем розроблено метод функціонального шифрування наборів даних-зображень, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах для вирішення задачі класифікації. У роботі [6] здобувачем проаналізовано загрози приватності в системах машинного навчання й розглянуто методи протидії їм. У роботі [7] здобувачем розроблено спосіб побудови матриці переходу, шляхом пошуку базисних елементів нормального базису на основі рекурентної формули.

Апробація результатів дисертації. Основні результати дисертаційного дослідження доповідалися та обговорювалися на наукових конференціях:

1. Тринадцята наукова конференція магістрантів і аспірантів «Прикладна математика та комп'ютинг» (ПМК' 2020), Київ, 18 - 20 листопада 2020 р., Київ, Україна.
2. XII Міжнародна науково-практична конференція молодих учених та студентів «Актуальні задачі сучасних технологій», 6-7 грудня 2023, Тернопіль, Україна.
3. XI науково-технічно конференція «Інформаційні моделі, системи та технології», 13-14 грудня 2023 р., Тернопіль, Україна.

Публікації. Основні наукові результати дисертаційної роботи опубліковано у 7 наукових працях, зокрема у 4 наукових статтях, з яких 1 статтю опубліковано у закордонному фаховому виданнях третього квартиля (Q3), яке проіндексоване в базі даних Scopus, 1 статтю опубліковано у виданні, яке проіндексоване в базі даних Web of science, і 2 статті опубліковано у фаховому виданні, включеному до переліку наукових фахових видань України з присвоєнням категорії «Б» та у 3 матеріалах наукових конференцій.

Структура роботи. Дисертаційна робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, що включає 105 найменувань, та 14 додатків. Загальний обсяг роботи становить 254 сторінки, у тому числі 141 сторінок основного тексту, 27 рисунків, 16 таблиць, 3 лістинги.

1. АНАЛІЗ ПРОГРАМНИХ СИСТЕМ ЗАХИСТУ ПРИВАТНИХ НАБОРІВ ДАНИХ У МАШИННОМУ НАВЧАННІ

1.1. Етичні аспекти використання систем штучного інтелекту

Штучний інтелект трансформує науку та суспільство в цілому. Технології штучного інтелекту змінюють процеси обробки та аналізу даних. Автономні та напів автономні системи частіше застосовуються у різних сферах (зокрема в медицині, транспорті та на виробництві). Оскільки штучний інтелект суттєво впливає на різні сфери суспільства, його поширення спричинило широкі дискусії щодо принципів і цінностей, яких необхідно дотримуватись під час розроблення та використання таких систем. Дискусії щодо того, що штучний інтелект може замінити значну частину професій, бути використаним зловмисниками, дозволить уникнути відповідальності або ненавмисно поширюватиме упередженість, підриваючи принцип справедливості, привертають увагу як науковців, так і суспільства впродовж останніх років. Проводяться дослідження на тему етичного штучного інтелекту, зокрема щодо ризиків та негативних наслідків (у тому числі ненавмисних), до яких може призвести використання штучного інтелекту.

Реагуючи на суспільний запит, національні та міжнародні організації створюють спеціальні комітети з експертами зі штучного інтелекту, яким доручають розробку політик щодо регулювання штучного інтелекту. Прикладами таких організацій є експертна група високого рівня зі штучного інтелекту (призначена Європейською комісією), група експертів зі штучного інтелекту Організації економічного співробітництва та розвитку, консультативна рада з етичного використання штучного інтелекту та даних у Сінгапурі та спеціальний комітет зі штучного інтелекту в Палаті лордів Сполученого Королівства (Великобританії) [1]. Подібні дії здійснюються й в приватному секторі, в професійних асоціаціях та некомерційних організаціях, зокрема власні рекомендації та принципи використання

штучного інтелекту публікували компанії Google і SAP, організації Асоціація обчислювальної техніки (ACM), Access Now та Amnesty International.

Під час розроблення та використання систем штучного інтелекту постають багато питань пов’язаних з етичними аспектами їх використання. Серед основних питань виділяють прозорість, справедливість, нешкідливість, відповідальність, приватність, «доброзичливість», свобода, довіра, гідність, стабільність та солідарність [1]. Характеристика цих принципів представлена у табл. 1.1. Розглянемо докладніше найпоширеніші принципи.

Принцип прозорості передбачає пояснення деталей розроблених систем. Передусім прозорість – це спосіб мінімізувати шкоду та покращити штучний інтелект. Для досягнення більшої прозорості, пропонується розкривати інформацію тим, хто розробляє або розгортає системи штучного інтелекту [1]. Такою інформацією може бути особливість використання штучного інтелекту, вихідний код, використані дані для розроблення системи, доказова база ефективності застосування штучного інтелекту, обмеження, закони, відповідальність, деталі щодо інвестицій в системи штучного інтелекту та можливий вплив використання таких систем. Дотримання цього принципу є сприяє довірі до таких систем.

Принцип справедливості виражається в термінах запобігання, моніторингу або пом’якшення небажаних упереджень та дискримінації [1]. Підкреслюється важливість справедливого доступу до даних, штучного інтелекту та його переваг. Крім цього, розглядається ризик упередженості в наборах даних, підкреслюється важливість отримання та обробки точних, повних і різноманітних, особливо навчальних, даних.

Таблиця 1.1

Етичні аспекти використання штучного інтелекту

Принцип	Поши-реність	Суть
Прозорість	Висока	Необхідність пояснення деталей розроблених систем
Справедливість	Висока	Запобігання, моніторинг або пом’якшення небажаних упереджень та дискримінації
Нешкідливість	Висока	Системи не повинні завдавати передбачуваної чи ненавмисної шкоди
Відповідальність	Висока	Визначення розподілу відповідальності, а також засобів правового захисту користувачів
Приватність	Середня	Забезпечення захисту та безпеки даних
«Доброзичливість»	Середня	Використання систем для людського добробуту та процвітання
Свобода	Середня	Вільне використання технологій та їх поширення (без маніпуляцій і стеження за користувачами)
Довіра	Середня	Достовірність досліджень та надійність технологій, фахівців та організацій
Гідність	Низька	Уникнення шкоди під час взаємодії людини і штучного інтелекту
Стабільність	Низька	Розроблення та розгортання систем, які стабільно обробляють дані та чиї ідеї залишаються дійсними з часом
Солідарність	Низька	Наслідки використання штучного інтелекту для ринку праці

Прикладом випадків, що викликають дискусії щодо порушення принципу справедливості, є використання даних, що захищені авторським правом. Наприклад, американське видання New York Times подало позов

проти компаній OpenAI та Microsoft за порушення авторських прав, оскільки чатботи ChatGPT та Copilot навчались на даних видання й за певними запитами генерують вихідні дані, що дослівно відтворюють контент, що був використаний для навчання (без посилання на першоджерело). Також прикладом проблем з принципом справедливості є використання контенту згенерованого штучним інтелектом як власних результатів (зокрема, під час навчання або як авторських творів для комерційної вигоди). Одним з варіантів вирішення такої проблеми розглядається маркування контенту згенерованого штучним інтелектом.

Принцип нешкідливості передбачає, що штучний інтелект не повинен завдавати передбачуваної чи ненавмисної шкоди. Зокрема, необхідно уникати конкретних ризиків або потенційної шкоди (наприклад, навмисне зловживання через кібервійну, хакерство), а також запропонувати стратегії управління ризиками. Рекомендації щодо запобігання шкоди передбачають технічні заходи та стратегії управління для уникнення втручання на рівні досліджень штучного інтелекту, проєктування, розроблення та розгортання технологій. Технічні рішення можуть реалізовуватись шляхом використання вбудованої оцінки якості даних або розроблення системи за принципом приватності за задумом (privacy by design). Стратегії управління можуть реалізовуватись через активну співпрацю зацікавлених осіб, розроблення й дотримання законодавства, а також встановлення узгоджених практик розроблення та нагляду за розробленими системами [1]. Прикладами порушення принципу нешкідливості є використання штучного інтелекту для шахрайства, зокрема, створення несправжніх новин чи зловмисних обманів з використанням технології deepfake, що дозволяє синтезувати мультимедійний контент (наприклад, обличчя, голоси, мову) для створення маніпулятивних матеріалів, які виглядатимуть так, ніби вони створені реальною людиною.

Принцип відповідальності, незважаючи на поширеність терміну «відповідальний штучний інтелект», не має чіткого визначення. Він

полягає в роз'ясненні розподілу відповідальності за певні ситуації (якщо їх можна передбачити заздалегідь), а також зосереджується на засобах правового захисту користувачів. Відповідальними за дії та рішення штучного інтелекту є різні учасники: розробники штучного інтелекту, проєктувальники, організації, що розробляють й використовують такі системи. Існують розбіжності щодо того, чи люди завжди повинні бути єдиними суб'єктами, які несуть остаточну відповідальність за технологічні артефакти, однак, станом на зараз остаточне рішення за ними. Яскравим прикладом принципу відповідальності є необхідність прийняття остаточного рішення людиною, особливо в питаннях, що стосуються життя й здоров'я людини. Наприклад, в медицині остаточне рішення щодо використання того чи іншого способу лікування приймається лікарем, незважаючи на рекомендації або висновки систем штучного інтелекту, які можуть враховуватись.

Принцип приватності пов'язують із захистом та безпекою даних, що використовуються при розробленні програмних систем штучного інтелекту. Приватність, відповідно до етики штучного інтелекту, – це цінність, яку необхідно підтримувати і захищати. Для збереження приватності передбачається використання технічних рішень (такі як диференційована конфіденційність, приватності за задумом (privacy by design), мінімізація даних і контроль доступу); додаткових досліджень і обізнаності про розроблені системи (зокрема, про використання даних в них); нормативних підходів. Прикладом втрати приватності є виявлення конфіденційної інформації про особу, на основі висновків системи аналізу даних та штучного інтелекту. Більш детальний аналіз загроз приватності даних у системах аналізу даних та штучного інтелекту проведено в підрозділі 1.2, а існуючих методів збереження приватності в підрозділі 1.3.

Принцип «доброзичливості» полягає у використанні систем для людського добробуту та процвітання. Невизначеність стосується суб'єктів, які мають отримати вигоду від штучного інтелекту: представники

приватного сектору наголошують на перевагах штучного інтелекту для клієнтів, хоча багато політик вимагають, щоб штучний інтелект був спільним і приносив користь всім. Стратегії досягнення цього включають узгодження штучного інтелекту з людськими цінностями, просування наукового погляду розуміння світу, мінімізацію концентрації влади; більш тісну роботу з «постраждалими» людьми, мінімізацію конфлікту інтересів; підтвердження «доброзичливості» через попит і відгуки споживачів, а також розроблення нових показників і вимірювань добробуту людини [1].

Принцип свободи фокусується на вільному використанні технологій штучного інтелекту, їх поширенню, однак, без маніпуляцій і стеження за користувачами. Вважається, що свобода та автономія сприяють прозорості та передбачуваності штучного інтелекту, активно розширюючи знання людей про штучний інтелект, надаючи повідомлення та згоду, або, навпаки, активно утримуючись від збору та поширення дані за відсутності інформованої згоди [1].

Принцип довіри полягає в достовірності досліджень та надійності технологій, а також фахівців та організацій, що розробляють й використовують штучний інтелект. Він передбачає використання надійних «принципів проєктування» та підкреслює важливість довіри клієнтів. Довіра до рекомендацій, суджень і використання штучного інтелекту необхідна для того, щоб штучний інтелект «реалізував свій потенціал, що змінює світ» [1]. Рекомендації щодо побудови й підтримки довіри включають освіту, надійність, підзвітність, процеси моніторингу та оцінки цілісності систем штучного інтелекту з плином часу, а також інструменти та методи, що забезпечують відповідність нормам і стандартам. Також довірі сприяє діалог із зацікавленими сторонами, обізнаність про цінність використання персональних даних та уникнення шкідливих наслідків.

Розглянуті етичні аспекти є важливими для безпечного й правомірного використання систем штучного інтелекту. Важливо їх враховувати при розробленні та використанні програмних систем

інтелектуального аналізу даних. Основним фокусом дисертації є принцип приватності, а саме використання приватних наборів даних під час розроблення програмних систем інтелектуального аналізу даних.

1.2. Аналіз загроз приватності даних у системах аналізу даних та штучного інтелекту

Усвідомлення загроз систем машинного навчання є передумовою для розроблення методів захисту, що їм протидіятимуть. Зважаючи на це, варто проаналізувати й класифікувати такі загрози. Атаки на системи машинного навчання можна поділити за етапом на якому вони відбуваються: під час навчання чи формування передбачень. Напади на системи під час навчання є досить частими, оскільки більшість моделей перенавчають систему новими даними. Прикладом цього є соціальні мережі, які аналізують поведінку користувача й адаптуються залежно від неї.

Існують різні категорії атак на моделі машинного навчання залежно від мети зловмисника (шпигунство, диверсія, шахрайство) і етапів машинного навчання (навчання та використання в реальних сценаріях). Їх також можна назвати атаками на алгоритм і атаками на модель відповідно.

Основними загрозами для систем машинного навчання є наступні типи атак на системи машинного навчання [2-6]:

- атаки логічного висновку (inference attacks);
- атаки на інверсію моделі (model inversion attacks);
- атаки з висновком про членство (membership inference attacks);
- атаки з крадіжкою моделі (model stealing attacks);
- атаки з отруєнням моделі (model poisoning attacks).

Важливо усвідомлювати наведені загрози й вживати заходів для їх запобігання, як-от використання методів збереження приватності, впровадження надійних заходів безпеки та регулярний моніторинг і оцінка продуктивності моделей машинного навчання, щоб переконатися, що вони

не роблять упереджених або несправедливих прогнозів. Тож розглянемо їх докладніше, а на основних методах протидії зосередимось у підрозділі 1.3.

1.2.1. Атаки логічного висновку

Атаки логічного висновку (inference attacks) – це вид загроз безпеці, коли зловмисник намагається визначити конфіденційну інформацію, аналізуючи реакції системи на певні запити або дії. У цьому випадку, атакуючий не отримує прямий доступ до приватних даних, але намагається визначити їх через відповіді системи.

Атака логічного висновку може включати в себе спроби визначити деталі навчальних даних чи навіть витягти інформацію про окремі точки даних, проводячи повторні запити до моделі. Прикладом цього в медичних системах є висновок щодо використання конкретного профілю пацієнта, пов’язаного з захворюванням, у процесі навчання класифікатора. У контексті втрати приватності, атака логічного висновку може включати в себе визначення чутливих деталей про осіб через аналіз агрегованих статистик.

Захист від атак логічного висновку включає в себе правильне шифрування, анонімізацію даних та ретельний контроль за відповідями системи, щоб мінімізувати ризик витоку приватної інформації.

1.2.2. Атаки інверсії моделі

Атаки інверсії моделі (model inversion) – це тип загроз конфіденційності в машинному навчанні, коли зловмисник використовує вихідні дані моделі машинного навчання, щоб отримати конфіденційну інформацію про вхідні дані, які використовуються для навчання моделі [2]. Такі атаки спираються на те, що моделі машинного навчання розроблені для вилучення шаблонів із вхідних даних і використання їх для прогнозування. Під час атаки інверсії моделі зловмисник може отримати доступ до висновку (результату) моделі машинного навчання та використовувати його

для отримання інформації про вхідні дані, які використовувалися для навчання моделі.

Розглянемо модель розпізнавання облич, що навчена на наборі даних зображень з обличчями, які містять відповідні гендерні мітки. Зловмисник може ввести зображення обличчя та використовувати вихідні дані моделі (тобто передбачену стать), щоб отримати інформацію про особу на зображенні.

Ще одним прикладом атаки інверсії моделі є відновлення приватної інформації особи, такої як її номер соціального страхування, номери кредитних карток та іншої конфіденційної інформації, з навченої моделі машинного навчання. Це може статися, якщо модель навчена на наборі даних, який містить як конфіденційну інформацію особи, так і неконфіденційну. Використавши вихідні дані моделі, зловмисник може реконструювати конфіденційну інформацію про особу.

Для запобігання атакам інверсії моделі важливо захищати конфіденційність вхідних даних, що використовуються для навчання моделей. Цього можна досягти використовуючи методи збереження конфіденційності, такі як анонімізація чи диференціальна приватність. Це дозволить змінити вхідні дані таким чином, щоб зловмиснику було важко відновити конфіденційну інформацію. Також системи штучного інтелекту необхідно регулярно моніторити та оцінювати, з метою запобігання упередженим або несправедливим прогнозам, які можуть розкрити приватну інформацію.

1.2.3. Атаки з висновком про членство

Атаки з висновком про членство (membership inference) – це тип загроз конфіденційності в машинному навчанні, де зловмисник намагається визначити, чи були дані конкретної особи включені до навчального набору даних, який використовується для розроблення моделі машинного навчання. Ці атаки використовують той факт, що моделі машинного

навчання розроблені для узагальнення вхідних даних з метою прогнозування нових, невідомих даних [5].

Під час атаки з висновком про членство зловмисник може отримати доступ до виходу моделі машинного навчання та використовувати його, щоб визначити, чи використовувалися дані певної особи для навчання моделі. Наприклад, розглянемо модель машинного навчання, яка навчена на наборі даних медичних записів і використовується для прогнозування ризику певного стану здоров'я. Зловмисник може ввести дані певної особи в модель і використати вихідні дані, щоб визначити, чи включені дані цієї особи в навчальний набір даних.

Цей тип атаки може бути особливо небезпечним у додатках пов'язаних з медичними дослідженнями чи кримінальними розслідуваннями, де включення даних особи до навчального набору даних може виявити приватну інформацію про неї. Крім того, атаки на приналежність можуть використовуватися для підриву конфіденційності осіб у навчальному наборі даних, дозволяючи зловмисникам визначити, чи використовувалася їх конфіденційна інформація для навчання моделі.

Для запобігання атакам з висновком про членство, необхідно вжити заходів для збереження конфіденційності вхідних даних, що використовуються у процесі навчання моделей машинного навчання. Наприклад, диференціальна конфіденційність може використовуватись, щоб порушити вхідні дані таким чином, аби зловмиснику було важко визначити, чи були дані конкретної особи включені до навчального набору даних. Іншим прикладом методів збереження приватності є використання безпечних багатосторонніх обчислень, що дозволяють зберігати конфіденційну інформацію в зашифрованому вигляді протягом усього навчального процесу. Крім того, моделі машинного навчання слід регулярно відстежувати та оцінювати, щоб переконатися, що вони не роблять упереджених або несправедливих прогнозів, які можуть розкрити приватну інформацію.

1.2.4. Атаки з крадіжкою моделі

Атаки з крадіжкою моделі (model stealing) – це тип загрози конфіденційності в машинному навчанні, коли зловмисник прагне отримати копію навченої моделі машинного навчання, з метою її використання для зловмисних цілей [2]. Це може статися, коли навчена модель публікується у відкритому доступі, наприклад, у хмарних системах машинного навчання, або коли зловмисник отримує доступ до моделі неавторизованим способом.

Отримавши копію моделі машинного навчання, зловмисник може використовувати її для прогнозування або зворотного проектування навчальних даних, використаних для створення моделі. Це може розкрити приватну інформацію про осіб, чиї дані використовувалися для навчання моделі, або дозволити зловмиснику використовувати модель для створення зловмисних прогнозів. Наприклад, зловмисник може використати викрадену модель, щоб створити фальшиві сповіщення про шахрайство з кредитною карткою або підробити медичні діагнози, що потенційно може завдати шкоди окремим особам або організаціям.

Для запобігання атакам крадіжки моделі, важливо впровадити надійні заходи безпеки для захисту навчених моделей, наприклад шифрування та контроль доступу. Також слід регулярно відстежувати та оцінювати моделі машинного навчання, щоб переконатися, що вони не використовуються зі зловмисною метою, а про будь-яку підозрілу діяльність слід повідомляти та негайно розслідувати її. Крім того, важливо ретельно розглянути наслідки обміну навченими моделями машинного навчання та ділитися моделями лише з довіреними сторонами, які мають законну потребу в інформації.

На додаток до цих профілактичних заходів фахівці з машинного навчання також можуть використовувати такі методи, як диференціальна конфіденційність і безпечні багатосторонні обчислення, щоб забезпечити конфіденційність і безпеку вхідних даних, що використовуються для навчання моделі, зменшуючи ризик розкриття або викрадення конфіденційної інформації. Ці методи можуть допомогти запобігти

отриманню зловмисниками конфіденційної інформації з навчальних даних і використанню її для викрадення навченої моделі.

1.2.5. Атаки з отруєнням моделі

Атаки з отруєнням моделі (model poisoning) – це тип загрози конфіденційності в машинному навчанні, при якому зловмисник маніпулює навчальними даними, що використовуються для створення моделі машинного навчання, щоб змусити модель поводитись неправильно [2, 3]. Цей тип атаки можна здійснити шляхом додавання, модифікації або видалення навчальних прикладів таким чином, який вплине на поведінку моделі.

Наприклад, розглянемо систему виявлення шахрайства в транзакціях кредитних карток. Якщо зловмисник зробить зміни в навчальних даних, що використовуються для створення моделі машинного навчання, шляхом додавання великої кількісті шахрайських транзакцій до навчальних даних, то модель класифікуватиме більшість транзакцій як законні, навіть у випадках, коли вони є шахрайськими. Як наслідок, модель не зможе виявити справжні шахрайські транзакції, що спричинить значну шкоду власникам системи.

Для запобігання атакам отруєння моделі важливо впроваджувати надійні заходи безпеки для захисту навчальних даних, які використовуються для створення моделей машинного навчання. Наприклад, впровадження контролю доступу, щоб обмежити доступ до даних, а також регулярний моніторинг і перевірка навчальних даних для виявлення будь-яких підозрілих змін. Крім того, фахівці з машинного навчання повинні ретельно перевіряти та очищати навчальні дані, перш ніж використовувати їх для створення моделей, щоб мінімізувати ризик маніпулювання даними або їх пошкодження. Крім цього, фахівці з машинного навчання також можуть використовувати такі методи, як диференціальна приватність і безпечні багатосторонні обчислення, щоб забезпечити конфіденційність і безпеку навчальних даних. Ці методи можуть допомогти запобігти

маніпулюванню даними, які використовуються для навчання моделі, і переконатися, що модель не отруєна чи іншим чином нескомпрометована.

Важливим є врахування наслідків атак з отруєнням моделі під час розгортання моделей машинного навчання в реальних системах. Це можна реалізувати завдяки регулярному моніторингу та оцінці моделей для виявлення будь-яких ознак зловмисної поведінки та вживанню відповідних заходів для зменшення ризику заподіяння шкоди. Крім того, беручи до уваги потенціал атак із отруєнням моделі, необхідно активно працювати над їх запобіганням, використовуючи надійні механізми безпеки та впроваджуючи методи збереження приватності.

1.3. Аналіз алгоритмічно-програмних методів збереження приватності в машинному навчанні

Розроблення й аналіз методів захисту приватних наборів даних у системах з використанням штучного інтелекту є областю досліджень багатьох науковців й провідних IT компаній (Google, Apple, Microsoft). Це простежується, як в значній кількості наукових статей на цю тему в останні роки; так і в проведенні різноманітних воркшопів та конференцій для представлення й обговорення останніх досліджень (наприклад, на постійній основі є щорічний воркшоп PPML (Privacy-Preserving Machine Learning), що присвячений виняткова питанням збереження приватності даних у машинному навчанні). У зв'язку з цим, актуальною задачею є комплексний порівняльний аналіз методів збереження приватності. Основними методами збереження приватності в машинному навчанні є [7, 8]: генерація синтетичних наборів даних, обробка приватних наборів даних (анонімізація, диференційна приватність, гомоморфне шифрування) та федеративне навчання. Проведемо комплексний порівняльний аналіз цих методів.

1.3.1. Методи генерації синтетичних даних

Синтетичними даними називаються штучні дані, які, на відміну від реальних даних, є згенерованими за допомогою різних алгоритмів та

статичних методів моделювання характеристик й закономірностей реальних даних. Їх ідея полягає в утворенні нових точок даних, тож це не просто модифікація існуючих даних, а їх імітація. Проте, анонімність синтетичних даних не є аксіомою. Такі дані використовують для навчання моделі з наміром перенести результати її навчання на реальні дані. Побудова надійного генератора синтетичних даних дозволить досягнути таких переваг:

- швидка та дешева генерація даних у бажаній кількості;
- використання синтетичних даних для заміни сегментів приватних даних, що входять до реального набору даних;
- генерація точних міток для даних у специфічних випадках, коли задача отримання реальних даних є складною.

Шляхи побудови генератора синтетичних даних є різними. Зокрема, вони можуть включати комбінування реальних даних або доменну рандомізацію. Найпоширенішими генеративними моделями є прихована модель Маркова, Баєсівська мережа, варіаційний автокодувальник, агентне моделювання та генеративні конкуруючі нейронні мережі [9-11].

Розглянемо докладніше одну з таких моделей, а саме генеративні конкуруючі мережі (Generative adversarial networks – GANs). Ця генеративна модель запропонована Яном Гудфелоу та іншими дослідниками у 2014 році [12]. Вона представляє собою архітектуру глибокого навчання без вчителя, основна ідея якої полягає в тому, щоб поставити одну нейронну мережу проти іншої у грі з нульовою сумою. Згідно з архітектурою, одна мережа генерує нові дані, використовуючи генеративні методики, а інша оцінює їх справжність, використовуючи дискримінаційні алгоритми.

Основною метою дискримінаційних алгоритмів є класифікація вхідних даних. Інакше кажучи, базуючись на особливостях вхідних зразків, алгоритмам необхідно розрізнити мітку, якій відповідає екземпляр даних. Наприклад, у системі визначення порушень правил інформаційної платформи (таких як використання ненормативної лексики) основною

метою є визначення наявності порушень, на основі аналізу даних. Формально таку задачу можна сформулювати за допомогою обчислення умовної ймовірності $p(y|x)$, яка визначає ймовірність отримання мітки y при вхідних даних x.

Задача генеративних моделей є значно складнішою, ніж аналогічних дискримінаційних моделей. Вони працюють протилежним чином: передбачають параметри, які повинні мати дані, щоб вони належали певній мітці. У розглянутому прикладі з визначенням порушень правил інформаційної платформи, метою генеративних моделей є обчислення й збільшення ймовірності того, що дані містять елементи, що порушують правила інформаційної платформи. Математично це можна сформулювати як умовна ймовірність $p(x|y)$, тобто ймовірність наявності ознак вхідних даних x, що дозволять отримати мітку y. Варто зауважити, що ймовірність $p(x|y)$ у генеративних моделей залишатись сталою, при фіксуванні ваг моделі, тож вони можуть використовуватись як класифікатори, однак, їх можливості значно більші, ніж класифікація вхідних даних.

На рис. 1.1 зображено класичну схему моделі генеративної конкуруючої мережі. Відповідно до схеми, є дві основні складові мережі: генератор та дискримінатор. Перша складова навчається генерації реалістичних даних, а інша розпізнаванню хибних (підроблених) даних.

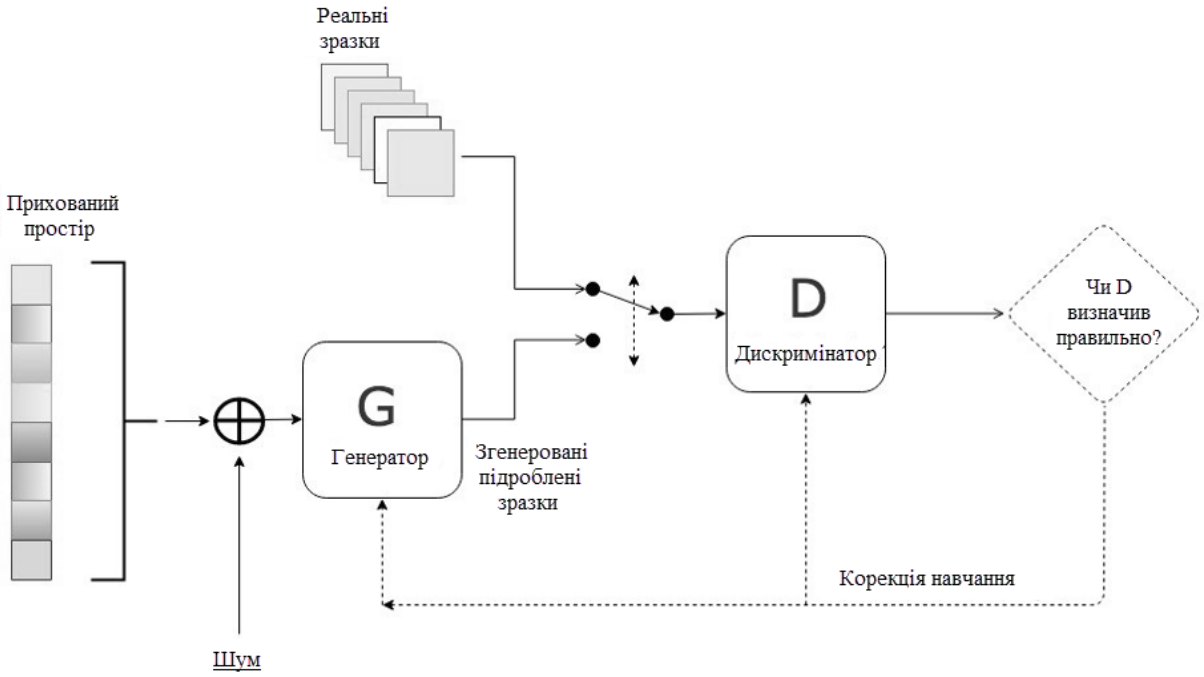


Рис. 1.1. Схема моделі генеративної конкуруючої мережі [13]

Навчання генеративних нейронних мереж, відповідно до зображеної схеми, відбувається у наступній послідовності:

1. Формування шуму з випадкового розподілу та його подання на вхід генератора *G*, з метою генерації екземпляру-підробки *x* (у такому випадку мітка $y = 0$). Як наслідок, отримуємо пару даних (x, y).
2. Подання отриманого підробленого зразку даних $(x, 0)$ та реального екземпляра даних $(x, 1)$, незалежно один від одного, на вхід дискримінатора *D*.
3. Обчислення функції втрат дискримінатора *D*. Оскільки дискримінатор є мережею, що вирішує задачу бінарної класифікації, як реальних *x*, так і підроблених *x*, то поєднуємо їх як спільну втрату.
4. Обчислення функції втрат генератора *G*.
5. Коригування ваг моделей нейронних мереж з використанням визначених для них функцій втрат.
6. Застосування алгоритмів оптимізації. Прикладом яких є градієнтний спуск, RMSprop, ADAM.

7. Повторення кроків 1-6 впродовж певної кількості епох навчання.

Важливим елементом навчання є функції втрат. Метою конкуруючих нейронних мереж є повторення розподілу ймовірностей, тому їх функції втрат мають відображати розподіл між реальними даними і тими, що були згенеровані мережею-генератором. Існують різні підходи до визначення відмінностей між розподілами даних для таких мереж, проте така задача є актуальною й перебуває у фокусі багатьох дослідників. Одним з найпоширеніших підходів до визначення втрат для архітектури GAN є функція мінімакс втрат. Метою генератора є мінімізація цієї функції, а дискримінатора – максимізація. Цю функцію втрат формулюють наступним чином [12, 14]: $\min_G \max_D L(D,G) = E_x\left[\log\left(D\left(x\right)\right)\right] + E_z\left[\log\left(1 - D\left(G\left(z\right)\right)\right)\right]$. У цій формулі E_x позначає очікуване значення для всіх зразків даних з реального набору, а E_z позначає очікуване значення для всіх підроблених зразків даних (створених генератором G). $G(z)$ позначає результат роботи генератора при шумі z, $D(x)$ є оцінкою дискримінатора щодо ймовірності того, що зразок даних x реальний, а $D(G(z))$ – оцінка ймовірності, що підроблений екземпляр є реальним, зроблена дискримінатором. У основі цієї формули визначення перехресної ентропії між розподілами реальних та згенерованих даних. Оскільки генератор не впливає безпосередньо на вираз $\log(D(x))$ з розглянутої функції, то мінімізація втрат генератором, є еквівалентною мінімізацію виразу $\log(1 - D(G(z)))$. Однак, у статті [12], де було запропоновано конкуруюче навчання нейронних мереж, зазначено, що результатом використання такої функції втрат у певних випадках може бути «застрягання» генеративної конкуруючої нейронної мережі, оскільки дискримінатор виконуватиме занадто просту задачу. Одним з механізмів уникнення такої ситуації є зміна функції втрат для генератора на $\log(D(G(z)))$, яку він буде максимізовувати.

Застосування генеративних конкуруючих нейронних мереж має великий потенціал, зважаючи на можливість імітації будь-якого розподілу

даних після відповідного навчання. Прикладами використання таких мереж є генерація реалістичних зображень (приклад яких представлено на рис. 1.2), відновлення пошкоджених зображень, покращення якості існуючих зображень, генерація контенту для відеоігор та віртуальної реальності, створення текстів, музики та звуків.

Рис. 1.2. Приклад реалістичних зображень, що згенеровані конкуруючою нейронною мережею [15]

При цьому варто зауважити, що існують й недоліки генеративних конкуруючих нейронних мереж. Одним з них є складність навчання, оскільки чутливість таких мереж до гіперпараметрів є високою, а також модель може бути незбіжною або існуватиме дисбаланс між мережами генератора та дискримінатора. Ще одним недоліком є не реальність даних, а лише їх реалістичність, тобто може виникнути ситуація (особливо для малих наборів даних), коли лише певний обмежений набір екземплярів даних створюється генератором.

1.3.2. Аналіз методів попередньої обробки наборів даних

Попередня обробка наборів даних є ще однією групою методів збереження приватності, серед яких анонімізація даних, диференційна приватність (differential privacy) та гомоморфне шифрування. Проаналізуємо ці методи докладніше.

Анонімізація даних

Анонімізація даних – це процес збереження приватності даних, у ході якого видаляються або змінюються ідентифікатори, які пов’язують вхідні

дані з певною особою. Іншими назвами цього процесу, що використовуються не так часто, є маскування, обфускація та деідентифікація даних. Процес анонімізації даних застосовується у багатьох галузях людського життя, особливо тих, що використовують конфіденційну, секретну або чутливу інформацію. Класичними прикладами таких галузей є медицина, військова та фінансова сфери. Використання анонімізованих даних, сприяє збереженню приватності даних. Прикладом цього є механізм оцінки виконання абітурієнтами тесту ЗНО (зовнішнє незалежне оцінювання). Спершу робота кожного учасника анонімізується (ідентифікаційні дані кодуються з використанням певного шифру) і лише потім подаються для перевірки комп'ютером або експертами (у випадку творчого завдання). При цьому програми й люди, що перевіряють роботу, не можуть ідентифікувати, хто написав роботу, що сприяє об'єктивності й унеможливлює фальсифікації результатів. Після визначення результатів, шифри декодуються й їм у відповідник знову ставляться ідентифікаційні дані користувача, що дозволяє кожному учаснику отримати свій результат тестування.

Існують різні способи забезпечення анонімності даних, серед яких перестановка даних, придушення атрибутів або записів, підміна даних (частковим випадком якої є псевдонімізація), дисперсія чисел і дат маскування символів та узагальнення [16, 17].

Перестановка даних – це перестановка певних значень атрибутів, з метою збереження їх в зміненому наборі даних, однак, з унеможливленням їх асоціації з початковими даними.

Придушення атрибутів або записів – видалення ідентифікуючих даних з набору, якщо вони несуттєві або інші способи анонімізації не можна використати.

Процес підміни даних полягає в заміні значень атрибуту випадковими підробленими значеннями, що є схожими за форматом. Прикладом підміни даних є заміна номерів кредитних карток випадковим числом з 16 цифр.

Псевдонімізація є частковим випадком цього процесу. Вона передбачає заміну ідентифікуючих атрибутів кодовими значеннями, які формуються за узгодженим правилом. Варто зауважити, що залежно від правила формування псевдонімів, такі псевдоніми можуть бути як необоротними (коли неможливо отримати початкові дані, знаючи правило кодування), так і оборотними.

Дисперсія дат і чисел – це зміна значень атрибута даних на певний інтервал. Дані представлені у вигляді чисел та дат найчастіше використовується як параметри пошукових запитів до набору даних, тож вони є важливими з точки зору статистики. Однак, додавання або віднімання якогось однакового значення до значення атрибуту легко виявити й декодувати, тож варто додавати або віднімати не конкретне значення, а випадкове, що належить визначеному довірчому інтервалу. Це забезпечить як анонімізацію даних атрибуту, так розподіл таких даних.

Маскування символів полягає в заміні символів у значеннях атрибута даних. Прикладом цього є використання сталих символів «x» або «*». Маскування найчастіше використовується лише до певних символів атрибута, тобто воно є частковим.

Спосіб узагальнення полягає в заміні конкретних даних більш загальними, що все одно міститимуть корисну для аналізу інформацію. Внаслідок цього процесу точність даних знизиться. Найчастіше узагальнення використовують для діапазонів даних (наприклад, шляхом представлення значення віку людину як елементу з певного діапазону значень) та зменшення точності початкової інформації (наприклад, перетворення повної адреси проживання на менш точну: район або місто). Ще однією назвою цього способу є перекодування даних. Застосування узагальнення даних є доцільним, якщо в перетвореному вигляді дані будуть все ще корисні для аналізу.

Для більш детального розгляду базових способів анонімізації даних, використаємо структуровані табличні дані, які задані у табл. 1.2.

Таблиця 1.2

Початковий набір вхідних даних

Користувач	Вік	Електронна пошта	Поштовий індекс	Адреса	Кількість замовлень
Роман	21	roman@gmai.com	04128	вулиця Авіаційна 3	4
Володимир	33	volodymyr@i.ua	02230	вулиця Шевченка 9	7
Максим	19	maksym@ukr.net	03062	вулиця Авіаційна 12А	1
Марія	26	mariia@gmai.com	04210	проспект Івасюка 10Б	9

У табл. 1.3 представлений змінений (анонімізований) набір, отриманий в результаті застосування способів перестановки даних (для атрибуту *Користувач*), придушення атрибутів (для атрибуту *Адреса*), дисперсії чисел (*Вік*) та маскування символів (*Поштовий індекс* і *Електронна пошта*).

Таблиця 1.3

Змінений набір даних, внаслідок застосування способів анонімізації (перестановка даних, придушення атрибутів, дисперсія чисел та маскування символів)

Користувач	Вік	Електронна пошта	Поштовий індекс	Кількість замовлень
Марія	22	*****@gmai.com	041xx	4
Максим	31	*********@i.ua	022xx	7
Роман	18	********@ukr.net	030xx	1
Володимир	28	******@gmai.com	042xx	9

Застосувавши інші способи анонімізації до початкових даних, можна отримати модифікований набір, поданий у табл. 1.4. У цьому випадку до даних, було застосовано узагальнення даних (для атрибутів *Вік* та *Адреса*), маскування символів (*Електронна пошта*) та псевдонімізацію (атрибут *Користувач*).

Підсумовуючи розглянуті приклади, можна стверджувати, що головною перевагою анонімізації є приховання чутливих елементів початкових даних. Проте, навіть при застосуванні зазначених способів приховування ідентифікуючої інформації, модифіковані дані можуть бути повторно ідентифіковані (деанонімізовані). У такому випадку методи деанонімізації поєднують дані з різних джерел (наприклад, реєстрів, частина з яких є у відкритому доступі) для виявлення особистої інформації в модифікованому наборі даних. Зважаючи на це, анонімізація даних надає помилкове відчуття безпеки, як стверджують її критики [18].

Таблиця 1.4

Змінений набір даних, внаслідок способів анонімізації (узагальнення даних, маскування символів та псевдонімізація)

Користувач	Вік	Електронна пошта	Поштовий індекс	Адреса	Кількість замовлень
User 3	< 23	*****@gmai.com	04128	вулиця Авіаційна	4
User 1	> 29	******@i.ua	02230	вулиця Шевченка	7
User 4	< 23	********@ukr.net	03062	вулиця Авіаційна	1
User 2	23-29	******@gmai.com	04210	проспект Івасюка	9

Диференційна приватність

Диференційна приватність – метод збереження приватності даних, основною ідеєю якого є додавання випадкового шуму до початкових даних, з метою забезпечення конфіденційності. Завдання цього методу полягає в забезпеченні строгих статистичних гарантій того, що базуючись на модифікованих даних за допомогою рандомізованого алгоритму, висновок про приватні дані зробити не можна. Тобто, якщо p_1 це ймовірність того, що значення s є результатом певного запиту до початкового набору даних D_1 і p_2 – ймовірність того, що результатом цього самого запиту до модифікованого набору даних D_2 є таке ж значення s, то абсолютна різниця цих ймовірностей має знаходитись в межах статистичної похибки (значення менше певного значення ε). Сформулюємо математичне визначення диференційної приватності [19]. Позначимо D_1 – початковий приватний набір даних; A – рандомізований алгоритм, що опрацьовує початковий приватний набір даних; D_2 – змінений набір даних; а $E(A)$ – область значень цього алгоритму. Також визначимо певне фіксоване значення $\varepsilon > 0$. Тоді

алгоритм A називається ε-диференційно приватним, коли для кожного запису з наборів даних D_1 та D_2, відмінність яких є лише в одному записі для всіх підмножин $S \subseteq E(A)$, виконується $p\{A(D_1) \in S\} \leq e^{\varepsilon} \cdot p\{A(D_2) \in S\}$, де p позначає ймовірність, що отримана з випадковості рандомізованого алгоритму.

Забезпечити досягнення такого результату можна, додавши випадковий шум, наприклад з розподілом Гауса чи Лапласа. Слід відзначити, що важливим є обсяг доданого шуму до початкових даних, оскільки більша кількість шуму призводить до меншої інформативності (корисності) зміненого набору даних, як наслідок до меншої точності результатів аналізу даних. Це твердження проілюстроване на рис. 1.3.

Рис. 1.3. Залежність інформативності даних, від обсягу доданого шуму [20]

Високий рівень конфіденційності даних методом диференційної приватності досягається, завдяки застосуванню значної кількості шуму до зображення, однак, у такому випадку зразок даних майже повністю позбувається своїх характеристик, що робить неможливим подальший аналіз таких даних. З огляду на це, важливим є пошук балансу між конфіденційністю і корисністю (збереженню початкових особливостей) даних. Крім цього, одним з недоліків диференційної приватності є потреба у великій множині даних (більшій, ніж для методу анонімізації) для забезпечення ефективності застосування методу. Незважаючи на недоліки,

метод диференційної приватності є практичним. Його використовують для певних задач високотехнологічні компанії (такі як Google, Apple, Uber та Facebook). Крім цього, у 2020 році цей підхід застосовувався США під час проведення перепису населення.

Гомоморфне шифрування

Гомоморфне шифрування – це форма шифрування даних, при якій виконання обчислень на зашифрованих даних даватимуть результати, розшифрована версія яких є ідентичною результатам виконання обчислень на початкових даних [21]. В основі такого шифрування лежить поняття гомоморфізму, яке формально визначається як відображення двох алгебраїчних структур одного типу: функція f: $A \to B$ між двома множинами A та B, структура яких однакова, є гомоморфною, якщо виконується наступне: $f(xy) = f(x)f(y)$ для $\forall x,\ y \in A$. Наприклад, $|xy| = |x||y|$, $(xy)^c = x^c y^c$ та $c(x+y) = cx + cy$ це гомоморфні функції.

Поєднавши визначення шифрування, яке буде докладніше розглянуто у пункті 3.1.1, та гомоморфізму, сформульованого вище, можна математично визначити гомоморфне шифрування. Нехай $a \cdot b \equiv 1$, тоді формулу шифрування можна подати як $E(x) = x^b \bmod n$, а дешифрування – $D(x) = y^a \bmod n$. При цьому, щоб шифрування було гомоморфним, має виконуватись наступна рівність (за визначанням гомоморфізму): $E(x_1) \cdot E(x_2) = x_1^{\,b} \cdot x_2^{\,b} \bmod n = (x_1 \cdot x_2)^b \bmod n = E(x_1 \cdot x_2)$, де x_1, x_2 – елементи початкового тексту над якими виконуються операції. Таким чином, будь-яку довільну гомоморфну функцію можна побудувати поєднуючи кілька операцій (наприклад, додавання та множення).

Розрізняють частково гомоморфне шифрування та повністю гомоморфне шифрування. Якщо схема шифрування дозволяє виконувати будь-які арифметичні операції над зашифрованими даними (без розкриття початкового тексту), то це схема повністю гомоморфного шифрування.

Якщо ж лише обмежена кількість арифметичних операцій може бути виконана над зашифрованими даними, то такі схеми називаються частково гомоморфними. Перші схеми шифрування з'явились у 1978 році, однак, у той час вони були лише частково гомоморфними. Перша повністю гомоморфна схема шифрування побудована у 2009 році Крейгом Джентрі. З того часу, актуальність досліджень у напрямі гомоморфного шифрування суттєво зросла.

Однією з перших бібліотек, що реалізує різні алгоритми гомоморфного шифрування (серед яких схеми CKKS і BFV) та надає зручний механізм для проведення обчислень з зашифрованими даними, є бібліотека SEAL (Simple Encrypted Arithmetic Library) [22]. Вона була представлена у 2015 році компанією Майкрософт як бібліотека з відкритим кодом. Мовою реалізації бібліотеки є C++, що дозволяє швидше виконувати обчислення, порівняно з іншими високорівневими мовами програмування. У системах, які працюють зі штучним інтелектом, такі бібліотеки намагаються застосовувати, однак, необхідність великої кількості пам'яті та значної обчислювальної потужності пристроїв є суттєвими технічними перешкодами для поширеного використання методів гомоморфного шифрування. Робота з зашифрованими даними потребує в мільйони разів більших ресурсів, порівняно з обробкою оригінальних даних. Проте, у зв'язку з значним розвитком технологій в останні 15 років, є можливість застосовувати методи гомоморфного шифрування у практичних задачах, хоч і з обмеженнями на ресурси.

Таким чином, головна перевага методу гомоморфного шифрування – це виконання обчислень безпосередньо над зашифрованими даними, результатом розшифрування яких буде таке саме значення, як при виконанні обчислень над оригінальними даними. Як наслідок ефективність навчання систем штучного інтелекту буде однаковою як на оригінальних даних, так і на зашифрованих даних. Проте, значний обсяг зашифрованоих даних,

порівняно з початковими, є основним недоліком. Наприклад, 1-у Мб початкових даних може відповідати 10 Гб зашифрованих даних.

Ще одним гомоморфним підходом до збереження приватності є протокол конфіденційного обчислення (MPC – multi-party computation) [23]. Його основним завданням є забезпечення конфіденційності розподілених даних. Припустимо є множина учасників (суб'єктів) комунікації P_1, …, P_n. У кожного з них є приватні дані, які позначимо як x_i. Основною метою комунікації є обчислення спільної функції $y = f\left(x_1,...,x_n\right)$, яка є комбінацією даних учасників. Важливими умовами комунікації суб'єктів є безпека між учасниками та коректність даних. Побудова такого конфіденційного протоколу може здійснюватись різними способами, беручи до уваги кількість учасників та структуру вхідних даних. Одним з найпростіших прикладів такого протоколу є задача обчислення суми значень усіх учасників $\sum_{i=1}^{n} x_i$. Нехай значення кожного учасника не перевищує значення m ($x_i < m$), тоді згенеруємо випадковим чином число $r < m$ для першого учасника. Інші суб'єкти комунікації не повинні знати значення числа r. Далі кожен з суб'єктів комунікації почергово обчислює $\tilde{x}_i = x_i + \tilde{x}_{i-1}$, за умови що $\tilde{x}_0 = r$, та передає наступному суб'єкту. Після обчислення відповідного значення n-м учасником комунікації, загальну суму легко знайти за формулою $y = \sum_{i=1}^{n} x_i = \tilde{x}_n - r$. Варто зауважити, що приватні дані учасників комунікації не розкриваються при використанні такого протоколу. Цей підхід має низку переваг, серед яких готовність до комерційного використання та точність результату, при цьому немає потреби досягати компромісу між безпекою даних та їх корисністю (що характерно анонімізації й диференційній приватності, що були розглянуті вище), оскільки учасники комунікації не взаємодіють з приватними даними інших учасників. Проте, варто зауважити, що протокол конфіденційного

обчислення є обчислювально витратним, оскільки існує постійна потреба в генерації випадкових чисел. Крім цього, затрати на комунікацію є великими, адже необхідно здійснювати передачу інформації між учасниками.

1.3.3. Федеративне навчання

Розроблення систем штучного інтелекту з використанням класичних підходів передбачає наявність набору даних для навчання на одному пристрої. У такому випадку навчання є централізованим. Проте, у статтях [24, 25] дослідники компанії Google запропонували підхід федеративного навчання (Federated Learning). Ідея такого навчання полягає в навчанні моделі на розподілених локальних пристроях (наприклад, телефонах, планшетах, персональних комп'ютерах чи серверах), при якому немає необхідності збору даних на одному централізованому сервері. Підхід федеративного навчання є відмінним, як від класичних підходів, у яких всі екземпляри даних для навчання спершу збираються на одному сервері, так і від інших децентралізованих підходів, локальні вибірки яких рівномірно розподілені між пристроями. Основними принципами федеративного навчання є:

- децентралізація (навчання моделі здійснюється на локальних пристроях чи серверах, дані з яких не передаються на центральний сервер);
- локальне навчання (модель оновлюється локально на кожному пристрої, використовуючи доступні їй дані);
- синхронізація моделі (після локального оновлення моделей на пристроях або серверах вони синхронізуються, обмінюючись лише параметрами моделі, а не навчальними даними);
- приватність та безпека (дані користувачів залишаються на локальних пристроях, що сприяє збереженню приватності та безпеці даних).

Для глибшого розуміння підходу федеративного навчання розглянемо його приклад, що представлений на рис. 1.4 [26]. Припустимо необхідно класифікувати геометричні фігури, серед яких є коло, овал, ромб, квадрат та зірка. За наявності достатньої кількості пристроїв (у прикладі це телефон чи планшет), де буде проходити навчання, можна використати підхід федеративного навчання.

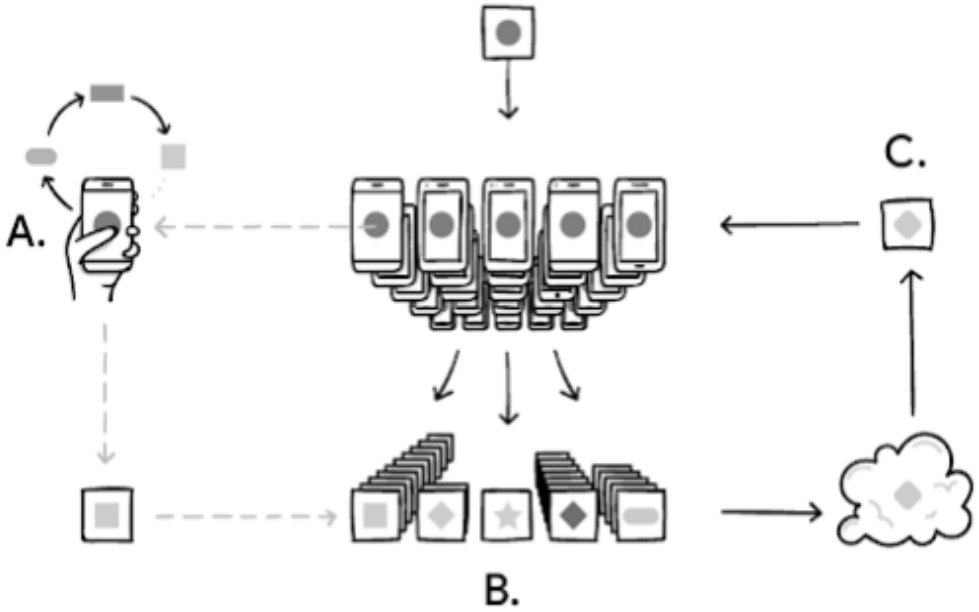


Рис. 1.4. Підхід федеративного навчання

Одним з найважливіших процесів у розглянутому підході є усереднення оновлень моделі, отриманих від різних користувачів (пристроїв). З огляду на це, варто розглянути докладніше алгоритм, за яким можна усереднювати оновлення. Спершу сформулюємо формулу оновлення моделі з використанням скінченної суми, яка використовується як в лінійній та логістичній регресії, так і в алгоритмах нейронних мереж. Її можна подати як обчислення $\min_{w \in R^d} f(w)$, де $f(w) \overset{def}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w)$. У машинному навчанні, функція $f_i(w) = l(w; x_i; y_i)$

представляє собою втрату точності передбачення моделі з параметрами *w* для зразка даних $\{x_i, y_i\}$. Нехай, кількість користувачів, що навчають модель, складає *K*. Позначимо P_k набір точок даних моделі $\{1,\ldots,n\}$, які зберігаються в *k*-го користувача, тоді множина $\{P_k\}_{k=1}^{K}$ позначає розподіл індексів таких даних, а $n_k = |P_k|$. З огляду на такі позначення, функція втрат для спільної моделі-класифікатора визначається як лінійна комбінація локальних функцій втрат моделі [24, 25]:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \overset{def}{=} \sum_{k=1}^{K} \frac{n_k}{n} \cdot \frac{1}{n_k} \sum_{i \in P_k} f_i(w).$$

Варто зазначити, що за такого підходу, локальні моделі можуть показувати як кращі результати передбачення, так і гірші, порівняно з спільною моделлю, оскільки навчання на локальних пристроях може бути нерівномірним (з різною кількістю даних), проте за рахунок усереднень оновлень спільна модель постійно покращується, як і локальні моделі, що показували результати, що гірші середніх. За допомогою федеративного навчання можна створювати моделі гарантуючи приватність, при цьому затримка й використання ресурсів серверу будуть меншими, оскільки значна частина навчання здійснюватиметься на локальних пристроях. Цей підхід дозволяє не лише оновлювати спільну модель, а й вдосконалювати з мінімальною затримкою локальну модель.

Підхід федеративного навчання не вирішує всі задачі систем машинного навчання, оскільки з його допомогою не можна, наприклад, навчити модель розпізнаванню різних порід собак навчаючи модель на маркованих прикладах. Крім цього, значна частина даних є централізованою й зберігається у хмарних сховищах, тож в таких випадках цей підхід застосовувати недоцільно. Зважаючи на це, виділимо класи задач, в яких використання федеративного навчання є ефективним. Першим класом задач є сценарії, в яких навчання здійснюється у реальному часі на багатьох

пристроях (мобільних телефонах, планшетах, персональних комп'ютерах) з використанням оригінальних даних. Ще одним класом задач, де застосування федеративного навчання є ефективним, є навчання на чутливих до конфіденційності або великих за обсягом (у порівнянні з вагами моделі) даних. У такому разі недоцільно зберігати дані для навчання в дата центрах, винятково з метою навчання моделі. Також федеративне навчання можна використовувати для навчання моделей з контрольними завданнями, мітки для яких можна створити, внаслідок взаємодії з користувачем. Прикладом такого сценарію є системи що спрямовані на захист від ботів, які використовують технологію reCAPTCHA, де необхідно визначити наявність певних елементів на рисунку.

Одним з прикладів успішного використання федеративного навчання є інтелектуальна клавіатура Google Gboard, яка використовується на пристроях з операційною системою Android [26]. Інтелектуальна клавіатура передбачає запит користувача й, локально працюючи з контекстом, оновлює модель залежно від того чи обрав користувач запропонований варіант. Після цього, беручи до уваги історію запитів користувача, формується оновлення моделі, результатом чого є покращення моделі, що передбачає запити користувача.

Ще одним прикладом використання підходу федеративного навчання є штап Owkin [27]. У цьому програмному забезпеченні розроблені моделі навчання біомедичних засобів, в основі яких є процес збору неоднорідних даних від медичних установ та фармацевтичних компаній. Такі дані збираються з використанням федеративного навчання та технології розподілених реєстрів (найвідоміший тип такого реєстру – blockchain).

1.3.4. Порівняльний аналіз збереження приватності в машинному навчанні

Підсумуємо розглянуту інформацію, провівши порівняльний аналіз методів збереження приватності в машинному навчанні за п'ятьма критеріями: складність, практичність, потреба у значній кількості даних для

використання методу, надійність (що показує наскільки значним є рівень приховування даних) та точність системи штучного інтелекту (з використанням методу збереження приватності). У табл. 1.5 представлені результати такого аналізу [28, 29].

Таблиця 1.5

Порівняльний аналіз методів збереження приватності

Метод	Складність	Практич-ність	Потреба у значній кількості даних	Надійність	Висока точність системи
Генерація синтетичних даних	+	+	+	+	+/–
Анонімізація даних	–	+	–	–	+
Диференційна приватність	+/–	+	+	+/–	+/–
Гомоморфне шифрування	+	–	–	+	+
Федеративне навчання	–	+/–	+	+	+

Отже, методи генерації синтетичних наборів даних практичні і надійні, однак задача побудови генератора таких даних є складаною та у більшості випадків потребує значної кількості вхідних даних (параметрів очікуваних зразків даних) для побудови достатньо точної системи штучного інтелекту. Метод анонімізації даних є досить простим, практичним й не вимагає великих обсягів вхідних даних, проте його надійність є недостатньою, тож його використання не є достатньою умовою для забезпечення високого рівня збереження приватності даних. Метод

диференційної приватності є практичним й потребує значної кількості вхідних даних, однак його надійність залежить від кількості шуму, що застосовується до даних: якщо шуму багато, то метод є надійним, але передбачення систем штучного інтелекту на таких даних є неточними; якщо шуму мало, то метод є точним, але ненадійним. Гомоморфне шифрування це надійний метод, який може використовуватись для побудови високоточних систем, однак, він є складним (у тому числі обчислювально витратним) і обмежується класом задач, де його можна застосовувати. Метод федеративного навчання є надійним і точним методом, він не розповсюджує локальні дані навчання, однак, його можна використовувати, якщо є принаймні кілька незалежних користувачів, які мають достатньо даних для навчання.

1.4. Аналіз вимог до програмного забезпечення

Розробленню програмного забезпечення передує збір та аналіз вимог. Вимоги дозволяють перевірити чи розроблене програмне забезпечення правильно виконує поставлену задачу, є надійним та містить всі заплановані функції. Ще одним важливим завданням у інженерії вимог є пріоритезація вимог, яка дозволяє відкинути несуттєві вимоги й зосередитись на найголовніших.

Вимоги поділяються на функціональні й нефункціональні. Функціональні вимоги, пов’язані з функціональним аспектом програмного забезпечення, й описують, що має робити програмна система. Нефункціональні вимоги – це неявні або очікувані характеристики програмного забезпечення, вони накладають обмеження на те, як система виконуватиме свої функції робитиме.

Базуючись на проведеному аналізі загроз приватності даних у системах аналізу даних та штучного інтелекту та аналізі алгоритмічно-програмних методів збереження приватності, сформулюємо вимоги до

програмного забезпечення захисту приватних наборів даних у задачах класифікації.

Сформулюємо функціональні вимоги до програмного забезпечення відсортовані за зменшенням пріоритетності:

- система повинна забезпечувати збереження приватності користувацьких даних, шляхом використання децентралізованих підходів до розробки систем штучного інтелекту;
- система повинна надавати можливість попередньої обробки даних, шляхом функціонального шифрування;
- система повинна надавати можливість вирішення задачі класифікації алгоритмом CNN;
- система повинна надавати можливість вирішення задачі класифікації лінійним класифікатором;
- система повинна надавати можливість зменшення розмірності вхідних даних.

Для нефункціональних вимог до програмного забезпечення використаємо класифікацію FURPS+. Згідно з нею, виділяються такі категорії вимог:

- F (Functionality) – функціональні вимоги (наведені вище);
- U (Usability) – вимоги щодо зручності використання (людський фактор, механізми допомоги користувачу, документація);
- R (Reliability) – вимоги щодо надійності (частота відмов, відновлюваність, передбачуваність);
- P (Performance) – вимоги щодо продуктивність (час відгуку, пропускна здатність, точність, доступність, використання ресурсів);
- S (Supportability) – вимоги щодо можливості підтримки (адаптованість, підтримка, інтернаціоналізація, конфігурованість).

Сформулюємо нефункціональні вимоги до програмного забезпечення за класифікацію FURPS+:

- U (Usability): система повинна підтримувати англійську мову;
- R (Reliability): система повинна надавати зберігати ваги навчених моделей;
- P (Performance): система повинна опрацювати запит користувача на класифікацію одного екземпляру даних, не більш як за 10 секунд;
- S (Supportability): система повинна мати модульну архітектуру (зокрема, модулі функціонального шифрування та класифікації даних).

1.5. Висновки

Розглянуто основні етичні аспекти використання систем штучного інтелекту та проблеми до яких може призвести їх ігнорування. Визначено, що завдання забезпечення захисту приватних наборів даних є важливим та актуальним, зважаючи на швидкість впровадження таких систем у більшість сфер життя.

Проаналізовано типи атак на системи штучного інтелекту, серед яких атаки інверсії, отруєння й логічного висновку, а також методи протидії атакам, що загрожують витоку приватних даних. Проведено комплексний аналіз методів збереження приватності, проаналізовано їх переваги та недоліки. Докладно розглянуто методи генерації синтетичних даних, попередньої обробки вхідних даних (анонімізацію даних, диференційну приватність, гомоморфне шифрування) і метод федеративного навчання. Розглянуті методи вирішують частину проблем приватності даних, але кожен з них має свої переваги та недоліки, які треба враховувати при вирішенні задачі. Розроблено функціональні та нефункціональні вимоги до програмного забезпечення захисту приватних наборів даних у задачах класифікації.

Отже, базуючись на проведеному аналізі, можна зробити висновок, що розроблення альтернативних і модифікація існуючих методів збереження приватності, що дозволять мінімізувати розглянуті недоліки, є актуальними напрямками подальших досліджень.

2. РОЗРОБЛЕННЯ АЛГОРИТМІЧНИХ МЕТОДІВ ВИКОНАННЯ ОПЕРАЦІЙ НАД ЕЛЕМЕНТАМИ ПОЛЯ $GF(p^m)$ У ПОЛІНОМІАЛЬНОМУ ТА НОРМАЛЬНОМУ БАЗИСАХ

2.1. Аналіз скінченних алгебраїчних структур у прикладних задачах

2.1.1. Математичні аспекти арифметики полів Галуа

Математичні аспекти полів Галуа ґрунтуються на теорії груп. Базовим поняттям зазначеної теорії є алгебраїчна структура (система) – непорожня множина з визначеним на ній набором операцій та відношень, що задовольняють певній системі аксіом [30]. При цьому множину з порожнім набором операцій та відношень вважають виродженою системою. Прикладами алгебраїчних систем є групоїд, моноїд, група, кільце, поле.

Найпростішою алгебраїчною структурою є групоїд (магма), який є множиною з визначеною на ній однією бінарною операцією (традиційно – множення), відносно якої ця множина є замкненою. Визначивши у групоїді нові операції та/або відношення, отримують різні алгебраїчні структури, серед яких найбільш вивченими є: квазігрупи, напівгрупи, групи і т.д.

Напівгрупа – групоїд, для якого задане відношення асоціативності: $a\cdot\left(b\cdot c\right)=\left(a\cdot b\right)\cdot c$. Якщо в такій множині задано нейтральний елемент для бінарної операції, тобто елемент, який при застосуванні цієї операції до іншого елемента множини його не змінить $a\cdot e=e\cdot a=a$, то така алгебраїчна структура матиме назву моноїд [31].

Група – моноїд, для кожного елементу якого визначено обернений елемент a^{-1}. Характерною особливістю оберненого елементу є те, що результатом виконання двомісної (бінарної) операції з початковим елементом є нейтральний елемент. Тобто $a\cdot a^{-1}=a^{-1}\cdot a=e$. Якщо операції групи задовольняють відношенню комутативності $a\cdot b=b\cdot a$, то вона називається абелевою [32].

Кільце – це абелева група, на якій визначено ще одну асоціативну бінарну операцію (найчастіше дві визначені операції – додавання та множення). Для цієї алгебраїчної структури обов'язковим є виконання дистрибутивного закону: $a \cdot (b + c) = a \cdot b + a \cdot c$, $(b + c) \cdot a = b \cdot a + c \cdot a$. Прикладом кільця є цілі числа. Якщо в кільці операція множення комутативна, то воно називається комутативним.

Тіло – кільце з одиницею (двохсторонній нейтральний елемент операції множення), в якому всі ненульові елементи мають обернений.

Поле – комутативне кільце, яке є тілом. Підсумовуючи, це множина, у якій визначено дві бінарні операції (наприклад, додавання та множення) та обернені їм (віднімання та ділення, крім ділення на нульовий елемент), а також операції задовольняють законам асоціативності, комутативності й дистрибутивності. Поля бувають скінченні (поле Галуа і його розширення) та нескінченні (множини раціональних, дійсних та комплексних чисел).

Отже, поле Галуа (позначається $GF(q)$) – це поле, яке складається зі скінченної кількості елементів. Кількість його елементів називається порядком q. Порядок поля є степенем будь-якого простого числа, тобто $q = p^m$, де p – просте число, а m – будь-яке натуральне. За такого позначення число p є характеристикою скінченного поля [30]. Найпростіше поле Галуа отримують при $m = 1$ – це поле лишків за модулем простого числа p: $GF(p) = \{0,1,\ldots,p-1\}$. У такому випадку 1 – нейтральний елемент відносно операції множення, а 0 – відносно додавання. Якщо $m > 1$, то поле $GF(p^m)$ називається розширенням поля Галуа $GF(p)$. Операції в такій алгебраїчній структурі виконуються за модулем незвідного многочлена (поліному, який не дорівнює константі та який не можна розкласти на множники у заданому полі). Для розширеного поля Галуа він є аналогом простого числа за модулем якого виконуються операції в $GF(p)$. У такому разі елементи поля представляються поліномами $\mathbf{a} = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1 x + a_0$, де

$a_i \in GF(p)$. Множина всіх таких поліномів називається кільцем многочленів. Елементи поля можна представляти використовуючи різні подання, зокрема найбільш поширеними є степеневе, поліноміальне, векторне й логарифмічне [33, 34]. Для побудови поля $GF(p^m)$, спочатку визначають твірний (примітивний) елемент α, а далі шляхом ділення елементів α^i на незвідний поліном (або підстановок з нього) знаходять шукану множину. Алгоритм побудови поля $GF(p^m)$ наведений на рис. 2.1.

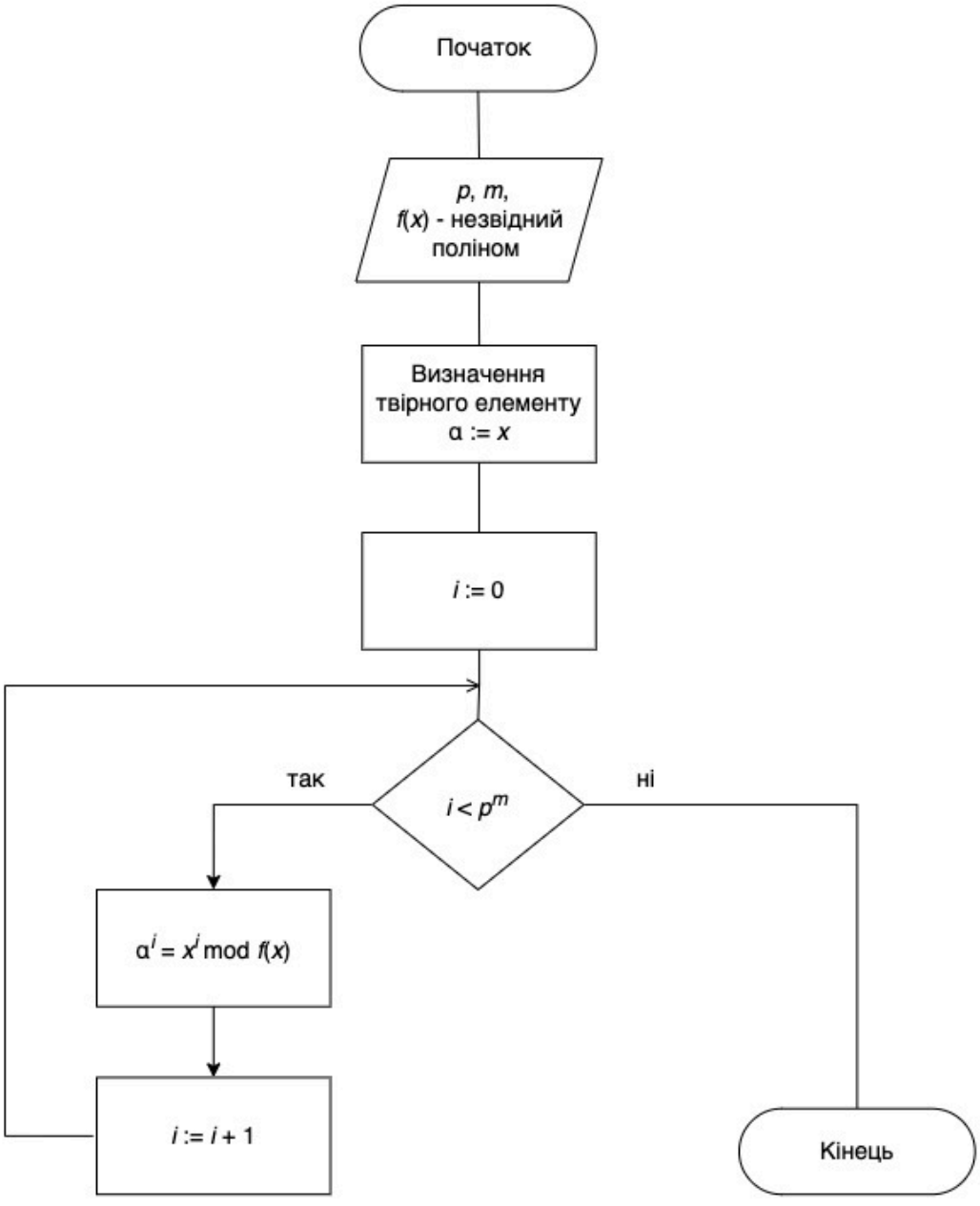


Рис. 2.1. Блок-схема алгоритму побудови поля $GF(p^m)$

Розглянемо приклад побудови поля Галуа. Нехай $p = 2$, $m = 3$, а незвідний многочлен – $1 + x + x^3$. Тоді для поля $GF(p^m)$ згідно алгоритму:

1. Визначаємо примітивний елемент. Для поліноміального базису він традиційно дорівнює $\boldsymbol{\alpha} = x = (0;1;0)$.

2. $\forall i = \overrightarrow{0 \ldots 2^3 - 1} = \overrightarrow{0 \ldots 7}$ обчислюємо α^i:

$$\boldsymbol{\alpha}^{\mathbf{0}} = 1 = (1;0;0),$$

$$\boldsymbol{\alpha}^{\mathbf{2}} = x^2 = (0;0;1),$$

$$\boldsymbol{\alpha}^{\mathbf{3}} = x^3 = x^3 + x^3 + x + 1 = x + 1 = (1;1;0),$$

$$\boldsymbol{\alpha}^{\mathbf{4}} = x^4 = x^3 \cdot x = (x+1) \cdot x = x^2 + x = (0;1;1),$$

$$\boldsymbol{\alpha}^{\mathbf{5}} = x^5 = x^4 \cdot x = (x^2 + x) \cdot x = x^3 + x^2 = x^2 + x + 1 = (1;1;1),$$

$$\boldsymbol{\alpha}^{\mathbf{6}} = x^6 = x^5 \cdot x = (x^2 + x + 1) \cdot x = x^3 + x^2 + x = x^2 + 1 = (1;0;1),$$

$$\boldsymbol{\alpha}^{\mathbf{7}} = x^7 = x^6 \cdot x = (x^2 + 1) \cdot x = x^3 + x = 1 = (1;0;0) = \boldsymbol{\alpha}^{\mathbf{1}}.$$

Отже, елементи поля $GF(2^3)$ мають вигляд:

1	x^1	x^2	Значення
0	0	0	0
1	0	0	1
0	1	0	2
1	1	0	3
0	1	1	4
1	1	1	5
1	0	1	6
1	0	0	7

Елементи скінченних полів можна подавати в різних базисах [35]. Основним базисом для виконання обчислень у полі Галуа є поліноміальний. У цьому базисі елементи представляються степенями елемента t: $A = \{1, t, t^2, \ldots, t^{m-1}\}$. Альтернативним способом представлення елементів є

нормальний базис, де вони представлені наступним чином $B = \left\{ \alpha_0 = t, \alpha_1 = t^p, \alpha_2 = t^{p^2}, \ldots, \alpha_{m-1} = t^{p^{m-1}} \right\}$.

2.1.2. Застосування полів Галуа в системах захисту та передачі інформації

Поля Галуа найбільш широко застосовуються у криптографії, при виправленні помилок передачі даних, у вейвлетних перетвореннях та для побудови фільтрів у цифровій обробці. Це спричинено тим, що кожен байт в комп'ютері можна представити як вектор в скінченному полі, а операції з використанням машинної арифметики реалізувати дуже просто. Також однією зі сфер застосування арифметики полів Галуа є методи збереження приватності даних у сучасних інформаційних системах, зокрема при побудові систем аналізу даних та штучного інтелекту. Наприклад, у гомоморфному шифруванні (наприклад, з використанням схеми Ель-Гамаля) та безпечних багатосторонніх обчисленнях [36-41]. Проаналізуємо використання скінчених полів при обробці даних і розглянемо особливості їх застосування.

Advanced Encryption Standard

У 2001 році на заміну DES Реймен і Даймен придумали більш складний алгоритм Rijndael, який після перемоги на відкритому конкурсі отримав назву Advanced Encryption Standard (AES). Цей алгоритм лежить в основі схем гомоморфного шифрування, таких як TFHE (fast fully homomorphic encryption) та CKKS (Cheon-Kim-Kim-Song).

Поле Галуа $GF(2^8)$ є основою для обчислень, які виконуються в AES з 128-бітним ключем. Перед шифруванням даних потрібно сформувати матрицю байтів (розбити дані на стани). Основна частина алгоритму шифрування складається з декількох етапів, а саме SubBytes, ShiftRows, MixColumns та AddRoundKey [36].

Розбиття байтів на стани – важливий процес для подальшого виконання операцій, але при цьому криптографічного значення в ньому

нема. У алгоритмі AES дані (повідомлення, що шифрується) розбивають на матрицю байтів із заздалегідь визначеним розміром для того, щоб кожен стан можна було шифрувати незалежно від інших. У AES з 128-бітним ключем, це матриця з 4 рядками і 4 стовпцями, де кожен запис – це 1 байт або 8 біт, що в сумі становить 16 байтів.

Етап SubBytes складається з двох кроків. Під час першого кроку кожен байт з матриці станів замінюється на мультиплікативно обернений елемент (якщо він дорівнює 0, то його не змінюють, оскільки такий байт немає мультиплікативно оберненого елемента). У другому кроці для кожного стану виконують афінне перетворення: $\tilde{\mathbf{x}} = A \cdot \mathbf{x} + \mathbf{b}$, де

$$A = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}, \; b = 99_{10} \left(\mathbf{b} = 01100011_2 \right).$$

У наступній фазі алгоритму ShiftRows виконується циклічний зсув рядків матриці станів вліво на кількість байт, що дорівнює номеру рядка. Таким чином, якщо елементи початкової матриця A позначити як $a_{ij}, i = \overrightarrow{0 \ldots 3}, \; j = \overrightarrow{0 \ldots 3}$, то в результаті зазначеної операції отримаємо матрицю $\tilde{A}$:

$$\tilde{A} = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{11} & a_{12} & a_{13} & a_{10} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{33} & a_{30} & a_{31} & a_{32} \end{pmatrix}.$$

Після цього виконується наступна стадія алгоритму – MixColumns, де виконується лінійна трансформація стовпців матриці станів. Кожен стовпець представляється поліномом 3-ої степені, який множиться на

заздалегідь визначений многочлен $c(x) = 3x^3 + x^2 + x + 2$. Результат цієї операції береться за модулем незвідного поліному $x^4 + 1$, оскільки всі обчислення виконуються в полі $GF(2^8)$.

Останнім етапом циклу є AddRoundKey. Це найважливіша фаза, яка забезпечує унікальність шифрування. Результат виконання повністю залежить від ключа користувача. На цьому етапі використовуючи алгоритм Rijndael's Key Schedule з основного ключа формується підключ, який має такий самий розмір як і стан. Як тільки цей підключ буде визначено, він додається до стану. Наступним кроком є розширення основного ключа для формування достатньої кількості підключів.

У ході алгоритму виконують 10 циклів, які складаються з розглянутих вище етапів. Для AES з ключами, що складаються з 192 і 256 біт, кількість циклів буде 12 і 14 відповідно [37].

Отже, в алгоритмі AES найбільш використовуваними операціями над елементами поля Галуа є множення й обчислення мультиплікативно оберненого елемента.

Криптографічні схеми гомоморфного шифрування

Основними схемами гомоморфного шифрування, в основі яких лежить арифметика скінченних полів є наступні [37]: CKKS, BFV, ElGamal.

Схема CKKS (Cheon-Kim-Kim-Song) – схема дещо гомоморфного шифрування (SHE), що використовується для арифметики наближених чисел [38]. У процесі її використання повідомлення *m* (вектор значень, на якому необхідно виконати певні обчислення), спочатку кодується в поліном відкритого тексту $p(X)$, а потім шифрується за допомогою відкритого ключа. Ця схема працює з поліномами, оскільки вони забезпечують хороший компроміс між безпекою та ефективністю порівняно з обчисленнями на векторах. Після того, як повідомлення *m* зашифровано в *c*, схема CKKS надає механізм виконання певної кількості операцій над зашифрованими даними (зокрема, додавання, множення та обертання). Відповідно результат

дешифрування після виконання операцій, буде ідентичним результату виконання операцій на початкових даних.

Схема BFV (Brakerski/Fan-Vercauteren) – це схема повністю гомоморфного шифрування (FHE), заснована на проблемі кільцевого навчання з помилками (RLWE) [37]. У такій схемі зашифровані тексти містять помилку, яка збільшується з кожною гомоморфною операцією і повинна залишатися нижче певного порогу правильності.

Схема ElGamal – криптосистема частково гомоморфного шифрування (PHE) з відкритим ключем, яка використовує асиметричне шифрування для шифрування повідомлень [39, 40]. Вона використовується для безпечної комунікації й цифрових підписів.

Коди Ріда-Соломона

Коди Ріда-Соломона використовуються для виправлення помилок, які виникають після втрати пакетів даних, під час їх передачі в комп'ютерній мережі. Зокрема, одним з поширених прикладів їх застосування є виправлення помилок при зчитуванні QR-кодів.

Кодування і декодування даних у кодах Ріда-Соломона відбувається з використанням полів $GF(p^m)$. Значення m – це розмір слова кодування у даному контексті. Кожен пакет даних розподіляється на слова довжиною m бітів для кожного з яких обчислюється контрольне значення. У зв'язку з цим для параметра m використовують значення, які є степенями двійки (8, 16, 32).

Потік пакетів, що передається в комп'ютерній мережі, складається з двох частин: пакетів даних і пакетів для перевірки, які мають однакову фіксовану довжину [41]. Група прямої корекції помилок складається з n пакетів даних та з k контрольних пакетів, при цьому $n+k \le 2^m$. У такому випадку, успішне отримання будь-яких n пакетів (які є пакетами для перевірки й пакетами з даними) є достатньою умовою для відновлення початкової інформації (з n пакетів з даними).

Алгоритм виправлення помилок (кодування й декодування повідомлень) потребує створення матриці з $n+k$ рядків та n стовпців. Квадратна частина матриці з n рядків – одинична матриця I, а будь-які n з $n+k$ рядків є лінійно незалежними. У зв'язку з цим, будь-який набір з n рядків має обернену матрицю. Шукану матрицю отримують з матриці Вандермонда, яка має вищенаведену властивість лінійної незалежності будь-яких n з $n+k$ рядків. Для будь-яких n та k матриця має наступний вигляд:

$$\begin{pmatrix} 0^0 & 0^1 & 0^2 & \dots & 0^{n-1} \\ 1^0 & 1^1 & 1^2 & \dots & 1^{n-1} \\ 2^0 & 2^1 & 2^2 & \dots & 2^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ \left(2^m-1\right)^0 & \left(2^m-1\right)^1 & \left(2^m-1\right)^2 & \dots & \left(2^m-1\right)^{n-1} \end{pmatrix}.$$

Ця матриця задовольняє другу властивість, але перші n її рядків не утворюють одиничну матрицю, тому ця умова досягається за допомогою лінійних перетворень, які не порушують властивість лінійної незалежності будь-яких n рядків. Отриману матрицю позначають літерою D.

Алгоритм корекції помилок передачі даних з використанням матриці D є достатньо простим. Припустимо, що отримано n пакетів, включаючи дані та пакети для перевірки. Необхідно вилучити n рядків з матриці D, які відповідають кількості отриманих пакетів. Цю матрицю традиційно позначають як D'. Оскільки рядки цієї матриці є лінійно незалежними, то існує обернена матриця в полі $GF(2^m)$. Наступним кроком алгоритму є обчислення матриці D'^{-1}. Добуток цієї матриці D'^{-1} на змішане слово (яке складається з даних і контрольних слів) дозволяє відновити шукані слова даних $d_0,\dots,d_{n-1}$.

Криптографія на еліптичних кривих

Одним з основних розділів сучасної криптографії є еліптична криптографія. Ця галузь є основою для побудови електронного цифрового

підпису, криптографічних протоколів та сертифікатів. Обчислення в полі Галуа є основою для еліптичних кривих. Загальна формула еліптичної кривої *E* [34]:

$$E:\ y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6.$$

У криптографії використовують спрощені формули для задання кривої залежно від поля $GF(p^m)$. Так, при $m = 1$, тобто для поля $\mathbb{Z}_p$ лишків за модулем простого непарного числа *p*, використовують криву:

$$E:\ y^2 = x^3 + ax + b \pmod p,$$

де *a*, *b* – точки еліптичної кривої, які є елементами скінченного поля $GF(p)$ і задовольняють рівність $4a^3 + 27b^2 \neq 0 \pmod p$.

Таким чином, група точок еліптичної кривої *E* над полем $GF(p)$ – це безліч пар $(x; y)$, які лежать на *E* і об'єднані з нульовим елементом:

$$E\left(GF(p)\right) = 0 \cup \left\{(x; y) \in GF(p) \middle| y^2 \equiv x^3 + ax + b \pmod p\right\}.$$

Іншим важливим полем для еліптичної криптографії є $GF(2^m)$. Для такого поля використовують сингулярні $y^2 + ay = x^3 + bx + c$ й несингулярні $y^2 + axy = x^3 + bx^2 + c$ криві. Особливість сингулярних кривих полягає в простоті обчислення порядку поля Галуа, порівняно з несингулярними.

Основними операціями над еліптичними кривими є додавання точок та множення їх на скаляр. Залежно від виду еліптичної кривої використовують різні формули виконання операцій на ній. У табл. 2.1 наведено формули для додавання точок $P = (x_P; y_P)$ і $Q = (x_Q; y_Q)$, а також для множення точки $P = (x_P; y_P)$ на 2 (для виконання операції множення точки на скаляр $nP = P + P + \ldots + P$ використовують алгоритм, що складається з операцій подвоєння й додавання точок).

Отже, операції додавання й Фробеніуса є найбільш використовуваними в криптографії на еліптичних кривих.

Таблиця 2.1

Формули виконання операцій над еліптичною кривою

<table>
<tr><th>Еліптична крива</th><th>Додавання точок</th><th>Множення точки на 2</th></tr>
<tr><td rowspan="2">$y^2 = x^3 + Ax + B \pmod p$</td><td>$m = \frac{y_P - y_Q}{x_P - x_Q}$</td><td>$m = \frac{3{x_P}^2 + a}{2y_P}$
$P = Q$</td></tr>
<tr><td colspan="2">$x_R = m^2 - x_P - x_Q$
$y_R = y_P + m \cdot (x_R - x_P)$</td></tr>
<tr><td rowspan="2">$y^2 + axy = x^3 + bx^2 + c \pmod{2^m}$</td><td>$m = \frac{y_P + y_Q}{x_P + x_Q}$</td><td>$m = \frac{{x_P}^2 + y_P}{x_P}$
$P = Q$</td></tr>
<tr><td colspan="2">$x_R = m^2 + m + x_P + x_Q + b$
$y_R = (x_P + x_R) \cdot m + x_R + y_P$</td></tr>
</table>

2.2. Аналіз алгоритмічних методів виконання операцій над елементами поля $GF(p^m)$

2.2.1. Алгоритми переходу між базисами

Існують різні підходи для перетворення елементів поля Галуа з одного базису в інший. Розглянемо класичний алгоритм з використанням оберненої матриці для перетворення з поліноміального базису в нормальний [35, 42], блок-схему якого представлено на рис. 2.2:

1. Виразити базисні елементи нормального базису $B = \left\{\alpha_0 = t, \alpha_1 = t^p, \alpha_2 = t^{p^2}, \ldots, \alpha_{m-1} = t^{p^{m-1}}\right\}$ через базисні елементи поліноміального $A = \left\{1, t, t^2, \ldots, t^{m-1}\right\}$.

2. Сформувати матрицю переходу (елементи нормального базису розміщуються в стовпчик).
3. Обчислення оберненої матриці (математичні операції виконуються за правилами поля $GF(p^m)$).
4. Перетворити елемент, що заданий у поліноміальному базисі у нормальний базис за допомогою формули $\mathbf{x}_B = M^{-1} \cdot \mathbf{x}_A$.

Примітка. Якщо на 3-му кроці алгоритму обернена матриця не існує, то поліном не є нормальним. Отже, побудувати нормальний базис неможливо. У такому разі доцільно обрати інший незвідний поліном у заданому полі й почати алгоритм спочатку.

Алгоритм перетворення з нормального базису в поліноміальний буде відрізнятись лише останнім кроком, де замість формули $\mathbf{x}_B = M^{-1} \cdot \mathbf{x}_A$ використовується – $\mathbf{x}_A = M \cdot \mathbf{x}_B$.

Розглянемо приклад переходу з поліноміального базису в нормальний. Нехай потрібно перевести елемент $(1;1;0)$ поля $GF(2^3)$ з незвідним многочленом $x^3 + x^2 + 1$ у нормальний базис.

Елементи поля $GF(2^3)$ в поліноміальному базисі мають вигляд:

1	t^1	t^2	Значення
0	0	0	0
1	0	0	1
0	1	0	2
1	1	0	3
0	0	1	4
1	0	1	5
0	1	1	6
1	1	1	7

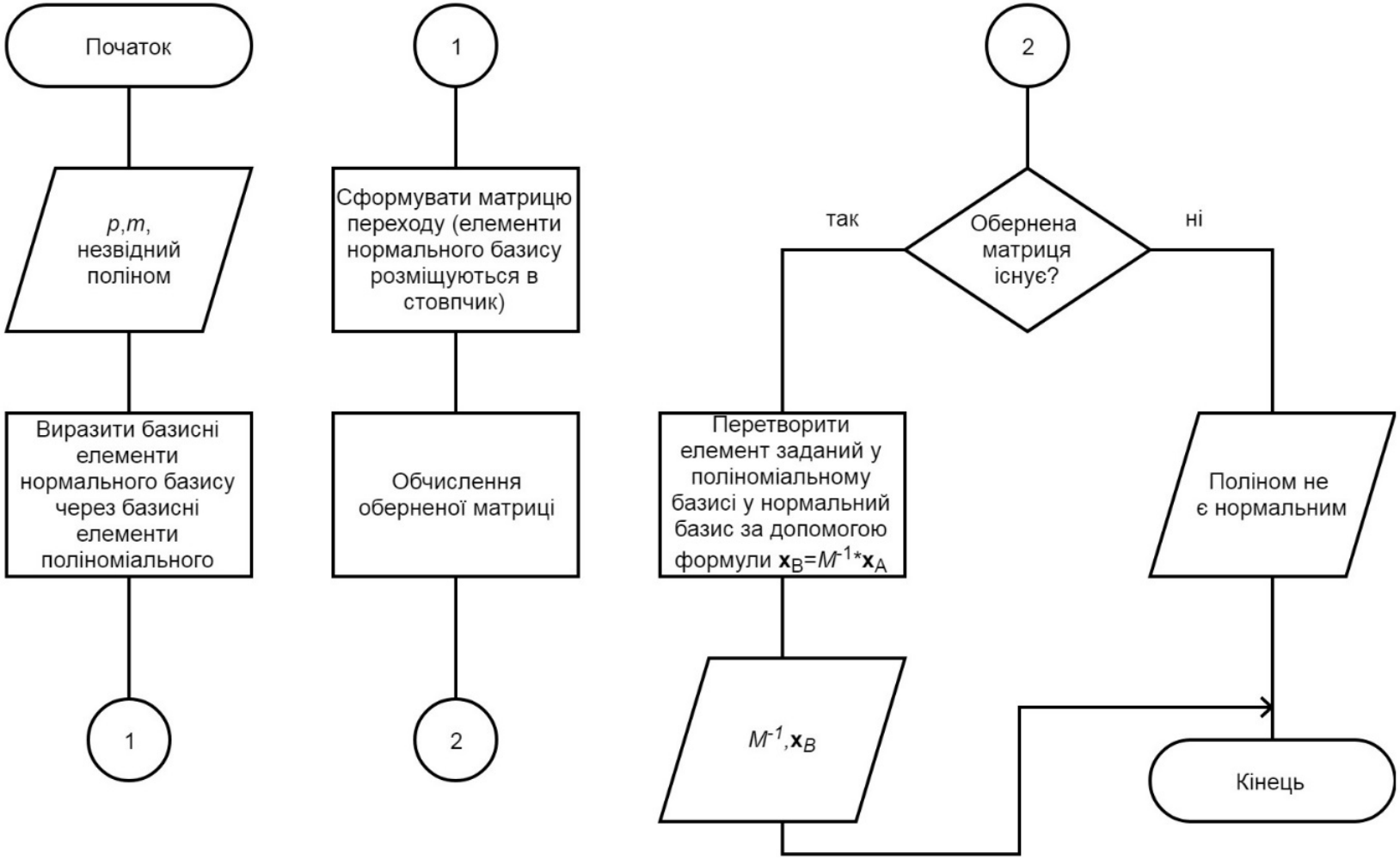


Рис. 2.2. Блок-схема алгоритму переходу між базисами

Базисними елементами для поліноміального базису є 1, t, t^2.

Згідно розглянутого алгоритму:

1. Виражаємо базисні елементи нормального базису через базисні елементи поліноміального.

   Елементи нормального базису мають вигляд:

$$B=\left\{\alpha_0=t,\alpha_1=t^2,\alpha_2=t^{2^2}=t^4\right\}.$$

   Для елементів нормального базису, що мають степінь більшу за $m-1$, значення в поліноміальному базисі можна знайти двома способами.

   1-й спосіб. Як остачу від ділення елемента в нормальному базисі на незвідний поліном:

$$\begin{array}{l|l} t^4+0t^3+0t^2+0t+0 & t^3+t^2+1 \\ \hline t^4+\ t^3+0t^2+\ t & t+1 \\ \hline \quad t^3+0t^2+\ t+0 & \\ \hline \quad t^3+\ t^2+0t+1 & \\ \hline \quad t^2+\ t+1 & \end{array}$$

   2-й спосіб. За допомогою підстановок (з незвідного полінома):

$$t^4=t\cdot t^3=t\cdot(t^2+1)=t^3+t=t^2+t+1.$$

   Тоді $t^4=t^2+t+1.$

   Отже, $B=\left\{\alpha_0=t,\alpha_1=t^2,\alpha_2=t^2+t+1\right\}$. У векторному поданні базисні елементи нормального базису мають вигляд: $(0;1;0)$, $(0;0;1)$, $(1;1;1)$.

2. З базисних елементів нормального базису сформуємо матрицю переходу:

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

3. Обчислення оберненої матриці

Можна використовувати будь-який метод обчислення оберненої матриці, але при цьому треба виконувати всі операції над елементами матриці у полі $GF(p)$. Для програмування зручно використовувати метод Гауса-Жордана, тому використаємо його й у цьому прикладі.

$$\left(\begin{array}{ccc|ccc} 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right) \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right) \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right) \sim$$

$$\left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right) \sim \left(\begin{array}{ccc|ccc} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right).$$

Отже, $M^{-1} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$.

4. Знайдемо шуканий елемент $(1;1;0)$ в нормальному базисі:

$$\mathbf{x}_B = M^{-1} \cdot \mathbf{x}_A = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$$

Знайдемо представлення всіх елементів поліноміального базису в нормальному за формулою $\mathbf{x}_B = M^{-1} \cdot \mathbf{x}_A$:

Елемент у поліноміальному базисі			Обернена матриця	Елемент у нормальному базисі		
1	t^1	t^2		t^1	t^2	$t^4 = t^2 + t + 1$
0	0	0	$\begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$	0	0	0
1	0	0		1	1	1
0	1	0		1	0	0
1	1	0		0	1	1
0	0	1		0	1	0
1	0	1		1	0	1
0	1	1		1	1	0
1	1	1		0	0	1

2.2.2. Порівняльний аналіз методів виконання операцій у поліноміальному та нормальному базисі

Основними операціями над елементами поля Галуа є: додавання, множення, обчислення мультиплікативно оберненого елемента, ділення, піднесення до степеня та операція Фробеніуса [30, 43-49].

Розглянемо виконання всіх зазначених операцій залежно від базису.

Додавання

Це найпростіша операція в полі Галуа. Вона виконується за наступною формулою [30]:

$$c = (a + b) \bmod p .$$

Обчислювальна складність виконання даної операції $O(m)$.

При $p = 2$ (у двійковому полі), виконання операції додавання значно спрощується. У такому випадку достатньо виконати побітову операцію XOR:

$$c = a \oplus b .$$

Аналогічно до додавання, операція віднімання виконується в обох базисах однаково з такою ж обчислювальною складністю $O(m)$, за формулою:

$$c = (a - b) \bmod p.$$

Відповідно для її виконання у двійковому полі також достатньо виконати бінарну операцію XOR:

$$c = a \oplus b.$$

Множення

У поліноміальному базисі операція множення в скінченному полі – це множення за модулем незвідного полінома, тобто [50, 51]:

1. За формулою $c = a \cdot b$ виконуємо множення елементів у поліноміальному базисі. При цьому, якщо розглянути множення в стовпчик, то порівняно з традиційними діями, не відбувається жодних переносів значень на наступний розряд, а дії над кожним розрядом виконуються за модулем p.
2. Обчислення остачі від ділення отриманого результату на незвідний поліном.

Розглянемо приклад. Нехай у полі $GF(2^3)$ з незвідним многочленом $x^3 + x^2 + 1$ необхідно помножити два полінома $\mathbf{a} = x + 1$ і $\mathbf{b} = x^2 + x$.

Представимо поліноми у координатному вигляді: $\mathbf{a} = (1;1;0)$, $b = (0;1;1)$.

Тоді згідно алгоритму:

1. Множимо поліноми:

$$\mathbf{a} \cdot \mathbf{b} = (x + 1) \cdot (x^2 + x) = x^3 + x^2 + x^2 + x = x^3 + x.$$

2. Обчислюємо остачу від ділення на незвідний многочлен:

$x^3+0x^2+\ x+0$	x^3+x^2+1
$x^3+\ x^2+0x+1$	1
$x^2+\ x+1$	

Отже, $(a\cdot b)\bmod f(x)=x^2+x+1$, де $f(x)$ – незвідний многочлен.

Множення у поліноміальному базисі за вищенаведеним алгоритмом має обчислювальну складність $O(m^2)$. Існують модифіковані алгоритми множення, що мають обчислювальну складність $O\left(m\log m\log\log m\right)$.

Операція множення у нормальному базисі виконується за допомогою додаткової матриці для множення *M*. Алгоритм у такому разі можна поділити на два етапи: обчислення матриці *M* та власне множення. Перша фаза виконується лише один раз, а для інших використовується вже знайдена матриця, оскільки вона залежить лише від поля Галуа.

Алгоритм знаходження матриці множення *M* для $p = 2$ [50]:

1. Визначаємо матрицю *C*:

$$C=\begin{pmatrix}0 & 1 & 0 & \dots & 0\\ 0 & 0 & 1 & \dots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & \dots & 1\\ c_0 & c_1 & c_2 & \dots & c_{m-1}\end{pmatrix},$$

де c_i – коефіцієнти нормального поліному $p(t)=t^m+c_{m-1}t^{m-1}+\ldots+c_1t+c_0$.

2. Обчислюємо матрицю $D=ACB$ над полем *GF*(2), де *A* – матриця, в якій стовпці – базисні елементи нормального базису в поліноміальному, матриця *B* – транспонована матриця переходу (алгоритм її знаходження наведено в пункті 2.2.1).

3. Обчислюємо значення коефіцієнтів матриці множення:

$$\mu_{ij} = d_{(j-i) \bmod m;(-i) \bmod m},$$

де $i, j = \overrightarrow{0 \ldots m-1}$.

4. Формуємо матрицю множення:

$$M = \begin{pmatrix} \mu_{00} & \mu_{01} & \cdots & \mu_{0;m-1} \\ \mu_{10} & \mu_{11} & \cdots & \mu_{1;m-1} \\ \vdots & \vdots & \ddots & \vdots \\ \mu_{m-1;0} & \mu_{m-1;1} & \cdots & \mu_{m-1;m-1} \end{pmatrix}.$$

Алгоритм множення елементів $\mathbf{a} = (a_0; a_1; \ldots; a_{m-1})$ і $\mathbf{b} = (b_0; b_1; \ldots; b_{m-1})$ поля $GF(2^m)$, блок-схему якого представлено на рис. 2.3, складається з наступних кроків [50, 52]:

1. Ініціалізуємо змінні: $\mathbf{x} = \mathbf{a}$, $\mathbf{y} = \mathbf{b}$.

2. Для $k = \overrightarrow{0 \ldots m-1}$:

   а) знаходимо $c_k = \mathbf{x} \cdot M \cdot \mathbf{y}^T$.

   б) робимо циклічний зсув вліво для векторів $\mathbf{x}$ та $\mathbf{y}$.

3. Результат множення – $\mathbf{c} = (c_0; c_1; \ldots; c_{m-1})$.

Обчислювальна складність операції множення у нормальному базисі у загальному випадку сягає $O(m^3)$, а в окремих її можна покращити до $O(m^2)$.

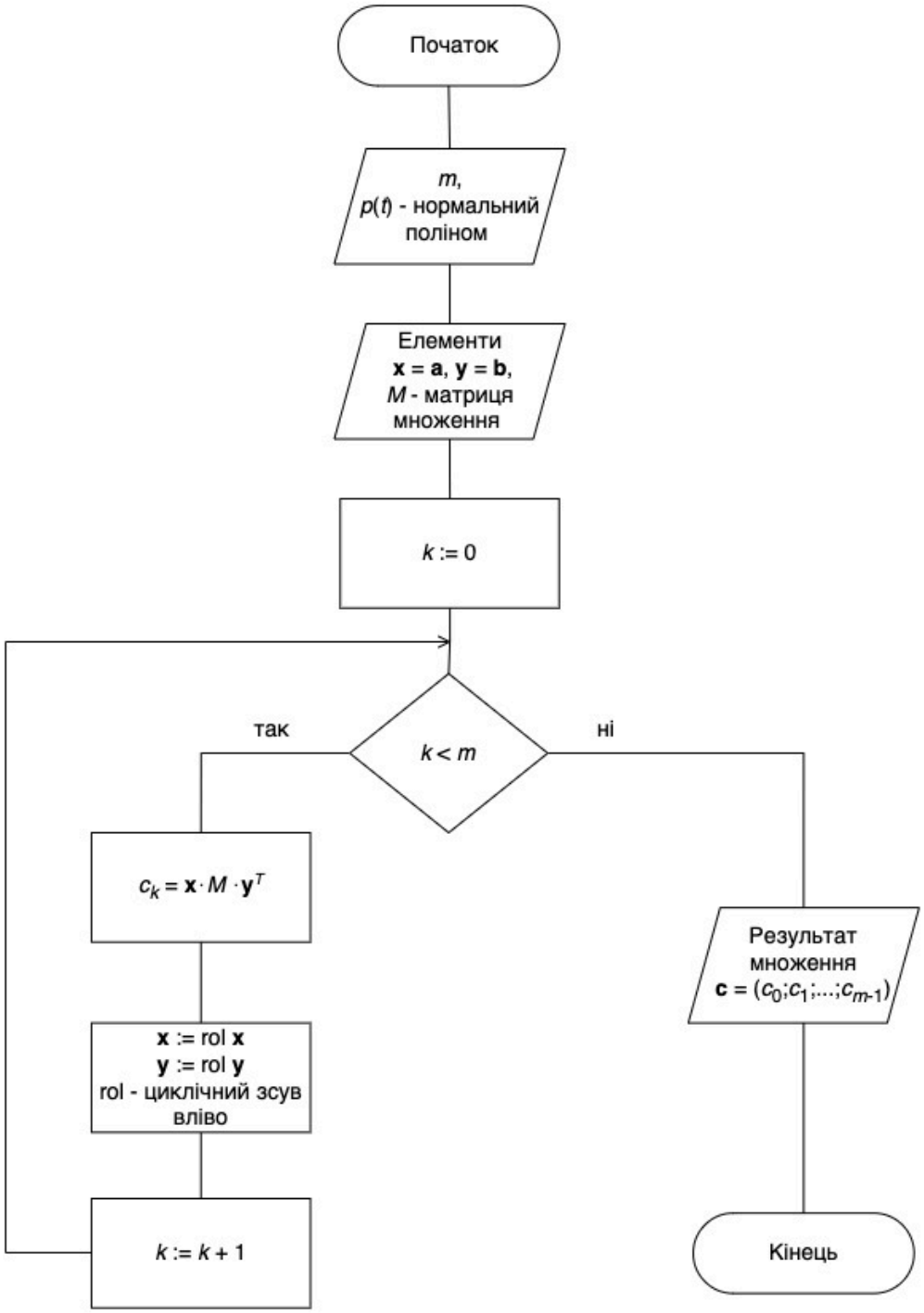


Рис. 2.3. Блок-схема алгоритму множення елементів у нормальному базисі

Обчислення мультиплікативно оберненого елемента

Мультиплікативно обернений елемент – це елемент для якого виконується рівність:

$$a \cdot a^{-1} = 1.$$

Обчислення мультиплікативно оберненого елемента у загальному випадку в обох базисах виконується двома способами: за розширеним алгоритмом Евкліда або за теоремою Ейлера [53-59].

Розширений алгоритм Евкліда знаходить частку безпосередньо, тому його можна використовувати як для ділення, так і для обчислення мультиплікативно оберненого елемента (у такому разі перший параметр $\alpha = 1$). Розглянемо кроки алгоритму для обчислення мультиплікативно оберненого елемента β^{-1} ($\beta \neq 0$):

1. Ініціалізуємо змінні:

$$r_0(t) = p(t), \text{ де } p(t) \text{ – нормальний поліном.}$$

$$r_1(t) = \beta.$$

$$s_0(t) = 0.$$

$s_1(t) = \alpha$, для обчислення мультиплікативно оберненого елемента $\alpha = 1$.

2. Виконуємо поки $r_1(t) \neq 0$:

а) $q(t) = r_0(t) / r_1(t)$;

б) $r_2(t) = r_0(t) + q(t) r_1(t)$;

в) $s_2(t) = s_0(t) + q(t) s_1(t)$;

г) $r_0(t) = r_1(t), r_1(t) = r_2(t)$;

д) $s_0(t) = s_1(t), s_1(t) = s_2(t)$.

3. $\beta^{-1} = s_0(t)$.

Альтернативний спосіб пошуку мультиплікативно оберненого елемента базується на використанні часткового випадку теореми Ейлера [60]. За теоремою Ейлера для будь-яких взаємно простих чисел *a* та *n*:

$$a^{\varphi(n)} \equiv 1 \pmod{n},$$

де $\varphi(n)$ – функція Ейлера, яка дорівнює кількості натуральних чисел менших m і взаємно простих з ним.

Для простого числа p: $\varphi(p) = p - 1$, тоді отримуємо рівність $a^{p-1} \equiv 1 \pmod{p}$, яка є малою теоремою Ферма. Знайдемо мультиплікативно обернений елемент β^{-1} використовуючи малу теорему Ферма:

$$\beta^{-1} = \beta^{\varphi(n)-1} = \beta^{k},$$

де $k \in \mathbb{N}$ і задовольняє рівність $k = (-1) \bmod r$. Зокрема, ця рівність виконується при $k = 2^m - 2$, тобто $\beta^{-1} = \beta^{2^m-2}$.

Отже, для обчислення мультиплікативно оберненого елементу можна використовувати частковий випадок алгоритму піднесення до степеня (для показника степеня $k = 2^m - 2$), який буде розглянутий нижче.

Ділення

Операція ділення реалізується шляхом множенням діленого на мультиплікативно обернений елемент до дільника $\mathbf{a}/\mathbf{b} = \mathbf{a} \cdot \mathbf{b}^{-1}$. Зважаючи на те, що множення та обчислення мультиплікативно оберненого елемента можна виконувати різними методами, то обчислювальна складність виконання операції ділення буде різною.

Піднесення до степеня

Піднесення до степеня k у нормальному й поліноміальному базисах можна виконувати за однаковим алгоритмом, блок-схема якого представлена на рис. 2.4. Алгоритм складається з наступних кроків [50, 59]:

1. Перетворимо степінь k у двійкову систему числення:
$$k = k_r k_{r-1} \ldots k_1 k_0.$$
2. Ініціалізуємо змінну $\mathbf{x} = \boldsymbol{\alpha}$, де $\boldsymbol{\alpha}$ – елемент поля $GF(p^m)$, який підносять до степеня k.
3. Для $i = \overrightarrow{r-1 \ldots 0}$:
   а) виконуємо множення $\mathbf{x} = \mathbf{x}^2$;

б) якщо $k_i = 1$, то $\mathbf{x} = \mathbf{a} \cdot \mathbf{x}$.

4. Результат операції – $\mathbf{x}$.

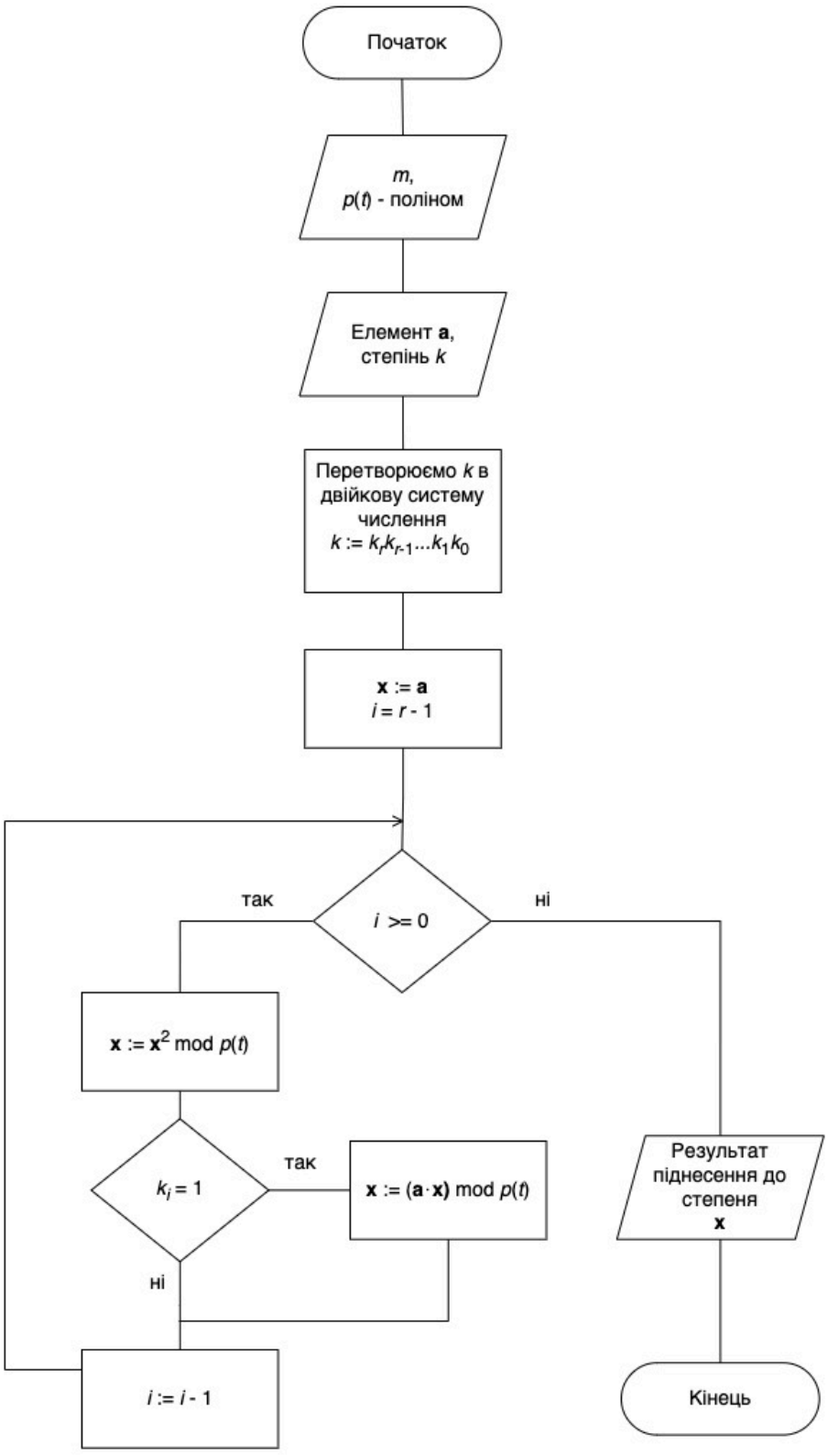


Рис. 2.4. Блок-схема алгоритму піднесення до степеня

Зазначена операція має різну обчислювальну складність залежно від базису, що використовується, адже множення, як розглянуто вище, залежно від представлення елементів виконується по різному. При цьому піднесення до квадрату в нормальному базисі можна замінити на операцію Фробеніуса для поля $GF(2^m)$.

Операція Фробеніуса

Тепер розглянемо докладно операцію Фробеніуса – піднесення полінома до степеня p^k. Часові витрати залежно від вибору базису відрізняються досить суттєво.

У нормальному базисі операцію Фробеніуса виконати дуже просто, це не потребує значних часових витрат. Завдяки представленню елементів цього базису $B=\left\{\alpha_0=t,\alpha_1=t^p,\alpha_2=t^{p^2},\ldots,\alpha_{m-1}=t^{p^{m-1}}\right\}$, достатньо виконати циклічний зсув на k розрядів. Залежно від подання елементів скінченного поля зсув потрібно виконувати у різному напрямку. Якщо елемент подати за зростанням степеня твірного елемента $\mathbf{a}=\left(a_0;a_1;\ldots;a_{m-1}\right)$, то циклічний зсув потрібно виконувати вправо $a^{p^k}=a_{m-k+1}a_{m-k+2}\ldots a_{m-1}a_0a_1\ldots a_{m-k}$; у іншому випадку, якщо $\mathbf{a}=\left(a_{m-1};a_{m-2};\ldots;a_0\right)$, зсув виконується вліво $a^{p^k}=a_{m-k}a_{m-k-1}\ldots a_1a_0a_{m-1}\ldots a_{m-k+1}$.

Розглянемо приклад виконання операції Фробеніуса у нормальному базисі. Нехай елемент $\mathbf{a}=\left(a_0;a_1;a_2\right)=\left(1;1;0\right)$ поля $GF(2^3)$ з незвідним многочленом x^3+x^2+1 потрібно піднести до степеня $z=p^k=2^2=4$. Тоді результат виконання операції:

$$\mathbf{a}^{2^2}=\mathbf{a}\gg 4=\left(a_2;a_0;a_1\right)=\left(0;1;1\right).$$

У поліноміальному базисі для виконання операції Фробеніуса немає швидкого алгоритму, її виконують як піднесення до степеня, яке докладно розглянуте вище.

Розглянемо приклад виконання операції Фробеніуса у поліноміальному базисі з вищенаведеними параметрами: елемент $\mathbf{a} = (a_0; a_1; a_2) = (1;1;0)$ поля $GF(2^3)$ з незвідним многочленом $x^3 + x^2 + 1$ потрібно піднести до степеня $z = p^k = 2^2 = 4$.

1. Перетворимо степінь z у двійкову систему числення:

$$z = 4_{10} = 100_2 = z_2 z_1 z_0.$$

2. $\mathbf{x} = x + 1.$

3. Для $i = \overrightarrow{1\ldots0}$:

а) $i = 1;$

б) $\mathbf{x} := \mathbf{x}^2 = (x+1)^2 = x^2 + 1;$

в) $z_1 = 0;$

г) $i := i - 1 = 1 - 1 = 0;$

д) $\mathbf{x} := \mathbf{x}^2 = (x^2+1)^2 = x^4 + 1 = x \cdot x^3 + 1 = x \cdot (x^2+1) + 1 =$

$= x^3 + x + 1 = x^2 + x.$

4. Отже, $\mathbf{a}^{2^2} = x^2 + x = (0;1;1).$

2.2.3. Аналіз обчислювальної складності виконання операцій

Узагальнене порівняння обчислювальної складності [61-64] виконання операцій у поліноміальному і нормальному базисах наведено в табл. 2.2.

Таким чином, обчислювальна складність операції додавання не залежить від базису, в якому подані елементи поля Галуа. Складність обчислення мультиплікативно оберненого елементу, виконання операції ділення та піднесення до степеня (у загальному випадку) є більшою у нормальному базисі, порівняно з поліноміальним. Водночас складність виконання операції Фробеніуса (піднесення до степеня p^k) є значно меншою в нормальному базисі, ніж в поліноміальному.

Таблиця 2.2

Порівняльна характеристика виконання операцій у базисах

Операції	Складність у поліноміальному базисі	Складність нормальному базисі
Додавання	$O(m)$	$O(m)$
Множення	$O\left(m\log m\log\log m\right)$ $O(m^2)$	$O(m^3)$ В окремих випадках $O(m^2)$
Обчислення мультиплікативно оберненого елементу	$O\left(m^2\left(\log\left(m-1\right)+1\right)\right)$	$O\left(m^3\log\left(m-1\right)\right)$
Ділення	$O\left(m^2\left(\log\left(m-1\right)+1\right)\right)$	$O\left(m^3\left(\log\left(m-1\right)+1\right)\right)$
Операція Фробеніуса (піднесення до степеня p^k)	$O\left(m^2\log p^k\right)$	$O(1)$
Піднесення до степеня	$O\left(m^2\log k\right)$	$O\left(m^3\log k\right)$

2.3. Алгоритмічний метод пошуку нормальних поліномів для міжбазисних перетворень елементів скінченних полів

Задання нормального поліному в полі Галуа є важливою умовою для побудови нормального базису. З огляду на це, розроблення методів пошуку нормального поліному є актуальною задачею.

Запропонований метод пошуку нормальних поліномів у полі $GF(2^m)$ полягає у виконанні таких кроків [42]:

1. Знайти всі прості числа в діапазоні $\left[2^m;2^{m+1}\right)$ (наприклад, за алгоритмом решето Ератосфена або з використанням генератора простого числа заданої довжини [65]).
2. Перетворити отримані прості числа у систему числення з основою 2. Результатом цього кроку є коефіцієнти поліномів.
3. Видалити з множини результатів поліноми слід, яких дорівнює 0. Результатом цього кроку будуть незвідні поліноми.
4. Спробувати знайти обернену матрицю для поля з незвідними поліномами взятими з кроку 2 (множину вхідних поліномів цього кроку можна суттєво зменшити видаливши з неї елементи, які містять коефіцієнт 0 при степені $m-1$, оскільки як довели експериментальні результати зазначені поліноми не є нормальними).
5. Многочлени для яких існує обернена матриця – нормальні.

Варто зауважити, що на кроці 3 знайдені не всі незвідні поліноми, але при цьому кількість пропущених многочленів є незначною відносно їх загальної кількості.

Розглянемо запропонований метод побудови нормальних многочленів для поля $GF(2^4)$:

1. Прості числа діапазону $\left[2^4;2^5\right)$: 17, 19, 23, 29, 31.
2. Перетворимо числа з десяткової системи числення у двійкову:

$$17_{10}=10001_2,$$

$$19_{10}=10011_2,$$

$$23_{10}=10111_2,$$

$$29_{10}=11101_2,$$

$$31_2=11111_2.$$

3. Знаходимо слід поліномів знайдених в кроці 2 за формулою $Tr(\mathbf{x}) = \sum_{i=0}^{m-1} a_i$:

$$Tr(10001) = 2 \bmod 2 = 0,$$

$$Tr(10011) = 3 \bmod 2 = 1,$$

$$Tr(10111) = 4 \bmod 2 = 0,$$

$$Tr(11101) = 4 \bmod 2 = 0,$$

$$Tr(11111) = 5 \bmod 2 = 1.$$

Поліноми 10011 та 11111 є незвідними.

4. За алгоритмом з підрозділу 2.1 шукаємо обернені матриці переходу. Така матриця існує для поліному 11111.

5. Отже, нормальний поліном – $x^4 + x^3 + x^2 + x + 1$.

У розглянутому прикладі порівняно з традиційним методом пошуку незвідних поліномів (через перевірку наявності остачі від ділення поточного полінома на всі незвідні з меншим степенем), у множині поліномів відсутній многочлен 11001, який є незвідним. Це пояснюється тим, що його двійкове представлення не відповідає простому числу в десятковій системі числення.

Розглянемо запропонований метод у полі більшої розмірності, зокрема для $GF(2^5)$:

1. Прості числа з діапазону $\left[2^5; 2^6\right)$: 37, 41, 43, 47, 53, 59, 61.

2. Перетворимо числа з десяткової системи числення у двійкову:

$$37_{10} = 100101_2,$$

$$41_{10} = 101001_2,$$

$$43_{10} = 101011_2,$$

$$47_{10} = 101111_2,$$

$$53_{10} = 110101_2,$$

$$59_{10} = 111011_2,$$

$$61_{10} = 111101_2.$$

3. Знаходимо слід поліномів знайдених в кроці 2 за формулою $Tr(\mathbf{x}) = \sum_{i=0}^{m-1} a_i$:

$$Tr\left(100101\right) = 3 \bmod 2 = 1,$$

$$Tr\left(101001\right) = 3 \bmod 2 = 1,$$

$$Tr\left(101011\right) = 4 \bmod 2 = 0,$$

$$Tr\left(101111\right) = 5 \bmod 2 = 1,$$

$$Tr\left(110101\right) = 4 \bmod 2 = 0,$$

$$Tr\left(111011\right) = 5 \bmod 2 = 1,$$

$$Tr\left(111101\right) = 5 \bmod 2 = 1.$$

Отже, поліноми 100101, 101001, 101111, 111011, 111101 є незвідними.

4. За алгоритмом з підрозділу 2.1 шукаємо обернені матриці переходу. Такі матриці існують для поліномів: 111011, 111101.

5. Отже, поліноми $x^5 + x^4 + x^3 + x + 1,\ x^5 + x^4 + x^3 + x^2 + 1.$

У розглянутому прикладі порівняно з традиційним методом пошуку незвідних поліномів, у множині поліномів відсутній многочлен 110111, який є незвідним. Це пояснюється тим, що його двійкове представлення не відповідає простому числу в десятковій системі числення.

2.4. Модифікований спосіб побудови матриці переходу між базисами

Побудова матриці переходу (алгоритм з пункту 2.2.1) базується на вираженні базисних елементів нормального базису $B = \left\{\alpha_0 = t, \alpha_1 = t^p, \alpha_2 = t^{p^2}, \ldots, \alpha_{m-1} = t^{p^{m-1}}\right\}$ через поліноміальний, блок-схему виконання цього етапу наведено на рис. 2.5. Традиційно цей етап виконують

у два способи: діленням елемента t^{p^i} на незвідний поліном або підстановками з незвідного многочлена. Розглянемо ці способи докладніше для поля $GF(2^5)$ з незвідним поліномом $f(x) = x^5 + x^4 + x^3 + x^2 + 1$.

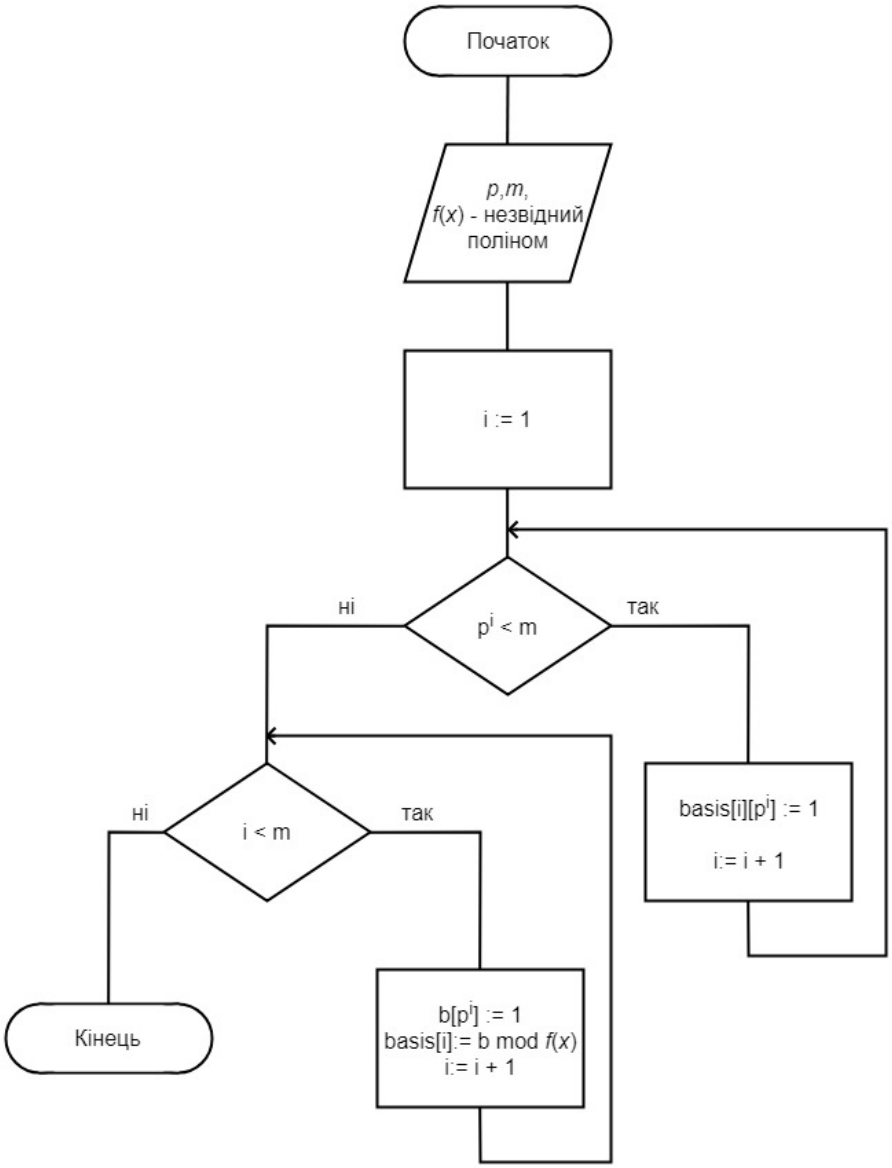


Рис. 2.5. Блок-схема етапу вираження базисних елементів нормального базису через поліноміальний

Для $p^i = 2^i < m$ додаткових обчислень виконувати непотрібно (елементи подаються за означенням): $\alpha_0 = x^{2^0} = x$, $\alpha_1 = x^{2^1} = x^2$,

$\alpha_2 = x^{2^2} = x^4$. Решту елементів треба обчислити використовуючи вищенаведені підходи.

Спосіб 1. Ділення на незвідний поліном

$$\alpha_3 = x^{2^3} = x^8 \bmod f(x).$$

$x^8 + 0x^7 + 0x^6 + 0x^5 + 0x^4 + 0x^3 + 0x^2 + 0x + 0$	$x^5 + x^4 + x^3 + x^2 + 0x + 1$
$x^8 + x^7 + x^6 + x^5 + 0x^4 + x^3$	$x^3 + x^2$
$x^7 + x^6 + x^5 + 0x^4 + x^3 + 0x^2$	
$x^7 + x^6 + x^5 + x^4 + 0x^3 + x^2$	
$x^4 + x^3 + x^2 + 0x + 0$	

$$\alpha_4 = x^{2^4} = x^{16} \bmod f(x) = x^3 + x^2 + x + 1.$$

У наведеному підході для обчислення остачі від ділення елемента на незвідний многочлен необхідно виділити пам'ять, яка пропорційна значенню $n^{p^{m-1}}$, де n – кількість пам'яті необхідної для збереження коефіцієнта елемента поля Галуа.

Обчислювальна складність заданого алгоритму дорівнює $O\left(m^{p^i}\right)$.

Спосіб 2. Підстановки з незвідного многочлена

$$\alpha_3 = x^{2^3} = x^8 = x^3 \cdot x^5 = x^3 \cdot \left(x^4 + x^3 + x^2 + 1\right) = x^7 + x^6 + x^5 + x^3 =$$

$$= \left(x^2 + x + 1\right) \cdot \left(x^4 + x^3 + x^2 + 1\right) + x^3 = x^6 + x + x^4 + 1 + x^3 =$$

$$= x \cdot \left(x^4 + x^3 + x^2 + 1\right) + x^4 + x^3 + x + 1 = x^5 + 1 = x^4 + x^3 + x^2,$$

$$\alpha_4 = x^{2^4} = x^{16} = x^{11} \cdot x^5 = x \cdot \left(x^4 + x^3 + x^2 + 1\right)^3 =$$

$$= x \cdot \left(\left(x^4 + x^3\right)^3 + \left(x^4 + x^3\right)^2 \cdot \left(x^2 + 1\right) + \left(x^4 + x^3\right) \cdot \left(x^2 + 1\right)^2 + \left(x^2 + 1\right)^3\right) =$$

$$= x \cdot \left(\left(x^{12} + x^{11} + x^{10} + x^9\right) + \left(x^{10} + x^6\right) + \left(x^8 + x^7 + x^4 + x^3\right) + \left(x^6 + x^4 + x^2 + 1\right)\right) =$$

$$= x \cdot \left(x^{12} + x^{11} + x^9 + x^8 + x^7 + x^3 + x^2 + 1 \right) = \ldots = x^3 + x^2 + x + 1.$$

Зазначений спосіб досить складно реалізувати програмно, оскільки зі зростанням параметра *m*, кількість додаткових обчислень експоненційно зростатиме.

Модифікований спосіб. Елемент нормального базису t^{p^i} можна обчислити за допомогою рекурентної формули $\alpha_{i+1} = t^{p^{i+1}} = t^{p^i \cdot p} = \prod_{j=1}^{p} t^{p^i} = \left(\alpha_i\right)^p$, що дозволить зменшити кількість виділеної пам'яті й пришвидшити виконання алгоритму [66].

$$\alpha_3 = x^{2^3} = \alpha_2 \cdot \alpha_2 = x^4 \cdot x^4 = x^8 \bmod f(x) = x^4 + x^3 + x^2,$$

$$\alpha_4 = x^{2^4} = \alpha_3{}^2 = \left(x^4 + x^3 + x^2\right)^2 = \left(x^8 + x^6 + x^2\right) \bmod f(x).$$

$x^8 + 0x^7 + x^6 + 0x^5 + 0x^4 + 0x^3 + x^2 + 0x + 0$	$x^5 + x^4 + x^3 + x^2 + 0x + 1$
$x^8 + x^7 + x^6 + x^5 + 0x^4 + x^3$	$x^3 + x^2 + x + 1$
$x^7 + 0x^6 + x^5 + 0x^4 + x^3 + x^2$	
$x^7 + x^6 + x^5 + x^4 + 0x^3 + x^2$	
$x^6 + 0x^5 + x^4 + x^3 + 0x^2 + 0x$	
$x^6 + x^5 + x^4 + x^3 + 0x^2 + x$	
$x^5 + 0x^4 + 0x^3 + 0x^2 + x + 0$	
$x^5 + x^4 + x^3 + x^2 + 0x + 1$	
$x^4 + x^3 + x^2 + x + 1$	

Отже, $\alpha_4 = x^3 + x^2 + x + 1$.

Для обчислення усіх елементів матриці переходу, використовуючи зазначений спосіб, необхідно виділити пам'ять пропорційну $p \cdot n$, де n – кількість пам'яті необхідної для збереження коефіцієнта елемента поля

Галуа. Це пояснюється тим, що саме така максимальна кількість коефіцієнтів буде результатом піднесення попереднього базисного елемента поля t^{p^i} до степеня p. Оскільки при вираженні елементів нормального базису через поліноміальні зазначеним способом, необхідно буде обчислювати остачу від ділення елемента $t^{p^{i+1}}$, максимальна довжина якого не перевищує $2m - 1$, то обчислювальна складність при реалізації зазначеного підходу дорівнює $O\left(m^p\right)$.

2.5. Висновки

Скінченні поля застосовуються у методах захисту приватних наборів даних, таких як гомоморфне шифрування та безпечні багатосторонні обчислення. Таким чином, скінченні поля застосовуються у таких прикладних задачах:

- блочне шифрування, яке використовує поля виду $GF(2^8)$;
- криптографія на еліптичних кривих, яка базується як на арифметиці бінарних полів Галуа великих порядків, так і на обчисленнях у скінченних полях за модулем простого числа;
- коди Ріда-Соломона, які використовують переважно бінарні поля Галуа.

Час виконання розглянутих операцій в скінченних полях залежить від базису в якому подаються елементи. Зважаючи на це, модифікації методу міжбазисних перетворень, а також поєднання різних базисів для розв'язання різних прикладних задач є актуальними.

На основі проведеного аналізу алгоритмів виконання операцій в обох базисах та розглянутих алгоритмів переходу між базисами можна зробити такі висновки:

- обчислювальна складність алгоритмів виконання операцій залежить від обраного базису;

- операції множення та обчислення мультиплікативно оберненого елементу виконуються швидше в поліноміальному базисі, а операція Фробеніуса в нормальному;
- використання різних базисів доцільно при виконанні певних типів задач, зокрема завдяки міжбазисним переходам можна прискорити виконання операції подвоєння точок в еліптичній криптографії.

Розроблено метод пошуку нормальних поліномів, який відрізняється від існуючого використанням простих чисел у десятковому представленні замість поліномів, що дозволяє зменшити обчислювальну складність пошуку нормальних многочленів з $O\left(n^3\right)$ до $O\left(n\log\left(\log n\right)\right)$ і, як наслідок, спростити міжбазисні перетворення у бінарних скінченних полях з метою пришвидшення виконання операцій над елементами поля у методах гомоморфного шифрування даних.

Розроблено модифікований спосіб для переходу між базисами, який полягає у використанні рекурентної формули $\alpha_{i+1} = t^{p^{i+1}} = \prod_{j=1}^{p} t^{p^i}$ замість обчислення остачі від ділення елемента $t^{p^{i+1}}$ на незвідний поліном, що дозволяє зменшити кількість використовуваної пам’яті з n^{p^i} до $p \cdot n$, а також обчислювальну складність з $O\left(m^{p^i}\right)$ до $O\left(m^p\right)$.

3. РОЗРОБЛЕННЯ АЛГОРИТМІЧНО-ПРОГРАМНОГО МЕТОДУ ЗАХИСТУ ПРИВАТНИХ НАБОРІВ ДАНИХ

3.1. Модифікована програмна модель шифрування даних

Функціональне шифрування – це тип шифрування, який забезпечує більш точний контроль над доступом до зашифрованих даних порівняно з традиційними методами шифрування [67, 68]. Метою функціонального шифрування є вибіркове розкриття певної інформації або функції, що містяться в зашифрованих даних, авторизованим сторонам, зберігаючи при цьому конфіденційність решти даних. Коли використовується традиційне шифрування даних, власник ключа може розшифрувати весь вміст повідомлення. Однак, функціональне шифрування дозволяє власнику ключа обчислювати лише певні функції на зашифрованих даних, при цьому не розкриваючи весь оригінальний вміст. Це досягається шляхом асоціювання різних ключів з різними функціями.

Форми функціонального шифрування є різними, зокрема прикладами є шифрування на основі атрибутів, предикатне шифрування та шифрування шляхом обчислення внутрішнього добутку. У шифрування на основі атрибутів політики доступу пов’язані з зашифрованими даними, а користувачеві надається ключ розшифрування, який пов’язаний з певними атрибутами. Користувач може розшифрувати дані, тільки якщо їхні атрибути задовольняють політиці доступу. Предикатне шифрування дозволяє розшифрувати повідомлення на основі того, чи задовольняє воно певний предикат або умову. Шифрування шляхом обчислення внутрішнього добутку дозволяє обчислити внутрішній добуток двох векторів даних (зашифрованого тексту та ключа шифрування), який можна використовувати у випадках, коли потрібно обчислити скалярний добуток, не розкриваючи окремих компонентів векторів [67, 68].

Функціональне шифрування застосовується в різних сферах, включаючи обчислення із збереженням конфіденційності, безпечний обмін

даними та контроль доступу в хмарних обчисленнях. Це забезпечує безпечну співпрацю, зберігаючи конфіденційність конфіденційної інформації. Однак реалізація функціональних систем шифрування може бути складною, і ефективність таких схем є предметом постійних досліджень.

Методи машинного навчання, зокрема нейронні мережі традиційно не є поширеними у методах захисту даних, оскільки навіть базові операції класичних алгоритмів, що застосовуються в криптографії (наприклад, XOR), є складними для таких технологій. У той же час системи, що базуються на використанні штучного інтелекту, застосовуються для зламу криптографічних систем, що захищають дані. Однак, останнім часом набули поширення генеративні конкуруючі нейронні мережі (GAN – generative adversarial networks), одним з експериментальних застосувань яких є модель шифрування, запропонована представниками компанії Google Brain [69, 70]. Модель шифрування наведено на рис. 3.1.

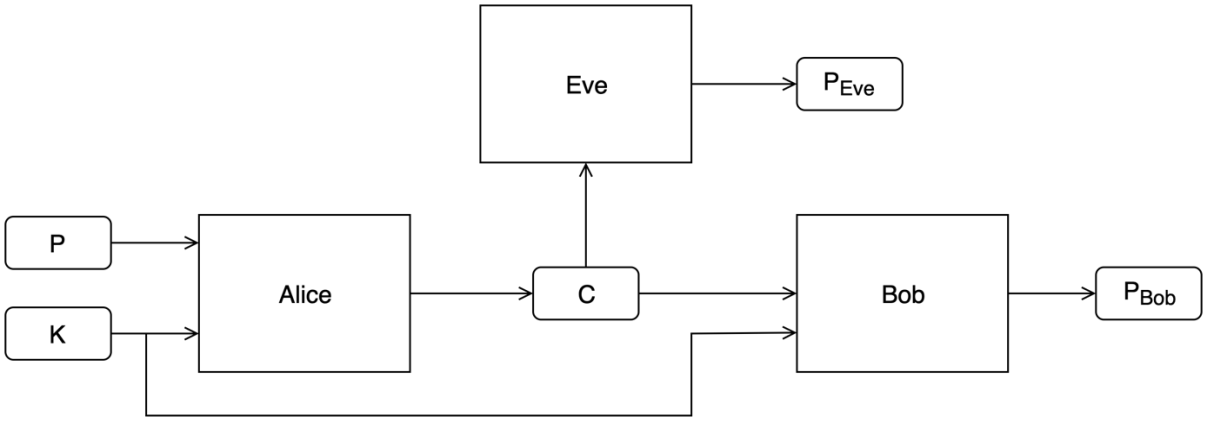


Рис. 3.1. Модель шифрування даних з використанням GAN

Особливістю цієї моделі є поєднання підходів симетричного шифрування та концепції генеративних конкуруючих нейронних мереж. Для аналізу цієї моделі доцільно розглянути кожен з підходів докладніше.

3.1.1. Симетричне шифрування

Симетричне шифрування – це тип шифрування, шифрування та дешифрування даних в якому відбувається за допомогою лише одного приватного ключа. Суб'єкти, які обмінюються даними з використанням симетричного шифрування, повинні на початку комунікації визначити єдиний ключ, що буде використовуватись для формування шифротексту. Зважаючи на те, що цей ключ повинен бути приватним, то іншою назвою цього типу шифрування є шифрування з приватним ключем (private key encryption).

Математично, шифр описується наступними формулами [71]:

$$E_K(P) = C,$$

$$D_K(C) = P,$$

де P – початковий текст, що шифрується (відкритий текст),

C – результат шифрування (шифротекст),

K – ключ шифрування,

E_K – функція, що виконує шифрування,

D_K – функція, що виконує дешифрування.

За умови використання однакового ключа для розглянутих перетворень (шифрування й дешифрування даних), шифр є симетричним і ключ шифрування в такому випадку має бути приватним. При використанні різних ключів для шифрування й дешифрування даних, шифрування є асиметричним і в такому разі один з ключів є приватним, а інший – загальнодоступний.

Розглянемо приклад симетричного шифру, щоб зрозуміти його принцип роботи. Одним з найпростіших і найстаріших таких шифрів є шифр Цезаря [72]. Цей шифр є шифром заміни, в якому кожна літера відкритого тексту зсувається на певну кількість позицій вниз або вгору по алфавіту.

Зважаючи на це, іншою назвою шифру Цезаря є шифр зсуву. Математично, такий шифр можна записати за допомогою наступних перетворень:

$$E(p) = (p + k) \bmod n,$$

$$D(c) = (c - k) \bmod n,$$

де p – літера алфавіту початкового тексту,

c – літера шифротексту,

k – ключ, що використовується для шифрування і відображає кількість позицій, на які має відбутись зсув в алфавіті під час шифрування,

n – кількість літер алфавіту.

Розглянемо приклад використання шифру Цезаря. Нехай повідомлення написано з використанням українського алфавіту, тоді кількість літер алфавіту (n) дорівнює 33. Обравши ключем шифрування $k = 5$, початковий текст $P =$ «таємна новина» в результаті шифрування перетвориться на текст $C =$ «чдїстд туєлтд». Якщо ж ключ шифрування $k = 7$, то результатом буде C = «щєкуфє фхзнфє». Дешифрування повідомлення виконати досить просто знаючи ключ шифрування, оскільки достатньо виконати зворотне перетворення (виконати зсув літер зашифрованого тексту в протилежну сторону).

Шифр Цезаря є моноалфавітним, оскільки у всіх місцях, де певна літера згадується, вона завжди буде замінена за одним і тим же принципом (на одну й ту ж нову літеру). Однак, це є одним з суттєвих недоліків таких шифрів. Як можна побачити, з прикладу, частина літер статистично частіше використовується в тому чи іншому алфавіті, тож шифр можна зламати використовуючи частотний аналіз. Хоча шифр Цезаря не є практичним у сучасному світі, він відображає основні принципи, що використовуються у симетричному шифруванні.

Симетричне шифрування має декілька суттєвих переваг, серед яких висока швидкість обробки даних та менша потреба використання обчислювальних ресурсах, порівняно з асиметричним шифруванням. Це

дозволяє використовувати симетричне шифрування для роботи з великими обсягами даних. Крім цього, існує можливість масштабування ключів шифрування (шляхом збільшення довжини) для забезпечення безпеки. Ще однією важливою перевагою є простота в реалізації, адже основою шифру є виконання простих операцій над даними.

Одним з найбільших недоліків симетричного шифрування є проблема обміну приватним ключем, між суб'єктами обміну інформації. Якщо такий ключ перехоплять, то зловмисники зможуть дешифрувати дані. Окрім проблеми обміну ключами, важливими завданнями є генерація, зберігання й знищення ключів шифрування; проте це є складною задачею, оскільки кількість ключів квадратично залежить від кількості суб'єктів обміну інформації. Одним з шляхів нівелювання цих недоліків є комбінація симетричних й асиметричних шифрів. Для передачі приватного ключа шифрування у такому випадку використовують асиметричний шифр, що є більш надійним; а для шифрування основного масиву даних використовують симетричний шифр, що є більш швидким.

Важливим для аналізу шифрування даних, з використанням моделі, що зображено на рис. 3.1, є метод обробки вхідного тексту. Текст може оброблятись шифром як послідовно (шифруючи байти або біти), так і паралельно (шляхом шифрування блоків). У першому випадку шифри називаються потоковими, а в другому – блочними. Потокові шифри опрацьовують текст посимвольно. Блочні шифри ділять вхідний текст на блоки певної довжини, шифрування яких відбувається окремо (й частіше всього паралельно). Однак, за такого підходу може виникнути ситуація, коли в останньому блоці недостатньо символів, тож у таких випадках зазвичай блок доповнюється нулями.

Одними з найвідоміших прикладів сучасних симетричних шифрів є блочні шифри AES (Advanced Encryption Standard), Twofish та IDEA (International Data Encryption Algorithm); а також потокові шифри Grain-128, ChaCha20 та Salsa20.

3.1.2. Генеративні конкуруючі нейронні мережі

Основна схема навчання генеративної конкуруючої мережі, вже розглянута в пункті 1.3.2, тож зараз докладніше проаналізуємо структуру її складових (мереж генератора й дискримінатора).

Основним підходом до реалізації генератора й дискримінатора є використання згорткових нейронних мереж (CNNs – convolutional neural networks). Такі мережі представляють собою клас глибоких нейронних мереж, що складаються хоча б з одного згорткового шару й реалізують метод прямого поширення інформації. Однією з основних сфер застосування таких мереж є робота з зображеннями.

Архітектура CNN мереж схожа на взаємозв'язок нейронів у мозку людини. Подразники впливають на окремі нейрони згорткової мережі, внаслідок чого вони реагують на них в обмеженому полі, відомому як рецептивне поле. Множина цих полів накладається, щоб охопити всю візуальну область зображення.

Згідно цієї архітектури, зображення розглядається як набір пікселів, що спершу розгладжується, а потім передається як багаторівневий перцептрон для класифікації.

У складі згорткової нейронної мережі є певна комбінація шарів [73]. Такими шарами можуть бути згортковий, агрегувальний, повністю з'єднані шари та шар активації. Проаналізуємо їх докладніше.

Одним з основних компонентів для побудови CNN архітектури, є згортковий шар (convolutional layer). Цей шар використовується для виявлення локальних особливостей вхідних даних. Це досягається за допомогою виконання операції згортки, яка є математичною операцією двох функцій $f(t)$ і $g(t)$, результатом якої є третя функція. Формально це перетворення можна описати наступною формулою: $(f*g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$.

Згортковий шар складається з набору фільтрів (ядер), сприятливе поле яких є невеликим, однак, вони поширюються на весь вхідний набір даних.

Прямий прохід даних через кожен фільтр згортає їх по ширині й висоті вхідного поля, що дозволяє створити двовимірну карту активації фільтру, обчисливши добуток між даними фільтру та входу. Внаслідок цього, нейронна мережа вивчає фільтри, що були активовані після виявлення певних характеристик у конкретному просторовому положенні на вході. На шар активації (нелінійну функцію) подається скалярний результат кожного згорткового шару, тому ці шари зазвичай об'єднують. Прикладами функції активації, що застосовувались раніше, є гіперболічний тангенс $f(x)=\tanh(x)$, $f(x)=\left|\tanh(x)\right|$ і сигмоїда $f(x)=\left(1+e^{-x}\right)^{-1}$. Однак, зараз найчастіше використовується функція активації ReLU (Rectified Linear Unit): $f(x)=\max(0,x)$, оскільки вона є простішою, тож її використання прискорює навчання та зменшує обчислювальні витрати. На рис. 3.2 проілюстровано принцип роботи згорткового шару.

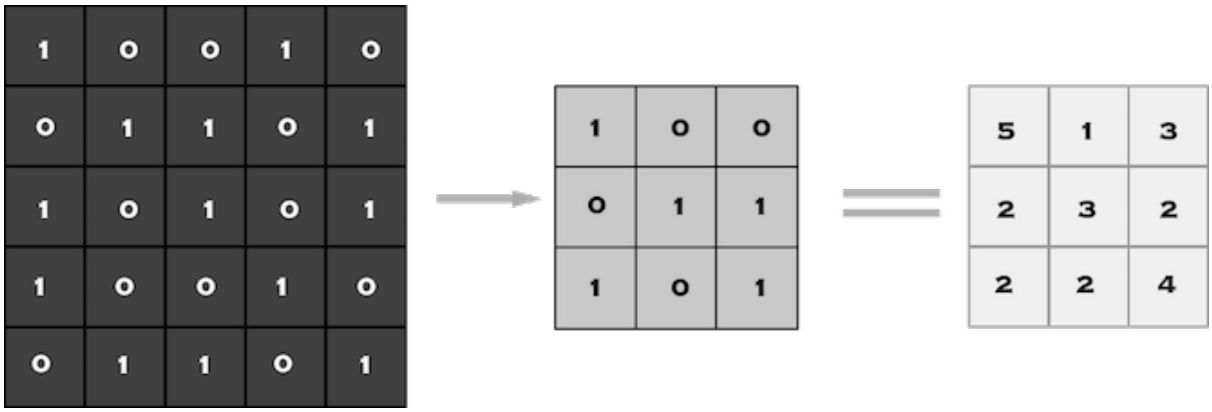


Рис. 3.2. Принцип роботи згорткового шару

Іншим типом шарів згорткової нейронної мережі є агрегувальний шар (pooling layer). Цей шар дозволяє зменшити розмірність даних, шляхом застосування нелінійного перетворення. Існують різні функції для такого агрегування, однак, найчастіше використовується агрегування шляхом максимізації (max pooling) [74]. Такий приклад агрегування даних наведено на рис. 3.3. На вхід агрегувального шару приходить матриця, яка

розділяється на чотири блоки. У кожному блоці знадиться максимальне значення й з них формується вихід агрегувального шару (матриця максимальних значень).

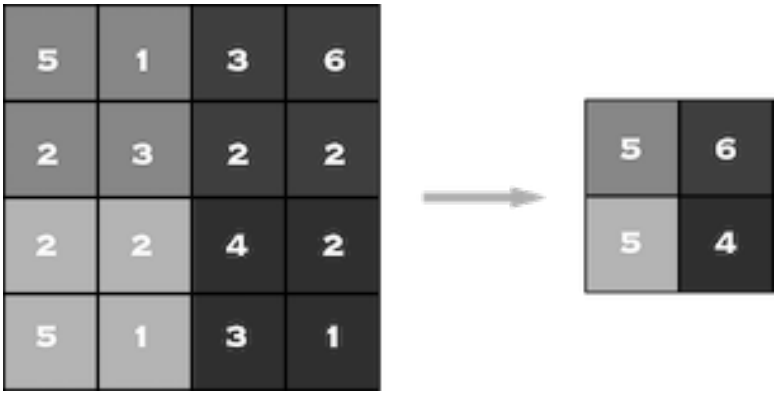


Рис. 3.3. Приклад обробки даних агрегувальним шаром

Повністю з’єднані шари (fully connected layers) також входять до згорткової нейронної мережі. Основною їх особливістю є те, що в цьому наборі шарів кожен параметр (нейрон) одного шару з’єднаний з кожним параметром (нейроном) іншого шару. Зважаючи на те, що часово-просторова складність зображення зменшується під час його обробки в згортковому та агрегувальному шарах, то це дозволяє класифікувати зображення, шляхом формування повністю з’єднаної мережі. На рис. 3.4 наведено приклад взаємодії повністю з’єднаних шарів згорткової мережі.

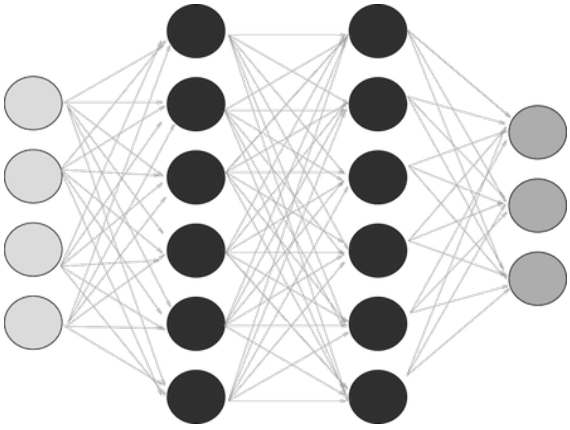


Рис. 3.4. Приклад взаємодії повністю з’єднаних шарів згорткової мережі

Ознайомившись, з кожною окремою складовою згорткової нейронної мережі, можна розглянути приклад роботи такої нейронної мережі в цілому. Розглянемо приклад обробки зображення згортковою нейронною мережею, наведений на рис. 3.5. Як показано на рисунку, спершу зображення обробляють у згорткових, активаційних та агрегувальних шарах (кількість яких залежить від складності зображення). У результаті обробки складність зображення зменшується, при цьому його характерні ознаки зберігаються. Після цього зображення класифікується, шляхом його обробки у повністю з'єднаних шарах, яке завершується обчисленням функції активації softmax:

$P\left(y=j\middle|\mathbf{x}\right)=\frac{e^{\mathbf{x}^T\mathbf{w}_j}}{\sum_{i=1}^{K}e^{\mathbf{x}^T\mathbf{w}_k}}$, де $\mathbf{w}_i$ – j-й вектор ваг моделі, $\mathbf{x}$ – вхідний вектор, а K – кількість можливих варіантів-результатів (класів класифікації).

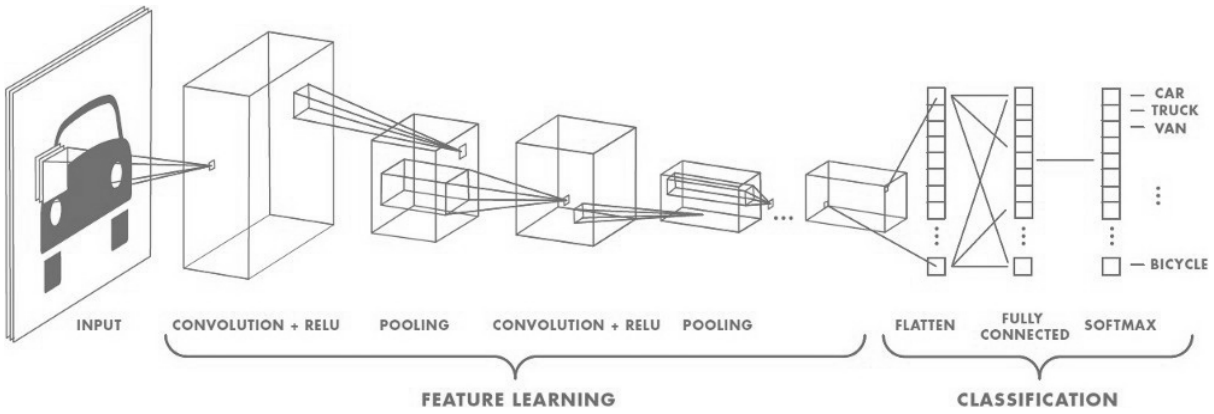


Рис. 3.5. Приклад обробки зображення згортковою нейронною мережею

Отже, мережі дискимінатора та генератора генеративної конкуруючої мережі доцільно побудувати з використанням згорткових нейронних мереж, оскільки вони вирішують задачу класифікації.

3.1.3. Складові моделі шифрування

У основі моделі, що зображена на рис. 3.1, взаємодія трьох нейронних мереж. Їх назви «Alice», «Bob», «Eve». Проаналізуємо їх задачі, вхідні й вихідні параметри та загальну структуру зазначених мереж.

Основним завданням мережі «Alice» є винайдення безпечного методу шифрування, що дозволить здійснювати безпечну комунікацію з мережею «Bob». На вхід мережі «Alice» приходить відкритий текст (*P*) і ключ шифрування (*K*), а вихідним параметром є результат шифрування (*C*).

Мережа «Eve» виступає у даній моделі зловмисником. Її мета дешифрувати текст з наміром отримання відкритого тексту, приймаючи як вхідний параметр лише шифротекст (*C*).

Мережа «Bob» намагається знайти метод дешифрування, що дозволить, приймаючи на вхід результат шифрування (*C*) та ключ шифрування (*K*), відтворити відкритий текст (*P*).

Відповідно до дослідження проведеного компанією Google Brain [69, 75], модель шифрування реалізовано за архітектурою «змішати та перетворити». Для цього використано шари згорткової мережі. Повністю з'єднаний шар згорткової мережі є першим шаром у цій архітектурі. Кількість входів і виходів цього шару однакова. Початковий текст, представлений набором біт, і набір біт ключа шифрування є вхідними параметрами цього повністю з'єднаного шару. Здатність цього шару змішувати біти ключа та початкового тексту дозволяє виражати кожен вихідний біт як лінійну комбінацію вхідних бітів. На наступному етапі дані проходять через послідовність згорткових шарів, результатом чого є шифротекст, розмір якого відповідає розміру відкритого тексту. Завданням цих згорткових шарів є застосовування певної функції, без її попереднього визначення, до набору бітів, які були попередньо змішані, шляхом поступового навчання. Порядок шарів даної моделі є зворотним у порівнянні з згортковою мережею, що застосовується в програмах обробки зображень. У нейронних мережах, призначених для обробки зображень, завданням згорткових шарів є використання просторових властивостей розташування пікселів на зображенні; а у випадку конкуруючої нейронної

криптографії важливою характеристикою, яку під час навчання вивчають нейронні мережі, є розташування біт.

Архітектура мережі «Alice» представлена на рис. 3.6. Спершу ця мережа утворює комбінацію двох *N*-бітних входів (початкового тексту та ключа шифрування), яка представлена у вигляді вектору бітових значень розміром $2N \times 1$. Далі цей вектор опрацьовується повністю з'єднаним шаром з розмірністю $2N \times 2N$, після чого результат проходить через послідовність чотирьох 1-*D* згорткових шарів. Фільтри для цих шарів мають розміри [4, 1, 2], [2, 2, 4], [1, 4, 4] та [1, 4, 1], з відповідними кроками 1, 2, 1, 1. Після кожного шару використовується сигмоїдна нелінійна функція активації, за винятком заключного шару, де використовується гіперболічний тангенс. Архітектура мережі «Bob» ідентична до мережі «Alice», тоді як мережа «Eve» не приймає ключа шифрування як вхідні дані (є лише шифротекст), тож її перший шар змінено на повністю з'єднаний шар розміром $N \times 2N$.

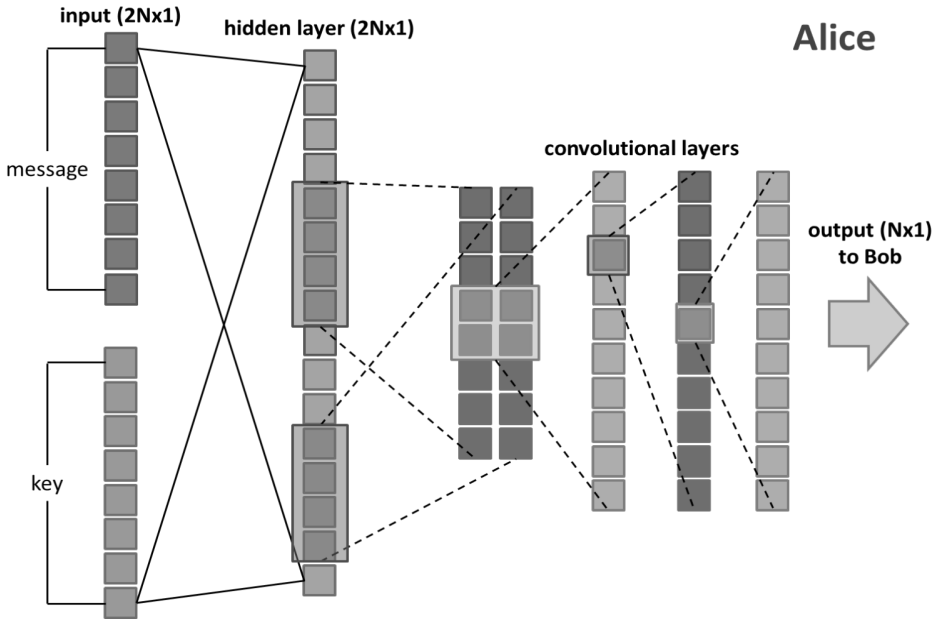


Рис. 3.6. Архітектура мережі «Alice»

Ідеєю моделі шифрування є спільне навчання мереж «Alice» і «Bob» для безпечного обміну даними, шляхом використання генеративних

методик конкуруючих нейронних мереж. При цьому одним з завдань навчання цих мереж є перемога над мережею «Eve», тобто мережа-зловмисник не повинна розшифрувати початкове повідомлення. Варто відзначити, що жодна з мереж заздалегідь не знає криптографічних методик, що сприяли б досягненню своїх завдань. Також важливим є те, що, відповідно до принципів генеративних конкуруючих мереж, мережі «Alice» і «Bob» прагнуть перемогти найкращу версію мережі «Eve», а не певну фіксовану версію цієї мережі. Діаграма діяльності для однієї ітерації навчання розглянутої моделі наведена на рис. 3.7.

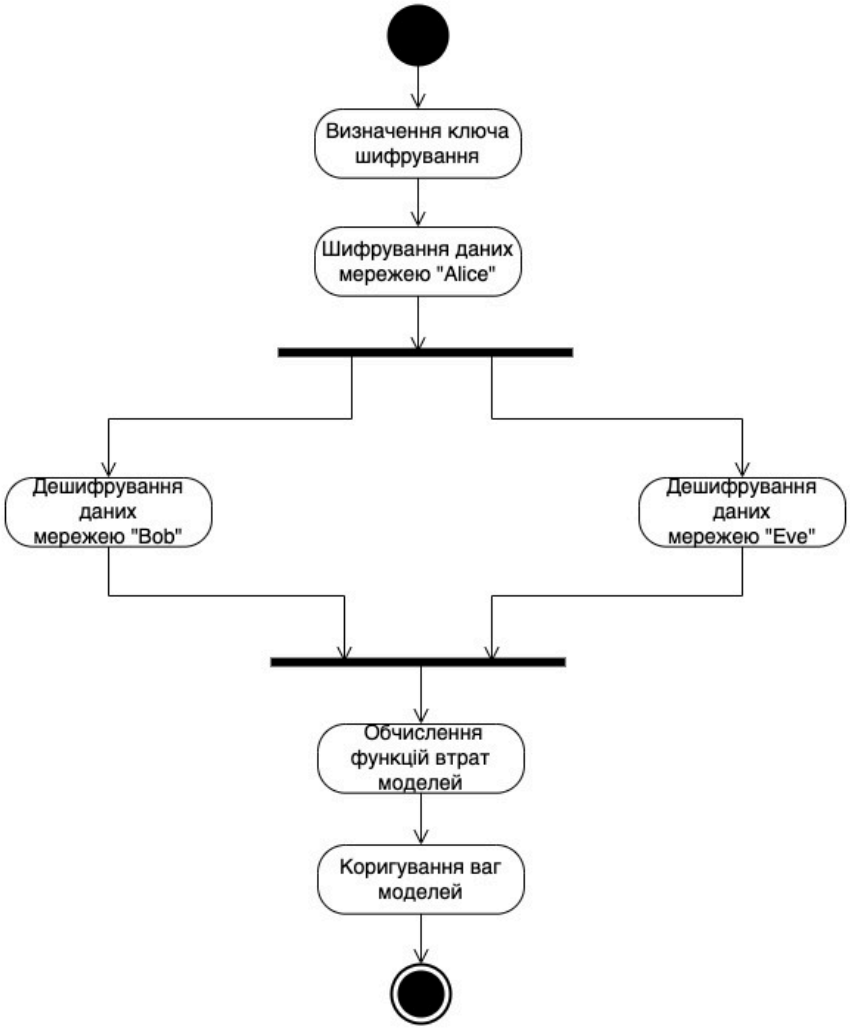


Рис. 3.7. Діаграма діяльності для однієї ітерації навчання моделі шифрування даних

У розглянутій моделі використовується дві різні функції втрат навчання [76-78].

Функцією втрат мережі «Eve» є абсолютна похибка дешифрування, яка визначається відстанню Хеммінга (d) між бітовим вектор початкового тексту (P) і текстом, який був дешифрований мережею «Eve» (P_{Eve}). Іншим способом визначення цієї функції втрат з огляду на бітове представлення тексту є норма l_1, яка обчислюється для результату виконання бітової операції XOR, що застосовується до зазначених векторів. Математично цю функцію можна записати за допомогою наступної формули:

$$L_{Eve} = d\left(P, P_{Eve}\right) = \sum_{j=1}^{N} \left| P_j - \left(P_{Eve}\right)_j \right| = \left\| P \oplus P_{Eve} \right\|_1 .$$

Функцію втрат для мереж «Alice» і «Bob» також визначено як абсолютну похибку дешифрування, тобто відстань Хеммінга (d) між бітовим вектор початкового тексту (P) і текстом, який був дешифрований мережею «Bob» (P_{Bob}). Однак, ця функція втрат уточнюється за допомогою додаткового параметру $((N/2 - L_{Eve})^2 / (N/2)^2$, у якому N – довжина вхідного тексту в бітах. Цей додатковий параметр дозволяє врахувати ефективність дешифрування, яке здійснює мережа «Eve». Отже, цю функцію втрат можна сформулювати наступним чином: $L_{Bob} = \left\| P \oplus P_{Bob} \right\|_1 + \dfrac{\left(\dfrac{N}{2} - L_{Eve}\right)^2}{\left(\dfrac{N}{2}\right)^2}$.

У дослідженні [69, 75] визначено гіперпараметри моделі шифрування, серед яких розмір порції даних (mini-batch size), оптимізатор та підхід до процесу навчання. Одна порція даних має розмір від 256 до 4096 бітів. Для вирішення задачі оптимізації використано оптимізатор Адама з коефіцієнтом швидкості навчання 0,0008. Процес навчання організовано почергово фіксуючи мережу «Eve» або мережі «Alice» і «Bob», тобто під час опрацювання однієї порції даних мережа «Eve» є фіксованою (не

навчається), в той час як мережі «Alice» і «Bob» навчаються; потім під час опрацювання двох інших порцій даних реалізується навчання мережі «Eve», при цьому мережі «Alice» та «Bob» залишаються незмінними.

На рис. 3.8 представлено результати навчання моделі шифрування, яка використовує конкуруючі нейронні мережі [69]. Згідно з графіком, перш ніж мережі «Bob» і «Eve» почнуть відтворювати початковий текст, проходить близько 8000 кроків навчання. Далі протягом ще приблизно 2000 кроків мережі «Alice» та «Bob» ускладнюють шифрування, завдяки чому похибка дешифрування мережі «Eve» починає підніматись. У цей момент комунікація стає більш захищеною й приблизно після 15000-го кроку навчання модель може використовуватись для шифрування.

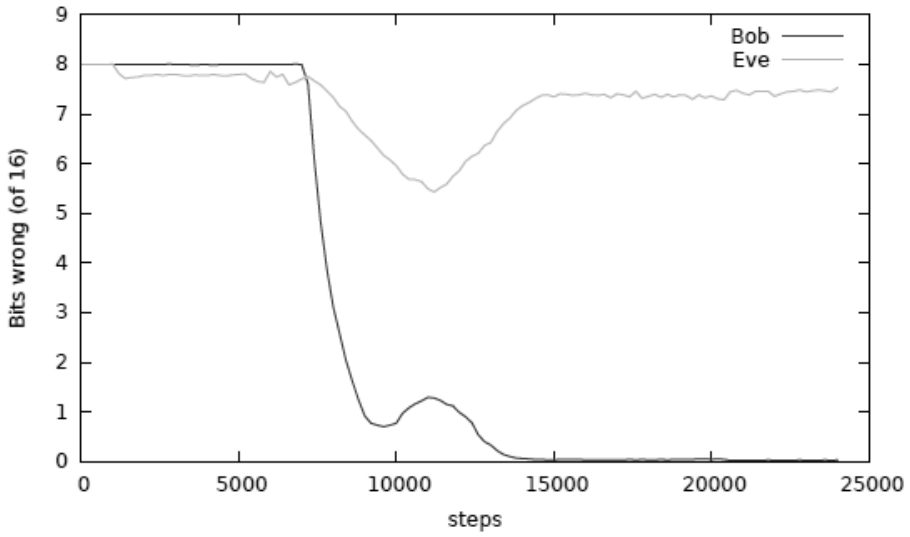


Рис. 3.8. Результати навчання моделі шифрування

Основною задачею класичної моделі шифрування є шифрування даних представлених, у вигляді набору біт. Однак, під час обробки візуальних даних, зазначений спосіб представлення інформації не є доцільним, оскільки такі дані найчастіше представляються як матриця пікселів. Зважаючи на це, вирішено модифікувати модель.

Запропоновано змінити структуру генеруючої мережі «Alice». Вхідні дані представляти у вигляді $N \times N$ - матриці пікселів зображення, розмірність якого було зменшено, та $N \times N$ - ключа шифрування. У такому випадку, прихований шар мережі – повністю з'єднаний шар розмірністю $N \times 2N$. Згорткові шари – це послідовність з трьох двовимірних згорткових шарів (Conv2D), функцією активації яких запропоновано використати сигмоїдну нелінійну функцію. Згідно запропонованої структури, вихідна інформація буде представлена матрицею N на N (зашифрованою матрицею пікселів зображення). По аналогії з класичною моделлю шифрування, структура мережі «Bob» має бути такою ж, як і в мережі «Alice»; в той час, як структура мережі «Eve», відрізнятиметься тільки першим шаром (адже на її вхід не буде подаватись матриця, що є ключем шифрування).

Отже, відмінністю запропонованої модифікації моделі шифрування даних від існуючої є використання двовимірних згорткових нейронних мереж, замість одновимірних, що спрощує її використання для візуальних даних (зокрема, зображень).

3.2. Алгоритмічно-програмний метод функціонального шифрування наборів даних

Базуючись на розглянутій модифікації програмної моделі шифрування даних, що використовує нейронні мережі, запропонованій в підрозділі 3.1, сформовано гіпотезу, що використовуючи подібні підходи можна функціонально зашифрувати вхідний набір приватних даних, що дозволить його використовувати в подальшому для аналізу.

Для досліджень обрано задачу класифікації, а даними, на яких сфокусовано дослідження для перевірки гіпотези обрано зображення, які представлені набором пікселів.

Діаграму діяльності розробленого методу функціонального шифрування наборів даних, що представлені у вигляді набору пікселів (екземплярів-зображень), наведено на рис. 3.9.

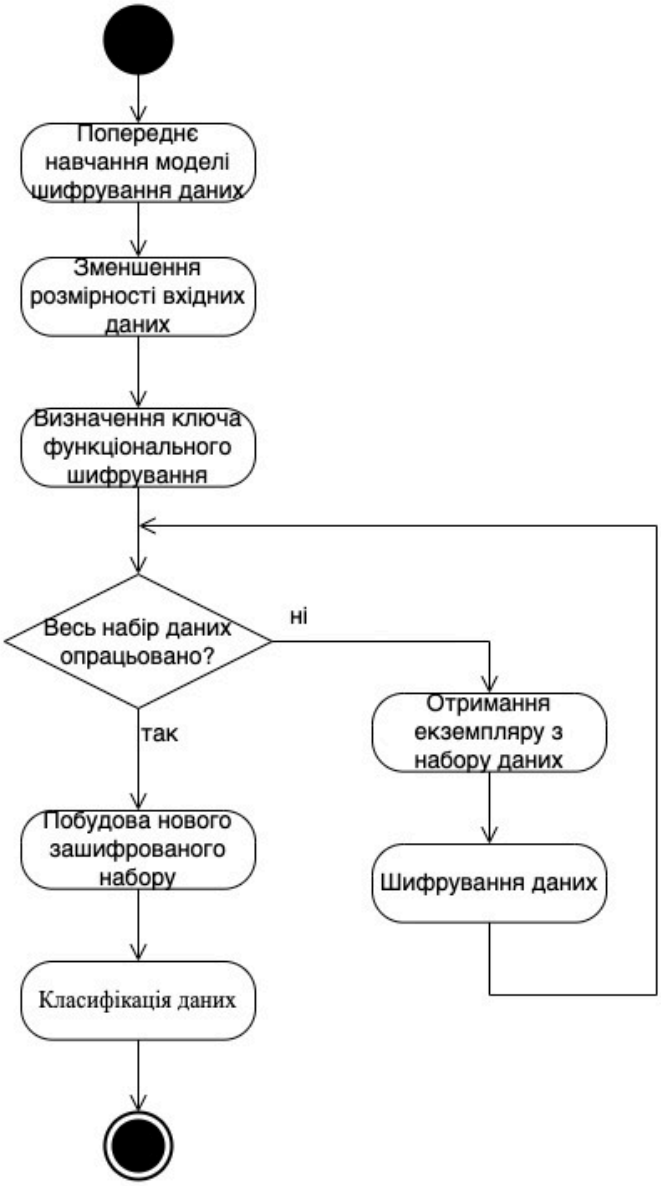


Рис. 3.9. Діаграма діяльності методу функціонального шифрування наборів даних

Розроблений метод полягає в виконанні наступних 4-х етапів [79]:

1. Здійснення попередньої обробки вхідних даних, шляхом зменшення розмірності екземплярів-зображень.

2. Застосування модифікованої моделі шифрування даних з використанням нейронних мереж до кожного зображення з набору даних. Основна ідея цієї моделі розглянута у підрозділі 3.1.
3. Побудова нового зашифрованого набору даних з зашифрованих зображень та ключа шифрування.
4. Класифікація на основі зашифрованого набору даних.

Розглянемо етапи методу докладніше.

Етап здійснення попередньої обробки вхідних даних полягає у зменшенні розмірності вхідних даних (кількості пікселів, що будуть опрацьовуватись), шляхом застосування комбінації згорткових та агрегувальних шарів до зображення. Це необхідно зробити з метою пришвидшення навчання нейронної мережі, оскільки вхідні дані зі зменшеною розмірністю будуть опрацьовуватись швидше.

Етап застосування модифікованої моделі шифрування даних передбачає використання навченої моделі для шифрування даних зменшеної розмірності. Запропоновано використовувати однаковий ключ шифрування для зменшення кількості даних, що потрібно зберігати. Результатом етапу застосування навченої моделі має бути зашифроване зображення. Варто зазначити, що передумовою для цього етапу є навчання модифікованої моделі шифрування зображень, яка використовує двовимірні згорткові шари й шифрує матриці пікселів вхідних зображень. Докладніше модифікація цієї моделі буде розглянута нижче.

Після застосування модифікованої моделі шифрування даних до кожного екземпляру даних, відбувається побудова нового (зашифрованого) набору даних для якого визначений ключ шифрування. Узагальнений процес побудови нового набору даних зображено на рис. 3.10.

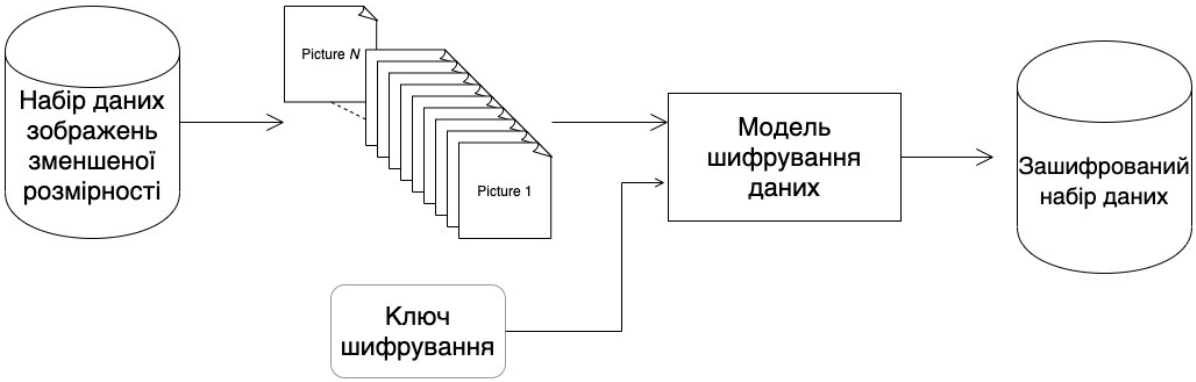


Рис. 3.10. Узагальнений процес побудови нового (зашифрованого) набору даних

Останнім етапом запропонованого методу є класифікація зашифрованих зображень, яку пропонується виконувати одним з наступних способів, безвідносно до методу класифікації, що використовується:

- класифікація зашифрованого набору даних без ключа, що використовувався для шифрування;
- класифікація з поєднанням зашифрованого набору даних й ключа шифрування.

Для першого способу, метод класифікації застосовується безпосередньо до зашифрованих даних, при цьому ступінь захищеності даних повинна бути кращою, тоді як точність класифікації – гіршою. При використанні другого способу, зашифровані зображення необхідно попереднього пропустити через додатковий повністю з'єднаний шар згорткової мережі, що побудований на основі ваг мережі «Bob», яка навчена дешифруванню даних з ключем шифрування й дозволить відтворити з незначною похибкою зображення зменшеної розмірності.

3.3. Аналіз метрик для оцінки алгоритмічно-програмних методів захисту наборів даних

Моніторинг і оцінка якості методів збереження приватності має важливе значення для забезпечення ефективного захисту приватних наборів

даних в машинному навчанні. Основними характеристиками, на які необхідно звернути увагу при розробці таких методів є оцінки втрати приватності, точності моделей машинного навчання, обчислювальних витрат та стійкості моделі до атак.

Оцінка втрати приватності дозволяє виміряти ступінь втрати конфіденційності, яка виникає в результаті використання методу збереження приватності. Її можна виміряти різними способами, серед яких інформаційна ентропія та розбіжність Кульбака-Лейблера.

Точність моделей машинного навчання можна оцінити за допомогою різних метрик, використання яких залежить від задачі, що вирішується. Докладніше метрики для класифікації розглянуті нижче.

Оцінка обчислювальних витрат є також важливою, оскільки методи захисту потребують додаткових ресурсів (зокрема, часу, пам’яті, пропускної здатності мережі). Таку оцінку можна зробити з використанням різним метрик, наприклад, час виконання програми, кількість ітерацій, обчислювальна складність алгоритму за нотацією Ландау та обсяг виділеної пам’яті.

Ще однією оцінкою, яку доцільно застосовувати до методів збереження приватності, є стійкість до атак. Докладно атаки на системи машинного навчання були розглянуті в підрозділі 1.2. Стійкість до атак можна оцінити різними способами, серед яких відсоток успіху атаки інверсії моделі або атак з висновком про членство [80].

Для проведення об’єктивної оцінки методів збереження приватності необхідно оцінювати комбінацією цих характеристик, оскільки треба впевнитись, що приватні дані не лише захищено, а й захищені дані дозволяють вирішувати задачі машинного навчання з високим рівнем точності.

Залежно від задачі машинного навчання, існують різні метрики для оцінки розроблених систем. Зокрема, існують різні метрики для оцінки вирішення задач класифікації, кластеризації, регресії та ранжування. Проте,

є метрики, що можуть використовуватися для кількох задач (наприклад, такою метрикою є точність). Оскільки запропонований метод шифрування даних буде використовуватись для задачі класифікації зображень, проаналізуємо метрики оцінки систем класифікації даних. Почнемо з метрик, що використовуються для бінарної класифікації, щоб зрозуміти логіку їх визначення, а далі узагальнимо розглянуті метрики для задачі класифікації з багатьма класами. У бінарній класифікації позитивний клас – один з двох класів, який представляє об'єкти або події, які система намагається визначити. Прикладом позитивного класу для системи фільтрації спаму електронної пошти є клас «спам», а клас «не спам» є негативним класом.

Основою для визначення метрик оцінки систем штучного інтелекту є матриця помилок. Структура матриці помилок подана в табл. 3.1.

Таблиця 3.1

Матриця помилок (confusion matrix)

		Передбачення	
		Позитивний клас	Негативний клас
Фактичний результат	Позитивний клас	True Positive (*TP*)	False Negative (*FN*)
	Негативний клас	False Positive (*FP*)	True Negative (*TN*)

Матриця помилок визначає чотири властивості системи машинного навчання, залежно від фактичного результату та отриманого передбачення. Кількість випадків (передбачень), коли модель правильно визначила позитивний клас, називається True Positive (*TP*). Кількість випадків, коли модель визначила негативний клас правильно це True Negative (*TN*). Кількість випадків, коли модель помилилася у визначенні позитивного

класу називається False Positive (*FP*). Кількість випадків, коли модель помилилася у визначенні негативного класу – False Negative (*FN*).

Точність моделі до певного значення (*Accuracy*) для задачі класифікації визначається як відношення кількості випадків правильної класифікації екземплярів даних ($N_{correct}$) до загальної кількості екземплярів даних (N_{total}) [81]:

$$Accuracy = \frac{N_{correct}}{N_{total}}.$$

Беручи до уваги, розглянуту термінологію, точність моделі до певного значення (*Accuracy*) для задачі бінарної класифікації визначається за наступною формулою:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{TP + TN}{N_{total}}.$$

Для побудови надійних і високоточних систем на невідомих даних, іноді недостатньо однієї метрики, оскільки вона може не в повній мірі відображати ефективність системи. Наприклад, метрику точності моделі до певного значення, недоцільно застосовувати для незбалансованих наборів даних. Для таких випадків, кращі результати покажуть інші метрики, зокрема прецизійність або точність позитивних передбачень (*Precision*) та повнота (*Recall*).

Прецизійність або точність позитивних передбачень (*Precision*) класу у задачі класифікації обчислюється як співвідношення кількості правильних позитивних результатів (*TP*) до загальної кількості елементів, мітка класу яких є позитивною. Формально це можна сформулювати наступним чином:

$$Precision = \frac{TP}{TP + FP}.$$

Якщо точність є високою, то модель передбачає значно більше доречних (релевантних) результатів, ніж недоречних.

Повнота (*Recall*) для задачі класифікації визначається як співвідношення правильних позитивних результатів (*TP*) до загальної

кількісті елементів, які відносяться до позитивного класу. Тобто це частота правильних рішень моделі, яку можна обчислити за формулою [82]:

$$Recall = \frac{TP}{TP + FN}.$$

Якщо рівень повноти моделі є високим, то така модель передбачає більшість доречних (релевантних) результатів.

Ще однією метрикою, яка часто використовується, є міра F_1. Вона дозволяє поєднати розглянуті метрики точності позитивних передбачень (*Precision* та *Recall*). Її обчислюють як середнє гармонійне цих метрик:

$$F_1 = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall}.$$

Використовуючи міру F_1, можна зменшити вплив екстремальних значень у наборі даних на оцінку моделі.

Також однією з метрик оцінки ефективності бінарної класифікації є ROC-крива, що є графіком, який дозволяє побачити ефективність моделі класифікації для всіх порогових параметрів. Параметри цієї кривої наступні:

- Рівень правильних позитивних результатів – True Positive Rate (*TPR*). Він є синонімом повноти, тож визначається за формулою:

$$TruePositiveRate(TPR) = \frac{TP}{TP + FN}.$$

- Рівень неправильних позитивних результатів – False Positive Rate (*FPR*), який обчислюється за формулою:

$$FalsePositiveRate(FPR) = \frac{FP}{FP + TN}.$$

На рис. 3.11 наведено приклад ROC-простору.

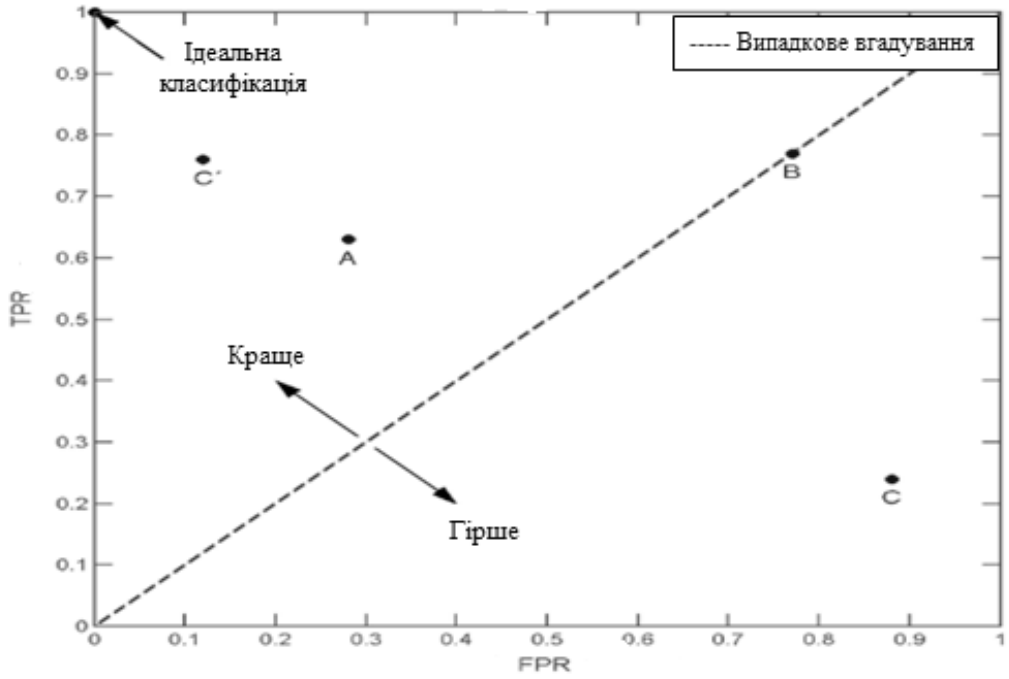


Рис. 3.11. Приклад ROC-простору

Відповідно до графіку точка (0;0) – стан класифікатора, який взагалі не робить передбачень приналежності екземпляру даних до позитивного класу; а точка (1;1) визначає стан моделі, коли вона завжди класифікує екземпляри даних як елементи позитивного класу. Пряма, що з'єднує ці дві точки називається прямою випадкового вгадування. Якщо результат цієї метрики належить цій прямі, то класифікація здійснюється неефективно, оскільки така ситуація рівнозначна вгадуванню класу. Прикладом такого результату на графіку є точка *B*. Якщо ж класифікатор показує ідеальні результати, то значення метрики має бути в точці (0;1). На розглянутому графіку найближче до цієї точки знаходиться точка *C′*. Важливою особливістю ROC-кривої є незалежність від збалансованості набору даних, як наслідок цього, вигляд кривої є сталим при зміні кількості екземплярів того чи іншого класу. Визначають також кількісну характеристику для цієї кривої, а саме міру *AUC*, яка обчислюється як площа області під кривою. Така характеристика є показником ефективності для класифікації: чим її значення вище, тим краще працює класифікатор.

Для узагальнення розглянутих метрик бінарної класифікації на багатокласову класифікацію необхідно розширити матрицю помилок. Матриця помилок для класифікації з багатьма класами, наведена у табл. 3.2.

Таблиця 3.2

Матриця помилок для багатокласової класифікації

		Передбачення			
		Клас 1	Клас 2	…	Клас N
Фактичний результат	Клас 1	X_{11}	X_{12}	…	X_{1N}
	Клас 2	X_{21}	X_{22}	…	X_{2N}
	…	…	…	…	…
	Клас N	X_{N1}	X_{N2}	…	X_{NN}

Для i-го класу класифікації з багатьма класами метрики оцінки ефективності моделі обчислюються за наступними формулами:

- $TP_i = X_{ii}$ – кількість випадків (передбачень), коли модель правильно визначила позитивний i-й клас;
- $TN_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{\substack{k=1 \\ k \neq i}}^{N} X_{jk}$ – кількість випадків, коли модель визначила негативний i-й клас правильно;
- $FP_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} X_{ji}$ – кількість випадків, коли модель помилилася у визначенні позитивного i-го класу;
- $FN_i = \sum_{\substack{j=1 \\ j \neq i}}^{N} X_{ij}$ – кількість випадків, коли модель помилилася у визначенні негативного i-го класу;

- $Accuracy_i = \frac{TP_i + TN_i}{N_{i\ total}} = \frac{\sum_{j=1}^{N}\sum_{k=1}^{N} X_{jk}}{N_{i\ total}}, j \neq i, k \neq i$ – точність моделі для *i*-го класу;
- $\left(Precision\right)_i = \frac{TP}{TP + FP_i} = \frac{\sum_{j=1}^{N} X_{jj}}{\sum_{j=1}^{N} X_{jj} + \sum_{\substack{j=1 \\ j \neq i}}^{N} X_{ji}}$ – прецизійність або точність позитивних передбачень для *i*-го класу;
- $\left(Recall\right)_i = \frac{\sum_{j=1}^{N} X_{jj}}{\sum_{j=1}^{N} X_{jj} + \sum_{\substack{j=1 \\ j \neq i}}^{N}\sum_{\substack{k=1 \\ k \neq i}}^{N} X_{jk}}$ – повнота для *i*-го класу.

Узагальнені метрики класифікації з багатьма класами обчислюються як середнє арифметичне значення відповідних метрик для кожного класу.

Ідеєю запропонованого методу функціонального шифрування наборів даних є забезпечення приватності даних, шляхом їх шифрування, таким чином, який дозволить вирішувати задачу класифікації з достатньою точністю, як на зашифрованих, так і на оригінальних даних. Крім цього, дослідження буде проводитись з використанням набору даних MNIST, який є збалансованим, тож основною метрикою для оцінки методу вирішено обрати точність (*Accuracy*) класифікації оригінальних та зашифрованих даних. При цьому варто провести оцінку точності використовуючи кілька алгоритмів для класифікації, щоб перевірити як обраний класифікатор впливатиме на результати запропонованого методу.

Основною метою модифікації моделі шифрування є спрощення обробки візуальних даних (зображень), тому для оцінки модифікованої моделі буде використана похибка дешифрування даних та тривалість її навчання (кількість кроків навчання для отримання достатньо навченої моделі).

3.4. Висновки

Докладно розглянуто математичне підґрунтя для побудови методу функціонального шифрування наборів даних з метою захисту приватних наборів даних, в основі якого використання нейронних мереж. Проаналізовано класичну модель шифрування даних (представлених набором біт), що поєднує в собі підходи симетричного шифрування та концепції генеративних конкуруючих нейронних мереж. Основу цієї моделі шифрування складають нейронні мережі, шари згорткової мережі яких розміщені в зворотному порядку. Завдяки цьому, мережі навчаються шифруванню самостійно, тобто без використання знань про криптографію та її алгоритми. Докладно розглянуто функції втрат та інші параметри, що використовувались при побудові моделі шифрування, для кожної з її складових (нейронних мереж).

Розроблено метод функціонального шифрування наборів даних, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах аналізу даних та штучного інтелекту шляхом зменшення їх розмірності й функціонального шифрування отриманих даних з використанням приватного ключа.

Розроблено модифікацію моделі шифрування даних, яка полягає у використанні двовимірних згорткових нейронних мереж і дозволяє застосовувати модель шифрування даних, що представлені набором пікселів, з яких складається зображення.

Проаналізовано метрики для оцінки запропонованого методу функціонального шифрування. Як наслідок, обрано метрику точності вирішення задачі класифікації даних, що будуть зашифровані запропонованим методом в порівнянні з класифікацією оригінальних даних. Для оцінки модифікованої моделі шифрування обрано тривалість навчання (кількість ітерацій) до досягнення достатнього рівня шифрування даних (похибка дешифрування мережі «Bob» повинна бути менше 0,1, а мережі «Eve» більше 0,49).

4. ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЗАХИСТУ ПРИВАТНИХ НАБОРІВ ДАНИХ

Для розроблення програмного забезпечення захисту приватних наборів даних, що дозволяє проаналізувати ефективність запропонованих методів вирішено розробити дві підсистеми:

- програмна системи виконання обчислень над елементами поля $GF(p^m)$ в процесі гомоморфного шифрування;
- програмна система вирішення задачі класифікації на приватних наборах даних.

Розробимо архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних та виконаємо проєктування згаданих програмних систем.

4.1. Архітектура програмної системи для вирішення задачі класифікації на основі приватних даних

Як було проаналізовано в підрозділі 1.3, основними способами захисту приватних наборів даних у задачах машинного навчання є генерація синтетичних наборів даних, обробка приватних наборів даних (анонімізація даних, диференціальна конфіденційність, гомоморфне шифрування) та федеративне навчання. Більшість із цих методів передбачає централізовану обробку навчальних даних системи. Іншими словами, дані потрібно збирати та зберігати на одному пристрої, який використовується для навчання системи інтелектуального аналізу даних. Однак існує й децентралізований архітектурний підхід – федеративне навчання. Ідея цього підходу полягає в тому, щоб навчити алгоритм штучного інтелекту на багатьох кінцевих пристроях або серверах, що містять локальні набори даних, які залишаються на пристрої під час навчання. Однак підхід федеративного навчання не вирішує всіх проблем систем машинного навчання, крім того, значна частина даних централізована і зберігається в хмарних сховищах, тому в

таких випадках використання такого підходу є недоцільним. Зважаючи на це, розробка альтернативної децентралізованої архітектури, яка мінімізує розглянуті обмеження, є актуальним завданням.

Основним завданням програмної системи для вирішення задачі класифікації на основі приватних даних є збереження приватності користувачів системи. З огляду на розроблені вимоги до програмного забезпечення в підрозділі 1.4, було розроблено архітектуру програмної системи, яка наведена на рис. 4.1.

В основі запропонованої архітектури програмної системи лежить клієнт-серверна архітектура та функціональне шифрування. Клієнт-серверна архітектура — це модель взаємодії комп'ютерних систем, де обчислювальні ресурси та функції розподілені між клієнтським і серверним компонентами. Основна ідея полягає в тому, що клієнти (користувачі) взаємодіють із серверами (зазвичай через мережу), щоб отримати доступ до ресурсів, які надає сервер. Важливою особливістю клієнт-серверної архітектури є розподіл обов'язків між клієнтськими та серверними програмами, що спрощує розробку, обслуговування та масштабування систем. Крім того, ця архітектура дозволяє підтримувати різні типи клієнтів і серверів, що робить її широко використовуваною в програмному забезпеченні. Функціональне шифрування – це тип шифрування, це тип шифрування, який забезпечує більш точний контроль над доступом до зашифрованих даних порівняно з традиційними методами шифрування. Докладніше функціональне шифрування розглянуто в підрозділі 3.1.

У розробленій архітектурі пропонується розмістити моделі шифрування та класифікатора на сервері (який може бути розташований локально або в хмарному середовищі), оскільки навчання таких компонентів вимагає значної кількості ресурсів. Крім того, в цьому випадку сервер дозволяє багатьом користувачам використовувати зазначені моделі.

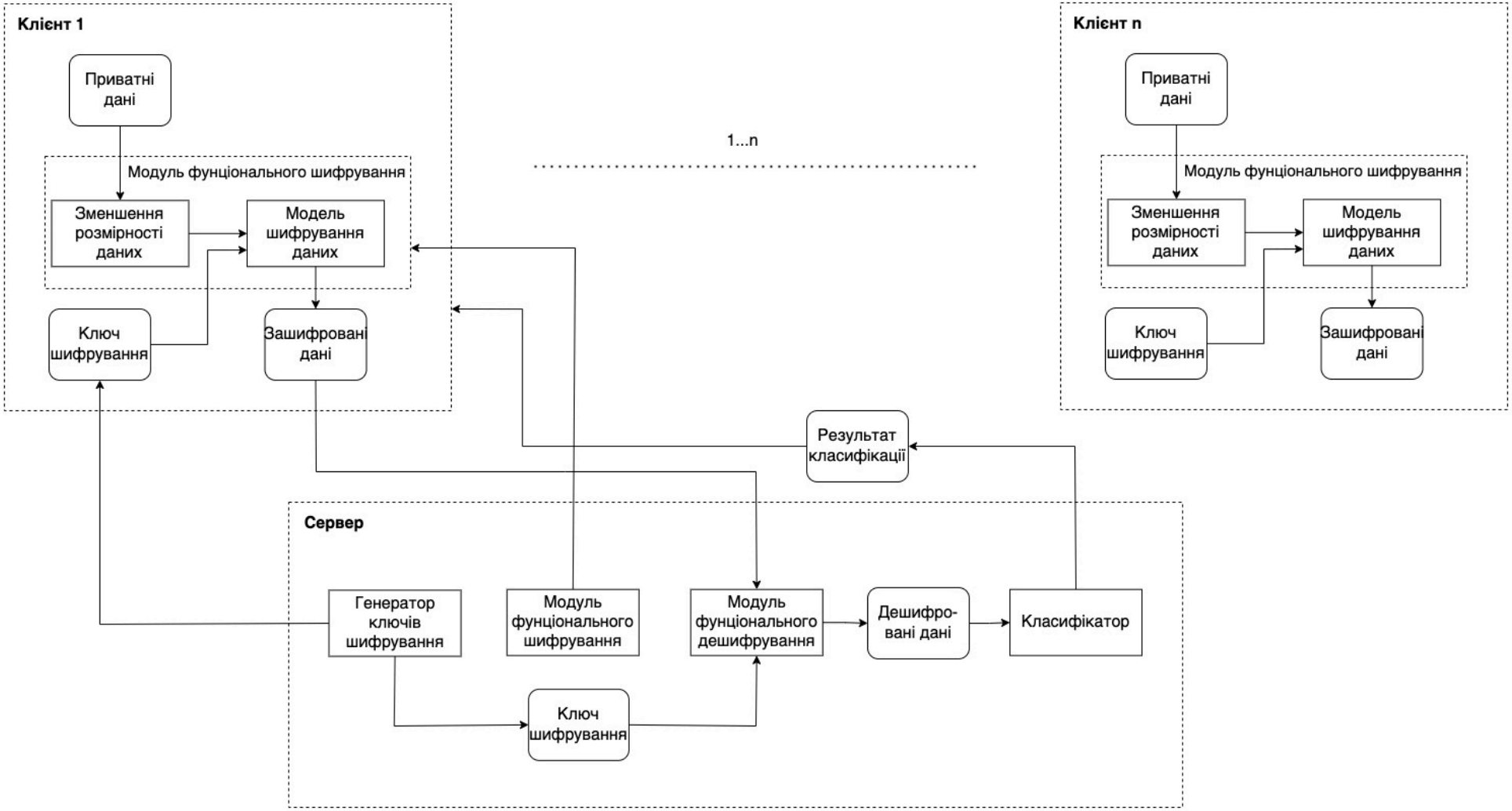


Рис. 4.1. Архітектура програмної системи для вирішення задачі класифікації на основі приватних даних

Відповідно до рис. 4.1, на сервері знаходиться генератор ключів шифрування, модуль функціонального шифрування, модуль функціонального дешифрування та класифікатор. Сервер опрацьовує зашифровані та дешифровані дані, а також генерує ключі шифрування.

На першому етапі клієнт завантажує собі модуль функціонального шифрування даних, попередньо навчений на даних заданої розмірності. Модуль функціонального шифрування реалізує метод функціонального шифрування, розроблений в підрозділі 3.2, й складається з підмодулю зменшення розмірності даних та моделі шифрування даних. Приватні дані користувача залишаються на клієнті. За потреби вони попередньо деанонімізовуються. Після цього відбувається зменшення розмірності приватних даних завантаженого модуля шифрування, де здійснюється зменшення розмірності вхідних даних та їх шифрування за допомогою програмної моделі розробленої в підрозділі 3.1. Ключ шифрування, що використовується в моделі, генерується на сервері й надсилається на клієнт. Результатом цього етапу є зашифровані дані.

Після цього сервер отримує зашифровані дані з клієнта й ключ шифрування з генератора ключів шифрування. Ці дані опрацьовуються модулем функціонального дешифрування, результатом чого є дешифровані дані, які можна використовувати для навчання, тестування й використання класифікатора. Результат роботи класифікатора надсилається на клієнт.

Кількість клієнтів, відповідно до розробленої архітектури системи, є необмеженою. Також може бути лише один клієнт. Це відрізняється від методів федеративного навчання, де важливо мати принаймні кілька незалежних користувачів, які мають достатню кількість даних для навчання. Структура клієнтів є однаковою, а ключ шифрування генерується окремий для кожного клієнту. Відповідно до розробленої архітектури, приватні дані не поширюються на сервер, за рахунок чого приватність користувачів системи зберігається.

Отже, запропоновано архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних, характерною особливістю якої є захист приватних наборів даних, шляхом функціонального шифрування, що відбувається на стороні клієнта, і дозволяє збільшити кількість наборів даних для навчання загальнодоступних систем аналізу даних і штучного інтелекту. Подальшими напрямками досліджень можуть бути розроблення різних реалізацій модулів, що використовуються в запропонованій архітектурі (зокрема, генератора ключів шифрування, модулів функціонального шифрування та дешифрування), а також проведення експериментальних дослідженнях ефективності застосування цієї архітектури для різних наборів даних і класифікаторів.

4.2. Проєктування програмної системи виконання обчислень над елементами поля $GF(p^m)$ в процесі гомоморфного шифрування

4.2.1. Архітектура системи

Для розроблення програмної системи обрано мову програмування C#, оскільки, вона надає можливості написання тексту програми як в об'єктно-орієнтованому, так і в функціональному стилі. Використовуючи її можна розділити логіку програмної системи на модулі й реалізувати графічний інтерфейс для користувача [83]. Для реалізації графічного інтерфейсу обрано графічну бібліотеку Windows Forms, адже головна задача програмної системи – проведення експериментальних досліджень, тобто головною вимогою до інтерфейсу користувача є зручність використання програми.

Розроблена система має модульну архітектуру, яка складається з таких модулів [84, 85]: виконання міжбазисних перетворень, виконання обчислень у поліноміальному базисі, виконання обчислень у нормальному базисі, проведення замірів часу виконання алгоритмів, генерація тестових даних, пошук нормальних поліномів, взаємодія з користувачем.

Діаграма класів, розроблених для реалізації програмного забезпечення, представлена у Додатоку В.

Під час розроблення програмної системи були використані такі шаблони проєктування [86, 87]: стратегія та шаблонний метод. Розглянемо докладно причини й особливості їх застосування.

Для виконання обчислень над елементами скінченних полів необхідно використовувати різні алгоритми залежно від заданого базису. З огляду на це, за допомогою шаблону проєктування «Стратегія» спроєктовано архітектуру ПЗ, яка дозволяє інкапсулювати базис елемента скінченного поля і, як наслідок, дозволяє замінювати конкретну реалізацію виконання операцій у базисі без застосування наслідування. Відповідний фрагмент діаграми класів з реалізацією алгоритмів виконання обчислень у різних базисах, представлено на рис. 4.2.

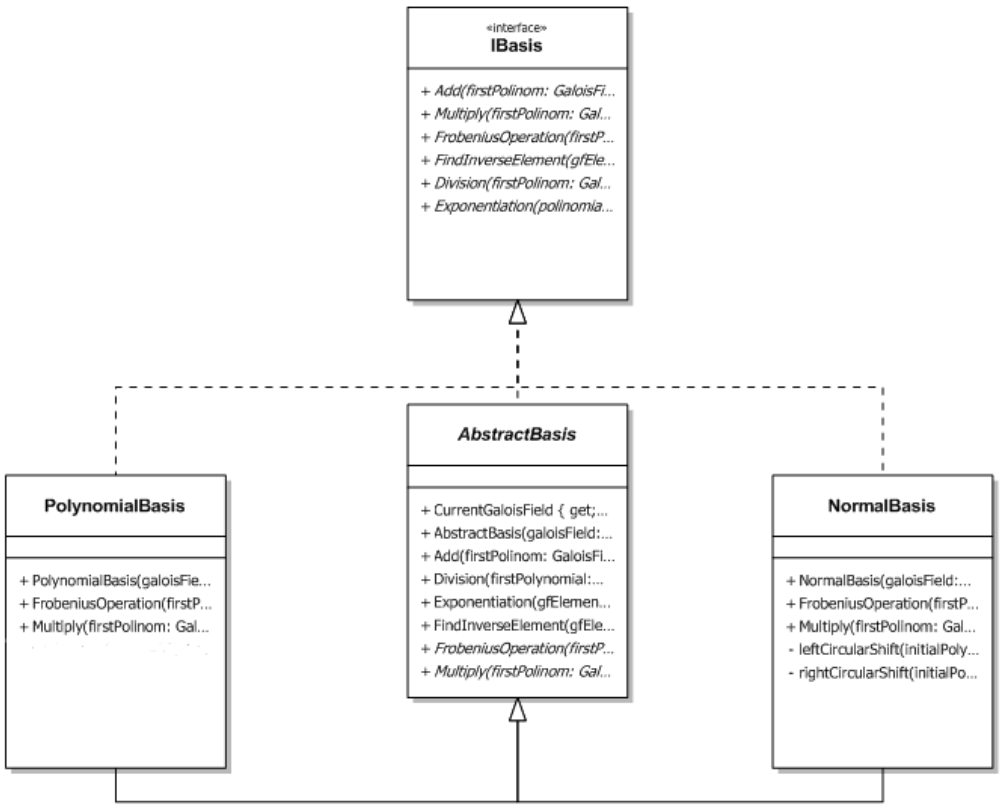


Рис. 4.2. Фрагмент діаграми класів, що реалізують алгоритми виконання обчислень у різних базисах

Аналіз часової складності виконання операцій над елементами скінченного поля, має однакову структуру, тож для реалізації цієї функціональності спроєктовано архітектуру ПЗ, яка реалізовує шаблон проєктування «Шаблонний метод», що дозволяє змінювати реалізацію методу виконання обчислень у підкласах не змінюючи кроки алгоритму в цілому [88]. Фрагмент діаграми класів з реалізацією цього архітектурного підходу зображено на рис. 4.3.

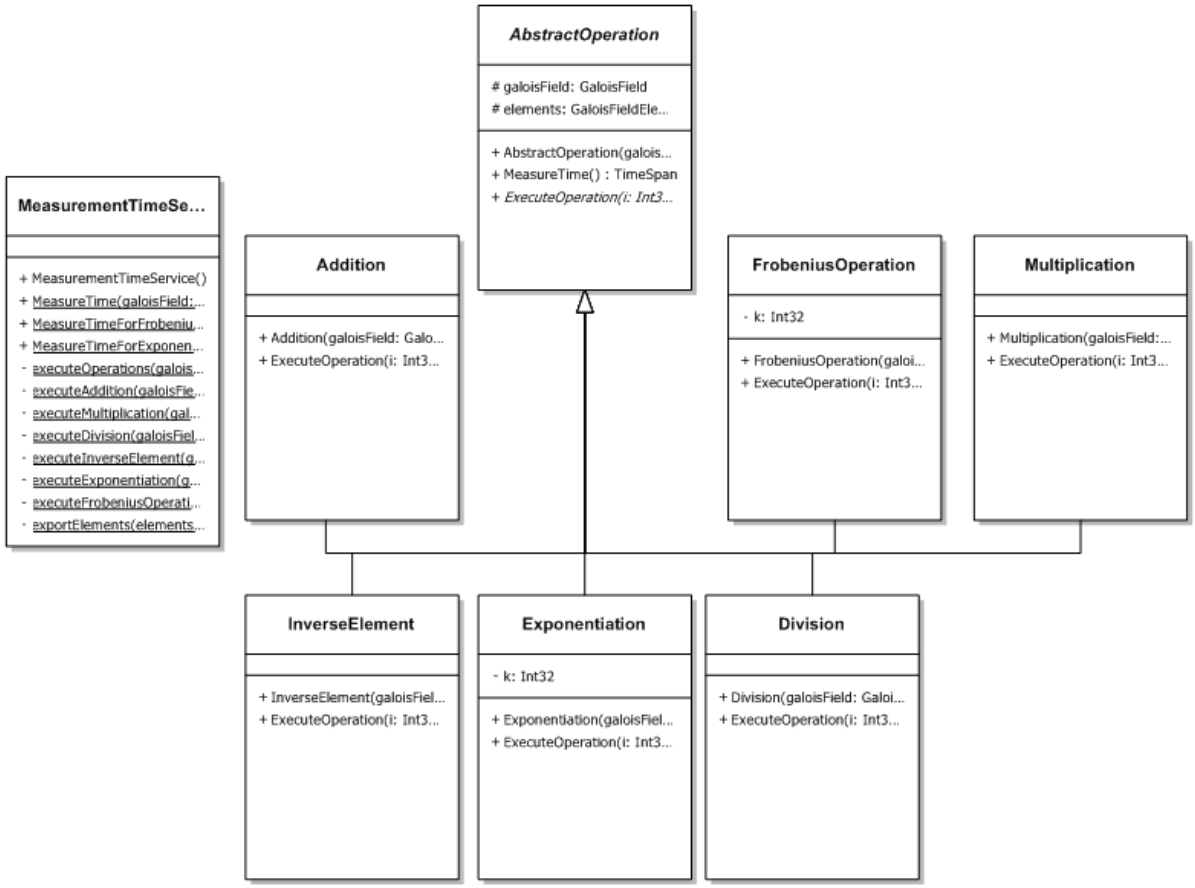


Рис. 4.3. Фрагмент діаграми класів, що реалізують аналіз часової складності виконання операцій у полі Галуа

Для взаємодії з програмною системою розроблено такі файлові інтерфейси:

- txt-файли з незвідними й нормальними поліномами;

- excel-файл з експериментальними результати виконання операцій;
- excel-файл з тестовими даними.

Файли з незвідними й нормальними поліномами використовуються у модулі пошуку нормальних поліномів та мають наступну структуру (приклади таких файлів наведені в Додатку З та в Додатку И):

1. Назва файлу задається за шаблоном «p m.txt», де параметри *p* та *m* визначають задане поле Галуа.
2. Коефіцієнти поліномів розміщуються за спаданням з розділювачем _ “ ”.
3. Кожен наступний многочлен розміщується з нового рядка.

Excel-файли з тестовими даними генеруються програмною системою для проведення експериментальних досліджень і використовуючи наступний інтерфейс:

1. Назва файлу відповідає шаблону «p m n.xlsx», де першими параметрами є характеристики поля Галуа, а останнім – кількість елементів, що генерується.
2. У вкладці «Irreducible Polynomial» розміщено нормальний поліном, коефіцієнти якого наведено за спаданням.
3. Вкладка «Matrix M» містить в собі матрицю переходу між базисами.
4. У вкладці «Inverse Matrix M» розміщено обернену матрицю переходу.
5. Згенеровані елементи поліноміального базису подаються за зростанням коефіцієнтів твірного елементу у вкладці «Elements In Polynomial Basis».
6. Відповідні їм елементи нормального базису представлені на вкладці «Elements In Normal Basis».

Файли з результатами експериментальних досліджень формується в модулі проведення замірів часу. Їх структура залежить від параметрів, що

використовуються при виконанні операцій. Так, для операцій, які не потребують додаткових параметрів формується excel-файл, який складається з наступних елементів:

1. Стовпці – операції, що досліджуються.
2. Рядки – базис, у якому задані елементи скінченного поля.
3. Час виконання операцій задається в мілісекундах.
4. Назва файлу відповідає шаблону – «Simple Operation – Result {0}», де {0} – це дата й час генерування файлу.

Формування файлів з експериментальними результатами виконання операцій з додатковим параметром реалізовано наступним чином:

1. Назва файлу відповідає шаблону – «{0} – Result {1}», де {0} – це метод, який досліджується, а {1} – дата й час генерування файлу.
2. У стовпцях розміщуються значення, які приймає додатковий параметр k.
3. Рядки – базис, у якому задані елементи скінченного поля.

4.2.2. Керівництво користувача

Інтерфейс користувача розробленої програмної системи представлений у Додатку Б. Він складається з 3-х вкладок: калькулятор у полі $GF(p^m)$, генерування вхідних даних та аналіз часової складності.

У вкладці «Калькулятор у полі $GF(p^m)$» система надає користувачу наступні можливості:

1. Виконання операції над елементами поля $GF(p^m)$: додавання, множення, операція Фробеніуса, обчислення мультиплікативно оберненого елементу, ділення та піднесення до степеня. Вхідними даними для зазначених операцій є:
    а) інформація про поле Галуа (General Information): параметри p, m, нормальний поліном;
    б) базис (поліноміальний, нормальний);
    в) обрана операція;

г) вхідні операнди;

д) для операцій Фробеніуса та піднесення до степеня важливий також додатковий параметр k.

2. Перетворення елемента поля $GF(p^m)$ з одного базису в інший. Вхідні дані:

   а) інформація про поле Галуа (General Information): параметри p, m, незвідний поліном;

   б) початковий многочлен;

   в) базис, у який потрібно перевести елемент;

   г) додатковий параметр «Calculate all and save results to file» дозволяє перевести всі елементи заданого поля в інший базис і записати цю інформацію в файл (важливо зазначити, що для великих значень параметра p^m ця функція є часовитратною).

У вкладці «Генерування вхідних даних» система надає таку функціональність:

1. Визначення нормальних поліномів шляхом аналізу файлу з незвідними многочленами.
2. Генерація елементів поля $GF(p^m)$. Вхідні параметри:

   а) інформація про поле Галуа: параметри p, m, нормальний поліном;

   б) кількість елементів, яку потрібно згенерувати.

У вкладці «Аналіз часової складності» система дозволяє отримати експериментальні результати виконання операцій у полі Галуа в поліноміальному й нормальному базисах. Вхідні дані:

1. Файл з поліномами у обох базисах.
2. Обрані операції час виконання, яких буде виміряно.
3. Для операцій Фробеніуса та піднесення до степеня додатково задається діапазон значень для параметра k.

4.2.3. Аналіз отриманих експериментальних результатів

Експериментальні дослідження здійснювались на персональному комп'ютері MacBook Pro, що має операційну систему macOS Venture, процесор Quad-Core Intel Core i5 з тактовою частотою 2.3 ГГц і оперативну пам'ять 8 Гб. Для роботи з користувацьким інтерфейсом, що написаний з використанням технології Windows Forms, використовувалась віртуальна машина з операційною системою Windows 10. Мова програмування C# використана для реалізації досліджуваних алгоритмів.

Виконання операцій у полі Галуа

Дослідження швидкості виконання операцій проведено з використанням таких вхідних даних: поле $GF(2^{24})$, нормальний поліном $x^{24}+x^{23}+x^{8}+x^{5}+x^{4}+x+1$, кількість згенерованих елементів – 100000.

Результати виконання операцій додавання, множення, обчислення мультиплікативно оберненого елемента й ділення наведені у табл. 4.1.

Таблиця 4.1

Результати виконання базових операцій у полі $GF(2^{24})$

Базис	Час виконання операції, мс			
	Додавання	Множення	Ділення	Обчислення мультиплікативно оберненого елементу
Нормальний	0,002047	1,218877	10,394179	9,262177
Поліноміальний	0,001912	0,109251	3,422463	3,299030

З отриманих експериментальних результатів можна встановити, що додавання є операцією незалежною від базисного представлення елемента й виконується за однаковий час. Операція множення у 10 разів швидше

виконується у поліноміальному базисі, ніж у нормальному. Обчислення мультиплікативно оберненого елемента займає у 2,5 рази більше часу для нормального базису, і, як наслідок, операція ділення в 3 рази швидше виконується у поліноміальному базисі.

Для проведення експериментальних досліджень виконання операції Фробеніуса у різних базисах використано вищенаведені вхідні дані з додатковим параметром k, який приймав значення від 1 до 5, 7, від 9 до 13. Результати виконання цієї операції наведені в табл. 4.2.

Таблиця 4.2

Результати виконання операції Фробеніуса у полі $GF(2^{24})$ для різних k

k	Базис	
	Нормальний	Поліноміальний
1	0,002436	0,114450
2	0,001679	0,225372
3	0,000997	0,339942
4	0,000994	0,451851
5	0,000992	0,557715
7	0,000982	0,777917
9	0,000972	0,989613
10	0,000990	1,095348
11	0,000996	1,205277
12	0,000999	1,310586
13	0,001014	1,417930

Аналізуючи результати наведені в табл. 4.2, можна зробити висновок, що операція Фробеніуса у нормальному базисі виконується за константний час, який в 47 разів менший за час виконання цієї операції у поліноміальному базисі для $k = 1$. Варто зауважити, що час виконання у поліноміальному базисі змінюється за лінійним законом відносно параметра k.

Експериментальні дослідження виконання операції піднесення до степеня виконувались на тих самих згенерованих елементах поля $GF(2^{24})$ з нормальний поліномом $x^{24}+x^{23}+x^8+x^5+x^4+x+1$, що й попередні заміри часу, однак, використовувались значення додаткового параметру k від 1 до 8, та від 10 до 13. Зазначені результати наведені на рис. 4.4.

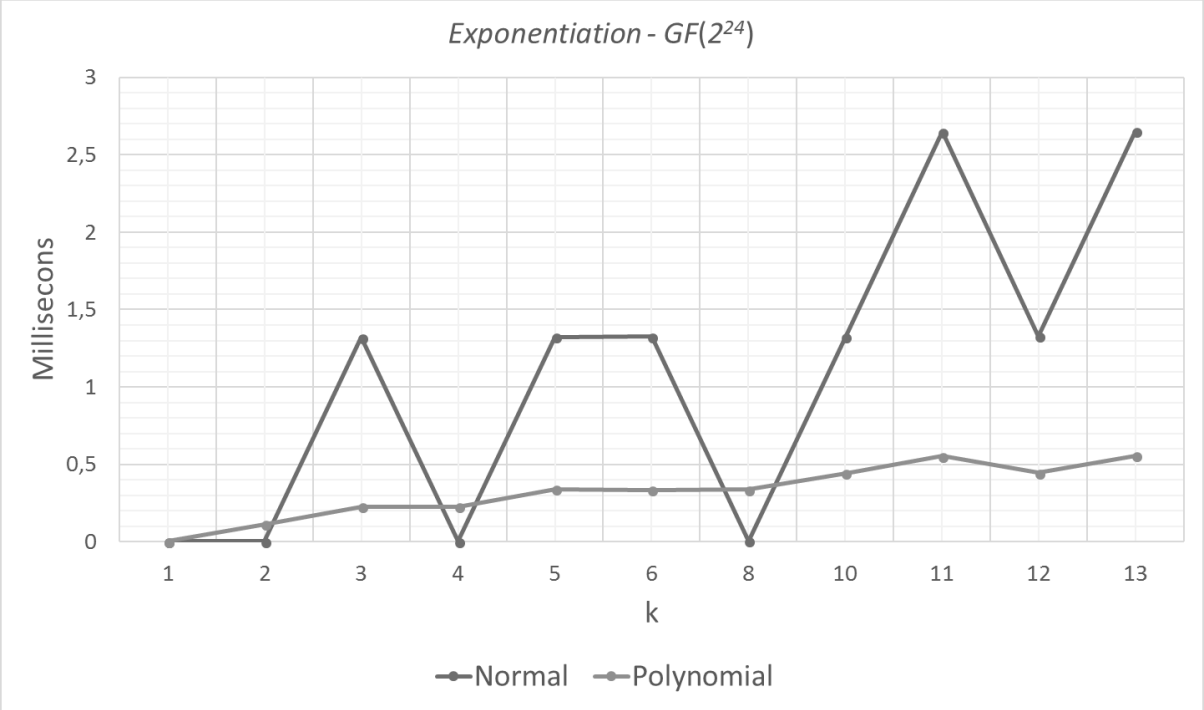


Рис. 4.4. Результати виконання операції піднесення до степеня у полі $GF(2^{24})$

Метод пошуку нормальних поліномів

Дослідження методу пошуку нормальних поліномів наведені у табл. 4.3.

З отриманих експериментальних результатів можна встановити, що пошук нормальних поліномів класичним методом для параметра $m < 10$ виконується за менший час, порівняно з запропонованим методом. Однак, приріст швидкодії запропонованого методу для параметра $m \geq 14$ є суттєвим: час пошуку нормальних поліномів запропонованим методом є меншим у понад 15 разів, порівняно з класичним методом. Для параметра $m = 16$ час пошуку нормальних поліномів запропонованим методом є у понад 150 разів меншим.

Таблиця 4.3

Результати пошуку нормальних поліномів

p	m	Час пошуку, мс	
		Класичний метод	Запропонований
2	8	8,24601	10,81466
2	10	47,79605	31,92607
2	14	18103,12345	315,0341
2	16	243103,12345	1514,77778

Модифікований спосіб побудови матриці переходу

Експериментальні результати швидкості побудови матриці переходу наведені у табл. 4.4.

Аналізуючи експериментальні результати, можна побачити, що запропонований спосіб показує більші значення часових показників для полів з параметром $m < 10$, порівняно з класичним способом ділення на незвідний поліном, однак, для полів з параметром $m > 12$ час виконання запропонованого способу є меншим у понад 5 разів порівняно з класичним способом. Таким чином, у полі $GF(2^{14})$ час побудови матриці переходу менший у 5 разів; у полі $GF(2^{16})$ – у 11, а для $m = 18$ – у 40 разів.

Таблиця 4.4

Результати побудови матриці переходу

p	m	Незвідний поліном	Час побудови, мс	
			Спосіб 1 (ділення на незвідний поліном)	Модифікований спосіб
2	8	$x^8 + x^7 + x^2 + x + 1$	0,0877433	0,1332861
2	10	$x^{10} + x^9 + x^4 + x + 1$	0,1923009	0,2208144
2	14	$x^{14} + x^{13} + x^3 + x^2 + 1$	2,016727	0,4689929
2	16	$x^{16} + x^{15} + x^4 + x + 1$	8,754337	0,6960259
2	18	$x^{18} + x^{17} + x^5 + x + 1$	37,6739521	0,8904218
2	20	$x^{20} + x^3 + 1$	164,2269162	1,1983284

4.3. Проєктування програмної системи вирішення задачі класифікації на приватних наборах даних

4.3.1. Аналіз технологій для розроблення системи

У процесі розроблення програмної системи вирішення задачі класифікації на приватних наборах даних, що реалізовує розроблений метод функціонального шифрування для захисту приватних наборів даних важливим є вибір технологій. Для розроблення вирішення поставленої задачі необхідно обрати мову програмування, технології для реалізації глибокого навчання та допоміжні бібліотеки (робота з наборами даних, засоби візуалізації результатів).

Проведемо докладний аналіз наявних технологій та визначимо, використання яких є найбільш доцільним для розроблення системи.

Мова програмування

Узагальнена характеристика мов програмування, що найчастіше використовуються для розроблення систем з використанням штучного інтелекту, наведена в табл. 4.5 [90-93].

Таблиця 4.5

Характеристика мов програмування

Мова	Переваги	Недоліки
Python	• велика й активна спільнота розробників; • широкий вибір бібліотек для роботи з даними; • широкий вибір фреймворків машинного навчання; • простий синтаксис; • крос-платформність;	• продуктивність (середній рівень); • динамічна типізація.
R	• наявність значної кількості методів для опрацювання даних; • відсутність плати за її використання; • крос-платформність;	• мала кількість засобів для роботи з алгоритмами машинного навчання; • низька продуктивність виконання програм; • високий «поріг входження», зважаючи на складний синтаксис;
C++	• найкраща продуктивність; • відсутність плати за її використання;	• відсутність збирача сміття (garbage collector); • складний синтаксис.

Відповідно до наведеної характеристики мов програмування, основними мовами для розроблення систем з використанням штучного інтелекту є Python, R та C++ [90]. Мова програмування Python використовується для розроблення систем з використанням машинного навчання, оскільки наявна велика й активна спільнота розробників, доступний широкий вибір бібліотек для роботи з даними і фреймворків для використання машинного навчання. Мова R – це спеціалізована мова для прогнозної аналітики, статистичного аналізу та візуалізації даних [92]. Основними перевагами якої є наявність значної кількості методів для опрацювання даних, відсутність плати за її використання та можливість виконувати текст програм незалежно від операційної системи. Іншою мовою, яка використовується для розроблення систем штучного інтелекту є мова програмування C++. Ця мова дозволяє досягти найкращої продуктивності для зазначених систем, тому найоптимальніше її використовувати в проєктах, де час і правильний розподіл ресурсів є критично важливим [91].

З огляду на проведений аналіз, мову Python обрано мовою для розроблення програмного забезпечення, оскільки вона надає широкий вибір засобів для побудови систем штучного інтелекту, має простий синтаксис, а також є найчастіше використовується для розроблення подібних систем.

Технології для реалізації глибокого навчання

Існує досить багато різних технологій для роботи з моделями глибокого навчання. Оскільки обрано мову програмування Python, то розглянемо найпопулярніші з технологій, що підтримують роботу з цією мовою: TensorFlow, PyTorch, Keras, Theano [94].

TensorFlow – це один з найпопулярніших фреймворків глибокого навчання [95]. Він є гнучким у налаштуванні й надає великий набір інтерфейсів різних рівнів абстракції для створення і навчання моделей машинного навчання. Важливою його перевагою є підтримка розподілених

обчислень, які дозволяють значно прискорити час навчання моделей. Крім цього, корисним є інструмент візуалізації TensorBoard, що використовується для налагодження, моніторингу та аналізу моделей. Іншою важливою перевагою є велика та активна спільнота розробників й наявність великої кількості навчальних ресурсів. Основними недоліками фреймворку TensorFlow є високий поріг входження, передусім при використанні низькорівневих інтерфейсів, й у деяких випадках виникають проблеми сумісності з різними версіями фреймворку та інших бібліотек.

PyTorch – це ще один популярний фреймворк глибокого навчання [96]. Основними його перевагами є простота використання, динамічний обчислювальний граф (який дозволяє змінювати структуру моделі під час її виконання, й забезпечує простоту використання під час створення складних моделей) і підтримка прискорення GPU (що дозволяє швидке навчання та виконання моделей). Проте, фреймворк PyTorch працює повільніше, ніж інші, для великомасштабних розподілених обчислювальних завдань. Крім цього, спільнота розробників є меншою, як і наявність доступних ресурсів для навчання, порівняно з TensorFlow.

Keras – це фреймворк, що надає високорівневий інтерфейс для побудови глибоких нейронних мереж, що написаний мовою Python. Він працює як надбудова над іншими фреймворками, такими як TensorFlow, Theano та PyTorch [97]. Він дозволяє розробникам швидко і просто будувати, тренувати та експериментувати з нейронними мережами, оскільки є зручним для користувача та надає простий інтерфейс для їх побудови. Зручність і простота інтерфейсу фреймворку Keras пришвидшує створення прототипів. Також фреймворк надає великий набір готових моделей, що можуть бути легко адаптовані до конкретного завдання. Основними недоліками цієї технології є обмежена гнучкість налаштування моделей та низька продуктивність на великих наборах даних.

Theano – це бібліотека мови Python, яка дозволяє ефективно працювати з математичними виразами, що включають багатовимірні масиви [98]. Вона дозволяє визначати, оптимізовувати та оцінювати вирази, пов’язані з глибоким навчанням. Бібліотека Theano є високопродуктивною, гнучкою й надає низькорівневі інтерфейси, що дозволяють змінювати параметри моделі. Крім цього, у бібліотеці Theano наявний значний набір готових моделей, які легко адаптувати для конкретних задач. Основними недоліками бібліотеки є високий поріг входження, значно менша спільнота розробників, як наслідок, менша кількість навчальних матеріалів, ніж в інших фреймворках. Також важливо відзначити, що час розроблення складних моделей з використанням цієї бібліотеки є довшим, порівняно з іншими розглянутими технологіями.

Беручи до уваги проведений аналіз, платформу TensorFlow та фреймворк Keras (її надбудову) обрано для розроблення ПЗ, що реалізує запропонований метод, оскільки ці технології надають необхідний рівень абстракції. Зокрема, ці технології дозволяють побудувати багатошарові нейронні мережі, використовувати різні метрики для оцінки моделі та зберігати й завантажувати ваги моделі.

Допоміжні бібліотеки

Для вирішення типових задач при розробленні систем з використанням штучного інтелекту обрано бібліотеки для попередньої обробки вхідних даних та візуалізації результатів.

Для попередньої обробки вхідних даних обрано бібліотеку Numpy. Ця бібліотека є базовою для наукових обчислень у мові Python. Вона використовується у великій кількості програм різного спрямування, від аналізу даних і машинного навчання до наукових досліджень й обробки сигналів, оскільки дозволяє легко опрацьовувати масиви даних, а також містить великий набір функцій лінійної алгебри [99].

Існує багато бібліотек, що вирішують задачу візуалізації отриманих результатів. Бібліотеки Matplotlib та Seaborn є найбільшими у мові Python. Бібліотека Matplotlib надає можливості побудови високоякісних графіків і діаграм, а також є незалежною від операційної системи й дозволяє працювати з бібліотекою Numpy, яка обрана для роботи з чисельним представленням даних. Бібліотека Seaborn краще працює з табличними даними, оскільки є інтегрованою з бібліотекою Pandas. Зважаючи на це, для розроблення ПЗ обрано бібліотеку Matplotlib.

4.3.2. Архітектура системи

Розроблена програмна система вирішення задачі класифікації на приватних наборах даних складається з двох модулів: модуль шифрування даних та модуль класифікації. У модулі шифрування даних розроблена логіка, що дозволяє здійснювати шифрування вхідних наборів даних (запропонованим методом і класичною моделлю шифрування). Реалізація модулю класифікації даних передбачає, що задача класифікації може бути вирішена як з використанням оригінальних, так і зашифрованих даних (з використання ключа шифрування й без нього). Діаграму класів розробленого програмного забезпечення подано на рис. 4.5.

Основними класами розробленої системи модулю шифрування даних є DatasetPreprocessor, ClassicCryptoNet, ModifiedCryptoNet. Клас ImageClassifier відноситься до модуля класифікації даних. Розглянемо ці класи детальніше.

Клас DatasetPreprocessor – клас, основною метою якого є інкапсуляція методу попередньої обробки набору даних. Єдиним методом класу є process_dataset, реалізація якого в класі відсутня за замовчуванням. Цей метод призначений для функціонального шифрування даних й визначається у класах-наслідниках.

Класи ClassicCryptoNet й ModifiedCryptoNet – класи, що реалізує шифрування даних. Клас ClassicCryptoNet реалізує логіку класичної моделі

що розглянута в підрозділі 3.1; тоді як, клас ModifiedCryptoNet є реалізацією модифікованої моделі, яка розроблена в підрозділі 3.1. Навчання генеративних конкуруючих нейронних мереж є досить часовитратним, тож для збереження результатів навчання реалізовано метод save_model в класах ClassicCryptoNet, ModifiedCryptoNet, який дозволяє зберігати стан навченої моделі (її ваги), та метод load_model, що завантажує ваги моделі в систему й відновлює її стан.

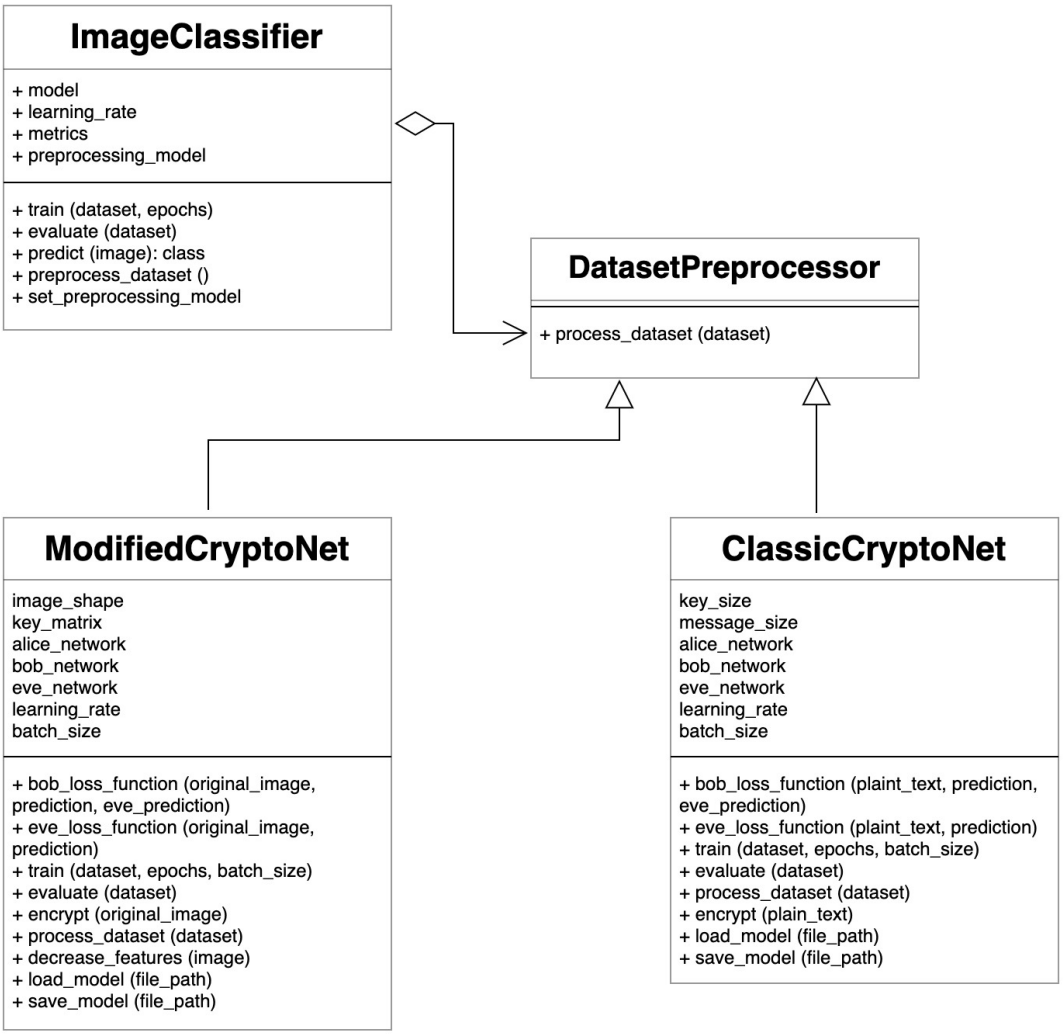


Рис. 4.5. Діаграма класів розробленої програмної системи

Клас ImageClassifier – клас, що призначений для класифікації зображень. Він містить метод processing_model, який дозволяє виконувати попередню обробку вхідних даних, шляхом виклику методу process_dataset

з класу DatasetPreprocessor. Завдяки цьому, можна виконувати класифікацію як оригінальних, так і зашифрованих даних.

Таким чином, для розроблення програмної системи використано шаблон проєктування «Стратегія» [84, 100] для інкапсуляції алгоритму попередньої обробки вхідного набору даних, який дозволяє підмінювати конкретну реалізацію в екземплярі класу класифікатор без наслідування класу.

4.3.3. Аналіз отриманих експериментальних результатів

Експериментальні дослідження здійснювались на персональному комп'ютері MacBook Pro, що має операційну систему macOS Venture, процесор Quad-Core Intel Core i5 з тактовою частотою 2.3 ГГц і оперативну пам'ять 8 Гб. Мова програмування Python використана для реалізації досліджуваних алгоритмів.

Набором даних для досліджень обрано набір даних MNIST, оскільки він є загальнодоступним, містить достатню кількість даних й є широко використовуваним у сфері штучного інтелекту. Набір даних складається з чорно-білих зразків рукописного написання арабських цифр (від 0 до 9). Кількість екземплярів навчального набору даних – 60000 зображень; набору для тестування – 10000 зображень. Розмірність зображеннь з набору даних є однаковою й складає 28 на 28 пікселів. Докладна характеристика набору даних MNIST з приклад екземплярів наведена у Додатку Й.

Варто відзначити, що обраний набір даних збалансований і складається з 10 класів. Як можна побачити у табл. 4.6 та табл. 4.7, кількість зображень кожної рукописної цифри варіюється від 5421 до 6742 в наборі даних для навчання й від 863 до 1127 в тестовому наборі даних. Це дозволяє провести навчання для вирішення задачі класифікації з високою точністю отриманих результатів.

Таблиця 4.6

Збалансованість навчального набору даних MNIST

Класи	«0»	«1»	«2»	«3»	«4»	«5»	«6»	«7»	«8»	«9»
Кількість екземплярів	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949

Таблиця 4.7

Збалансованість тестового набору даних MNIST

Класи	«0»	«1»	«2»	«3»	«4»	«5»	«6»	«7»	«8»	«9»
Кількість екземплярів	1001	1127	991	1032	980	863	1014	1069	945	978

Згідно з базовими підходами до розроблення програмного забезпечення з використанням штучного інтелекту [101], дані потрібно розділяти хоча б на дві незалежні частини: для навчання та для тестування. Однак, доцільним є застосування й ще однієї підмножини набору даних, а саме даних для затвердження (validation set), яка використовується на етапі навчання й дозволяє об’єктивно оцінити навчену модель в певний момент часу. Ця підмножина дозволяє регулювати гіперпараметри моделі під час навчання. Набір даних для затвердження не використовуються для навчання моделі, його використовують для перевірки моделі в кінці кожної епохи навчання. Для експериментальних досліджень проведено розподіл набору даних MNIST на три частини. Перший набір даних, який складає 5000 екземплярів даних (10 % з них – дані для затвердження (validation set)), сформовано для навчання моделі шифрування даних. Другий набір, що налічує інших 55000 екземплярів даних (20 % з них – дані для затвердження (validation set)), сформовано як навчальний набір даних для вирішення задачі класифікації. Третій набір даних налічує 10000 зображень з

навчального набору MNIST й використовується для тестування моделі вирішення задачі класифікації.

4.3.4. Метод функціонального шифрування

Експериментальні дослідження запропонованого методу функціонального шифрування проведено в два етапи: дослідження тривалості навчання класичної та модифікованої моделі шифрування; а також аналіз точності класифікації з використанням оригінальних та зашифрованих даних.

Під час дослідження тривалості навчання моделей шифрування даних з використанням нейронних мереж, коефіцієнт швидкості навчання (learning rate) для модифікованої моделі шифрування обрано таким, що дорівнює 0,0015. Обраним методом оптимізації для обох моделей є оптимізатор Адам, який є методом адаптивної оцінки моментів, оскільки для вирішення задачі оптимізації його результати є одними з кращих, відповідно до останніх досліджень. У результаті експериментів, розмір порції даних (batch size) обрано таким, що складає 150 екземплярів даних (зображень), а кількість епох навчання – 30, адже за такими параметрами результати виявились найкращими. На рис. 4.6 можна побачити експериментальні результати тривалості навчання класичної та модифікованої моделей шифрування даних.

Згідно з наведеним графіком, на початкових ітераціях навчання (1-1250) похибка дешифрування є високою, тобто нейронні мережі «Eve» та «Bob» обох моделей не можуть дешифрувати зображення. Під час ітерацій 1250-1750 мережі починають знаходити підхід до дешифрування, що дозволяє зменшити похибку дешифрування на 10-15 %. Протягом наступних 1000 ітерацій вплив ключа шифрування, що запропонований у модифікації моделі є суттєвішим. За рахунок цього, мережа «Bob» починає дешифрувати повідомлення з прийнятною похибкою, а похибка

мережі «Eve» поступово досягаю 50 %, що є відповідником вгадування інформації.

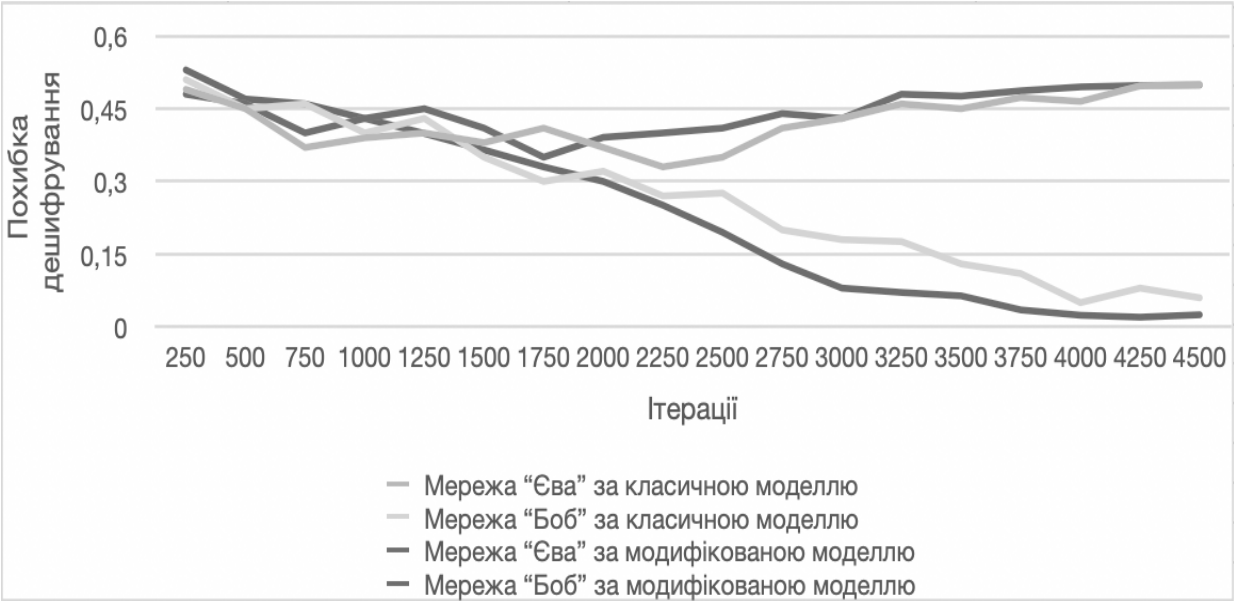


Рис. 4.6. Результати тривалості навчання класичної та модифікованої моделей шифрування даних

Отже, тривалість навчання за модифікованою є приблизно у 1,4 рази меншою, оскільки мережі «Bob» та «Eve» досягають потрібного значення похибки дешифрування раніше. Зокрема, значення похибки дешифрування мережі «Bob» є достатнім (менше 0,1) на 2850-й ітерації, в той час як в класичній моделі, таке значення досягається на 3950-й ітерації. Значення похибки дешифрування мережі «Eve» дорівнює необхідному (більше 0,49) починаючи з ітерації 3250 у моделі з модифікованою моделлю шифрування, тоді як у моделі з класичною – на 4250-й ітерації. З огляду на це, модифіковану модель можна використовувати для шифрування з 3250-ї ітерації, а класичну з 4250-ї.

Отже, запропонована модифікація моделі шифрування даних, екземпляри яких представлені у вигляді зображень, дозволяє скоротити тривалість навчання шифрування цих даних у 1,4 рази.

На рис. 4.7 подані експериментальні результати точності класифікації оригінального набору даних та зашифрованого набору, запропонованим методом функціонального шифрування. Вони проводились з використанням двох алгоритмів класифікації: алгоритму на основі згорткових мереж (CNN) та лінійного класифікатора. Відповідно до представленої діаграми, класифікація оригінальних даних за допомогою CNN-класифікатора показує на 4 % кращу точність, в порівнянні з класифікацією зашифрованих зображень з використанням інформації про ключ шифрування. При проведенні класифікації за допомогою лінійного класифікатора, для зашифрованих екземплярів даних з відомим ключем шифрування точність передбачень є на 2% меншою, ніж з використанням оригінальних даних. Однак, варто зазначити, що використання зашифрованого набору даних без ключа шифрування, показує недостатній ефект від навчання, оскільки точність класифікації зображень у цьому разі не досягає 64 %.

Рис. 4.7. Експериментальні результати точності класифікації оригінальних і зашифрованих даних

4.4. Аналіз шляхів інтеграції розроблених програмних систем

Одним з важливих завдань сучасної програмної системи є її інтеграція з іншими системами, тому що одне програмне забезпечення не може вирішити всі задачі з достатньою ефективністю. Оскільки у дисертації розроблено дві системи, одна з яких написана мовою C#, а інша – мовою Python, то проаналізуємо шляхи інтеграції систем, що використовують такі технології.

Одним з варіантів інтеграції є взаємодія через зовнішній інтерфейс. У цьому варіанті системи взаємодіють між собою за допомогою файлів, мережевих з'єднань (наприклад, сокетів) або API. Наприклад, програма написана мовою C# може генерувати певні дані й зберігати їх у файл, а потім програма написана мовою Python прочитає цей файл й буде використовувати ці дані. Іншим прикладом цього підходу є використання модулю subprocess у мові Python, завдяки якому можна запустити новий процес (програму мовою C#) й отримати результат його виконання. Приклад використання модулю subprocess [102] представлено на лістингу 4.1.

Лістинг 4.1. Приклад використання модулю subprocess для запуску коду, написаного мовою C#

```
      import shlex, subprocess
      command_line = input()
      /bin/vikings -input eggs.txt -output "spam spam.txt" -cmd "echo
'$MONEY'"
      args = shlex.split(command_line)
      print(args)
      ['/bin/vikings',   '-input',   'eggs.txt',   '-output',   'spam
spam.txt', '-cmd', "echo '$MONEY'"]
p = subprocess.Popen(args) # Success!
```

Іншим шляхом інтеграції розроблених систем є використання коду написаного мовою Python у програмній системі, що розроблена за

допомогою мови C#. Цього можна досягти за допомогою таких інструментів як Python.NET [103] або IronPython [104], які дозволяють запустити код Python у програмі C#. Завдяки цьому можна використовувати функції та бібліотеки мови Python у коді C#. На лістингу 4.2 представлено приклад використання бібліотеки IronPython.

Лістинг 4.2. Приклад використання бібліотеки IronPython

```
var eng = IronPython.Hosting.Python.CreateEngine();
var scope = eng.CreateScope();
eng.Execute(@"
def greetings(name):
    return 'Hello ' + name.title() + '!'
", scope);
dynamic greetings = scope.GetVariable("greetings");
System.Console.WriteLine(greetings("world"));
```

Ще одним подібним до попереднього варіантом інтеграції є використання мови C# у мові Python. За допомогою бібліотек на зразок Pythonnet можна імпортувати та використовувати бібліотеки та класи мови C# безпосередньо у коді програми, що написана мовою Python [105]. Приклад використання бібліотеки Pythonnet представлено на лістингу 4.3.

Лістинг 4.3. Приклад використання бібліотеки Pythonnet

```
import clr
dll = clr.FindAssembly('MyMath')  # returns path to dll
assembly = clr.AddReference('MyMath')
#print(type(assembly))                  #                  <class
'System.Reflection.RuntimeAssembly'>
#print(dir(assembly))
from MyMath import MyMathClass
from MyMath import MyMathClass as My


assert My.addInts(2, 3)         == 5
assert My.addInts(2.7, 7.8)     == 9
```

```
assert My.addDouble(11.2, 23.3) == 34.5
assert My.addString("hello", "world") == "hello world"
```

Альтернативним варіантом інтеграції програмних систем написаних різними мовами є архітектура мікросервісів. У системі, що складається з мікросервісів, деякі служби можуть бути написані мовою Python, а інші – мовою C#. Ці служби можуть взаємодіяти за допомогою інтерфейсів (API). Наприклад, кожен мікросервіс розроблений окремо на своїй власній мові та взаємодіє з іншими через RESTful API або системи повідомлень, такі як RabbitMQ.

Вибір методу залежить від потреб проєкту та переваг кожної мови для різних частин завдання. Вирішуючи, як поєднати Python і C#, важливо пам’ятати про такі речі, як продуктивність, обслуговування та вимоги проєкту.

Зважаючи на розглянуту інформацію, найоптимальнішим шляхом інтеграції розроблених систем є використання бібліотеки Pythonnet, що дозволить виконувати код написаній мовою C# на Python, оскільки розроблений метод функціонального шифрування є більш високорівневим відносно методів виконання операцій над елементами полів Галуа з використанням нормального й поліноміального базисів представлення. Крім цього, варто зазначити, що безпосереднє виконання коду є швидшим, ніж відповідна реалізація через інтерфейси (API), адже не потребує часу на формування й обробку відповіді, а також не потребує додаткових мережевих чи системних ресурсів.

4.5. Висновки

Отже, запропоновано архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних, характерною особливістю якої є захист приватних наборів даних, шляхом функціонального шифрування, що відбувається на стороні клієнта, і дозволяє збільшити

кількість наборів даних для навчання загальнодоступних систем аналізу даних і штучного інтелекту.

Розроблено програмне забезпечення захисту приватних наборів даних, що дозволяє дослідити розроблені методи захисту приватних наборів даних. Розроблене програмне забезпечення складається з двох незалежних програмних систем, шляхи інтеграції яких докладно проаналізовано.

Розроблено програмну систему, яка дозволяє виконувати обчислення над елементами поля $GF(p^m)$, проводити експериментальні дослідження, використовуючи поліноміальне й нормальне представлення елементів поля $GF(p^m)$, задавати різні значення вхідних параметрів p та m, а також генерувати різні набори тестових даних залежно від нормальних поліномів поля Галуа. У ході розроблення спроєктовано архітектуру ПЗ, особливістю якої є інкапсуляція базису елемента скінченного поля, яка дозволяє на основі шаблону «Стратегія» підмінювати конкретну реалізацію в екземплярі класу елементу поля Галуа без застосування наслідування. Розроблено спосіб побудови матриці переходу, час виконання якого для параметра $m > 12$ є у понад 5 разів меншим, порівняно з класичним способом (ділення $t^{p^{i+1}}$ на незвідний поліном). Таким чином, для $m = 16$ час побудови матриці переходу є в 11 разів меншим, ніж з використанням класичного способу. Розроблено метод пошуку нормальних поліномів для міжбазисних перетворень бінарних скінченних полів, який дає приріст швидкодії для параметра $m \geq 14$ у понад 15 разів. Так, для $m = 16$ час пошуку нормальних поліномів є більш, ніж у 150 разів меншим.

Другою розробленою програмною системою є система вирішення задачі класифікації на приватних наборах даних, що реалізує метод функціонального шифрування для захисту приватних наборів даних й дозволяє вирішувати задачу класифікації, використовуючи як оригінальні дані, так і зашифровані. Під час розроблення програмної системи використано шаблон проєктування «Стратегія» для інкапсуляції алгоритму

попередньої обробки вхідного набору даних. Це дозволяє без застосування наслідування замінювати конкретну реалізацію в екземплярі класу класифікатор. Розроблено метод функціонального шифрування наборів даних, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах аналізу даних та штучного інтелекту. Точність передбачень для задачі класифікації з використанням зашифрованого, запропонованим методом, набору даних з відомим ключем є меншою всього на 2-4 %, порівняно з класифікацією оригінального набору даних. Розроблено модифікацію моделі шифрування даних, що використовує двовимірні згорткові нейронні мережі, дозволяє скоротити у 1,4 рази тривалість навчання моделі шифрування зображень, у порівнянні з класичною моделлю, метою якої є шифрування бітів.

Проаналізовано шляхи інтеграції розроблених програмних систем, внаслідок чого використання бібліотеки Pythonnet, яке дозволить виконувати код написаний мовою C# на Python, визначено найоптимальнішим шляхом інтеграції; тому що розроблений метод функціонального шифрування є більш високорівневим відносно методів виконання операцій над елементами полів Галуа з використанням нормального й поліноміального базисів представлення.

ВИСНОВКИ

У дисертаційній роботі вирішено актуальну науково-технічну задачу удосконалення процесу розроблення загальнодоступних систем аналізу даних і штучного інтелекту на основі приватних даних, шляхом їх шифрування.

Результатом дисертаційного дослідження є отримання таких основних наукових результатів:

1. Проведено комплексний аналіз загроз приватності наборів даних, а також методів протидії їм. Проаналізовано переваги та недоліки таких методів збереження приватності у машинному навчанні.
2. Розроблено програмну систему, яка дозволяє виконувати обчислення над елементами поля $GF(p^m)$, проводити експериментальні дослідження, використовуючи поліноміальне й нормальне представлення елементів поля $GF(p^m)$, задавати різні значення вхідних параметрів p та m, а також генерувати набори тестових даних залежно від незвідних поліномів поля Галуа.
3. Показано, що для виконання операцій над елементами розширеного поля Галуа можна використовувати поліноміальний і нормальний базис. Залежно від обраного базису швидкість операцій відрізняється, зокрема в поліноміальному базисі у 4 рази швидше виконується операція множення, а обчислення мультиплікативно оберненого елемента – у 2 рази швидше. У той же час, в нормальному, операція Фробеніуса виконується за константний час незалежно від параметра k, при чому для $k = 1$ час виконання менший в 10 разів, порівняно з поліноміальним базисом.
4. Розроблено алгоритмічно-програмний метод пошуку нормальних поліномів для міжбазисних перетворень бінарних скінченних полів, незвідні поліноми у якому пропонується шукати як прості числа з

діапазону $\left[2^m;2^{m+1}\right)$ представлені у системі числення з основою 2. Це дає приріст швидкодії для параметра $m \geq 14$ у понад 15 разів.

5. Розроблено спосіб побудови матриці переходу, шляхом пошуку базисних елементів нормального базису на основі рекурентної формули $\alpha_{i+1}=t^{p^{i+1}}=t^{p^i \cdot p}=\left(\alpha_i\right)^p$, для зменшення витрат пам'яті та прискорення обчислень. Приріст швидкодії спостерігається при використанні зазначеного способу для параметра $m > 12$. Таким чином, для $m = 16$ час побудови матриці переходу є в 11 разів меншим, за стандартний спосіб.
6. Розроблено архітектуру програмної системи для вирішення задачі класифікації на основі приватних даних, характерною особливістю якої є захист приватних наборів даних, шляхом функціонального шифрування, що відбувається на стороні клієнта, і дозволяє збільшити кількість наборів даних для навчання загальнодоступних систем аналізу даних і штучного інтелекту.
7. Розроблено програмну систему вирішення задачі класифікації на приватних наборах даних, що реалізує метод функціонального шифрування для захисту приватних наборів даних й дозволяє вирішувати задачу класифікації, використовуючи як оригінальні дані, так і зашифровані. Під час розроблення програмної системи використано шаблон проєктування «Стратегія» для інкапсуляції алгоритму попередньої обробки вхідного набору даних.
8. Розроблено алгоритмічно-програмний метод функціонального шифрування наборів даних, особливістю якого є можливість використання приватних наборів даних в загальнодоступних системах аналізу даних та штучного інтелекту шляхом зменшення їх розмірності й функціонального шифрування отриманих даних з використанням приватного ключа. Точність передбачень для задачі класифікації з використанням зашифрованого, запропонованим

методом, набору даних з відомим ключем є меншою на 2-4 %, порівняно з класифікацією оригінального набору даних, що є допустимим рівнем втрати точності з метою збереження приватності.

9. Розроблено модифікацію програмної моделі шифрування даних, що використовує двовимірні згорткові нейронні мережі і дозволяє застосовувати модель шифрування даних, що представлені набором пікселів, з яких складається зображення. Модифікація моделі скорочує у 1,4 рази тривалість навчання моделі шифрування зображень, у порівнянні з класичною моделлю, метою якої є шифрування бітів.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Jobin A. Artificial Intelligence: the global landscape of ethics guidelines / A. Jobin, M. Ienca, E. Vayena. // Health Ethics & Policy Lab. — 2019.
2. Xu R. Privacy-preserving machine learning: Methods, challenges and directions / R. Xu, N. Baracaldo, J. Joshi. // arXiv preprint arXiv:2108.04417. — 2021 — DOI: 10.48550/arXiv.2108.04417.
3. Rigaki M. A Survey of Privacy Attacks in Machine Learning / M. Rigaki, S. Garcia. // ACM Computing Surveys. — 2023 — DOI 10.48550/arXiv.2007.07646.
4. A taxonomy and survey of attacks against machine learning / [N. Pitropakisa, E. Panaousisb, T. Giannetsos and others]. // Computer Science Review. — 2019. — DOI 10.1016/j.cosrev.2019.100199.
5. Membership Inference Attacks on Machine Learning: A Survey / [H. Hu, Z. Salcic, L. Sun and others]. — 2021 — DOI 10.48550/arXiv.2103.07853.
6. Practical Black-Box Attacks against Machine Learning / [N. Papernot, P. McDaniel, I. Goodfellow and others]. // Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security, Abu Dhabi, UAE. — 2016 — DOI 10.48550/arXiv.1602.02697.
7. Lauter K. Faculty Summit 2017: Private AI [Електронний ресурс] / Kristin Lauter // Microsoft Research. — 2017. — Режим доступу: https://www.microsoft.com/en-us/research/wp-content/uploads/2017/07/Private_AI_Kristin_Lauter.pdf.
8. Thaine P. Perfectly Privacy-Preserving AI [Електронний ресурс] / Patricia Thaine. — 2020. — Режим доступу: https://towardsdatascience.com/perfectly-privacy-preserving-ai-c14698f322f5.
9. Emam K. E. Practical Synthetic Data Generation / K. E. Emam, L. Mosquera, R. Hoptroff. — 2020. — 163 с. – O'Reilly Media, Inc. — ISBN 978-1492072744.

10. Synthetic Data for Machine Learning: Its Nature, Types, and Means of Generation [Електронний ресурс]. — 2022. — Режим доступу: https://www.altexsoft.com/blog/synthetic-data-generation/.
11. Synthetic Data Generation: Definition, Types, Techniques, and Tools [Електронний ресурс] — Режим доступу: https://www.turing.com/kb/synthetic-data-generation-techniques.
12. Generative Adversarial Nets [Text] / [I. J. Goodfellow, J. Pouget-Abadie, M. Mirza та ін.]. // Advances in neural information processing systems. — 2014. — Vol. 2 — P. 2672–2680.
13. Generative Adversarial Networks — Hot Topic in Machine Learning [Електронний ресурс]. — 2017. — Режим доступу: https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html.
14. Generative Adversarial Networks [Електронний ресурс] — Режим доступу: https://developers.google.com/machine-learning/gan.
15. Progressive Growing of GANs for Improved Quality, Stability, and Variation / T.Karras, T. Aila, S. Laine, J. Lehtinen. // International Conference on Learning Representations. — 2018 — DOI 10.48550/arXiv.1710.10196.
16. Data Anonymization Techniques [Електронний ресурс]. — 2019. — Режим доступу: https://www.solarwindsmsp.com/blog/data-anonymization-overview.
17. Guide to basic data anonymisation techniques [Електронний ресурс] // Personal Data Protection Commission Singapore (PDPC). — 2018. — Режим доступу: https://iapp.org/media/pdf/resource_center/Guide_to_Anonymisation.pdf.
18. Rubinstein I. S. Anonymization and risk / I. S. Rubinstein, W. Hartzog. // Washington Law Review — 2016. — №91. — С. 704–760.
19. Dwork C. The Algorithmic Foundations of Differential Privacy [Text] / C. Dwork, A. Roth. // Foundations and Trends® in Theoretical Computer Science. — 2014. — Vol. 9, №3-4. — С. 211–407. — DOI 10.1561/0400000042.

20. Sartor N. Explaining Differential Privacy in 3 Levels of Difficulty [Електронний ресурс] / Nicolas Sartor. — 2019. — Режим доступу: https://aircloak.com/explaining-differential-privacy/.
21. Minelli M. Fully homomorphic encryption for machine learning [Text] / Michele Minelli., 2018. — 157 p.
22. Microsoft SEAL: [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en-us/research/project/microsoft-seal/.
23. Lindell Y. Secure Multiparty Computation (MPC) [Електронний ресурс] / Yehuda Lindell — Режим доступу: https://eprint.iacr.org/2020/300.pdf.
24. Communication-efficient learning of deep networks from decentralized data. [Text] / [H. Brendan McMahan, E. Moore, D. Ramage та ін.]. — 2016.
25. Konečný J. Federated Optimization: Distributed Optimization Beyond the Datacenter [Електронний ресурс] / J. Konečný, B. McMahan, D. Ramage. — 2015. — Режим доступу: https://arxiv.org/pdf/1511.03575.pdf.
26. Brendan McMahan H. Federated Learning: Collaborative Machine Learning without Centralized Training Data [Електронний ресурс] / H. Brendan McMahan, D. Ramage. — 2017. — Режим доступу: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.
27. Kuchler H. Pharma groups combine to promote drug discovery with AI [Електронний ресурс] / Hannah Kuchler // Financial Times. — 2019. — Режим доступу: https://www.ft.com/content/ef7be832-86d0-11e9-a028-86cea8523dc2.
28. Северін А.І. Методи збереження приватності в машинному навчанні. / М.В. Онай, А.І. Северін // Вісник Хмельницького національного університету Серія: «Технічні науки». — 2023. — №6. — С. 274-280 — DOI: 10.31891/2307-5732-2023-329-6-274-280.
29. Северін А.І. Комплексний порівняльний аналіз методів збереження приватності в машинному навчанні. / М.В. Онай, А.І. Северін // Актуальні задачі сучасних технологій : зб. тез доповідей XII міжнар. наук.-практ. конф. Молодих учених та студентів, (Тернопіль, 6-7 грудня 2023) / М-во освіти і

науки України, Терн. націон. техн. ун-т ім. І. Пулюя [та ін.]. — Тернопіль: ФОП Паляниця В. А., 2023. — С. 406-407.

30. Lidl R. Finite Fields / R. Lidl, H. Niederreiter. — Cambridge: Cambridge University Press, 1996. — 755 p. DOI: 10.1017/CBO9780511525926.
31. Böhm J. Introduction to Algebraic Structures [Електронний ресурс] / J. Böhm, M. Marais. — 2019. — Режим доступу: https://agag-jboehm.math.rptu.de/~boehm/lehre/1920_MfI/introduction_to_algebraic_structures.pdf.
32. Castillo D. M. Group theory in algebraic structures [Електронний ресурс] / Daniel Macias Castillo. — 2020. — Режим доступу: https://www.cartagena99.com/recursos/alumnos/apuntes/estructuras_alg.pdf.
33. Онай, М.В. Спосіб перетворення многочленного подання елементів поля $GF(p^m)$ у степеневе / М.В. Онай, Ю.В. Вальчук // Шістнадцята всеукраїнська (одинадцята міжнародна) студентська наукова конференція з прикладної математики та інформатики СНКПМІ-2013: Тези доповідей, 11-12 квітня, 2013 р. — Львів : ЛНУ 2013. — С. 46-47.
34. Дичка І.А., Онай М.В. Спосіб зберігання в пам’яті ЕОМ різних форм подання елементів скінченного поля характеристики 2 // Комп’ютерні інтелектуальні системи та мережі. Матеріали VI Всеукраїнської WEB-конференції аспірантів, студентів та молодих вчених (19-21 березня 2013 р.). — Кривий Ріг: Криворізький національний університет, 2013. — С. 56-59.
35. Shuhong, G. Normal Bases over Finite Fields [Електронний ресурс]. / Gao Shuhong — Режим доступу: http://www.math.clemson.edu/~sgao/papers/thesis.pdf.
36. Advanced Encryption Standard (AES) [Електронний ресурс]. — Режим доступу: https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard.

37. Kim A. Revisiting Homomorphic Encryption Schemes for Finite Fields / A. Kim, Y. Polyakov, V. Zucca. // Advances in Cryptology — ASIACRYPT 2021. — 2022. — pp. 608-639.
38. Huynh D. CKKS explained: Part 1, Vanilla Encoding and Decoding [Електронний ресурс] / Daniel Huynh. – 2020. – Режим доступу до ресурсу: https://blog.openmined.org/ckks-explained-part-1-simple-encoding-and-decoding/.
39. A fully homomorphic encryption based on magic number fragmentation and El-Gamal encryption: Smart healthcare use case / [M. Kara, A. Laouid, M. A. Yagoub та ін.]. — 2021. — DOI: 10.1111/exsy.12767.
40. Westall, J. An Introduction to Galois Fields and Reed-Solomon Coding [Електронний ресурс] / J. Westall, J. Martin // School of Computing Clemson University Clemson, SC 29634-1906. — 2010. — Режим доступу: https://people.cs.clemson.edu/~westall/851/rs-code.pdf.
41. Fully Homomorphic Encryption from the Finite Field Isomorphism Problem / [Y. Doröz, J. Hoffstein, J. Pipher and others]. // Public-Key Cryptography — PKC 2018. — 2018. — pp. 125-155.
42. Modified Change-of-Basis Conversion Method in $GF(2^m)$ / I. A. Dychka, V. P. Legeza, M. V. Onai, A. I. Severin. // Radio Electronics, Computer Science, Control. — 2020. — №2. — С. 117–128 — DOI: 10.15588/1607-3274-2020-2-12.
43. Дичка, І.А. Організація спеціалізованих комп'ютерних систем для реалізації обчислень у скінченних полях / І.А. Дичка, В.І. Голуб, М.В. Онай // Вісник Східноукраїнського національного університету ім. В. Даля. — 2012. — №6 (177). — С. 268-278.
44. Дичка, І.А. Архітектура проблемно-орієнтованого процесора для реалізації арифметики скінченних полів / І.А. Дичка, М.В. Онай, О.В. Ващілін // Вісник Східноукраїнського національного університету ім. В. Даля. — 2012. — №12 (183) ч.2. — С. 99-106.

45. Дичка, І.А. Апаратна реалізація операторів та функцій в полях Галуа / І.А. Дичка, М.В. Онай, О.В. Ващілін // Вісник Хмельницького національного університету. — 2012. — Вип. 5. — С. 234-240.
46. Дичка, І.А. Апаратна реалізація обчислень у скінченних полях характеристики два / І.А. Дичка, М.В. Онай, Ю.В. Бухтіяров // Наукові вісті НТУУ “КПІ”. — 2013. — №6. — С. 20-27.
47. Дичка І.А., Онай М.В. Організація системи команд співпроцесора Галуа // Міжнародна науково-технічна конференція “Радіотехнічні поля, сигнали, апарати та системи”. Київ, 11-15 березня 2013 р.: матеріали конференції — Київ: 2013. — С. 212-213.
48. Дичка І.А., Онай М.В. Організація проблемно-орієнтованого процесора для реалізації операцій в полях Галуа виду $GF(2^m)$ // Тези доповідей Четвертої Міжнародної науково-практичної конференції “Методи та засоби кодування, захисту й ущільнення інформації” м. Вінниця, 23-25 квітня 2013 року. — Вінниця: ПП “ТД “Едельвейс і К”, 2013. — С. 270-273.
49. Онай, М.В. Спосіб апаратної реалізації операцій над елементами поля $GF(2^m)$ з використанням логарифма Зеча [Текст] / М.В. Онай, А.І. Дичка // Інтелектуальні технології в системному програмуванні (ІТСП-2015). IV Всеукраїнська науково-практична конференція молодих учених та студентів. Збірник наукових праць. Хмельницький, 22-24 квітня 2015 року. — Хмельницький : ПП Гонта А.С., 2015. — С. 51.
50. IEEE Standard Specifications for Public-Key Cryptography [Електронний ресурс]. — 2000. — Режим доступу: https://perso.telecom-paristech.fr/guilley/recherche/cryptoprocesseurs/ieee/00891000.pdf.
51. Дичка, І.А. Апаратна реалізація процедур множення і ділення многочленів у скінченних полях / І.А. Дичка, В.І. Голуб, М.В. Онай // Наукові вісті НТУУ “КПІ”. — 2012. — №5. — С. 61-66.
52. Глухов, В.С. Порівняння поліноміального і нормального базисів представлення елементів полів Галуа [Текст] / В.С. Глухов // Вісник

Національного університету "Львівська політехніка". — 2007. — № 591 : Комп'ютерні системи проектування. Теорія і практика. — С. 22–27.

53. Zhengbing Hu, I. A. Dychka, Onai Mykola, Bartkoviak Andrii, "The Analysis and Investigation of Multiplicative Inverse Searching Methods in the Ring of Integers Modulo M", International Journal of Intelligent Systems and Applications (IJISA), Vol.8, No.11, pp. 9-18, 2016. DOI: 10.5815/ijisa.2016.11.02.
54. Method of Performing Operations on the Elements of $GF(2^m)$ Using a Sparse Table / I. Dychka, M. Onai, A. Severin, C. Hu. // International Journal of Computer Network and Information Security (IJCNIS). — 2024. — Vol. 16, №1. — pp. 61-72 — DOI: 10.5815/ijcnis.2024.01.05.
55. Дичка І.А., Онай М.В. Способи знаходження мультиплікативного оберненого елемента в скінченних полях // Друга наукова конференція магістрантів та аспірантів присвячена 20-річчю факультету прикладної математики «Прикладна математика та комп’ютинг ПМК-2010»: Київ, 14-16 квіт. 2010 р.: зб. тез доп. / ред кол.: Дичка І.А. [та ін.] – К.: Просвіта, 2010. – С. 313-317.
56. Onai, M.V. Modification of Norton’s extended algorithm for search of multiplicative inverse element modulo *m* / M.V. Onai, T.P. Drozda // П’ятнадцята міжнародна наукова конференція імені академіка М. Кравчука, 15-17 травня, 2014 р., Київ : Матеріали конф. Т. 2. Алгебра. Геометрія. Математичний аналіз. — К. : НТУУ “КПІ”, 2014. — С. 25-26.
57. Дичка, І.А. Застосування *k*-арного методу Евкліда для пошуку мультплікативно оберненого елемента у кільці лишків за модулем m [Текст] / І.А. Дичка, М.В. Онай, А.Ю. Бартков’як // Матеріали статей П’ятої Міжнародної науково-практичної конференції «Інформаційні технології та комп’ютерна інженерія». м. Івано-Франківськ : п. Голіней О.М., 2015. — С. 151-153.
58. Онай, М.В. Знаходження мультиплікативно оберненого елемента у кільці лишків за довільним модулем методом Джої-Пейє [Текст] / М.В. Онай, А.Ю.

Бартков’як // Міжнародна науково-практична конференція «Проблеми інформатики та комп’ютерної техніки». Праці конференції. — Чернівці : Видавничий дім "Родовід", 2015. — С. 91-93.

59. Онай, М.В. Пошук мультиплікативно оберненого елементу у кільці лишків за довільним модулем методами, що грунтуються на модулярному піднесенні до степеня [Текст] / М.В. Онай, А.Ю. Бартков’як // Шістнадцята міжнародна наукова конференція імені академіка Михайла Кравчука, 14-15 травня, 2015 р., Київ : Матеріали конф. Т. 2. Алгебра. Геометрія. Математичний аналіз. — К. : НТУУ “КПІ”, 2015. — С. 139-141.
60. Lassak M. Fermat-Euler Theorem in Algebraic Number Fields / M. Lassak, S. Porubsky. // Journal of number theory. — 1996. — №60. — P. 254–290.
61. Introduction to Algorithms (3rd ed.) / T. H.Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. — 2009. — 1313 p. — MIT Press and McGraw-Hill — ISBN 978-0-262-03384-8.
62. Knuth D. E. The Art of Computer Programming, Vol. 1: Fundamental Algorithms (3rd ed.) / Donald E. Knuth. — 1997. — 672 p. — Addison-Wesley Professional — ISBN 978-0-201-89683-1.
63. Zindros D. A Gentle Introduction to Algorithm Complexity Analysis [Електронний ресурс] / Dionysis Zindros – Режим доступу: https://discrete.gr/complexity/.
64. Нотація Ландау та аналіз алгоритмів з прикладами на Python [Електронний ресурс] — 2019. — Режим доступу: https://devzone.org.ua/post/notaciia-landau-ta-analiz-algoritmiv-z-prikladami-na-python.
65. Онай, М.В. Спосіб генерування випадкового простого числа заданої довжини / М.В. Онай, О.С. Князькіна // П"ятнадцята всеукраїнська (десята міжнародна) студентська наукова конференція з прикладної математики та інформатики СНКПМІ-2012: Тези доповідей. — Львів : ЛНУ 2012. — С. 52-54.

66. Северін А.І. Модифікований підхід для побудови матриці міжбазисних перетвореннь у $GF(p^m)$. / М.В. Онай, А.І. Северін // Матеріали XI науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 13-14 грудня 2023 р.). — Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2023 — С. 110.
67. Panzade P. SoK: Privacy Preserving Machine Learning using Functional Encryption: Opportunities and Challenges [Електронний ресурс] / P. Panzade, D. Takabi. — 2022. — Режим доступу: https://arxiv.org/pdf/2204.05136.pdf.
68. Boneh D. Functional Encryption: Definitions and Challenges / D. Boneh, A. Sahai, B. Waters. // Lecture Notes in Computer Science. — Springer: Berlin, Heidelberg. — 2011. — Vol. 6597. — pp. 253-273 — DOI 10.1007/978-3-642-19571-6_16.
69. Abadi M. Learning to Protect Communications with Adversarial Neural Cryptography [Електронний ресурс] / M. Abadi, D. G. Andersen. — 2016. — Режим доступу: https://arxiv.org/abs/1610.06918.
70. Rodriguez J. Adversarial Neural Cryptography can Solve the Biggest Friction Point in Modern AI [Електронний ресурс] / Jesus Rodriguez. — 2018. — Режим доступу: https://towardsdatascience.com/adversarial-neural-cryptography-can-solve-the-biggest-friction-point-in-modern-ai-cc13b337f969.
71. Smart N. Cryptography: An Introduction / Nigel Smart. — London, St. Louis: Mcgraw-Hill College, 2004. — 433 p. — ISBN 978-0-07-709987-9, DOI 10.5555/1206247.
72. Schneier B. Applied Cryptography: Protocols, Algorithms and Source Code in C, 20th Anniversary Edition / Bruce Schneier. — 2015. — 784 p. — ISBN 978-1-119-09672-6.
73. CS231n Convolutional Neural Networks for Visual Recognition [Електронний ресурс] — Режим доступу: https://cs231n.github.io/convolutional-networks/.

74. Saha S. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way [Електронний ресурс] / Sumit Saha. — 2018. — Режим доступу: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.
75. Adversarial Neural Cryptography [Електронний ресурс]. — 2018. — Режим доступу: https://mathybit.github.io/adversarial-neural-crypto/#ai-based-encryption-poc.
76. Северін А. І. Метод шифрування даних з використанням нейронних мереж [Текст] / М. В. Онай, А. І. Северін. // Прикладна математика та комп'ютинг. ПМК, 2019 : дванадцята наук. конф. магістрантів та аспірантів, Київ, 13-15 лист. 2019 р. : зб. тез доп. — К.: Просвіта. — 2019. — С. 95–98 — ISBN 978-617-7010-16-5.
77. Triastcyn A. Generating Differentially Private Datasets Using GANs [Електронний ресурс] / A. Triastcyn, B. Faltings. — 2018. — Режим доступу: https://openreview.net/pdf?id=rJv4XWZA-.
78. Learning Perfectly Secure Cryptography to Protect Communications with Adversarial Neural Cryptography [Text] / [M. Coutinho, R. D. Albuquerque, F. Borges та ін.]. // Sensors (Basel). — 2018.
79. Северін А.І. Метод захисту набору даних зображень для вирішення задачі класифікації. / М.В. Онай, А.І. Северін // Прикладна математика та комп'ютинг. ПМК-2020 : тринадцята наук. конф. магістрантів та аспірантів, Київ, 18-20 листопада 2020 р. : зб. тез доп. / [ред кол.: Дичка І.А. та ін.] . — К. : Просвіта, 2020. — С. 221-226.
80. Song L. Systematic Evaluation of Privacy Risks of Machine Learning Models [Електронний ресурс] / L. Song, P. Mittal. — 2020. — Режим доступу: https://arxiv.org/pdf/2003.10595.pdf.
81. Singh B. Evaluation Metrics for Machine Learning Models [Електронний ресурс] / Bhajandeep Singh. — 2019. — Режим доступу:

https://heartbeat.fritz.ai/evaluation-metrics-for-machine-learning-models-d42138496366.

82. Провост Ф. Data Science для бізнесу. Як збирати, аналізувати і використовувати дані / Ф. Провост, Т. Фоусетт: пер. з англ. — Київ: Наш формат, 2019. — 400 с. — ISBN 978-617-7730-03-2.
83. C# Guide [Електронний ресурс]. — Режим доступу: https://docs.microsoft.com/en-us/dotnet/csharp.
84. Мартін Р. Чистий код: створення, аналіз, рефакторинг / Роберт Мартін., 2019. — 416 с. — ISBN 978-617-09-5285-1.
85. Мартін Р. С. Чиста архітектура / Роберт Сесіл Мартін., 2019. — 368 с. — ISBN 978-617-09-5286-8.
86. Будай, А. Дизайн-патерни — просто, як двері [Текст] / Андрій Будай. — Львів, 2012. — 90 с.
87. .NET Design Patterns [Електронний ресурс]. — Режим доступу: http://www.dofactory.com/net/design-patterns.
88. Bishop, J. C# 3.0 Design Patterns [Text] / Judith Bishop. — O'Reilly, 2008. — 290 p. — ISBN-10 0-596-52773-X; ISBN-13 978-0-596-52773-0.
89. Фрімен Е. Head First. Патерни проєктування / Е. Фрімен, Е. Робсон. — 672 с. — ISBN 978-617-09-6159-4.
90. Bodepudi R. Most Popular Programming Languages & Why They're Useful in Machine Learning [Електронний ресурс] / Rita Bodepudi. – 2023. – Режим доступу: https://neptune.ai/blog/programming-languages-machine-learning.
91. Selawsky J. Top five programming languages for AI and machine learning you should learn this year [Електронний ресурс] / John Selawsky. — 2019. — Режим доступу: https://www.itproportal.com/features/top-five-programming-languages-for-ai-and-machine-learning-you-should-learn-this-year/.
92. Beklemysheva A. Why Use Python for AI and Machine Learning? [Електронний ресурс] / Angela Beklemysheva — Режим доступу: https://steelkiwi.com/blog/python-for-ai-and-machine-learning/.

93. Sharma A. Top 9 Python Libraries for Machine Learning in 2020 [Електронний ресурс] / Aditya Sharma. — 2020. — Режим доступу: https://www.upgrad.com/blog/top-python-libraries-for-machine-learning/.
94. Kharkovyna O. Top 10 Best Deep Learning Frameworks in 2019 [Електронний ресурс] / Oleksii Kharkovyna. — 2019. — Режим доступу: https://towardsdatascience.com/top-10-best-deep-learning-frameworks-in-2019-5ccb90ea6de.
95. Tutorials | TensorFlow Core [Електронний ресурс] — Режим доступу: https://www.tensorflow.org/tutorials.
96. PyTorch documentation [Електронний ресурс] – Режим доступу: https://pytorch.org/docs/stable/index.html.
97. Keras API reference [Електронний ресурс] — Режим доступу: https://keras.io/api/.
98. Theano: A Python framework for fast computation of mathematical expressions [Електронний ресурс]. — 2016. — Режим доступу: https://arxiv.org/pdf/1605.02688.pdf.
99. NumPy documentation [Електронний ресурс] — Режим доступу: https://numpy.org/doc/stable/index.html.
100. Python Design Patterns Tutorial [Електронний ресурс] — Режим доступу: https://www.tutorialspoint.com/python_design_patterns/index.htm.
101. Machine Learning Guides [Електронний ресурс] — Режим доступу: https://developers.google.com/machine-learning/guides.
102. subprocess – Subprocess management [Електронний ресурс] – Режим доступу: https://docs.python.org/3/library/subprocess.html.
103. Embedding Python into .NET [Електронний ресурс] — Режим доступу: https://pythonnet.github.io/pythonnet/dotnet.html.
104. IronPython [Електронний ресурс] – Режим доступу: https://ironpython.net/.
105. Embedding .NET into Python [Електронний ресурс] — Режим доступу: https://pythonnet.github.io/pythonnet/python.html.

ДОДАТКИ

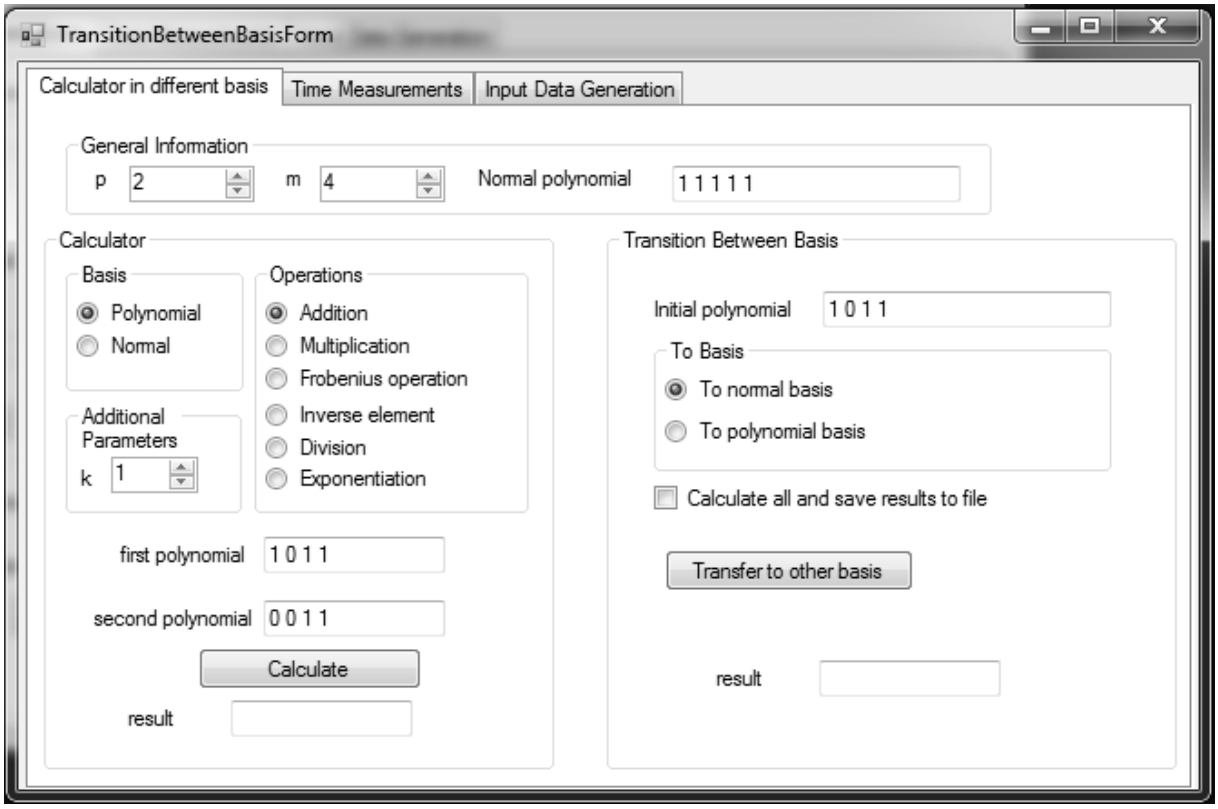
Додаток А. Список публікацій здобувача за темою дисертації

1. Modified Change-of-Basis Conversion Method in $GF(2^m)$ / I. A. Dychka, V. P. Legeza, M. V. Onai, A. I. Severin. // Radio Electronics, Computer Science, Control. — 2020. — №2. — С. 117–128 — DOI: 10.15588/1607-3274-2020-2-12.
2. Method of Performing Operations on the Elements of $GF(2^m)$ Using a Sparse Table / I. Dychka, M. Onai, A. Severin, C. Hu. // International Journal of Computer Network and Information Security (IJCNIS). — 2024. — Vol. 16, №1. — pp. 61-72 — DOI: 10.5815/ijcnis.2024.01.05.
3. Северін А.І. Методи збереження приватності в машинному навчанні. / М.В. Онай, А.І. Северін // Вісник Хмельницького національного університету Серія: «Технічні науки». — 2023. — №6. — С. 274-280 — DOI: 10.31891/2307-5732-2023-329-6-274-280.
4. A. Severin. Architecture of a software system for solving the classification problem based on private data. / M. Onai, A. Severin // Herald of Khmelnytskyi national university. Technical Sciences. — 2024. — №1. — pp. 244-247 — DOI: 10.31891/2307-5732-2024-331-36.
5. Северін А.І. Метод захисту набору даних зображень для вирішення задачі класифікації. / М.В. Онай, А.І. Северін // Прикладна математика та комп'ютинг. ПМК-2020 : тринадцята наук. конф. магістрантів та аспірантів, Київ, 18-20 листопада 2020 р. : зб. тез доп. / [ред кол.: Дичка І.А. та ін.] . — К. : Просвіта, 2020. — С. 221-226.
6. Северін А.І. Комплексний порівняльний аналіз методів збереження приватності в машинному навчанні. / М.В. Онай, А.І. Северін // Актуальні задачі сучасних технологій : зб. тез доповідей XII

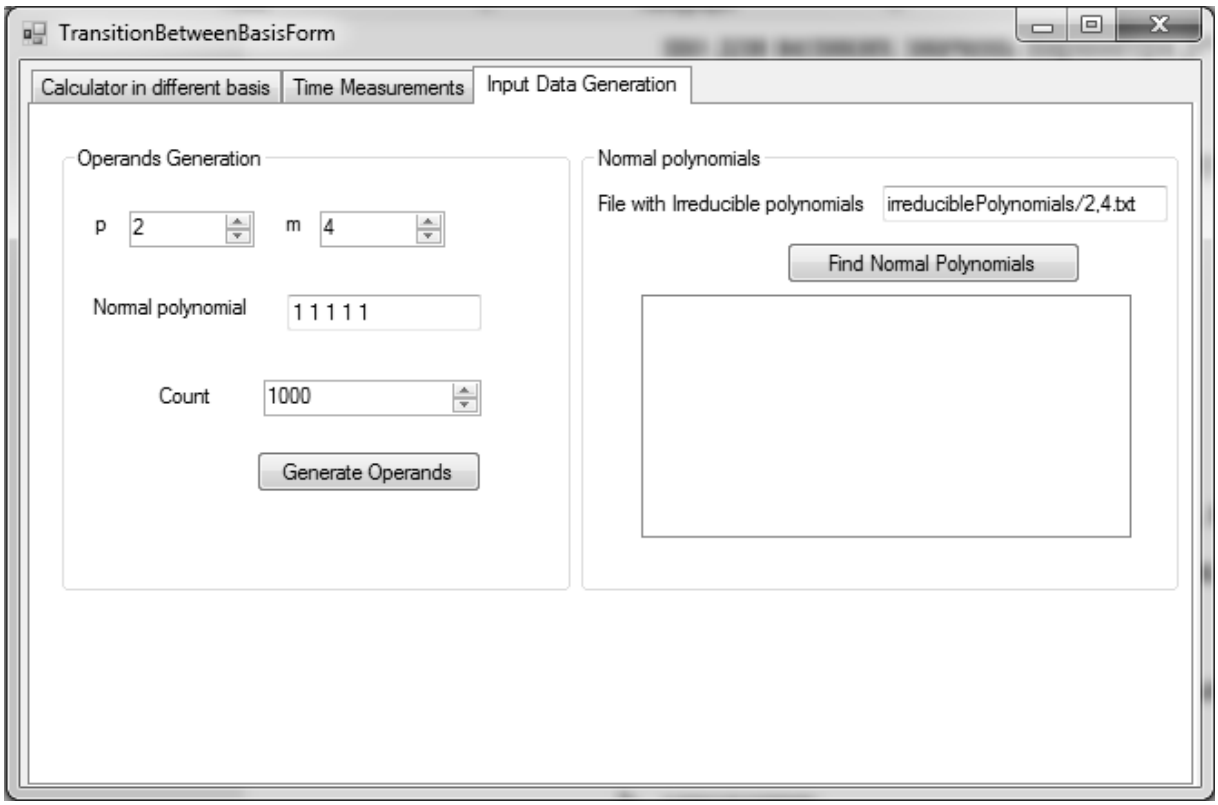
міжнар. наук.-практ. конф. Молодих учених та студентів, (Тернопіль, 6-7 грудня 2023) / М-во освіти і науки України, Терн. націон. техн. ун-т ім. І. Пулюя [та ін.]. — Тернопіль: ФОП Паляниця В. А., 2023. — С. 406-407.

7. Северін А.І. Модифікований підхід для побудови матриці міжбазисних перетвореннь у $GF(p^m)$. / М.В. Онай, А.І. Северін // Матеріали XI науково-технічної конференції «Інформаційні моделі, системи та технології» Тернопільського національного технічного університету імені Івана Пулюя, (Тернопіль, 13-14 грудня 2023 р.). — Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2023 — С. 110.

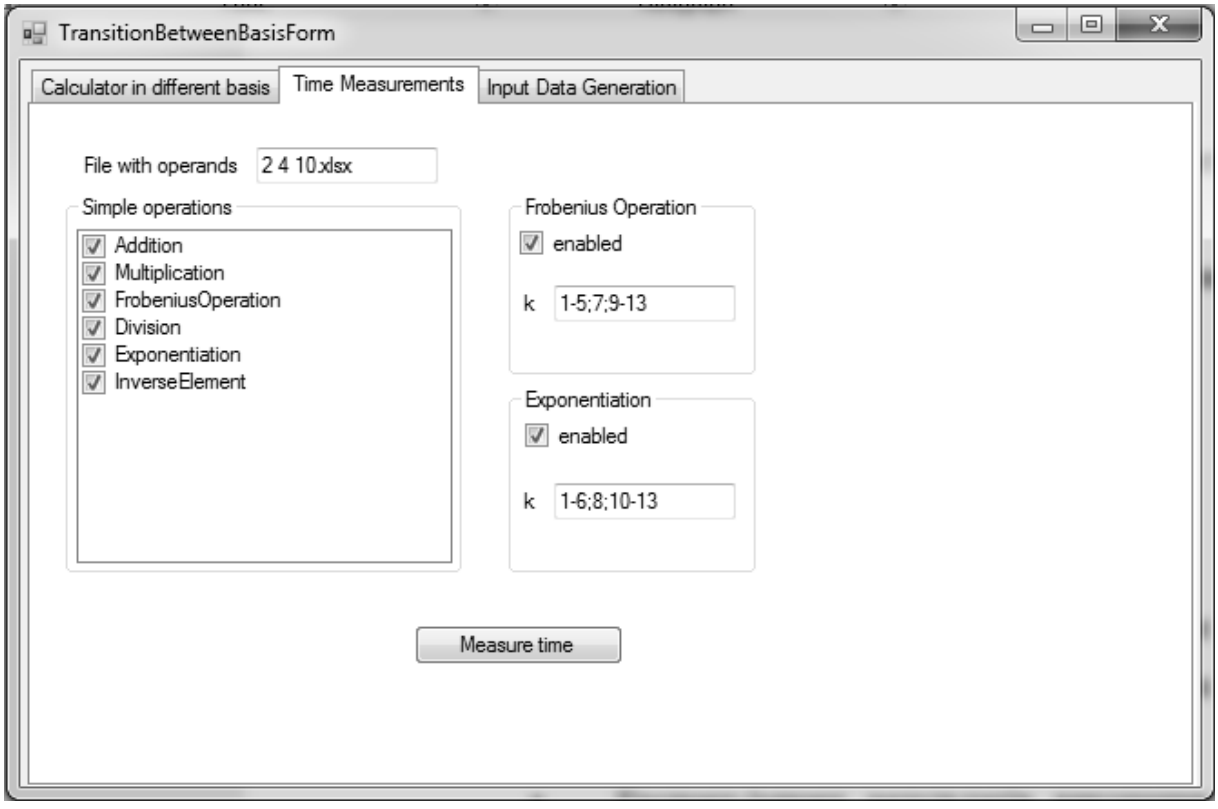
Додаток Б. Інтерфейс користувача програмної системи виконання обчислень над елементами поля $GF(p^m)$ в процесі гомоморфного шифрування



Вкладка «Калькулятор у полі $GF(p^m)$»

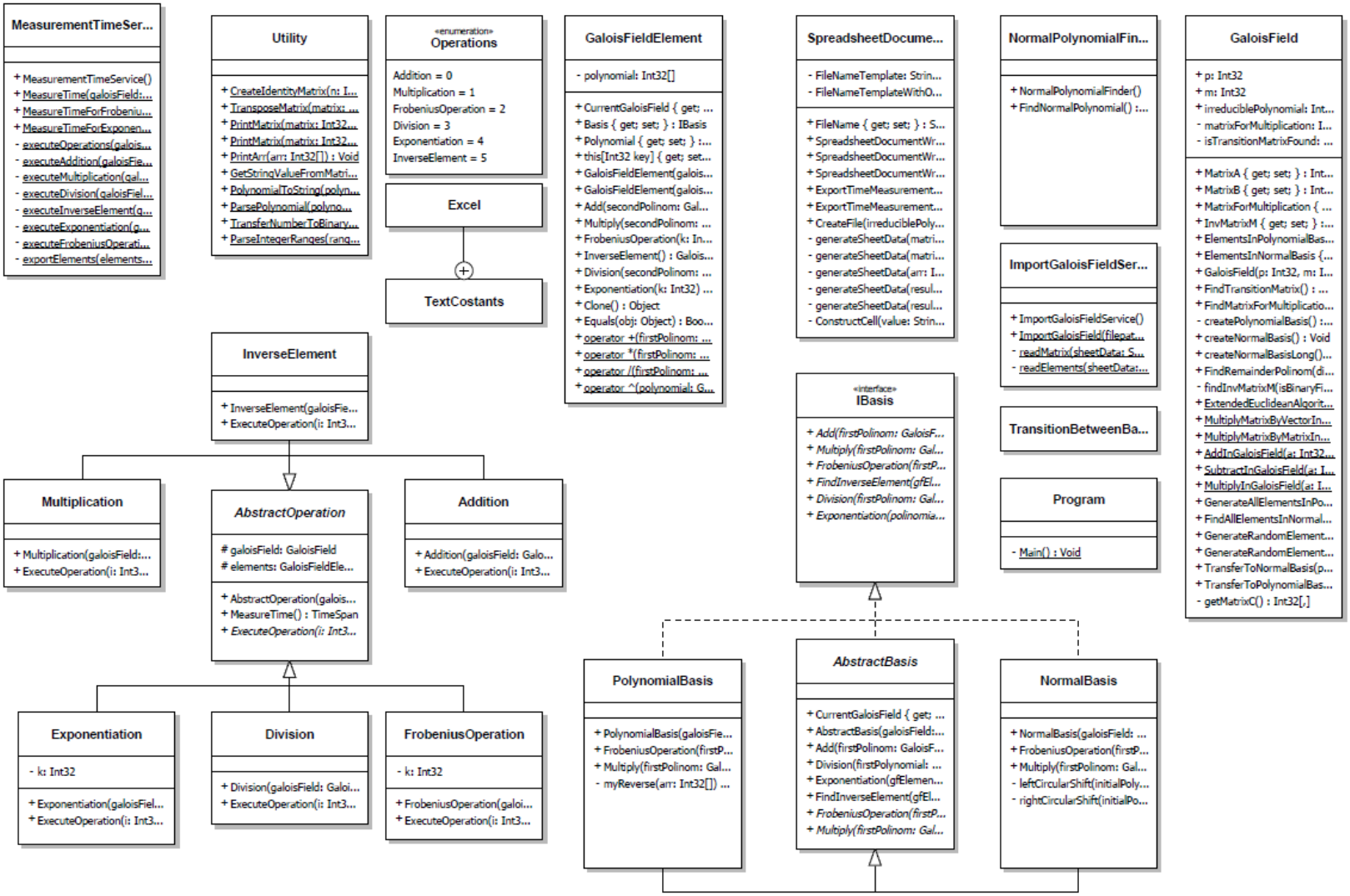


Вкладка «Генерування вхідних даних»



Вкладка «Аналіз часової складності»

Додаток В. Діаграма класів програмної системи виконання обчислень над елементами поля $GF(p^m)$ в процесі гомоморфного шифрування



Додаток Г. Фрагменти програмного коду для реалізації міжбазисних перетворень у розширеному полі Галуа

GaloisField class

```
public class GaloisField
{
      public int p;
      public int m;
      public int[] irreduciblePolynomial;

      /// <summary>
      /// Matrix with basic vectors of polynomial basis (in columns)
      /// </summary>
      public int[,] MatrixA { get; set; }

      /// <summary>
      /// Matrix with basis vectors of a normal basis (in columns)
expressed in terms of a polynomial basis
      /// </summary>
      public int[,] MatrixB { get; set; }

      private int[,] matrixForMultiplication;

      public int[,] MatrixForMultiplication
      {
            get
            {
                  if (matrixForMultiplication == null)
                        FindMatrixForMultiplication();

                  return matrixForMultiplication;
            }
            set
            {
                  matrixForMultiplication = value;
            }
      }

      /// <summary>
      /// Matrix is inverse to matrix B, it's the matrix of the transition
      /// </summary>
      public int[,] InvMatrixM { get; set; }

      public int[][] ElementsInPolynomialBasis { get; set; }

      public int[][] ElementsInNormalBasis { get; set; }

      private bool isTransitionMatrixFound = false;

      public GaloisField(int p, int m, int[] irreduciblePolynomial)
      {
            this.p = p;
            this.m = m;
            this.irreduciblePolynomial = irreduciblePolynomial;
      }

      public void FindTransitionMatrix()
```

```
        {
            createPolynomialBasis();

            createNormalBasis();

            findInvMatrixM(true);

            isTransitionMatrixFound = true;
        }

        public void FindMatrixForMultiplication()
        {
            if (!isTransitionMatrixFound)
                FindTransitionMatrix();

            MatrixForMultiplication = new int[m, m];

            var matrixD = MultiplyMatrixByMatrixInGaloisField(
                MultiplyMatrixByMatrixInGaloisField(
                    Utility.TransposeMatrix(MatrixB),
                    getMatrixC(),
                    p),
                Utility.TransposeMatrix(InvMatrixM),
                p);

            for (int i = 0; i < m; i++)
            {
                for(int j = 0; j < m; j++)
                {
                    MatrixForMultiplication[i, j] =
matrixD[SubtractInGaloisField(j, i, m), SubtractInGaloisField(0, i, m)];
                }
            }
        }

        private void createPolynomialBasis()
        {
            MatrixA = Utility.CreateIdentityMatrix(m);
        }

        /// <summary>
        /// Should be private
        /// </summary>
        public void createNormalBasis()
        {
            MatrixB = new int[m, m];
            MatrixB[1, 0] = 1;
            int p_i = p;
            int i = 1;
            while (p_i < m)
            {
                MatrixB[p_i, i] = 1;
                p_i *= p;
                i++;
            }

            // > update
            int[] prevPolynomial = new int[m];
            for (int j = 0; j < m; j++)
            {
                prevPolynomial[j] = MatrixB[j, i - 1];
            }
```

```
            // <

            while (i < m)
            {
                  GaloisFieldElement gfElem = new GaloisFieldElement(this,
prevPolynomial);
                  gfElem *= gfElem;

                  for (int j = 0; j < m; j++)
                        MatrixB[j, i] = gfElem[j];

                  prevPolynomial = gfElem.Polynomial;

                  /*
                  int[] pInDegreeN = new int[p_i + 1];
                  pInDegreeN[0] = 1;

                  int[] res = FindRemainderPolinom(pInDegreeN,
irreduciblePolynomial);

                  for (int j = 0; j < m; j++)
                        MatrixB[j, i] = res[res.Length - (j + 1)];

                  p_i *= p;
                  */
                  i++;
            }
      }

      /// <summary>
      /// Should be private (long)
      /// </summary>
      public void createNormalBasisLong()
      {
            MatrixB = new int[m, m];
            MatrixB[1, 0] = 1;
            int p_i = p;
            int i = 1;
            while (p_i < m)
            {
                  MatrixB[p_i, i] = 1;
                  p_i *= p;
                  i++;
            }

            while (i < m)
            {
                  int[] pInDegreeN = new int[p_i + 1];
                  pInDegreeN[0] = 1;

                  int[] res = FindRemainderPolinom(pInDegreeN,
irreduciblePolynomial);

                  for (int j = 0; j < m; j++)
                        MatrixB[j, i] = res[res.Length - (j + 1)];

                  p_i *= p;

                  i++;
            }
      }
```

```
      /// <summary>
      /// The search for the remainder from dividing one polynomial to
another
      /// </summary>
      /// <param name="dividedPolinom"></param>
      /// <param name="dividerPolinom"></param>
      /// <returns></returns>
      public  int[] FindRemainderPolinom(int[] dividedPolinom, int[]
dividerPolinom)
      {
            int[] result = (int[])dividedPolinom.Clone();

            for (int i = 0; dividedPolinom.Length - i >=
dividerPolinom.Length; i++)
            {
                  if (result[i] == 0) continue;

                  for (int j = 0, k = i; j < dividerPolinom.Length; j++,
 k++)
                  {
                        result[k] ^= dividerPolinom[j];
                  }
            }

            return result;
      }

      /// <summary>
      /// The search for the inverse of matrix B is based on the Gauss-
Jordan algorithm
      /// (but first, the check for the equality of 0 elements on the
diagonal is performed,
      /// in this case the first non-zero element in this column is
searched and
      /// the corresponding line is added to the line with zero in the
diagonal)
      /// </summary>
      /// <param name="p"></param>
      /// <param name="m"></param>
      private void findInvMatrixM(bool isBinaryField)
      {
            int[,] currentMatrix = (int[,])MatrixB.Clone();
            InvMatrixM = Utility.CreateIdentityMatrix(m);

            for (int k = 0; k < m; k++)
            {
                  if (currentMatrix[k, k] == 0)
                  {
                        // adding first non-zero line to kth line
                        bool flag = false;
                        for (int i = k + 1; i < m; i++)
                        {
                              if (currentMatrix[i, k] == 0) continue;

                              for (int j = 0; j < m; j++)
                              {
                                    if (isBinaryField)
                                    {
                                          currentMatrix[k, j] ^=
currentMatrix[i, j];
                                          InvMatrixM[k, j] ^= InvMatrixM[i,
j];
```

```
                                    }
                                    else
                                    {
                                          currentMatrix[k, j] =
      SubtractInGaloisField(currentMatrix[k, j], currentMatrix[i, j], p);
                                          InvMatrixM[k, j] =
      SubtractInGaloisField(InvMatrixM[k, j], InvMatrixM[i, j], p);
                                    }
                              }

                              flag = true;
                              break;
                        }
                        if (flag == false || ((k == m) && (currentMatrix[k,
k] == 0)))
                              throw new ApplicationException("it is not
normal polynomial");
                  }

                  // making 1 in the kth line
                  if (!isBinaryField)
                  {
                        int invElement = ExtendedEuclideanAlgorithm(p,
currentMatrix[k, k]);
                        for (int z = 0; z < m; z++)
                        {
                              currentMatrix[k, z] =
                                    MultiplyInGaloisField(currentMatrix[k,
z], invElement, p);
                              InvMatrixM[k, z] =
                                    MultiplyInGaloisField(InvMatrixM[k, z],
invElement, p);
                        }
                  }

                  // Zeroing the elements of the kth column of all rows
except k
                  for (int i = 0; i < m; i++)
                  {
                        if (i == k) continue;

                        if (currentMatrix[i, k] == 0) continue;

                        int coefficient = currentMatrix[i, k]; // uses only
for !isBinaryField

                        for (int j = 0; j < m; j++)
                        {
                              if (isBinaryField)
                              {
                                    currentMatrix[i, j] ^= currentMatrix[k,
j];
                                    InvMatrixM[i, j] ^= InvMatrixM[k, j];
                              }
                              else
                              {
                                    currentMatrix[i, j] =
SubtractInGaloisField(
                                          currentMatrix[i, j],
```

```
      MultiplyInGaloisField(currentMatrix[k, j], coefficient, p),
                                        p);
                                  InvMatrixM[i, j] =
SubtractInGaloisField(
                                        InvMatrixM[i, j],

      MultiplyInGaloisField(InvMatrixM[k, j], coefficient, p),
                                        p);
                            }
                        }
                  }
            }
      }

      public static int ExtendedEuclideanAlgorithm(int a, int b)
      {
            int u = a, A = 1, B = 0;
            int v = b, C = 0, D = 1;

            while (v != 0)
            {
                  int q = u / v;
                  int t1 = u - q * v, t2 = A - q * C, t3 = B - q * D;
                  u = v;
                  A = C;
                  B = D;
                  v = t1;
                  C = t2;
                  D = t3;
            }

            D = u;

            int x = A;
            int y = B;
            if (y >= 0)
                  return y;
            else
                  return y + a;
      }

      /// <summary>
      /// Only for square matrix
      /// </summary>
      /// <param name="matrix"></param>
      /// <param name="vector"></param>
      /// <returns></returns>
      public static int[] MultiplyMatrixByVectorInGaloisField(int[,]
matrix, int[] vector, int p)
      {
            int[] result = new int[vector.Length];

            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                  for (int j = 0; j < matrix.GetLength(1); j++)
                  {
                        result[i] = AddInGaloisField(
                              result[i],
                              MultiplyInGaloisField(matrix[i, j], vector[j],
p),
                              p);
```

```
                }
            }

            return result;
        }

        /// <summary>
        ///
        /// </summary>
        /// <param name="matrixA"></param>
        /// <param name="matrixB"></param>
        /// <param name="p"></param>
        /// <returns></returns>
        public static int[,] MultiplyMatrixByMatrixInGaloisField(int[,]
matrixA, int[,] matrixB, int p)
        {
            if (matrixA.GetLength(1) != matrixB.GetLength(0))
                throw new ApplicationException("You cannot multiply these
matrices");

            var result = new int[matrixA.GetLength(0),
matrixB.GetLength(1)];

            for (int i = 0; i < matrixA.GetLength(0); i++)
            {
                for (int j = 0; j < matrixB.GetLength(1); j++)
                {
                    result[i, j] = 0;

                    for (int k = 0; k < matrixB.GetLength(0); k++)
                    {
                        result[i, j] = AddInGaloisField(
                            result[i, j],
                            MultiplyInGaloisField(matrixA[i, k],
matrixB[k, j], p),
                            p);
                    }
                }
            }

            return result;
        }

        public static int AddInGaloisField(int a, int b, int p)
        {
            return (a + b) % p;
        }

        public static int SubtractInGaloisField(int a, int b, int p)
        {
             int res = (a - b) % p;
            return res < 0 ? res + p : res;
        }

        public static int MultiplyInGaloisField(int a, int b, int p)
        {
            return (a * b) % p;
        }

        public void GenerateAllElementsInPolynomialBasis()
        {
            int cnt = (int)Math.Pow(p, m);
```

```
            ElementsInPolynomialBasis = new int[cnt][];

            int[] initialPolinom = new int[m];
            ElementsInPolynomialBasis[0] = new int[m];

            Array.Copy(initialPolinom, ElementsInPolynomialBasis[0], m);

            for (int i = 1; i < cnt; i++)
            {
                  ElementsInPolynomialBasis[i] = new int[m];

                  for (int j = 0; j < m; j++)
                  {
                        initialPolinom[j] = (initialPolinom[j] + 1) % p;

                        if (initialPolinom[j] != 0)
                              break;
                  }

                  Array.Copy(initialPolinom, ElementsInPolynomialBasis[i],
m);
            }
      }

      public void FindAllElementsInNormalBasis()
      {
            int cnt = ElementsInPolynomialBasis.Length;
            ElementsInNormalBasis = new int[cnt][];

            for (int i = 0; i < cnt; i++)
            {
                  ElementsInNormalBasis[i] =
                        MultiplyMatrixByVectorInGaloisField(InvMatrixM,
ElementsInPolynomialBasis[i], p);
            }
      }

      public void GenerateRandomElementsInAllBasis(int n)
      {
            GenerateRandomElementsInPolynomialBasis(n);
            FindAllElementsInNormalBasis();
      }

      public void GenerateRandomElementsInPolynomialBasis(int n)
      {
            ElementsInPolynomialBasis = new int[n][];

            var polinom = new int[m];

            var random = new Random();

            for (int i = 0; i < n; i++)
            {
                  ElementsInPolynomialBasis[i] = new int[m];

                  for (int j = 0; j < m; j++)
                  {
                        polinom[j] = random.Next(p);
                  }

                  Array.Copy(polinom, ElementsInPolynomialBasis[i], m);
            }
```

```
        }

        public int[] TransferToNormalBasis(int[] polynomial)
        {
                return MultiplyMatrixByVectorInGaloisField(InvMatrixM,
polynomial, p);
        }

        public int[] TransferToPolynomialBasis(int[] polynomial)
        {
                return MultiplyMatrixByVectorInGaloisField(MatrixB, polynomial,
p);
        }

        private int[,] getMatrixC()
        {
                var matrix = new int[m, m];

                for(int i = 0; i < m - 1; i++)
                {
                        matrix[i, i + 1] = 1;
                        matrix[m - 1, i] = irreduciblePolynomial[m - i];
                }
                matrix[m - 1, m - 1] = irreduciblePolynomial[1];

                return matrix;
        }
}
```

Додаток Ґ. Фрагменти програмного коду для реалізації елементу скінченного поля

GaloisFieldElement class

```
public class GaloisFieldElement : ICloneable
{
	public GaloisField CurrentGaloisField { get; set; }

	public IBasis Basis { get; set; }

	private int[] polynomial;

	public int[] Polynomial
	{
		get
		{
			return polynomial;
		}
		set
		{
			polynomial = value; //(int[])value.Clone();
		}
	}

	public GaloisFieldElement(GaloisField galoisField, int[] polynomial)
	{
		this.CurrentGaloisField = galoisField;
		this.polynomial = polynomial; //(int[])polynomial.Clone();
		this.Basis = new PolynomialBasis(galoisField);
	}

	public GaloisFieldElement(GaloisField galoisField, int[] polynomial,
IBasis basis)
	{
		this.CurrentGaloisField = galoisField;
		this.polynomial = polynomial; //(int[])polynomial.Clone();
		this.Basis = basis;
	}

	public GaloisFieldElement Add(GaloisFieldElement secondPolinom)
	{
		return Basis.Add(this, secondPolinom);
	}

	public GaloisFieldElement Multiply(GaloisFieldElement secondPolinom)
	{
		return Basis.Multiply(this, secondPolinom);
	}

	public GaloisFieldElement FrobeniusOperation(int k)
	{
		return Basis.FrobeniusOperation(this, k);
	}

	public GaloisFieldElement InverseElement()
	{
		return Basis.FindInverseElement(this);
```

```
        }

        public GaloisFieldElement Division(GaloisFieldElement secondPolinom)
        {
            return Basis.Division(this, secondPolinom);
        }

        public GaloisFieldElement Exponentiation(int k)
        {
            return Basis.Exponentiation(this, k);
        }

        #region Operators

        public static GaloisFieldElement operator +(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom)
        {
            return firstPolinom.Add(secondPolinom);
        }

        public static GaloisFieldElement operator *(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom)
        {
            return firstPolinom.Multiply(secondPolinom);
        }

        public static GaloisFieldElement operator /(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom)
        {
            return firstPolinom.Division(secondPolinom);
        }

        public static GaloisFieldElement operator ^(GaloisFieldElement
polynomial, int k)
        {
            return polynomial.Exponentiation(k);
        }

        #endregion

        public int this[int key]
        {
            get
            {
                return polynomial[key];
            }
            set
            {
                polynomial[key] = value;
            }
        }

        #region ICloneable Members

        public object Clone()
        {
            return this.MemberwiseClone();
        }

        #endregion
}
```

Додаток Д. Фрагменти програмного коду для виконання обчислень у поліноміальному й нормальному базисах

IBasis interface

```
public interface IBasis
{
      GaloisFieldElement Add(GaloisFieldElement firstPolinom,
GaloisFieldElement secondPolinom);

      GaloisFieldElement Multiply(GaloisFieldElement firstPolinom,
GaloisFieldElement secondPolinom);

      GaloisFieldElement FrobeniusOperation(GaloisFieldElement
firstPolinom, int k);

      GaloisFieldElement FindInverseElement(GaloisFieldElement gfElement);

      GaloisFieldElement Division(GaloisFieldElement firstPolinom,
GaloisFieldElement secondPolinom);

      GaloisFieldElement Exponentiation(GaloisFieldElement polinomial, int
k);
      }
```

AbstractBasis class

```
public abstract class AbstractBasis : IBasis
{
      public GaloisField CurrentGaloisField { get; set; }

      public AbstractBasis(GaloisField galoisField)
      {
            this.CurrentGaloisField = galoisField;
      }

      public virtual GaloisFieldElement Add(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom)
      {
            GaloisField galoisField = firstPolinom.CurrentGaloisField;

            if (!galoisField.Equals(secondPolinom.CurrentGaloisField))
                  throw new ApplicationException("Different galois
fields");

            int m = galoisField.m;
            int p = galoisField.p;
            int[] resultPolinom = new int[m];

            for (int i = 0; i < m; i++)
            {
                  resultPolinom[i] = (firstPolinom[i] + secondPolinom[i]) %
p;
            }

            return new GaloisFieldElement(galoisField, resultPolinom,
firstPolinom.Basis);
```

```
      }

      public virtual GaloisFieldElement Division(GaloisFieldElement
firstPolynomial, GaloisFieldElement secondPolynomial)
      {
            return Multiply(firstPolynomial,
FindInverseElement(secondPolynomial));
      }

      /**
       * RL
       **/
      public virtual GaloisFieldElement Exponentiation(GaloisFieldElement
gfElement, int k)
      {
            var b = Utility.TransferNumberToBinaryBasis(k);
            var r = b.Length - 1;

            var result = gfElement;

            for (int i = r - 1; i >= 0; i--)
            {
                  //result *= result;
                  result = result.FrobeniusOperation(1);

                  if (b[r - i] == 1)
                  {
                        result *= gfElement;
                  }
            }

            return result;
      }

      public virtual GaloisFieldElement
FindInverseElement(GaloisFieldElement gfElement)
      {
            var m = gfElement.CurrentGaloisField.m;
            var b = Utility.TransferNumberToBinaryBasis(m - 1);
            var r = b.Length - 1;

            var polynomialN = (GaloisFieldElement)gfElement.Clone();
            var k = 1;

            for (int i = r - 1; i >= 0; i--)
            {
                  var polynomialM = polynomialN;

                  //
                  //for (int j = 1; j <= k; j++)
                  //{
                  //    polynomialM *= polynomialM;
                  //}
                  // only for 2 ^ m
                  polynomialM = polynomialM.FrobeniusOperation(k);
                  //

                  polynomialN = polynomialM * polynomialN;
                  k *= 2;

                  if (b[r - i] == 1)
                  {
```

```
                        // only for 2^m
                        //polynomialN *= polynomialN * gfElement;
                        polynomialN = polynomialN.FrobeniusOperation(1);
                        polynomialN *= gfElement;
                        //
                        k++;
                  }
            }

            return polynomialN.FrobeniusOperation(1);//polynomialN *
polynomialN;
      }

      public abstract GaloisFieldElement
FrobeniusOperation(GaloisFieldElement firstPolinom, int k);
      public abstract GaloisFieldElement Multiply(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom);
      }
```

NormalBasis class

```
public class NormalBasis : AbstractBasis
{
      public NormalBasis(GaloisField galoisField) : base(galoisField)
      {
      }

      public override GaloisFieldElement
FrobeniusOperation(GaloisFieldElement firstPolinom, int k)
      {
            return new GaloisFieldElement(firstPolinom.CurrentGaloisField,
rightCircularShift(firstPolinom.Polynomial, k),
                  new NormalBasis(CurrentGaloisField));
      }

      public override GaloisFieldElement Multiply(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom)
      {
            var galoisField = firstPolinom.CurrentGaloisField;
            var resultPolynomial = new int[galoisField.m];

            int[] leftPolynomial = null;
            int[] rightPolynomial = null;

            for (int i = 0; i < galoisField.m; i++)
            {
                  leftPolynomial = (i == 0) ?
(int[])firstPolinom.Polynomial.Clone() : leftCircularShift(leftPolynomial,
1);
                  rightPolynomial = (i == 0) ?
(int[])secondPolinom.Polynomial.Clone() :
leftCircularShift(rightPolynomial, 1);

                  var tmpRigthMultiplication =
GaloisField.MultiplyMatrixByVectorInGaloisField(
                        galoisField.MatrixForMultiplication,
                        rightPolynomial,
                        galoisField.p);

                  for (int j = 0; j < galoisField.m; j++)
```

```
                {
                      resultPolynomial[i] = GaloisField.AddInGaloisField(
                            resultPolynomial[i],

      GaloisField.MultiplyInGaloisField(leftPolynomial[j],
tmpRigthMultiplication[j], galoisField.p),
                            galoisField.p);
                }
            }

            return new GaloisFieldElement(galoisField, resultPolynomial,
new NormalBasis(CurrentGaloisField));
      }

      private int[] leftCircularShift(int[] initialPolynomial, int k)
      {
            int[] result = new int[initialPolynomial.Length];

            for(int i = 0; i < initialPolynomial.Length; i++)
            {
                  result[i] = initialPolynomial[(i + k) %
initialPolynomial.Length];
            }

            return result;
      }

      private int[] rightCircularShift(int[] initialPolynomial, int k)
      {
            int[] result = new int[initialPolynomial.Length];

            for (int i = 0; i < initialPolynomial.Length; i++)
            {
                  result[(i + k) % initialPolynomial.Length] =
initialPolynomial[i];
            }

            return result;
      }
}
```

PolynomialBasis class

```
public class PolynomialBasis : AbstractBasis
{
      public PolynomialBasis(GaloisField galoisField) : base(galoisField)
      {
      }

      public override GaloisFieldElement
FrobeniusOperation(GaloisFieldElement firstPolinom, int k)
      {
            var resultPolinomial = new
GaloisFieldElement(firstPolinom.CurrentGaloisField,
(int[])firstPolinom.Polynomial.Clone());
            for (int i = 0; i < k; i++)
            {
                  resultPolinomial *= resultPolinomial;
            }

            return resultPolinomial;
```

```
      }

      public override GaloisFieldElement Multiply(GaloisFieldElement
firstPolinom, GaloisFieldElement secondPolinom)
      {
            GaloisField galoisField = firstPolinom.CurrentGaloisField;

            if (!galoisField.Equals(secondPolinom.CurrentGaloisField))
                  throw new ApplicationException("Different galois
fields");

            int m = galoisField.m;
            int p = galoisField.p;
            GaloisFieldElement resultPolinom = new
GaloisFieldElement(galoisField, new int[2 * m - 1]);

            for (int i = 0; i < m; i++)
            {
                  resultPolinom[i] = (firstPolinom[i] * secondPolinom[0]) %
p;
            }

            for (int i = 1; i < m; i++)
            {
                  GaloisFieldElement currentGaloisFieldElement = new
GaloisFieldElement(galoisField, new int[m + i]);

                  for (int j = 0; j < m; j++)
                  {
                        currentGaloisFieldElement[j + i] = (firstPolinom[j]
* secondPolinom[i]) % p;
                  }

                  for (int j = 0; j <
currentGaloisFieldElement.Polynomial.Length; j++)
                  {
                        resultPolinom[j] = (resultPolinom[j] +
currentGaloisFieldElement[j]) % p;
                  }
            }

            Array.Reverse(resultPolinom.Polynomial);

            resultPolinom.Polynomial =

      galoisField.FindRemainderPolinom(resultPolinom.Polynomial,
galoisField.irreduciblePolynomial);

            int[] result = new int[m];
            for (int i = 0; i < m; i++)
            {
                  result[i] = resultPolinom[resultPolinom.Polynomial.Length
- m + i];
            }
            resultPolinom.Polynomial = result;

            Array.Reverse(resultPolinom.Polynomial);
            return resultPolinom;
      }

      private int[] myReverse(int[] arr) // to delete if not necessary
      {
```

```
            int i = arr.Length - 1;

            while (arr[i] == 0)
            {
                  i--;
            }

            int[] result = new int[i + 1];
            for (; i >= 0; i--)
            {
                  result[result.Length - i - 1] = arr[i];
            }

            return result;
      }
}
```

Додаток Е. Фрагменти програмного коду для реалізації допоміжних функцій

Utility class

```
    static class Utility
    {
        public static int[,] CreateIdentityMatrix(int n)
        {
            int[,] matrix = new int[n, n];

            for (int i = 0; i < n; i++)
                matrix[i, i] = 1;

            return matrix;
        }

        public static int[,] TransposeMatrix(int[,] matrix)
        {
            int[,] result = new int[matrix.GetLength(1),
matrix.GetLength(0)];

            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    result[j, i] = matrix[i, j];
                }
            }

            return result;
        }

        public static void PrintMatrix(int[,] matrix, string name)
        {
            Console.WriteLine();
            Console.WriteLine(name);
            PrintMatrix(matrix);
        }

        public static void PrintMatrix(int[,] matrix)
        {
            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    Console.Write("{0,4:D}", matrix[i, j]);
                }
                Console.WriteLine();
            }
        }

        public static void PrintArr(int[] arr)
        {
            foreach (int b in arr)
            {
                Console.Write("{0,3}", b);
            }
```

```
            Console.WriteLine();
        }

        public static string GetStringValueFromMatrix(int[,] matrix, string
name)
        {
            StringBuilder stringBuilder = new StringBuilder();

            stringBuilder.AppendLine(name);

            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    stringBuilder.AppendFormat("{0,4:D}", matrix[i, j]);
                }
                stringBuilder.AppendLine();
            }

            return stringBuilder.ToString();
        }

        public static string PolynomialToString(int[] polynomial)
        {
            StringBuilder stringBuilder = new StringBuilder();

            for (int i = 0; i < polynomial.Length; i++)
            {
                stringBuilder.Append(polynomial[i]);
                stringBuilder.Append(' ');
            }

            stringBuilder.Remove(stringBuilder.Length - 1, 1);

            return stringBuilder.ToString();
        }

        public static int[] ParsePolynomial(string polynomial)
        {
            string[] numbers = polynomial.Split(' ');
            int[] resultPolynomial = new int[numbers.Length];

            for (int i = 0; i < numbers.Length; i++)
            {
                resultPolynomial[i] = int.Parse(numbers[i]);
            }

            return resultPolynomial;
        }

        public static int[] TransferNumberToBinaryBasis(int number)
        {
            var list = new LinkedList<int>();
            var n = 2;

            while (number >= n)
            {
                list.AddFirst(number % n);
                number /= 2;
            }

            list.AddFirst(number);
```

```
            int[] result = list.SkipWhile(element => element ==
0).ToArray(); // it's not necessary to skip

            return result;
        }

        /// <summary>
        /// example 1-5;7;9-11
        /// </summary>
        /// <param name="range"></param>
        /// <returns></returns>
        public static List<int> ParseIntegerRanges(string ranges)
        {
            List<int> result = new List<int>();

            string[] rangeList = ranges.Split(';');

            foreach(var range in rangeList)
            {
                if(range.Contains('-'))
                {
                    string[] r = range.Split('-');
                    int firstIndex = int.Parse(r[0]);
                    int lastIndex = int.Parse(r[1]);

                    for(int i = firstIndex; i <= lastIndex; i++)
                    {
                        result.Add(i);
                    }
                } else
                {
                    result.Add(int.Parse(range));
                }
            }

            return result;
        }
    }
```

Додаток Є. Фрагменти програмного коду для експорту результатів експериментальних досліджень

Operations enum

```
    public enum Operations
    {
        Addition, Multiplication, FrobeniusOperation, Division,
Exponentiation, InverseElement
    }
```

TextCostants class

```
    public static class TextCostants
    {
        #region excel file
        public static class Excel
        {
            public const string IrreduciblePolynomialTab = "Irreducible
Polynomial";
            public const string MatrixMTab = "Matrix M";
            public const string InverseMatrixMTab = "Inverse Matrix M";
            public const string ElementsInPolynomialBasisTab = "Elements In
Polynomial Basis";
            public const string ElementsInNormalBasisTab = "Elements In
Normal Basis";
        }
        #endregion
    }
```

SpreadsheetDocumentWriter class

```
  internal class SpreadsheetDocumentWriter
    {
        private const string FileNameTemplate = "{0} {1}.xlsx";
        private const string FileNameTemplateWithOperandsCount = "{0} {1}
{2}.xlsx";

        public string FileName { get; set; }

        public SpreadsheetDocumentWriter(int p, int m)
        {
            FileName = String.Format(FileNameTemplate, p, m);
        }

        public SpreadsheetDocumentWriter(int p, int m, int n)
        {
            FileName = String.Format(FileNameTemplateWithOperandsCount, p,
m, n);
        }

        public SpreadsheetDocumentWriter(string fileName)
        {
            this.FileName = fileName;
        }
```

```
        public void ExportTimeMeasurement(int p, Dictionary<String,
Dictionary<Operations, double>> results)
        {
            using (SpreadsheetDocument document =
SpreadsheetDocument.Create(FileName, SpreadsheetDocumentType.Workbook))
            {
                WorkbookPart workbookPart = document.AddWorkbookPart();
                workbookPart.Workbook = new Workbook();
                Sheets sheets = workbookPart.Workbook.AppendChild(new
Sheets());

                WorksheetPart worksheetPart =
workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(results));
                Sheet sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 1, Name =
String.Format("p={0}", p) };
                sheets.Append(sheet);
            }
        }

        public void ExportTimeMeasurement(int p, Dictionary<String,
Dictionary<int, double>> results)
        {
            using (SpreadsheetDocument document =
SpreadsheetDocument.Create(FileName, SpreadsheetDocumentType.Workbook))
            {
                WorkbookPart workbookPart = document.AddWorkbookPart();
                workbookPart.Workbook = new Workbook();
                Sheets sheets = workbookPart.Workbook.AppendChild(new
Sheets());

                WorksheetPart worksheetPart =
workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(results));
                Sheet sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 1, Name =
String.Format("p={0}", p) };
                sheets.Append(sheet);
            }
        }

        public void CreateFile(int[] irreduciblePolynomial, int[,] matrixM,
            int[,] invMatrixM, int [][] elementsInPolynomialBasis, int[][]
elementsInNormalBasis)
        {
            using (SpreadsheetDocument document =
SpreadsheetDocument.Create(FileName, SpreadsheetDocumentType.Workbook))
            {
                WorkbookPart workbookPart = document.AddWorkbookPart();
                workbookPart.Workbook = new Workbook();
                Sheets sheets = workbookPart.Workbook.AppendChild(new
Sheets());

                WorksheetPart worksheetPart =
workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(irreduciblePolynomial));
```

```
                Sheet sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 1, Name =
TextCostants.Excel.IrreduciblePolynomialTab };
                sheets.Append(sheet);

                worksheetPart = workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(matrixM));
                sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 2, Name =
TextCostants.Excel.MatrixMTab };
                sheets.Append(sheet);

                worksheetPart = workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(invMatrixM));
                sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 3, Name =
TextCostants.Excel.InverseMatrixMTab };
                sheets.Append(sheet);

                worksheetPart = workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(elementsInPolynomialBasis));
                sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 4, Name =
TextCostants.Excel.ElementsInPolynomialBasisTab };
                sheets.Append(sheet);

                worksheetPart = workbookPart.AddNewPart<WorksheetPart>();
                worksheetPart.Worksheet = new
Worksheet(generateSheetData(elementsInNormalBasis));
                sheet = new Sheet() { Id =
workbookPart.GetIdOfPart(worksheetPart), SheetId = 5, Name =
TextCostants.Excel.ElementsInNormalBasisTab };
                sheets.Append(sheet);
            }
        }

        private SheetData generateSheetData(int[,] matrix)
        {
            SheetData sheetData = new SheetData();

            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                Row row = new Row();

                for (int j = 0; j < matrix.GetLength(1); j++)
                {
                    row.Append(ConstructCell(matrix[i, j].ToString(),
CellValues.Number));
                }

                sheetData.AppendChild(row);
            }

            return sheetData;
        }

        private SheetData generateSheetData(int[][] matrix)
        {
            SheetData sheetData = new SheetData();
```

```
            for (int i = 0; i < matrix.GetLength(0); i++)
            {
                Row row = new Row();

                for (int j = 0; j < matrix[i].GetLength(0); j++)
                {
                    row.Append(ConstructCell(matrix[i][j].ToString(),
CellValues.Number));
                }

                sheetData.AppendChild(row);
            }

            return sheetData;
        }

        private SheetData generateSheetData(int[] arr)
        {
            SheetData sheetData = new SheetData();

            Row row = new Row();

            for (int i = 0; i < arr.GetLength(0); i++)
            {
                row.Append(ConstructCell(arr[i].ToString(),
CellValues.Number));
            }

            sheetData.AppendChild(row);

            return sheetData;
        }

        private SheetData generateSheetData(Dictionary<String,
Dictionary<Operations, double>> results)
        {
            SheetData sheetData = new SheetData();

            Row row = new Row();

            row.Append(ConstructCell("m", CellValues.String));
            row.Append(ConstructCell("Basis", CellValues.String));

            foreach (var operation in Enum.GetNames(typeof(Operations)))
            {
                row.Append(ConstructCell(operation, CellValues.String));
            }

            sheetData.AppendChild(row);

            foreach (var result in results)
            {
                row = new Row();

                string[] fixedElements = result.Key.Split(' ');
                foreach (var element in fixedElements)
                {
                    row.Append(ConstructCell(element, CellValues.String));
                }
                //row.Append(ConstructCell(result.Key.m.ToString(),
CellValues.String));
```

```
                foreach (var operation in
Enum.GetValues(typeof(Operations)))
                {
                    if (result.Value.TryGetValue((Operations)operation, out
double milliseconds))
                        row.Append(ConstructCell(milliseconds.ToString(),
CellValues.String));
                    else
                        row.Append(ConstructCell("-", CellValues.String));
                }

                sheetData.AppendChild(row);
            }

            return sheetData;
        }

        private SheetData generateSheetData(Dictionary<string,
Dictionary<int, double>> results)
        {
            SheetData sheetData = new SheetData();

            Row row = new Row();

            row.Append(ConstructCell("m", CellValues.String));
            row.Append(ConstructCell("Basis", CellValues.String));

            foreach (int i in results.First().Value.Keys)
            {
                row.Append(ConstructCell(i.ToString(), CellValues.String));
            }

            sheetData.AppendChild(row);

            foreach (var result in results)
            {
                row = new Row();

                string[] fixedElements = result.Key.Split(' ');
                foreach (var element in fixedElements)
                {
                    row.Append(ConstructCell(element, CellValues.String));
                }
                //row.Append(ConstructCell(result.Key.m.ToString(),
CellValues.String));

                foreach (var i in result.Value.Keys)
                {
                    row.Append(ConstructCell(result.Value[i].ToString(),
CellValues.String));
                }

                sheetData.AppendChild(row);
            }

            return sheetData;
        }

        private Cell ConstructCell(string value, CellValues dataType)
        {
            return new Cell()
```

```
            {
                CellValue = new CellValue(value),
                DataType = new EnumValue<CellValues>(dataType)
            };
        }
    }
```

Додаток Ж. Фрагменти програмного модулю взаємодії з користувачем

TransitionBetweenBasisForm class

```
    public partial class TransitionBetweenBasisForm : Form
    {
        public TransitionBetweenBasisForm()
        {
            InitializeComponent();
            SetDefaultValues();
        }

        private void SetDefaultValues()
        {
            txtIrreduciblePolynomial.Text = Utility.PolynomialToString(new
int[] { 1, 1, 1, 1, 1 });
            txtFirstPolynomial.Text = Utility.PolynomialToString(new int[]
{ 1, 0, 1, 1 });
            txtSecondPolynomial.Text = Utility.PolynomialToString(new int[]
{ 0, 0, 1, 1 });
            txtInitialPolynomial.Text = Utility.PolynomialToString(new
int[] { 1, 0, 1, 1 });
            updParameterP.Value = 2;
            updParametrM.Value = 4;
            rdoPolynomialBasis.Checked = true;
            rdoToNormalBasis.Checked = true;
            rdoAddition.Checked = true;
            updParameterK.Value = 1;

cklSimpleOperations.Items.AddRange(Enum.GetNames(typeof(Operations)));
            for (int i = 0; i < cklSimpleOperations.Items.Count; i++)
            {
                cklSimpleOperations.SetItemChecked(i, true);
            }
        }

        private GaloisField getGaloisField(decimal p, decimal m, string
normalPolynomial)
        {
            int[] irreduciblePolynomial =
Utility.ParsePolynomial(normalPolynomial.Trim());
            return new GaloisField((int)p, (int)m, irreduciblePolynomial);
        }

        private void btnCalculate_Click(object sender, EventArgs e)
        {
            GaloisField gfField = getGaloisField(updParameterP.Value,
updParametrM.Value, txtIrreduciblePolynomial.Text);

            var gfElement1 = new GaloisFieldElement(gfField,
Utility.ParsePolynomial(txtFirstPolynomial.Text));
            var gfElement2 = new GaloisFieldElement(gfField,
Utility.ParsePolynomial(txtSecondPolynomial.Text));

            if (rdoNormalBasis.Checked)
            {
                gfElement1.Basis = new NormalBasis(gfField);
```

```
                gfElement2.Basis = gfElement1.Basis;
            }

            GaloisFieldElement result;

            if (rdoAddition.Checked)
                result = gfElement1 + gfElement2;
            else if (rdoMultiplication.Checked)
                result = gfElement1 * gfElement2;
            else if (rdoFrobeniusOperation.Checked)
                result =
gfElement1.FrobeniusOperation((int)updParameterK.Value);
            else if (rdoInverseElement.Checked)
                result = gfElement1.InverseElement();
            else if (rdoDivision.Checked)
                result = gfElement1 / gfElement2;
            else if (rdoExponentiation.Checked)
                result = gfElement1 ^ (int)updParameterK.Value;
            else
                result = new GaloisFieldElement(gfField, new int[0]);

            txtCalculationResult.Text =
Utility.PolynomialToString(result.Polynomial);
        }

        private void btnTransferToOtherBasis_Click(object sender, EventArgs
e)
        {
            GaloisField galoisField = getGaloisField(updParameterP.Value,
updParametrM.Value, txtIrreduciblePolynomial.Text);
            galoisField.FindTransitionMatrix();

            if (chkSaveToFile.Checked)
            {
                galoisField.GenerateAllElementsInPolynomialBasis();
                galoisField.FindAllElementsInNormalBasis();
                SpreadsheetDocumentWriter excel = new
SpreadsheetDocumentWriter(galoisField.p, galoisField.m);
                excel.CreateFile(galoisField.irreduciblePolynomial,
                    galoisField.MatrixB,
                    galoisField.InvMatrixM,
                    galoisField.ElementsInPolynomialBasis,
                    galoisField.ElementsInNormalBasis);

                MessageBox.Show("file successfully created");
            }

            int[] initialPolynomial =
Utility.ParsePolynomial(txtInitialPolynomial.Text);
            var result = (rdoToPolynomialBasis.Checked) ?
                galoisField.TransferToPolynomialBasis(initialPolynomial) :
                galoisField.TransferToNormalBasis(initialPolynomial);

            txtTransitionResult.Text = Utility.PolynomialToString(result);
        }

        private void btnMeasureTime_Click(object sender, EventArgs e)
        {
            string filepath = txtImportFile.Text;

            var galoisField =
ImportGaloisFieldService.ImportGaloisField(filepath);
```

```
            //var stopwatch = Stopwatch.StartNew(); ;
            galoisField.FindMatrixForMultiplication();
            //stopwatch.Stop();

//MessageBox.Show((stopwatch.Elapsed.TotalMilliseconds).ToString());

            // to check refactor
            List<string> operations = new List<string>();
            foreach (var item in cklSimpleOperations.CheckedItems)
            {
                operations.Add(item.ToString());
            }

            var dictionary =
MeasurementTimeService.MeasureTime(galoisField, operations);

            var excelWriter = new
SpreadsheetDocumentWriter(String.Format("Simple Operation - Result
{0}.xlsx", DateTime.Now.ToString("yyyy-MM-dd HH mm ss")));
            excelWriter.ExportTimeMeasurement(galoisField.p, dictionary);

            if(chkFrobenius.Checked)
            {
                List<int> frobeniusOperationParameterK =
Utility.ParseIntegerRanges(txtFrobeniusParameterK.Text);
                Dictionary<String, Dictionary<int, double>> results =

MeasurementTimeService.MeasureTimeForFrobeniusOperation(galoisField,
frobeniusOperationParameterK);

                excelWriter = new
SpreadsheetDocumentWriter(String.Format("Frobenius - Result {0}.xlsx",
DateTime.Now.ToString("yyyy-MM-dd HH mm ss")));
                excelWriter.ExportTimeMeasurement(galoisField.p, results);
            }

            if(chkExponentiation.Checked)
            {
                List<int> exponentiationOperationParameterK =
Utility.ParseIntegerRanges(txtExponentiationParameterK.Text);
                Dictionary<String, Dictionary<int, double>> results =

MeasurementTimeService.MeasureTimeForExponentiation(galoisField,
exponentiationOperationParameterK);

                excelWriter = new
SpreadsheetDocumentWriter(String.Format("Exponentiation - Result {0}.xlsx",
DateTime.Now.ToString("yyyy-MM-dd HH mm ss")));
                excelWriter.ExportTimeMeasurement(galoisField.p, results);
            }

            MessageBox.Show($"File successfully created -
{excelWriter.FileName}");
        }

        private void btnGenerateOperands_Click(object sender, EventArgs e)
        {
            GaloisField galoisField =
getGaloisField(updParameterPInInputDataGeneration.Value,
                updParameterMInInputDataGeneration.Value,
                txtNormalPolynomialInInputDataGeneration.Text);
```

```
            galoisField.FindTransitionMatrix();
            int n = (int)updCount.Value;

            galoisField.GenerateRandomElementsInAllBasis(n);

            var excel = new SpreadsheetDocumentWriter(galoisField.p,
galoisField.m, n);
            excel.CreateFile(galoisField.irreduciblePolynomial,
                galoisField.MatrixB,
                galoisField.InvMatrixM,
                galoisField.ElementsInPolynomialBasis,
                galoisField.ElementsInNormalBasis);

            MessageBox.Show($"File successfully created -
{excel.FileName}");
        }

        private void btnFindNormalPolynomials_Click(object sender,
EventArgs e)
        {
            lstNormalPolynomials.Items.Clear();

            string resultString = txtIrreduciblePolynomials.Text;
            resultString =
resultString.Substring(resultString.LastIndexOf('/') +
1).Replace(".txt","");
            string[] numbers = resultString.Split(',');

            int p = int.Parse(numbers[0]);
            int m = int.Parse(numbers[1]);

            string line;
            StreamReader file = new
StreamReader(txtIrreduciblePolynomials.Text);
            while ((line = file.ReadLine()) != null)
            {
                var gfField = getGaloisField(p, m, line);

                try
                {
                    gfField.FindTransitionMatrix();

                    lstNormalPolynomials.Items.Add(line);
                } catch
                {
                }
            }

            file.Close();

            using (StreamWriter file1 = new StreamWriter(@".\output5.txt"))
            {
                foreach (var line1 in lstNormalPolynomials.Items)
                {
                    file1.WriteLine(line1.ToString());

//file.WriteLine(Convert.ToString(int.Parse(line.Trim()), 2));
                }
            }
        }
```

```
        private void lstNormalPolynomials_SelectedIndexChanged(object
sender, EventArgs e)
        {
            txtNormalPolynomialInInputDataGeneration.Text =
lstNormalPolynomials.SelectedItem.ToString();
        }

        private static void exportElements(int[][] elements, string
filename)
        {
            using (StreamWriter file = new StreamWriter(filename))
            {
                foreach (var element in elements)
                {
                    file.WriteLine(Utility.PolynomialToString(element));
                }
            }
        }

        private void btnCreateTransitionMatrix1_Click(object sender,
EventArgs e)
        {
            var n = 1000;

            //var galoisField = new GaloisField(2, 8, new int[] { 1, 1, 0,
0, 0, 0, 1, 1, 1 });
            //var galoisField = new GaloisField(2, 10, new int[] { 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1 });
            var galoisField = new GaloisField(2, 14, new int[] { 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1 });
            //var galoisField = new GaloisField(2, 16, new int[] { 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1 });
            //var galoisField = new GaloisField(2, 18, new int[] { 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1 });
            //var galoisField = new GaloisField(2, 20, new int[] { 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1 });

            var stopwatch = Stopwatch.StartNew();
            for(int i = 0; i < n; i++)
            {
                galoisField.createNormalBasisLong();
            }
            stopwatch.Stop();
            MessageBox.Show((stopwatch.Elapsed.TotalMilliseconds /
n).ToString());
            stopwatch.Reset();
        }

        private void btnCreateMyTransitionMatrix_Click(object sender,
EventArgs e)
        {
            var n = 1000;

            //var galoisField = new GaloisField(2, 8, new int[] { 1, 1, 0,
0, 0, 0, 1, 1, 1 });
            //var galoisField = new GaloisField(2, 10, new int[] { 1, 1, 0,
0, 0, 0, 1, 0, 0, 1, 1 });
            var galoisField = new GaloisField(2, 14, new int[] { 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1 });
            //var galoisField = new GaloisField(2, 16, new int[] { 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1 });
```

```
            //var galoisField = new GaloisField(2, 18, new int[] { 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1 });
            //var galoisField = new GaloisField(2, 20, new int[] { 1, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1 });

            var stopwatch = Stopwatch.StartNew();
            for (int i = 0; i < n; i++)
            {
                galoisField.createNormalBasis();
            }
            stopwatch.Stop();
            MessageBox.Show((stopwatch.Elapsed.TotalMilliseconds /
n).ToString());
            stopwatch.Reset();
        }

        private void btnFindNormalPolynomial_Click(object sender, EventArgs
e)
        {
            int n = 10;

            var stopwatch = Stopwatch.StartNew();
            for (int i = 0; i < n; i++)
            {
                long twoPowM = (long)Math.Pow(2,
(double)updMForNormalPolynomial.Value);
                long num = twoPowM * 2;
                var primes = Eratosfen.findPrimes((ulong)twoPowM * 2);
                var polynomials = new List<string>();
                var resultPolynomials = new List<string>();
                primes.Where(prime => prime >
twoPowM).ToList().ForEach(prime => polynomials.Add(Convert.ToString(prime,
2)));

                foreach (var polynomial in polynomials)
                {
                    if (!(polynomial[0] == '1' && polynomial[1] == '1'))
continue;

                    int count = 0;

                    foreach (var ch in polynomial)
                    {
                        if (ch == '1')
                            count++;
                    }

                    if (count % 2 != 0)
                    {
                        resultPolynomials.Add(polynomial);
                    }
                }

                //   var resultPrime = new StringBuilder();

                var res = new List<string>();
                var sb = new StringBuilder();

                foreach (var resultPolynomial in resultPolynomials)
                {

                    foreach (var ch in resultPolynomial)
```

```
                    {
                        sb.Append(ch);
                        sb.Append(' ');
                    }

                    res.Add(sb.ToString());

                    sb.Clear();
                    //         resultPrime.Append(resultPolynomial + "\n");
                }
                //  MessageBox.Show(resultPrime.ToString());


                //  resultPrime.Clear();
                //   foreach (var resultPolynomial in res)
                // {
                //   resultPrime.Append(resultPolynomial + "\n");
                // }
                // MessageBox.Show(resultPrime.ToString());



                lstNormalPolynomials.Items.Clear();
                foreach (var line in res)
                {
                    var gfField = getGaloisField(2,
updMForNormalPolynomial.Value, line);

                    try
                    {
                        gfField.FindTransitionMatrix();

                        lstNormalPolynomials.Items.Add(line);
                    }
                    catch
                    {
                    }
                }
            }

            MessageBox.Show((stopwatch.Elapsed.TotalMilliseconds /
n).ToString());
            stopwatch.Reset();


            using (StreamWriter file1 = new
StreamWriter(String.Format(@".\normal m={0}.txt",
updMForNormalPolynomial.Value)))
            {
                foreach (var line1 in lstNormalPolynomials.Items)
                {
                    file1.WriteLine(line1.ToString());
                }
            }
        }

        /*
        int p = 2, m = 7;
        int[] polynomial = new int[] { 1, 1, 0, 0, 0, 0, 0, 1 };

        var gfField = new GaloisField(p, m, polynomial);
        gfField.FindTransitionMatrix();
```

```
        int n = 10000;
        int[][] gfElements = new int[n][];

        string line;
        StreamReader file = new StreamReader(@"Division-
TransitionBetweenBasis.PolynomialBasis.txt");
        var i = 0;
        while ((line = file.ReadLine()) != null)
        {
            gfElements[i] =
gfField.TransferToNormalBasis(Utility.ParsePolynomial(line));
            i++;
        }

        file.Close();

        exportElements(gfElements, @"Division-
TransitionBetweenBasis.PolynomialBasis-transition.txt");
        */
    }

    partial class TransitionBetweenBasisForm
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be
disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.tpNormalBasisCalc = new System.Windows.Forms.TabPage();
            this.grpGeneralInfo = new System.Windows.Forms.GroupBox();
            this.lblIrreduciblePolynomial = new
System.Windows.Forms.Label();
            this.txtIrreduciblePolynomial = new
System.Windows.Forms.TextBox();
            this.updParametrM = new System.Windows.Forms.NumericUpDown();
            this.updParameterP = new System.Windows.Forms.NumericUpDown();
            this.lblParameterM = new System.Windows.Forms.Label();
            this.lblParameterP = new System.Windows.Forms.Label();
```

```
            this.grpTransitionBetweenBasis = new
System.Windows.Forms.GroupBox();
            this.txtTransitionResult = new System.Windows.Forms.TextBox();
            this.lblTransitionResult = new System.Windows.Forms.Label();
            this.grpToBasis = new System.Windows.Forms.GroupBox();
            this.rdoToPolynomialBasis = new
System.Windows.Forms.RadioButton();
            this.rdoToNormalBasis = new System.Windows.Forms.RadioButton();
            this.lblInitialPolynomial = new System.Windows.Forms.Label();
            this.txtInitialPolynomial = new System.Windows.Forms.TextBox();
            this.chkSaveToFile = new System.Windows.Forms.CheckBox();
            this.btnTransferToOtherBasis = new
System.Windows.Forms.Button();
            this.grpCalculator = new System.Windows.Forms.GroupBox();
            this.grpAdditionalParameters = new
System.Windows.Forms.GroupBox();
            this.updParameterK = new System.Windows.Forms.NumericUpDown();
            this.lblParameterK = new System.Windows.Forms.Label();
            this.grpBasis = new System.Windows.Forms.GroupBox();
            this.rdoNormalBasis = new System.Windows.Forms.RadioButton();
            this.rdoPolynomialBasis = new
System.Windows.Forms.RadioButton();
            this.txtCalculationResult = new System.Windows.Forms.TextBox();
            this.lblResult = new System.Windows.Forms.Label();
            this.btnCalculate = new System.Windows.Forms.Button();
            this.grpOperations = new System.Windows.Forms.GroupBox();
            this.rdoExponentiation = new
System.Windows.Forms.RadioButton();
            this.rdoDivision = new System.Windows.Forms.RadioButton();
            this.rdoInverseElement = new
System.Windows.Forms.RadioButton();
            this.rdoFrobeniusOperation = new
System.Windows.Forms.RadioButton();
            this.rdoMultiplication = new
System.Windows.Forms.RadioButton();
            this.rdoAddition = new System.Windows.Forms.RadioButton();
            this.txtSecondPolynomial = new System.Windows.Forms.TextBox();
            this.lblSecondPolynomial = new System.Windows.Forms.Label();
            this.txtFirstPolynomial = new System.Windows.Forms.TextBox();
            this.lblFirstPolynomial = new System.Windows.Forms.Label();
            this.tabMenu = new System.Windows.Forms.TabControl();
            this.tpTimeMeasurements = new System.Windows.Forms.TabPage();
            this.btnCreateMyTransitionMatrix = new
System.Windows.Forms.Button();
            this.btnCreateTransitionMatrix = new
System.Windows.Forms.Button();
            this.grpExponentiation = new System.Windows.Forms.GroupBox();
            this.txtExponentiationParameterK = new
System.Windows.Forms.TextBox();
            this.lblExponentiationParameterK = new
System.Windows.Forms.Label();
            this.chkExponentiation = new System.Windows.Forms.CheckBox();
            this.grpFrobeniusOperation = new
System.Windows.Forms.GroupBox();
            this.txtFrobeniusParameterK = new
System.Windows.Forms.TextBox();
            this.lblFrobeniusParameterK = new System.Windows.Forms.Label();
            this.chkFrobenius = new System.Windows.Forms.CheckBox();
            this.grpSimpleOperations = new System.Windows.Forms.GroupBox();
            this.cklSimpleOperations = new
System.Windows.Forms.CheckedListBox();
            this.txtImportFile = new System.Windows.Forms.TextBox();
```

```
            this.label1 = new System.Windows.Forms.Label();
            this.btnMeasureTime = new System.Windows.Forms.Button();
            this.tpOperandsGeneration = new System.Windows.Forms.TabPage();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.btnGenerateOperands = new System.Windows.Forms.Button();
            this.updCount = new System.Windows.Forms.NumericUpDown();
            this.lblCount = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.txtNormalPolynomialInInputDataGeneration = new
System.Windows.Forms.TextBox();
            this.updParameterMInInputDataGeneration = new
System.Windows.Forms.NumericUpDown();
            this.updParameterPInInputDataGeneration = new
System.Windows.Forms.NumericUpDown();
            this.label3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.grpNormalPolynomials = new
System.Windows.Forms.GroupBox();
            this.lstNormalPolynomials = new System.Windows.Forms.ListBox();
            this.txtIrreduciblePolynomials = new
System.Windows.Forms.TextBox();
            this.lblIrreduciblePolynomials = new
System.Windows.Forms.Label();
            this.btnFindNormalPolynomials = new
System.Windows.Forms.Button();
            this.updMForNormalPolynomial = new
System.Windows.Forms.NumericUpDown();
            this.btnFindNormalPolynomial = new
System.Windows.Forms.Button();
            this.tpNormalBasisCalc.SuspendLayout();
            this.grpGeneralInfo.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.updParametrM)).BeginInit()
;

((System.ComponentModel.ISupportInitialize)(this.updParameterP)).BeginInit(
);
            this.grpTransitionBetweenBasis.SuspendLayout();
            this.grpToBasis.SuspendLayout();
            this.grpCalculator.SuspendLayout();
            this.grpAdditionalParameters.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.updParameterK)).BeginInit(
);
            this.grpBasis.SuspendLayout();
            this.grpOperations.SuspendLayout();
            this.tabMenu.SuspendLayout();
            this.tpTimeMeasurements.SuspendLayout();
            this.grpExponentiation.SuspendLayout();
            this.grpFrobeniusOperation.SuspendLayout();
            this.grpSimpleOperations.SuspendLayout();
            this.tpOperandsGeneration.SuspendLayout();
            this.groupBox1.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.updCount)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.updParameterMInInputDataGe
neration)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.updParameterPInInputDataGe
neration)).BeginInit();
            this.grpNormalPolynomials.SuspendLayout();
```

```
((System.ComponentModel.ISupportInitialize)(this.updMForNormalPolynomial)).
BeginInit();
            this.SuspendLayout();
            //
            // tpNormalBasisCalc
            //
            this.tpNormalBasisCalc.Controls.Add(this.grpGeneralInfo);

this.tpNormalBasisCalc.Controls.Add(this.grpTransitionBetweenBasis);
            this.tpNormalBasisCalc.Controls.Add(this.grpCalculator);
            this.tpNormalBasisCalc.Location = new System.Drawing.Point(4,
22);
            this.tpNormalBasisCalc.Name = "tpNormalBasisCalc";
            this.tpNormalBasisCalc.Padding = new
System.Windows.Forms.Padding(3);
            this.tpNormalBasisCalc.Size = new System.Drawing.Size(636,
371);
            this.tpNormalBasisCalc.TabIndex = 1;
            this.tpNormalBasisCalc.Text = "Calculator in different basis";
            this.tpNormalBasisCalc.UseVisualStyleBackColor = true;
            //
            // grpGeneralInfo
            //

this.grpGeneralInfo.Controls.Add(this.lblIrreduciblePolynomial);

this.grpGeneralInfo.Controls.Add(this.txtIrreduciblePolynomial);
            this.grpGeneralInfo.Controls.Add(this.updParametrM);
            this.grpGeneralInfo.Controls.Add(this.updParameterP);
            this.grpGeneralInfo.Controls.Add(this.lblParameterM);
            this.grpGeneralInfo.Controls.Add(this.lblParameterP);
            this.grpGeneralInfo.Location = new System.Drawing.Point(18,
14);
            this.grpGeneralInfo.Name = "grpGeneralInfo";
            this.grpGeneralInfo.Size = new System.Drawing.Size(510, 46);
            this.grpGeneralInfo.TabIndex = 9;
            this.grpGeneralInfo.TabStop = false;
            this.grpGeneralInfo.Text = "General Information";
            //
            // lblIrreduciblePolynomial
            //
            this.lblIrreduciblePolynomial.AutoSize = true;
            this.lblIrreduciblePolynomial.Location = new
System.Drawing.Point(224, 18);
            this.lblIrreduciblePolynomial.Name =
"lblIrreduciblePolynomial";
            this.lblIrreduciblePolynomial.Size = new
System.Drawing.Size(92, 13);
            this.lblIrreduciblePolynomial.TabIndex = 12;
            this.lblIrreduciblePolynomial.Text = "Normal polynomial";
            //
            // txtIrreduciblePolynomial
            //
            this.txtIrreduciblePolynomial.Location = new
System.Drawing.Point(334, 18);
            this.txtIrreduciblePolynomial.Name =
"txtIrreduciblePolynomial";
            this.txtIrreduciblePolynomial.Size = new
System.Drawing.Size(159, 20);
            this.txtIrreduciblePolynomial.TabIndex = 11;
            //
```

```
            // updParametrM
            //
            this.updParametrM.Location = new System.Drawing.Point(140, 18);
            this.updParametrM.Name = "updParametrM";
            this.updParametrM.Size = new System.Drawing.Size(69, 20);
            this.updParametrM.TabIndex = 10;
            //
            // updParameterP
            //
            this.updParameterP.Location = new System.Drawing.Point(35, 18);
            this.updParameterP.Name = "updParameterP";
            this.updParameterP.Size = new System.Drawing.Size(69, 20);
            this.updParameterP.TabIndex = 9;
            //
            // lblParameterM
            //
            this.lblParameterM.AutoSize = true;
            this.lblParameterM.Location = new System.Drawing.Point(119,
18);
            this.lblParameterM.Name = "lblParameterM";
            this.lblParameterM.Size = new System.Drawing.Size(15, 13);
            this.lblParameterM.TabIndex = 8;
            this.lblParameterM.Text = "m";
            //
            // lblParameterP
            //
            this.lblParameterP.AutoSize = true;
            this.lblParameterP.Location = new System.Drawing.Point(14, 18);
            this.lblParameterP.Name = "lblParameterP";
            this.lblParameterP.Size = new System.Drawing.Size(13, 13);
            this.lblParameterP.TabIndex = 7;
            this.lblParameterP.Text = "p";
            //
            // grpTransitionBetweenBasis
            //

this.grpTransitionBetweenBasis.Controls.Add(this.txtTransitionResult);

this.grpTransitionBetweenBasis.Controls.Add(this.lblTransitionResult);
            this.grpTransitionBetweenBasis.Controls.Add(this.grpToBasis);

this.grpTransitionBetweenBasis.Controls.Add(this.lblInitialPolynomial);

this.grpTransitionBetweenBasis.Controls.Add(this.txtInitialPolynomial);

this.grpTransitionBetweenBasis.Controls.Add(this.chkSaveToFile);

this.grpTransitionBetweenBasis.Controls.Add(this.btnTransferToOtherBasis);
            this.grpTransitionBetweenBasis.Location = new
System.Drawing.Point(316, 66);
            this.grpTransitionBetweenBasis.Name =
"grpTransitionBetweenBasis";
            this.grpTransitionBetweenBasis.Size = new
System.Drawing.Size(314, 299);
            this.grpTransitionBetweenBasis.TabIndex = 10;
            this.grpTransitionBetweenBasis.TabStop = false;
            this.grpTransitionBetweenBasis.Text = "Transition Between
Basis";
            //
            // txtTransitionResult
            //
```

```
            this.txtTransitionResult.Location = new
System.Drawing.Point(117, 238);
            this.txtTransitionResult.Name = "txtTransitionResult";
            this.txtTransitionResult.Size = new System.Drawing.Size(100,
20);
            this.txtTransitionResult.TabIndex = 22;
            //
            // lblTransitionResult
            //
            this.lblTransitionResult.AutoSize = true;
            this.lblTransitionResult.Location = new
System.Drawing.Point(58, 241);
            this.lblTransitionResult.Name = "lblTransitionResult";
            this.lblTransitionResult.Size = new System.Drawing.Size(32,
13);
            this.lblTransitionResult.TabIndex = 21;
            this.lblTransitionResult.Text = "result";
            //
            // grpToBasis
            //
            this.grpToBasis.Controls.Add(this.rdoToPolynomialBasis);
            this.grpToBasis.Controls.Add(this.rdoToNormalBasis);
            this.grpToBasis.Location = new System.Drawing.Point(26, 61);
            this.grpToBasis.Name = "grpToBasis";
            this.grpToBasis.Size = new System.Drawing.Size(252, 74);
            this.grpToBasis.TabIndex = 20;
            this.grpToBasis.TabStop = false;
            this.grpToBasis.Text = "To Basis";
            //
            // rdoToPolynomialBasis
            //
            this.rdoToPolynomialBasis.AutoSize = true;
            this.rdoToPolynomialBasis.Location = new
System.Drawing.Point(6, 42);
            this.rdoToPolynomialBasis.Name = "rdoToPolynomialBasis";
            this.rdoToPolynomialBasis.Size = new System.Drawing.Size(117,
17);
            this.rdoToPolynomialBasis.TabIndex = 1;
            this.rdoToPolynomialBasis.TabStop = true;
            this.rdoToPolynomialBasis.Text = "To polynomial basis";
            this.rdoToPolynomialBasis.UseVisualStyleBackColor = true;
            //
            // rdoToNormalBasis
            //
            this.rdoToNormalBasis.AutoSize = true;
            this.rdoToNormalBasis.Location = new System.Drawing.Point(6,
19);
            this.rdoToNormalBasis.Name = "rdoToNormalBasis";
            this.rdoToNormalBasis.Size = new System.Drawing.Size(99, 17);
            this.rdoToNormalBasis.TabIndex = 0;
            this.rdoToNormalBasis.TabStop = true;
            this.rdoToNormalBasis.Text = "To normal basis";
            this.rdoToNormalBasis.UseVisualStyleBackColor = true;
            //
            // lblInitialPolynomial
            //
            this.lblInitialPolynomial.AutoSize = true;
            this.lblInitialPolynomial.Location = new
System.Drawing.Point(23, 38);
            this.lblInitialPolynomial.Name = "lblInitialPolynomial";
            this.lblInitialPolynomial.Size = new System.Drawing.Size(83,
13);
```

```
            this.lblInitialPolynomial.TabIndex = 19;
            this.lblInitialPolynomial.Text = "Initial polynomial";
            //
            // txtInitialPolynomial
            //
            this.txtInitialPolynomial.Location = new
System.Drawing.Point(119, 35);
            this.txtInitialPolynomial.Name = "txtInitialPolynomial";
            this.txtInitialPolynomial.Size = new System.Drawing.Size(159,
20);
            this.txtInitialPolynomial.TabIndex = 18;
            //
            // chkSaveToFile
            //
            this.chkSaveToFile.AutoSize = true;
            this.chkSaveToFile.Location = new System.Drawing.Point(26,
141);
            this.chkSaveToFile.Name = "chkSaveToFile";
            this.chkSaveToFile.Size = new System.Drawing.Size(191, 17);
            this.chkSaveToFile.TabIndex = 17;
            this.chkSaveToFile.Text = "Calculate all and save results to
file";
            this.chkSaveToFile.UseVisualStyleBackColor = true;
            //
            // btnTransferToOtherBasis
            //
            this.btnTransferToOtherBasis.Location = new
System.Drawing.Point(32, 177);
            this.btnTransferToOtherBasis.Name = "btnTransferToOtherBasis";
            this.btnTransferToOtherBasis.Size = new
System.Drawing.Size(137, 23);
            this.btnTransferToOtherBasis.TabIndex = 16;
            this.btnTransferToOtherBasis.Text = "Transfer to other basis";
            this.btnTransferToOtherBasis.UseVisualStyleBackColor = true;
            this.btnTransferToOtherBasis.Click += new
System.EventHandler(this.btnTransferToOtherBasis_Click);
            //
            // grpCalculator
            //
            this.grpCalculator.Controls.Add(this.grpAdditionalParameters);
            this.grpCalculator.Controls.Add(this.grpBasis);
            this.grpCalculator.Controls.Add(this.txtCalculationResult);
            this.grpCalculator.Controls.Add(this.lblResult);
            this.grpCalculator.Controls.Add(this.btnCalculate);
            this.grpCalculator.Controls.Add(this.grpOperations);
            this.grpCalculator.Controls.Add(this.txtSecondPolynomial);
            this.grpCalculator.Controls.Add(this.lblSecondPolynomial);
            this.grpCalculator.Controls.Add(this.txtFirstPolynomial);
            this.grpCalculator.Controls.Add(this.lblFirstPolynomial);
            this.grpCalculator.Location = new System.Drawing.Point(6, 66);
            this.grpCalculator.Name = "grpCalculator";
            this.grpCalculator.Size = new System.Drawing.Size(281, 299);
            this.grpCalculator.TabIndex = 9;
            this.grpCalculator.TabStop = false;
            this.grpCalculator.Text = "Calculator";
            //
            // grpAdditionalParameters
            //
            this.grpAdditionalParameters.Controls.Add(this.updParameterK);
            this.grpAdditionalParameters.Controls.Add(this.lblParameterK);
            this.grpAdditionalParameters.Location = new
System.Drawing.Point(12, 97);
```

```
            this.grpAdditionalParameters.Name = "grpAdditionalParameters";
            this.grpAdditionalParameters.Size = new System.Drawing.Size(97,
61);
            this.grpAdditionalParameters.TabIndex = 18;
            this.grpAdditionalParameters.TabStop = false;
            this.grpAdditionalParameters.Text = "Additional Parameters";
            //
            // updParameterK
            //
            this.updParameterK.Location = new System.Drawing.Point(25, 29);
            this.updParameterK.Name = "updParameterK";
            this.updParameterK.Size = new System.Drawing.Size(48, 20);
            this.updParameterK.TabIndex = 10;
            //
            // lblParameterK
            //
            this.lblParameterK.AutoSize = true;
            this.lblParameterK.Location = new System.Drawing.Point(6, 34);
            this.lblParameterK.Name = "lblParameterK";
            this.lblParameterK.Size = new System.Drawing.Size(13, 13);
            this.lblParameterK.TabIndex = 0;
            this.lblParameterK.Text = "k";
            //
            // grpBasis
            //
            this.grpBasis.Controls.Add(this.rdoNormalBasis);
            this.grpBasis.Controls.Add(this.rdoPolynomialBasis);
            this.grpBasis.Location = new System.Drawing.Point(12, 19);
            this.grpBasis.Name = "grpBasis";
            this.grpBasis.Size = new System.Drawing.Size(98, 72);
            this.grpBasis.TabIndex = 17;
            this.grpBasis.TabStop = false;
            this.grpBasis.Text = "Basis";
            //
            // rdoNormalBasis
            //
            this.rdoNormalBasis.AutoSize = true;
            this.rdoNormalBasis.Location = new System.Drawing.Point(6, 37);
            this.rdoNormalBasis.Name = "rdoNormalBasis";
            this.rdoNormalBasis.Size = new System.Drawing.Size(58, 17);
            this.rdoNormalBasis.TabIndex = 1;
            this.rdoNormalBasis.TabStop = true;
            this.rdoNormalBasis.Text = "Normal";
            this.rdoNormalBasis.UseVisualStyleBackColor = true;
            //
            // rdoPolynomialBasis
            //
            this.rdoPolynomialBasis.AutoSize = true;
            this.rdoPolynomialBasis.Location = new System.Drawing.Point(6,
19);
            this.rdoPolynomialBasis.Name = "rdoPolynomialBasis";
            this.rdoPolynomialBasis.Size = new System.Drawing.Size(75, 17);
            this.rdoPolynomialBasis.TabIndex = 0;
            this.rdoPolynomialBasis.TabStop = true;
            this.rdoPolynomialBasis.Text = "Polynomial";
            this.rdoPolynomialBasis.UseVisualStyleBackColor = true;
            //
            // txtCalculationResult
            //
            this.txtCalculationResult.Location = new
System.Drawing.Point(103, 260);
            this.txtCalculationResult.Name = "txtCalculationResult";
```

```
            this.txtCalculationResult.Size = new System.Drawing.Size(100,
20);
            this.txtCalculationResult.TabIndex = 16;
            //
            // lblResult
            //
            this.lblResult.AutoSize = true;
            this.lblResult.Location = new System.Drawing.Point(44, 263);
            this.lblResult.Name = "lblResult";
            this.lblResult.Size = new System.Drawing.Size(32, 13);
            this.lblResult.TabIndex = 15;
            this.lblResult.Text = "result";
            //
            // btnCalculate
            //
            this.btnCalculate.Location = new System.Drawing.Point(85, 231);
            this.btnCalculate.Name = "btnCalculate";
            this.btnCalculate.Size = new System.Drawing.Size(123, 23);
            this.btnCalculate.TabIndex = 14;
            this.btnCalculate.Text = "Calculate";
            this.btnCalculate.UseVisualStyleBackColor = true;
            this.btnCalculate.Click += new
System.EventHandler(this.btnCalculate_Click);
            //
            // grpOperations
            //
            this.grpOperations.Controls.Add(this.rdoExponentiation);
            this.grpOperations.Controls.Add(this.rdoDivision);
            this.grpOperations.Controls.Add(this.rdoInverseElement);
            this.grpOperations.Controls.Add(this.rdoFrobeniusOperation);
            this.grpOperations.Controls.Add(this.rdoMultiplication);
            this.grpOperations.Controls.Add(this.rdoAddition);
            this.grpOperations.Location = new System.Drawing.Point(116,
19);
            this.grpOperations.Name = "grpOperations";
            this.grpOperations.Size = new System.Drawing.Size(151, 139);
            this.grpOperations.TabIndex = 13;
            this.grpOperations.TabStop = false;
            this.grpOperations.Text = "Operations";
            //
            // rdoExponentiation
            //
            this.rdoExponentiation.AutoSize = true;
            this.rdoExponentiation.Location = new System.Drawing.Point(6,
110);
            this.rdoExponentiation.Name = "rdoExponentiation";
            this.rdoExponentiation.Size = new System.Drawing.Size(95, 17);
            this.rdoExponentiation.TabIndex = 5;
            this.rdoExponentiation.TabStop = true;
            this.rdoExponentiation.Text = "Exponentiation";
            this.rdoExponentiation.UseVisualStyleBackColor = true;
            //
            // rdoDivision
            //
            this.rdoDivision.AutoSize = true;
            this.rdoDivision.Location = new System.Drawing.Point(6, 93);
            this.rdoDivision.Name = "rdoDivision";
            this.rdoDivision.Size = new System.Drawing.Size(62, 17);
            this.rdoDivision.TabIndex = 4;
            this.rdoDivision.TabStop = true;
            this.rdoDivision.Text = "Division";
            this.rdoDivision.UseVisualStyleBackColor = true;
```

```
            //
            // rdoInverseElement
            //
            this.rdoInverseElement.AutoSize = true;
            this.rdoInverseElement.Location = new System.Drawing.Point(6,
75);
            this.rdoInverseElement.Name = "rdoInverseElement";
            this.rdoInverseElement.Size = new System.Drawing.Size(100, 17);
            this.rdoInverseElement.TabIndex = 3;
            this.rdoInverseElement.TabStop = true;
            this.rdoInverseElement.Text = "Inverse element";
            this.rdoInverseElement.UseVisualStyleBackColor = true;
            //
            // rdoFrobeniusOperation
            //
            this.rdoFrobeniusOperation.AutoSize = true;
            this.rdoFrobeniusOperation.Location = new
System.Drawing.Point(6, 55);
            this.rdoFrobeniusOperation.Name = "rdoFrobeniusOperation";
            this.rdoFrobeniusOperation.Size = new System.Drawing.Size(118,
17);
            this.rdoFrobeniusOperation.TabIndex = 2;
            this.rdoFrobeniusOperation.TabStop = true;
            this.rdoFrobeniusOperation.Text = "Frobenius operation";
            this.rdoFrobeniusOperation.UseVisualStyleBackColor = true;
            //
            // rdoMultiplication
            //
            this.rdoMultiplication.AutoSize = true;
            this.rdoMultiplication.Location = new System.Drawing.Point(6,
37);
            this.rdoMultiplication.Name = "rdoMultiplication";
            this.rdoMultiplication.Size = new System.Drawing.Size(86, 17);
            this.rdoMultiplication.TabIndex = 1;
            this.rdoMultiplication.TabStop = true;
            this.rdoMultiplication.Text = "Multiplication";
            this.rdoMultiplication.UseVisualStyleBackColor = true;
            //
            // rdoAddition
            //
            this.rdoAddition.AutoSize = true;
            this.rdoAddition.Location = new System.Drawing.Point(6, 19);
            this.rdoAddition.Name = "rdoAddition";
            this.rdoAddition.Size = new System.Drawing.Size(63, 17);
            this.rdoAddition.TabIndex = 0;
            this.rdoAddition.TabStop = true;
            this.rdoAddition.Text = "Addition";
            this.rdoAddition.UseVisualStyleBackColor = true;
            //
            // txtSecondPolynomial
            //
            this.txtSecondPolynomial.Location = new
System.Drawing.Point(121, 205);
            this.txtSecondPolynomial.Name = "txtSecondPolynomial";
            this.txtSecondPolynomial.Size = new System.Drawing.Size(100,
20);
            this.txtSecondPolynomial.TabIndex = 12;
            //
            // lblSecondPolynomial
            //
            this.lblSecondPolynomial.AutoSize = true;
```

```
            this.lblSecondPolynomial.Location = new
System.Drawing.Point(25, 208);
            this.lblSecondPolynomial.Name = "lblSecondPolynomial";
            this.lblSecondPolynomial.Size = new System.Drawing.Size(94,
13);
            this.lblSecondPolynomial.TabIndex = 11;
            this.lblSecondPolynomial.Text = "second polynomial";
            //
            // txtFirstPolynomial
            //
            this.txtFirstPolynomial.Location = new
System.Drawing.Point(121, 170);
            this.txtFirstPolynomial.Name = "txtFirstPolynomial";
            this.txtFirstPolynomial.Size = new System.Drawing.Size(100,
20);
            this.txtFirstPolynomial.TabIndex = 10;
            //
            // lblFirstPolynomial
            //
            this.lblFirstPolynomial.AutoSize = true;
            this.lblFirstPolynomial.Location = new System.Drawing.Point(40,
173);
            this.lblFirstPolynomial.Name = "lblFirstPolynomial";
            this.lblFirstPolynomial.Size = new System.Drawing.Size(75, 13);
            this.lblFirstPolynomial.TabIndex = 9;
            this.lblFirstPolynomial.Text = "first polynomial";
            //
            // tabMenu
            //
            this.tabMenu.Controls.Add(this.tpNormalBasisCalc);
            this.tabMenu.Controls.Add(this.tpTimeMeasurements);
            this.tabMenu.Controls.Add(this.tpOperandsGeneration);
            this.tabMenu.Location = new System.Drawing.Point(4, 3);
            this.tabMenu.Name = "tabMenu";
            this.tabMenu.SelectedIndex = 0;
            this.tabMenu.Size = new System.Drawing.Size(644, 397);
            this.tabMenu.TabIndex = 8;
            //
            // tpTimeMeasurements
            //

this.tpTimeMeasurements.Controls.Add(this.btnCreateMyTransitionMatrix);

this.tpTimeMeasurements.Controls.Add(this.btnCreateTransitionMatrix);
            this.tpTimeMeasurements.Controls.Add(this.grpExponentiation);

this.tpTimeMeasurements.Controls.Add(this.grpFrobeniusOperation);
            this.tpTimeMeasurements.Controls.Add(this.grpSimpleOperations);
            this.tpTimeMeasurements.Controls.Add(this.txtImportFile);
            this.tpTimeMeasurements.Controls.Add(this.label1);
            this.tpTimeMeasurements.Controls.Add(this.btnMeasureTime);
            this.tpTimeMeasurements.Location = new System.Drawing.Point(4,
22);
            this.tpTimeMeasurements.Name = "tpTimeMeasurements";
            this.tpTimeMeasurements.Padding = new
System.Windows.Forms.Padding(3);
            this.tpTimeMeasurements.Size = new System.Drawing.Size(636,
371);
            this.tpTimeMeasurements.TabIndex = 2;
            this.tpTimeMeasurements.Text = "Time Measurements";
            this.tpTimeMeasurements.UseVisualStyleBackColor = true;
            //
```

```
            // btnCreateMyTransitionMatrix
            //
            this.btnCreateMyTransitionMatrix.Location = new
System.Drawing.Point(462, 168);
            this.btnCreateMyTransitionMatrix.Name =
"btnCreateMyTransitionMatrix";
            this.btnCreateMyTransitionMatrix.Size = new
System.Drawing.Size(147, 23);
            this.btnCreateMyTransitionMatrix.TabIndex = 13;
            this.btnCreateMyTransitionMatrix.Text = "Create Transition
Matrix My";
            this.btnCreateMyTransitionMatrix.UseVisualStyleBackColor =
true;
            this.btnCreateMyTransitionMatrix.Click += new
System.EventHandler(this.btnCreateMyTransitionMatrix_Click);
            //
            // btnCreateTransitionMatrix
            //
            this.btnCreateTransitionMatrix.Location = new
System.Drawing.Point(462, 79);
            this.btnCreateTransitionMatrix.Name =
"btnCreateTransitionMatrix";
            this.btnCreateTransitionMatrix.Size = new
System.Drawing.Size(147, 23);
            this.btnCreateTransitionMatrix.TabIndex = 12;
            this.btnCreateTransitionMatrix.Text = "Create Transition
Matrix";
            this.btnCreateTransitionMatrix.UseVisualStyleBackColor = true;
            this.btnCreateTransitionMatrix.Click += new
System.EventHandler(this.btnCreateTransitionMatrix1_Click);
            //
            // grpExponentiation
            //

this.grpExponentiation.Controls.Add(this.txtExponentiationParameterK);

this.grpExponentiation.Controls.Add(this.lblExponentiationParameterK);
            this.grpExponentiation.Controls.Add(this.chkExponentiation);
            this.grpExponentiation.Location = new System.Drawing.Point(261,
155);
            this.grpExponentiation.Name = "grpExponentiation";
            this.grpExponentiation.Size = new System.Drawing.Size(136,
103);
            this.grpExponentiation.TabIndex = 11;
            this.grpExponentiation.TabStop = false;
            this.grpExponentiation.Text = "Exponentiation";
            //
            // txtExponentiationParameterK
            //
            this.txtExponentiationParameterK.Location = new
System.Drawing.Point(25, 53);
            this.txtExponentiationParameterK.Name =
"txtExponentiationParameterK";
            this.txtExponentiationParameterK.Size = new
System.Drawing.Size(100, 20);
            this.txtExponentiationParameterK.TabIndex = 3;
            this.txtExponentiationParameterK.Text = "1-6;8;10-13";
            //
            // lblExponentiationParameterK
            //
            this.lblExponentiationParameterK.AutoSize = true;
```

```
            this.lblExponentiationParameterK.Location = new
System.Drawing.Point(6, 56);
            this.lblExponentiationParameterK.Name =
"lblExponentiationParameterK";
            this.lblExponentiationParameterK.Size = new
System.Drawing.Size(13, 13);
            this.lblExponentiationParameterK.TabIndex = 2;
            this.lblExponentiationParameterK.Text = "k";
            //
            // chkExponentiation
            //
            this.chkExponentiation.AutoSize = true;
            this.chkExponentiation.Checked = true;
            this.chkExponentiation.CheckState =
System.Windows.Forms.CheckState.Checked;
            this.chkExponentiation.Location = new System.Drawing.Point(9,
19);
            this.chkExponentiation.Name = "chkExponentiation";
            this.chkExponentiation.Size = new System.Drawing.Size(64, 17);
            this.chkExponentiation.TabIndex = 1;
            this.chkExponentiation.Text = "enabled";
            this.chkExponentiation.UseVisualStyleBackColor = true;
            //
            // grpFrobeniusOperation
            //

this.grpFrobeniusOperation.Controls.Add(this.txtFrobeniusParameterK);

this.grpFrobeniusOperation.Controls.Add(this.lblFrobeniusParameterK);
            this.grpFrobeniusOperation.Controls.Add(this.chkFrobenius);
            this.grpFrobeniusOperation.Location = new
System.Drawing.Point(261, 49);
            this.grpFrobeniusOperation.Name = "grpFrobeniusOperation";
            this.grpFrobeniusOperation.Size = new System.Drawing.Size(136,
100);
            this.grpFrobeniusOperation.TabIndex = 10;
            this.grpFrobeniusOperation.TabStop = false;
            this.grpFrobeniusOperation.Text = "Frobenius Operation";
            //
            // txtFrobeniusParameterK
            //
            this.txtFrobeniusParameterK.Location = new
System.Drawing.Point(25, 50);
            this.txtFrobeniusParameterK.Name = "txtFrobeniusParameterK";
            this.txtFrobeniusParameterK.Size = new System.Drawing.Size(100,
20);
            this.txtFrobeniusParameterK.TabIndex = 2;
            this.txtFrobeniusParameterK.Text = "1-5;7;9-13";
            //
            // lblFrobeniusParameterK
            //
            this.lblFrobeniusParameterK.AutoSize = true;
            this.lblFrobeniusParameterK.Location = new
System.Drawing.Point(6, 53);
            this.lblFrobeniusParameterK.Name = "lblFrobeniusParameterK";
            this.lblFrobeniusParameterK.Size = new System.Drawing.Size(13,
13);
            this.lblFrobeniusParameterK.TabIndex = 1;
            this.lblFrobeniusParameterK.Text = "k";
            //
            // chkFrobenius
            //
```

```
            this.chkFrobenius.AutoSize = true;
            this.chkFrobenius.Checked = true;
            this.chkFrobenius.CheckState =
System.Windows.Forms.CheckState.Checked;
            this.chkFrobenius.Location = new System.Drawing.Point(6, 19);
            this.chkFrobenius.Name = "chkFrobenius";
            this.chkFrobenius.Size = new System.Drawing.Size(64, 17);
            this.chkFrobenius.TabIndex = 0;
            this.chkFrobenius.Text = "enabled";
            this.chkFrobenius.UseVisualStyleBackColor = true;
            //
            // grpSimpleOperations
            //

this.grpSimpleOperations.Controls.Add(this.cklSimpleOperations);
            this.grpSimpleOperations.Location = new
System.Drawing.Point(17, 49);
            this.grpSimpleOperations.Name = "grpSimpleOperations";
            this.grpSimpleOperations.Size = new System.Drawing.Size(218,
209);
            this.grpSimpleOperations.TabIndex = 8;
            this.grpSimpleOperations.TabStop = false;
            this.grpSimpleOperations.Text = "Simple operations";
            //
            // cklSimpleOperations
            //
            this.cklSimpleOperations.FormattingEnabled = true;
            this.cklSimpleOperations.Location = new System.Drawing.Point(6,
19);
            this.cklSimpleOperations.Name = "cklSimpleOperations";
            this.cklSimpleOperations.Size = new System.Drawing.Size(207,
184);
            this.cklSimpleOperations.TabIndex = 9;
            //
            // txtImportFile
            //
            this.txtImportFile.Location = new System.Drawing.Point(122,
23);
            this.txtImportFile.Name = "txtImportFile";
            this.txtImportFile.Size = new System.Drawing.Size(100, 20);
            this.txtImportFile.TabIndex = 7;
            this.txtImportFile.Text = "2 4 10.xlsx";
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(24, 26);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(92, 13);
            this.label1.TabIndex = 6;
            this.label1.Text = "File with operands";
            //
            // btnMeasureTime
            //
            this.btnMeasureTime.Location = new System.Drawing.Point(209,
286);
            this.btnMeasureTime.Name = "btnMeasureTime";
            this.btnMeasureTime.Size = new System.Drawing.Size(115, 22);
            this.btnMeasureTime.TabIndex = 5;
            this.btnMeasureTime.Text = "Measure time";
            this.btnMeasureTime.UseVisualStyleBackColor = true;
```

```
            this.btnMeasureTime.Click += new
System.EventHandler(this.btnMeasureTime_Click);
            //
            // tpOperandsGeneration
            //
            this.tpOperandsGeneration.Controls.Add(this.groupBox1);

this.tpOperandsGeneration.Controls.Add(this.grpNormalPolynomials);
            this.tpOperandsGeneration.Cursor =
System.Windows.Forms.Cursors.Default;
            this.tpOperandsGeneration.Location = new
System.Drawing.Point(4, 22);
            this.tpOperandsGeneration.Name = "tpOperandsGeneration";
            this.tpOperandsGeneration.Padding = new
System.Windows.Forms.Padding(3);
            this.tpOperandsGeneration.Size = new System.Drawing.Size(636,
371);
            this.tpOperandsGeneration.TabIndex = 3;
            this.tpOperandsGeneration.Text = "Input Data Generation";
            this.tpOperandsGeneration.UseVisualStyleBackColor = true;
            //
            // groupBox1
            //
            this.groupBox1.Controls.Add(this.btnGenerateOperands);
            this.groupBox1.Controls.Add(this.updCount);
            this.groupBox1.Controls.Add(this.lblCount);
            this.groupBox1.Controls.Add(this.label2);

this.groupBox1.Controls.Add(this.txtNormalPolynomialInInputDataGeneration);

this.groupBox1.Controls.Add(this.updParameterMInInputDataGeneration);

this.groupBox1.Controls.Add(this.updParameterPInInputDataGeneration);
            this.groupBox1.Controls.Add(this.label3);
            this.groupBox1.Controls.Add(this.label4);
            this.groupBox1.Location = new System.Drawing.Point(15, 22);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(280, 282);
            this.groupBox1.TabIndex = 10;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "Operands Generation";
            //
            // btnGenerateOperands
            //
            this.btnGenerateOperands.Location = new
System.Drawing.Point(107, 168);
            this.btnGenerateOperands.Name = "btnGenerateOperands";
            this.btnGenerateOperands.Size = new System.Drawing.Size(124,
23);
            this.btnGenerateOperands.TabIndex = 15;
            this.btnGenerateOperands.Text = "Generate Operands";
            this.btnGenerateOperands.UseVisualStyleBackColor = true;
            this.btnGenerateOperands.Click += new
System.EventHandler(this.btnGenerateOperands_Click);
            //
            // updCount
            //
            this.updCount.Location = new System.Drawing.Point(111, 129);
            this.updCount.Maximum = new decimal(new int[] {
            1410065408,
            2,
            0,
```

```
            0});
            this.updCount.Name = "updCount";
            this.updCount.Size = new System.Drawing.Size(120, 20);
            this.updCount.TabIndex = 14;
            this.updCount.Value = new decimal(new int[] {
            1000,
            0,
            0,
            0});
            //
            // lblCount
            //
            this.lblCount.AutoSize = true;
            this.lblCount.Location = new System.Drawing.Point(51, 131);
            this.lblCount.Name = "lblCount";
            this.lblCount.Size = new System.Drawing.Size(35, 13);
            this.lblCount.TabIndex = 13;
            this.lblCount.Text = "Count";
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(14, 82);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(92, 13);
            this.label2.TabIndex = 12;
            this.label2.Text = "Normal polynomial";
            //
            // txtNormalPolynomialInInputDataGeneration
            //
            this.txtNormalPolynomialInInputDataGeneration.Location = new
System.Drawing.Point(124, 82);
            this.txtNormalPolynomialInInputDataGeneration.Name =
"txtNormalPolynomialInInputDataGeneration";
            this.txtNormalPolynomialInInputDataGeneration.Size = new
System.Drawing.Size(107, 20);
            this.txtNormalPolynomialInInputDataGeneration.TabIndex = 11;
            this.txtNormalPolynomialInInputDataGeneration.Text = "1 1 1 1
1";
            //
            // updParameterMInInputDataGeneration
            //
            this.updParameterMInInputDataGeneration.Location = new
System.Drawing.Point(142, 36);
            this.updParameterMInInputDataGeneration.Name =
"updParameterMInInputDataGeneration";
            this.updParameterMInInputDataGeneration.Size = new
System.Drawing.Size(69, 20);
            this.updParameterMInInputDataGeneration.TabIndex = 10;
            this.updParameterMInInputDataGeneration.Value = new decimal(new
int[] {
            4,
            0,
            0,
            0});
            //
            // updParameterPInInputDataGeneration
            //
            this.updParameterPInInputDataGeneration.Location = new
System.Drawing.Point(37, 36);
            this.updParameterPInInputDataGeneration.Name =
"updParameterPInInputDataGeneration";
```

```
            this.updParameterPInInputDataGeneration.Size = new
System.Drawing.Size(69, 20);
            this.updParameterPInInputDataGeneration.TabIndex = 9;
            this.updParameterPInInputDataGeneration.Value = new decimal(new
int[] {
            2,
            0,
            0,
            0});
            //
            // label3
            //
            this.label3.AutoSize = true;
            this.label3.Location = new System.Drawing.Point(121, 36);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(15, 13);
            this.label3.TabIndex = 8;
            this.label3.Text = "m";
            //
            // label4
            //
            this.label4.AutoSize = true;
            this.label4.Location = new System.Drawing.Point(16, 36);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(13, 13);
            this.label4.TabIndex = 7;
            this.label4.Text = "p";
            //
            // grpNormalPolynomials
            //

this.grpNormalPolynomials.Controls.Add(this.btnFindNormalPolynomial);

this.grpNormalPolynomials.Controls.Add(this.updMForNormalPolynomial);

this.grpNormalPolynomials.Controls.Add(this.lstNormalPolynomials);

this.grpNormalPolynomials.Controls.Add(this.txtIrreduciblePolynomials);

this.grpNormalPolynomials.Controls.Add(this.lblIrreduciblePolynomials);

this.grpNormalPolynomials.Controls.Add(this.btnFindNormalPolynomials);
            this.grpNormalPolynomials.Location = new
System.Drawing.Point(301, 22);
            this.grpNormalPolynomials.Name = "grpNormalPolynomials";
            this.grpNormalPolynomials.Size = new System.Drawing.Size(329,
282);
            this.grpNormalPolynomials.TabIndex = 4;
            this.grpNormalPolynomials.TabStop = false;
            this.grpNormalPolynomials.Text = "Normal polynomials";
            //
            // lstNormalPolynomials
            //
            this.lstNormalPolynomials.FormattingEnabled = true;
            this.lstNormalPolynomials.Location = new
System.Drawing.Point(33, 97);
            this.lstNormalPolynomials.Name = "lstNormalPolynomials";
            this.lstNormalPolynomials.Size = new System.Drawing.Size(270,
134);
            this.lstNormalPolynomials.TabIndex = 6;
            this.lstNormalPolynomials.SelectedIndexChanged += new
System.EventHandler(this.lstNormalPolynomials_SelectedIndexChanged);
```

```
            //
            // txtIrreduciblePolynomials
            //
            this.txtIrreduciblePolynomials.Location = new
System.Drawing.Point(166, 22);
            this.txtIrreduciblePolynomials.Name =
"txtIrreduciblePolynomials";
            this.txtIrreduciblePolynomials.Size = new
System.Drawing.Size(157, 20);
            this.txtIrreduciblePolynomials.TabIndex = 5;
            this.txtIrreduciblePolynomials.Text =
"irreduciblePolynomials/2,4.txt";
            //
            // lblIrreduciblePolynomials
            //
            this.lblIrreduciblePolynomials.AutoSize = true;
            this.lblIrreduciblePolynomials.Location = new
System.Drawing.Point(6, 25);
            this.lblIrreduciblePolynomials.Name =
"lblIrreduciblePolynomials";
            this.lblIrreduciblePolynomials.Size = new
System.Drawing.Size(154, 13);
            this.lblIrreduciblePolynomials.TabIndex = 4;
            this.lblIrreduciblePolynomials.Text = "File with Irreducible
polynomials";
            //
            // btnFindNormalPolynomials
            //
            this.btnFindNormalPolynomials.Location = new
System.Drawing.Point(113, 53);
            this.btnFindNormalPolynomials.Name =
"btnFindNormalPolynomials";
            this.btnFindNormalPolynomials.Size = new
System.Drawing.Size(162, 23);
            this.btnFindNormalPolynomials.TabIndex = 3;
            this.btnFindNormalPolynomials.Text = "Find Normal Polynomials";
            this.btnFindNormalPolynomials.UseVisualStyleBackColor = true;
            this.btnFindNormalPolynomials.Click += new
System.EventHandler(this.btnFindNormalPolynomials_Click);
            //
            // updMForNormalPolynomial
            //
            this.updMForNormalPolynomial.Location = new
System.Drawing.Point(59, 250);
            this.updMForNormalPolynomial.Name = "updMForNormalPolynomial";
            this.updMForNormalPolynomial.Size = new System.Drawing.Size(69,
20);
            this.updMForNormalPolynomial.TabIndex = 16;
            this.updMForNormalPolynomial.Value = new decimal(new int[] {
            2,
            0,
            0,
            0});
            //
            // btnFindNormalPolynomial
            //
            this.btnFindNormalPolynomial.Location = new
System.Drawing.Point(173, 250);
            this.btnFindNormalPolynomial.Name = "btnFindNormalPolynomial";
            this.btnFindNormalPolynomial.Size = new
System.Drawing.Size(130, 23);
            this.btnFindNormalPolynomial.TabIndex = 17;
```

```
            this.btnFindNormalPolynomial.Text = "Find Normal Polynomial";
            this.btnFindNormalPolynomial.UseVisualStyleBackColor = true;
            this.btnFindNormalPolynomial.Click += new
System.EventHandler(this.btnFindNormalPolynomial_Click);
            //
            // TransitionBetweenBasisForm
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(650, 400);
            this.Controls.Add(this.tabMenu);
            this.Name = "TransitionBetweenBasisForm";
            this.Text = "TransitionBetweenBasisForm";
            this.tpNormalBasisCalc.ResumeLayout(false);
            this.grpGeneralInfo.ResumeLayout(false);
            this.grpGeneralInfo.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.updParametrM)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.updParameterP)).EndInit();
            this.grpTransitionBetweenBasis.ResumeLayout(false);
            this.grpTransitionBetweenBasis.PerformLayout();
            this.grpToBasis.ResumeLayout(false);
            this.grpToBasis.PerformLayout();
            this.grpCalculator.ResumeLayout(false);
            this.grpCalculator.PerformLayout();
            this.grpAdditionalParameters.ResumeLayout(false);
            this.grpAdditionalParameters.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.updParameterK)).EndInit();
            this.grpBasis.ResumeLayout(false);
            this.grpBasis.PerformLayout();
            this.grpOperations.ResumeLayout(false);
            this.grpOperations.PerformLayout();
            this.tabMenu.ResumeLayout(false);
            this.tpTimeMeasurements.ResumeLayout(false);
            this.tpTimeMeasurements.PerformLayout();
            this.grpExponentiation.ResumeLayout(false);
            this.grpExponentiation.PerformLayout();
            this.grpFrobeniusOperation.ResumeLayout(false);
            this.grpFrobeniusOperation.PerformLayout();
            this.grpSimpleOperations.ResumeLayout(false);
            this.tpOperandsGeneration.ResumeLayout(false);
            this.groupBox1.ResumeLayout(false);
            this.groupBox1.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.updCount)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.updParameterMInInputDataGe
neration)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.updParameterPInInputDataGe
neration)).EndInit();
            this.grpNormalPolynomials.ResumeLayout(false);
            this.grpNormalPolynomials.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.updMForNormalPolynomial)).
EndInit();
            this.ResumeLayout(false);

        }
```

```
        #endregion
        private System.Windows.Forms.TabPage tpNormalBasisCalc;
        private System.Windows.Forms.GroupBox grpTransitionBetweenBasis;
        private System.Windows.Forms.GroupBox grpToBasis;
        private System.Windows.Forms.RadioButton rdoToPolynomialBasis;
        private System.Windows.Forms.RadioButton rdoToNormalBasis;
        private System.Windows.Forms.Label lblInitialPolynomial;
        private System.Windows.Forms.TextBox txtInitialPolynomial;
        private System.Windows.Forms.CheckBox chkSaveToFile;
        private System.Windows.Forms.Button btnTransferToOtherBasis;
        private System.Windows.Forms.GroupBox grpCalculator;
        private System.Windows.Forms.GroupBox grpBasis;
        private System.Windows.Forms.RadioButton rdoNormalBasis;
        private System.Windows.Forms.RadioButton rdoPolynomialBasis;
        private System.Windows.Forms.TextBox txtCalculationResult;
        private System.Windows.Forms.Label lblResult;
        private System.Windows.Forms.Button btnCalculate;
        private System.Windows.Forms.GroupBox grpOperations;
        private System.Windows.Forms.RadioButton rdoFrobeniusOperation;
        private System.Windows.Forms.RadioButton rdoMultiplication;
        private System.Windows.Forms.RadioButton rdoAddition;
        private System.Windows.Forms.TextBox txtSecondPolynomial;
        private System.Windows.Forms.Label lblSecondPolynomial;
        private System.Windows.Forms.TextBox txtFirstPolynomial;
        private System.Windows.Forms.Label lblFirstPolynomial;
        private System.Windows.Forms.TabControl tabMenu;
        private System.Windows.Forms.TextBox txtTransitionResult;
        private System.Windows.Forms.Label lblTransitionResult;
        private System.Windows.Forms.RadioButton rdoInverseElement;
        private System.Windows.Forms.RadioButton rdoDivision;
        private System.Windows.Forms.GroupBox grpAdditionalParameters;
        private System.Windows.Forms.NumericUpDown updParameterK;
        private System.Windows.Forms.Label lblParameterK;
        private System.Windows.Forms.RadioButton rdoExponentiation;
        private System.Windows.Forms.TabPage tpTimeMeasurements;
        private System.Windows.Forms.Button btnMeasureTime;
        private System.Windows.Forms.TabPage tpOperandsGeneration;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox txtImportFile;
        private System.Windows.Forms.GroupBox grpSimpleOperations;
        private System.Windows.Forms.CheckedListBox cklSimpleOperations;
        private System.Windows.Forms.GroupBox grpNormalPolynomials;
        private System.Windows.Forms.TextBox txtIrreduciblePolynomials;
        private System.Windows.Forms.Label lblIrreduciblePolynomials;
        private System.Windows.Forms.Button btnFindNormalPolynomials;
        private System.Windows.Forms.ListBox lstNormalPolynomials;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Button btnGenerateOperands;
        private System.Windows.Forms.NumericUpDown updCount;
        private System.Windows.Forms.Label lblCount;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox
txtNormalPolynomialInInputDataGeneration;
        private System.Windows.Forms.NumericUpDown
updParameterMInInputDataGeneration;
        private System.Windows.Forms.NumericUpDown
updParameterPInInputDataGeneration;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label lblParameterP;
        private System.Windows.Forms.Label lblParameterM;
        private System.Windows.Forms.NumericUpDown updParameterP;
```

```
        private System.Windows.Forms.NumericUpDown updParametrM;
        private System.Windows.Forms.TextBox txtIrreduciblePolynomial;
        private System.Windows.Forms.Label lblIrreduciblePolynomial;
        private System.Windows.Forms.GroupBox grpGeneralInfo;
        private System.Windows.Forms.GroupBox grpFrobeniusOperation;
        private System.Windows.Forms.GroupBox grpExponentiation;
        private System.Windows.Forms.TextBox txtFrobeniusParameterK;
        private System.Windows.Forms.Label lblFrobeniusParameterK;
        private System.Windows.Forms.CheckBox chkFrobenius;
        private System.Windows.Forms.TextBox txtExponentiationParameterK;
        private System.Windows.Forms.Label lblExponentiationParameterK;
        private System.Windows.Forms.CheckBox chkExponentiation;
        private System.Windows.Forms.Button btnCreateMyTransitionMatrix;
        private System.Windows.Forms.Button btnCreateTransitionMatrix;
        private System.Windows.Forms.Button btnFindNormalPolynomial;
        private System.Windows.Forms.NumericUpDown updMForNormalPolynomial;
    }
```

Додаток З. Приклади txt-файлів з знайденими нормальними поліномами для полів з $m < 8$

Примітка: коефіцієнти поліномів розміщуються за спаданням

Знайдені нормальні поліноми для $GF(2^3)$

1 1 0 1

Знайдені нормальні поліноми для $GF(2^4)$

1 1 1 1 1

Знайдені нормальні поліноми для $GF(2^5)$

1 1 1 0 1 1

1 1 1 1 0 1

Знайдені нормальні поліноми для $GF(2^7)$

1 1 0 0 0 0 0 1

1 1 0 1 0 0 1 1

1 1 1 0 0 1 0 1

1 1 1 0 1 1 1 1

Додаток И. Приклад txt-файлу з знайденими нормальними поліномами для поля $GF(2^{18})$

Примітка: коефіцієнти поліномів розміщуються за спаданням

1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1 1	1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1
1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 1	1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1
1 1 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 1 1 1	1 1 0 0 0 0 1 0 0 0 1 1 0 1 1 1 0 0 1
1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 1 0 1 1	1 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1 1 1 1
1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1	1 1 0 0 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1
1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1	1 1 0 0 0 0 1 0 0 1 1 1 1 0 0 1 1 1 1
1 1 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 0 1	1 1 0 0 0 0 1 0 1 0 0 0 0 0 0 1 0 1 1
1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 1 0 1	1 1 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1
1 1 0 0 0 0 0 0 0 1 1 1 0 0 1 0 0 0 1	1 1 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1
1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 1 1	1 1 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1 1
1 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 1 1	1 1 0 0 0 0 1 0 1 1 0 0 0 0 0 1 1 1 1
1 1 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 0 0 1	1 1 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1 0 1
1 1 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 1	1 1 0 0 0 0 1 0 1 1 0 1 1 0 1 1 0 1 1
1 1 0 0 0 0 0 0 0 1 1 1 1 0 1 1 0 1 1 1	1 1 0 0 0 0 1 0 1 1 0 1 1 1 1 1 0 0 1
1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 1 1 0 1	1 1 0 0 0 0 1 0 1 1 1 1 0 1 0 0 1 1 1
1 1 0 0 0 0 0 1 0 0 0 1 1 0 0 0 0 1 1	1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 1 1
1 1 0 0 0 0 0 1 0 1 1 0 0 0 1 1 1 0 1	1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1
1 1 0 0 0 0 0 1 0 1 1 0 1 0 0 0 1 1 1	1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 1
1 1 0 0 0 0 0 1 1 0 1 0 1 0 0 1 0 1 1	1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 1 1 1
1 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 1	1 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 0 1
1 1 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 0 1	1 1 0 0 0 0 1 1 0 0 1 0 0 1 1 0 0 1 1
1 1 0 0 0 0 0 1 1 1 1 0 1 0 0 0 0 1	1 1 0 0 0 0 1 1 0 1 0 1 1 1 0 1 1 0 1
1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 1	1 1 0 0 0 0 1 1 0 1 1 0 1 1 1 0 0 1 1
1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 0 0 0 1	1 1 0 0 0 0 1 1 0 1 1 0 1 1 1 1 0 0 1

1 1 0 0 0 0 1 1 0 1 1 1 0 0 0 0 1 0 1

1 1 0 0 0 0 1 1 0 1 1 1 1 1 1 1 1 0 1

1 1 0 0 0 0 1 1 1 0 1 1 1 0 1 0 1 0 1

1 1 0 0 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1

1 1 0 0 0 0 1 1 1 1 0 0 1 0 0 0 0 1 1

1 1 0 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1

1 1 0 0 0 0 1 1 1 1 1 0 0 1 1 1 0 0 1

1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 0 1 1 1

1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 1

1 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 1 1 1

1 1 0 0 0 1 0 0 0 0 1 0 0 0 1 1 0 0 1

1 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1 1 0 1

1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0 1 0 1

1 1 0 0 0 1 0 0 0 1 0 1 0 1 0 1 0 1 1

1 1 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 1

1 1 0 0 0 1 0 0 0 1 1 1 1 0 0 0 0 1 1

1 1 0 0 0 1 0 0 1 0 0 0 1 1 0 1 1 0 1

1 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 0 1

1 1 0 0 0 1 0 0 1 1 0 1 0 0 1 1 1 1 1

1 1 0 0 0 1 0 0 1 1 1 0 1 1 0 1 1 0 1

1 1 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1

1 1 0 0 0 1 0 0 1 1 1 1 0 0 0 0 0 1 1

1 1 0 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 1

1 1 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 1

1 1 0 0 0 1 0 1 0 1 1 1 1 0 0 1 1 0 1

1 1 0 0 0 1 0 1 1 0 0 0 0 0 0 1 0 0 1

1 1 0 0 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1

1 1 0 0 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1

1 1 0 0 0 1 0 1 1 0 1 0 1 0 1 1 0 1 1

1 1 0 0 0 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1

1 1 0 0 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1 1

1 1 0 0 0 1 0 1 1 1 0 0 0 1 0 0 1 0 0 1

1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 0 1 0 1

1 1 0 0 0 1 0 1 1 1 0 0 1 1 0 0 1 0 0 1

1 1 0 0 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1 1

1 1 0 0 0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1

1 1 0 0 0 1 0 1 1 1 1 0 0 0 0 0 0 1 1 1

1 1 0 0 0 1 0 1 1 1 1 0 0 1 0 0 1 0 0 1

1 1 0 0 0 1 0 1 1 1 1 0 1 0 0 1 0 0 0 1

1 1 0 0 0 1 0 1 1 1 1 0 1 1 0 1 1 0 1 1

1 1 0 0 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1

1 1 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 0 1

1 1 0 0 0 1 1 1 1 0 1 1 0 1 1 1 1 0 1

1 1 0 0 0 1 1 1 1 1 0 0 1 0 0 1 0 1 1

1 1 0 0 0 1 1 1 1 1 0 1 0 1 1 0 1 1 1

1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1

1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 0 1 1 1

1 1 0 0 1 0 0 0 0 0 1 1 0 0 0 0 0 1 1

1 1 0 0 1 0 0 0 0 1 0 0 0 1 1 1 0 1 1

1 1 0 0 1 0 0 0 0 1 0 0 0 1 1 1 1 0 1

1 1 0 0 1 0 0 0 0 1 1 1 0 0 0 1 0 1 1

1 1 0 0 1 0 0 0 1 0 0 0 0 1 0 1 1 1 1

1 1 0 0 1 0 0 0 1 0 0 0 1 1 0 1 0 1 1

1 1 0 0 1 0 0 0 1 0 1 1 0 1 0 1 0 0 1

1 1 0 0 1 0 0 0 1 0 1 1 1 0 1 1 1 0 1

1 1 0 0 1 0 0 0 1 1 0 0 1 0 0 1 0 1 1

1 1 0 0 1 0 0 0 1 1 0 1 1 1 0 0 0 0 1

1 1 0 0 1 0 0 0 1 1 0 1 1 1 1 0 0 1 1

1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 0 1 1 1

1 1 0 0 1 0 0 1 0 0 0 0 0 1 1 1 0 1 1

1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1

1 1 0 0 1 0 0 1 0 0 0 1 0 0 1 0 1 1 1

1 1 0 0 1 0 0 1 0 0 1 1 1 0 0 0 0 1 1

1 1 0 0 1 0 0 1 0 1 0 0 0 1 1 0 0 1 1

1 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 0 0 1

1 1 0 0 1 0 0 1 0 1 1 0 0 1 0 1 1 1 1

1 1 0 0 1 0 0 1 0 1 1 1 0 0 1 1 1 0 1

1 1 0 0 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1

1 1 0 0 1 0 0 1 1 0 0 1 0 0 1 1 1 1 1

1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 1 1

1 1 0 0 1 0 0 1 1 0 1 0 0 0 1 0 1 1 1

1 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1 0 1

1 1 0 0 1 0 0 1 1 1 0 0 0 0 1 1 1 0 1

1 1 0 0 1 0 0 1 1 1 0 0 0 1 1 1 0 1 1

1 1 0 0 1 0 0 1 1 1 0 0 1 1 1 1 1 0 1

1 1 0 0 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1

1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 1

1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1 1

1 1 0 0 1 0 1 0 0 0 1 0 1 0 0 0 0 0 1

1 1 0 0 1 0 1 0 0 0 1 1 0 0 1 0 1 0 1

1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 1 0 0 1

1 1 0 0 1 0 1 0 0 1 1 1 1 0 1 0 1 0 1

1 1 0 0 1 0 1 0 0 1 0 1 0 0 1 0 0 0 1

1 1 0 0 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1

1 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0 0 1 1

1 1 0 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1

1 1 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 1 1

1 1 0 0 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1

1 1 0 0 1 0 1 1 0 0 0 0 0 1 0 1 0 1 1

1 1 0 0 1 0 1 1 0 0 0 0 0 1 1 1 0 0 1

1 1 0 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1 1

1 1 0 0 1 0 1 1 0 0 0 1 1 1 0 1 0 1 1

1 1 0 0 1 0 1 1 0 1 0 0 0 0 0 0 1 1 1

1 1 0 0 1 0 1 1 0 1 0 1 0 0 1 0 1 1 1

1 1 0 0 1 0 1 1 1 0 0 0 1 0 1 0 0 0 1

1 1 0 0 1 0 1 1 1 1 1 0 0 1 0 1 1 1 1

1 1 0 0 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1

1 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1 1

1 1 0 0 1 1 0 0 0 0 0 1 0 0 1 1 0 1 1

1 1 0 0 1 1 0 0 0 0 0 1 0 1 0 0 0 0 1

1 1 0 0 1 1 0 0 0 0 0 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 0 0 1 0 0 0 1 1 0 1
1 1 0 0 1 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 0 0 1 1 0 0 1
1 1 0 0 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 1
1 1 0 0 1 1 0 0 0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1 1 0 0 1 0 1 0 0 0 1 0 0 1
1 1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 1 1 1
1 1 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 1 1 0 0 1 1 1 0 0 1 0 1 0 1 0 0 1 1 1
1 1 0 0 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1
1 1 0 0 1 1 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 0 0 0 0 1
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1
1 1 0 0 1 1 0 0 0 1 1 1 1 1 0 1 0 0 1 1 1 0 0 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1
1 1 0 0 1 1 0 0 1 0 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0 0 0 0 0 1 1
1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0 1 1 0 1 1
1 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 0 1 1 1 0 0 1
1 1 0 0 1 1 0 0 1 1 0 1 1 0 0 0 1 1 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1 0 1 1 1
1 1 0 0 1 1 0 0 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 0 1 0 1 1
1 1 0 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1 1 1
1 1 0 0 1 1 0 1 0 0 0 1 0 0 1 1 1 1 1 1 0 0 1 1 1 0 0 0 1 1 0 1 0 0 1 1
1 1 0 0 1 1 0 1 0 0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1 1
1 1 0 0 1 1 0 1 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 0 1 1
1 1 0 0 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 0 0 1 0 0 0 1 1 1
1 1 0 0 1 1 0 1 0 1 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 1
1 1 0 0 1 1 0 1 0 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 0 0 0 1 0 1 0 1 1
1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1 1 0 1
1 1 0 0 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 0 0 1 1 1 1 0 1 0 1 0 1 0 0 0 1
1 1 0 0 1 1 0 1 1 0 0 0 0 1 1 0 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1
1 1 0 0 1 1 0 1 1 0 1 1 0 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1
1 1 0 0 1 1 0 1 1 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1
1 1 0 0 1 1 0 1 1 1 0 0 1 0 0 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
1 1 0 0 1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1

1 1 0 1 0 0 0 0 0 0 1 0 0 1 0 1 1 1 1
1 1 0 1 0 0 0 0 0 0 1 1 0 1 0 0 1 1 1
1 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 1 1
1 1 0 1 0 0 0 0 0 1 0 1 1 0 1 0 1 0 1
1 1 0 1 0 0 0 0 0 1 1 1 0 1 1 1 1 0 1
1 1 0 1 0 0 0 0 0 1 1 1 1 1 1 1 0 0 1
1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 1 0 1
1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1
1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1 1
1 1 0 1 0 0 0 0 1 1 1 1 0 0 0 1 0 0 1
1 1 0 1 0 0 0 0 1 1 1 1 0 1 1 0 1 0 1
1 1 0 1 0 0 0 1 0 0 1 0 1 1 1 0 0 0 1
1 1 0 1 0 0 0 1 0 0 1 1 1 1 1 0 0 1 1
1 1 0 1 0 0 0 1 0 1 0 1 0 0 0 1 0 1 1
1 1 0 1 0 0 0 1 0 1 1 1 0 0 1 1 1 0 1
1 1 0 1 0 0 0 1 1 0 0 0 0 1 0 1 1 0 1
1 1 0 1 0 0 0 1 1 0 0 0 0 1 1 0 1 0 1
1 1 0 1 0 0 0 1 1 0 1 1 0 0 0 1 0 0 1
1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 0 1 1
1 1 0 1 0 0 0 1 1 1 0 0 0 0 1 0 1 0 1
1 1 0 1 0 0 0 1 1 1 0 0 1 0 1 1 0 1 1
1 1 0 1 0 0 0 1 1 1 1 0 0 0 1 1 1 0 1
1 1 0 1 0 0 0 1 1 1 1 0 1 0 0 1 1 0 1
1 1 0 1 0 0 0 1 1 1 1 0 1 1 1 0 0 0 1
1 1 0 1 0 0 1 0 0 0 0 0 0 1 0 1 0 0 1
1 1 0 1 0 0 1 0 0 0 0 0 0 1 1 1 0 1 1
1 1 0 1 0 0 1 0 0 0 1 1 0 1 1 0 0 0 1
1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 1
1 1 0 1 0 0 1 0 0 1 1 0 1 0 0 1 0 0 1
1 1 0 1 0 0 1 0 0 1 1 1 0 0 0 0 0 1 1
1 1 0 1 0 0 1 0 0 1 1 1 0 1 0 1 1 0 1
1 1 0 1 0 0 1 0 1 0 0 0 0 1 0 1 0 1 1
1 1 0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 1 1
1 1 0 1 0 0 1 0 1 0 0 1 0 1 1 0 1 1 1
1 1 0 1 0 0 1 0 1 0 1 0 1 0 1 0 0 0 1
1 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 0 0 1
1 1 0 1 0 0 1 0 1 1 0 1 1 1 1 0 1 1 1
1 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 1 1 1
1 1 0 1 0 0 1 0 1 1 1 1 0 1 0 0 1 0 1
1 1 0 1 0 0 1 0 1 1 1 1 1 0 0 0 0 1 1
1 1 0 1 0 0 1 0 1 1 1 1 1 1 0 0 0 0 1
1 1 0 1 0 0 1 1 0 0 0 0 1 0 0 1 0 1 1
1 1 0 1 0 0 1 1 0 0 0 1 0 0 1 0 1 0 1
1 1 0 1 0 0 1 1 0 0 0 1 1 0 1 0 0 0 1
1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 0 1 0 1
1 1 0 1 0 0 1 1 0 0 1 0 0 1 1 0 0 0 1
1 1 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1 1
1 1 0 1 0 0 1 1 0 0 1 0 1 1 1 0 0 1 1
1 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1 0 1
1 1 0 1 0 0 1 1 0 1 0 0 1 1 0 1 1 0 1
1 1 0 1 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1
1 1 0 1 0 0 1 1 0 1 1 0 0 0 1 1 0 1 1
1 1 0 1 0 0 1 1 0 1 1 0 0 1 1 1 1 1 1
1 1 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1
1 1 0 1 0 0 1 1 1 0 0 1 1 0 0 1 1 0 1
1 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 0 0 1
1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 0 0 1 1
1 1 0 1 0 0 1 1 1 1 0 0 0 0 0 1 0 0 1

1 1 0 1 0 0 1 1 1 1 0 0 0 0 1 0 0 0 1
1 1 0 1 0 0 1 1 1 1 0 0 1 0 1 0 0 1 1
1 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1
1 1 0 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1
1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 0 1 1
1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0 1 1
1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 1 0 1
1 1 0 1 0 1 0 0 0 0 1 0 1 0 1 1 0 0 1
1 1 0 1 0 1 0 0 0 0 1 1 0 0 0 1 1 0 1
1 1 0 1 0 1 0 0 0 0 1 1 0 1 0 0 1 0 1
1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 1 1 1 1
1 1 0 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0 1
1 1 0 1 0 1 0 0 1 0 0 0 1 0 0 1 1 0 1
1 1 0 1 0 1 0 0 1 0 1 0 0 0 1 1 0 0 1
1 1 0 1 0 1 0 0 1 0 1 1 0 1 0 0 1 1 1
1 1 0 1 0 1 0 0 1 1 0 0 0 0 1 0 1 0 1
1 1 0 1 0 1 0 0 1 1 0 0 1 1 1 1 1 1 1
1 1 0 1 0 1 0 1 0 0 0 0 1 0 1 0 0 1 1
1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 0 1 1 1
1 1 0 1 0 1 0 1 1 0 1 0 0 0 1 1 1 0 1
1 1 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0 1
1 1 0 1 0 1 0 1 1 1 0 1 1 1 0 0 1 1
1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 0 0 1
1 1 0 1 0 1 1 0 0 0 0 1 1 1 1 1 1 1 1
1 1 0 1 0 1 1 0 0 0 1 1 1 0 0 0 0 0 1
1 1 0 1 0 1 1 0 0 1 0 0 0 1 1 0 0 0 1
1 1 0 1 0 1 1 0 0 1 0 0 1 0 0 0 1 0 1
1 1 0 1 0 1 1 0 0 1 0 0 1 1 0 0 1 1 1
1 1 0 1 0 1 1 0 0 1 0 1 1 0 1 1 1 1 1
1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 1 0 0 1
1 1 0 1 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1
1 1 0 1 0 1 1 0 1 0 0 1 0 0 0 0 1 0 1
1 1 0 1 0 1 1 0 1 0 1 1 0 1 1 1 0 0 1
1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 0 1 1
1 1 0 1 0 1 1 0 1 1 0 1 0 0 0 1 0 1 1
1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 1 1
1 1 0 1 0 1 1 0 1 1 1 0 0 1 0 0 0 1 1
1 1 0 1 0 1 1 0 1 1 1 0 1 1 0 1 1 1 1
1 1 0 1 0 1 1 0 1 1 1 0 1 1 1 0 0 1 1
1 1 0 1 0 1 1 0 1 1 1 1 0 0 1 0 0 0 1
1 1 0 1 0 1 1 0 1 1 1 1 0 0 1 1 0 0 1
1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1
1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1
1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 0 1
1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1
1 1 0 1 1 0 0 0 0 0 0 0 1 1 1 1 0 0 1
1 1 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 1
1 1 0 1 1 0 0 0 0 1 0 0 0 1 0 0 1 1 1
1 1 0 1 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1
1 1 0 1 1 0 0 0 0 1 0 1 1 1 0 1 1 0 1
1 1 0 1 1 0 0 0 0 1 0 1 1 1 1 1 0 0 1

1 1 0 1 1 0 0 0 0 1 1 0 0 1 0 0 0 1 1

1 1 0 1 1 0 0 0 0 1 1 0 1 1 0 1 0 1 1

1 1 0 1 1 0 0 0 0 1 1 1 0 0 1 1 1 0 1

1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 1 1

1 1 0 1 1 0 0 0 1 0 0 0 0 1 0 1 1 0 1

1 1 0 1 1 0 0 0 1 0 0 1 0 1 0 1 0 0 1

1 1 0 1 1 0 0 0 1 1 0 1 1 0 0 1 1 0 1

1 1 0 1 1 0 0 0 1 1 0 1 1 1 1 0 1 1 1

1 1 0 1 1 0 0 0 1 1 1 1 0 0 0 0 0 0 1

1 1 0 1 1 0 0 0 1 1 1 1 0 1 1 1 0 1 1

1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 0 0 1 1

1 1 0 1 1 0 0 1 0 0 0 0 0 0 0 0 1 0 1

1 1 0 1 1 0 0 1 0 0 1 1 0 0 1 0 0 0 1

1 1 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 0 1

1 1 0 1 1 0 0 1 0 1 1 0 0 0 0 0 0 1 1

1 1 0 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 1

1 1 0 1 1 0 0 1 1 0 0 0 0 0 0 0 1 1 1

1 1 0 1 1 0 0 1 1 0 0 0 0 1 1 1 0 1 1

1 1 0 1 1 0 0 1 1 0 1 0 0 0 0 0 1 1 1

1 1 0 1 1 0 0 1 1 1 0 1 0 1 1 0 0 0 1

1 1 0 1 1 0 0 1 1 1 1 0 0 1 0 0 0 1 1

1 1 0 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1

1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1

1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1

1 1 0 1 1 0 1 0 0 0 1 0 1 1 0 1 0 1 1

1 1 0 1 1 0 1 0 0 0 1 1 0 1 0 0 0 0 1

1 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1 0 1 1

1 1 0 1 1 0 1 0 0 1 1 0 0 0 1 1 0 1 1

1 1 0 1 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1

1 1 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1 0 1

1 1 0 1 1 0 1 0 1 1 0 1 0 1 1 0 1 0 1

1 1 0 1 1 0 1 0 1 1 0 1 0 0 0 1 0 1 1

1 1 0 1 1 0 1 0 1 1 0 1 1 1 1 0 0 0 1

1 1 0 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1

1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 1

1 1 0 1 1 0 1 1 1 0 0 0 1 0 1 0 1 0 1

1 1 0 1 1 0 1 1 1 0 0 0 1 0 0 0 1 1

1 1 0 1 1 0 1 1 1 0 0 1 0 1 1 0 1 1

1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 0 1

1 1 0 1 1 0 1 1 1 1 1 0 0 1 0 1 1 0 1

1 1 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1

1 1 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 1

1 1 0 1 1 1 0 0 0 0 0 1 0 0 1 0 0 1 1

1 1 0 1 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1

1 1 0 1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 1

1 1 0 1 1 1 0 0 0 0 1 0 0 0 1 1 0 0 1

1 1 0 1 1 1 0 0 0 0 1 1 0 0 0 1 0 1 1

1 1 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 1

1 1 0 1 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1

1 1 0 1 1 1 0 0 0 1 1 0 0 0 0 1 1 1 1

1 1 0 1 1 1 0 0 0 1 1 0 0 1 0 0 0 0 1

1 1 0 1 1 1 0 0 0 1 1 0 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 1 1 0 0 1
1 1 0 1 1 1 0 0 1 0 0 0 0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 1 0 0 0 0 0 1 1 1
1 1 0 1 1 1 0 0 1 0 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 1 1 1 0 0 1 1 0 1 1
1 1 0 1 1 1 0 0 1 0 1 0 0 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1
1 1 0 1 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 0 1 0 1 1
1 1 0 1 1 1 0 0 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 1 0 1 0 1
1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 1 0 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 0 0 1
1 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1 0 1 1 1 1 0 1 1 1 0 1 1 0 1 1 0 1 0 0 1 1
1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1
1 1 0 1 1 1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 1
1 1 0 1 1 1 0 1 0 0 1 0 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 1 1
1 1 0 1 1 1 0 1 0 0 1 0 1 1 1 0 1 1 1 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1
1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 0 0 0 1 1 1 0 1 1 1 1 0 1 0 0 0 0 1 1 0 0 1
1 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 0 0 1 1 0 0 1 1 1
1 1 0 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 0 0 1 0 1 1 1
1 1 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 1 1 0 1 1 0 0 0 1 1
1 1 0 1 1 1 0 1 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 0 0 1 0 1
1 1 0 1 1 1 0 1 1 0 1 0 0 0 1 0 0 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 1 1
1 1 0 1 1 1 0 1 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1 0 1 0 0 0 0 1
1 1 0 1 1 1 0 1 1 0 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1
1 1 0 1 1 1 0 1 1 1 0 0 0 0 1 0 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1
1 1 0 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1
1 1 0 1 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1
1 1 0 1 1 1 0 1 1 1 0 0 0 1 0 0 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1 0 1
1 1 0 1 1 1 0 1 1 1 0 0 1 0 1 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1
1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 0 1
1 1 0 1 1 1 0 1 1 1 1 0 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 1
1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1
1 1 0 1 1 1 0 1 1 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1
1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 0 0 1
1 1 0 1 1 1 0 1 1 1 1 0 0 0 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1
1 1 0 1 1 1 0 1 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1

1 1 1 0 0 0 0 0 1 0 1 0 1 1 1 1 1 0 1
1 1 1 0 0 0 0 0 1 1 0 1 0 1 0 1 1 1 1
1 1 1 0 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1
1 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 1
1 1 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 1 1
1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 0 1
1 1 1 0 0 0 0 1 0 0 0 1 0 1 1 1 1 1 1
1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 0 1 1
1 1 1 0 0 0 0 1 0 0 0 1 1 1 1 1 1 0 1
1 1 1 0 0 0 0 1 0 0 1 1 0 0 0 0 1 1 1
1 1 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1 0 1
1 1 1 0 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1
1 1 1 0 0 0 0 1 1 0 0 1 0 1 1 1 1 0 1
1 1 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1
1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1
1 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1
1 1 1 0 0 0 0 1 1 1 0 1 0 0 1 1 1 0 1
1 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 0 0 1
1 1 1 0 0 0 1 0 0 0 1 1 0 1 1 1 1 0 1
1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 1 1 0 1
1 1 1 0 0 0 1 0 0 0 1 1 0 1 0 1 0 0 1
1 1 1 0 0 0 1 0 0 1 0 1 1 1 0 1 1 0 1
1 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 1 0 1
1 1 1 0 0 0 1 0 0 1 0 0 1 1 1 1 0 0 1
1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1
1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 1 1
1 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 0 1
1 1 1 0 0 0 1 1 0 0 0 0 1 1 1 1 1 1 1
1 1 1 0 0 0 1 1 0 0 1 0 1 1 1 1 0 1 1
1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 1
1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1
1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 0 0 1 1
1 1 1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 0 1
1 1 1 0 0 0 1 1 1 0 1 1 1 1 0 0 0 1 1
1 1 1 0 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1
1 1 1 0 0 0 1 1 1 1 0 0 0 1 1 0 1 1 1
1 1 1 0 0 0 1 1 1 1 0 0 1 0 1 1 0 0 1
1 1 1 0 0 0 1 1 1 1 0 1 0 1 1 1 0 1 1
1 1 1 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1
1 1 1 0 0 0 1 1 1 1 1 0 0 1 0 1 1 0 1
1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 1 0 0 0 0 1 0 0 1 0 0 1 1 1
1 1 1 0 0 1 0 0 0 1 0 0 0 0 0 1 1 1 1
1 1 1 0 0 1 0 0 0 1 0 0 1 1 0 0 1 0 1
1 1 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1
1 1 1 0 0 1 0 0 0 1 1 0 1 1 1 0 0 1 1
1 1 1 0 0 1 0 0 0 1 1 1 1 1 1 0 0 1 0 1
1 1 1 0 0 1 0 0 1 0 0 0 1 1 0 1 1 1 1
1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 0 0 0 1
1 1 1 0 0 1 0 1 0 0 0 0 0 1 0 1 1 0 1
1 1 1 0 0 1 0 1 0 0 0 1 0 0 0 0 1 1 1
1 1 1 0 0 1 0 1 0 0 0 1 1 1 0 1 1 0 1
1 1 1 0 0 1 0 1 0 0 1 0 0 1 1 0 1 1 1
1 1 1 0 0 1 0 1 0 0 1 0 1 1 1 0 0 1 1
1 1 1 0 0 1 0 1 0 0 1 1 1 1 1 0 0 1 0 1
1 1 1 0 0 1 0 1 0 0 1 1 1 1 1 1 1 0 1

1 1 1 0 0 1 0 1 0 1 0 0 1 1 1 1 1 1 1
1 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 0 0 1
1 1 1 0 0 1 0 1 0 1 1 0 0 1 0 0 0 0 1
1 1 1 0 0 1 0 1 0 1 1 1 0 0 1 1 1 1 1
1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1
1 1 1 0 0 1 0 1 1 0 0 1 0 0 1 0 1 1 1
1 1 1 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 1
1 1 1 0 0 1 0 1 1 1 0 0 0 0 0 0 0 1 1
1 1 1 0 0 1 0 1 1 1 0 1 0 1 1 0 1 1 1
1 1 1 0 0 1 0 1 1 1 0 1 1 0 1 1 0 1 1
1 1 1 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1 0 0 1 0 1 1 1 1 0 1 1 0 0 1 1 1
1 1 1 0 0 1 0 1 1 1 1 0 1 1 0 1 0 1 1
1 1 1 0 0 1 0 1 1 1 1 1 0 1 0 1 1 0 1
1 1 1 0 0 1 1 0 0 0 0 1 0 1 0 1 0 0 1
1 1 1 0 0 1 1 0 0 0 0 1 0 1 1 0 0 0 1
1 1 1 0 0 1 1 0 0 0 0 1 1 1 1 0 0 1 1
1 1 1 0 0 1 1 0 0 0 1 1 0 1 0 1 0 1 1
1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1
1 1 1 0 0 1 1 0 0 1 0 0 0 0 1 0 0 1 1
1 1 1 0 0 1 1 0 0 1 0 0 1 1 0 0 0 0 1
1 1 1 0 0 1 1 0 0 1 0 0 1 0 1 1 1 1 1
1 1 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 1 1
1 1 1 0 0 1 1 0 0 1 1 0 0 1 0 1 1 0 1
1 1 1 0 0 1 1 0 1 0 0 0 0 1 1 1 0 1 1
1 1 1 0 0 1 1 0 1 0 1 0 1 1 0 0 1 0 1
1 1 1 0 0 1 1 0 1 1 0 1 0 1 1 1 0 1 1
1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1

1 1 1 0 0 1 1 0 1 1 1 0 1 1 0 1 0 1 1
1 1 1 0 0 1 1 0 1 1 1 1 0 0 0 0 0 1 1
1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 1
1 1 1 0 0 1 1 1 0 0 0 0 1 1 1 0 0 1 1
1 1 1 0 0 1 1 1 0 0 1 0 0 0 0 1 0 0 1
1 1 1 0 0 1 1 1 0 0 1 1 0 1 0 1 1 1 1
1 1 1 0 0 1 1 1 0 1 0 0 0 0 1 0 0 0 1
1 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1
1 1 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 0 1
1 1 1 0 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1
1 1 1 0 0 1 1 1 0 0 0 1 1 0 0 0 1 1
1 1 1 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1
1 1 1 0 0 1 1 1 1 0 0 1 1 1 0 0 0 0 1
1 1 1 0 0 1 1 1 1 0 1 0 0 0 1 0 1 0 1
1 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1
1 1 1 0 0 1 1 1 1 0 0 0 1 0 0 0 1 1
1 1 1 0 0 1 1 1 1 0 1 1 0 1 1 1 1 1
1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 1 1 0 1
1 1 1 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 0 1
1 1 1 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1
1 1 1 0 1 0 0 0 0 0 0 0 1 1 1 0 1 0 0 1
1 1 1 0 1 0 0 0 0 1 0 1 0 0 0 1 1 0 1
1 1 1 0 1 0 0 0 0 1 0 1 0 0 1 1 1 1 1
1 1 1 0 1 0 0 0 0 1 0 1 0 1 0 1 1 1 1
1 1 1 0 1 0 0 0 0 1 1 0 1 0 0 0 1 0 1
1 1 1 0 1 0 0 0 1 0 0 0 0 0 1 0 1 1 1
1 1 1 0 1 0 0 0 1 0 0 0 1 0 1 1 0 0 1
1 1 1 0 1 0 0 0 1 0 0 1 1 0 1 1 1 0 1
1 1 1 0 1 0 0 0 1 0 1 0 1 0 1 1 0 1 1

1 1 1 0 1 0 0 0 1 0 1 0 1 1 0 0 0 0 1
1 1 1 0 1 0 0 0 1 1 1 0 1 1 1 0 1 1 1
1 1 1 0 1 0 0 1 0 0 0 1 0 1 1 0 0 0 1
1 1 1 0 1 0 0 1 0 0 1 1 0 1 1 0 0 1 1
1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1 0 0 1
1 1 1 0 1 0 0 1 0 1 0 0 1 1 1 0 0 1 1
1 1 1 0 1 0 0 1 0 1 0 1 1 1 1 1 0 1 1
1 1 1 0 1 0 0 1 1 0 1 0 1 1 1 1 1 0 1
1 1 1 0 1 0 0 1 1 0 1 1 0 1 0 0 0 1 1
1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 0 0 1 1
1 1 1 0 1 0 0 1 1 1 1 1 0 0 1 1 0 1 1
1 1 1 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 1
1 1 1 0 1 0 1 0 0 0 0 1 1 0 0 0 1 0 1
1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1
1 1 1 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1
1 1 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0 1 1
1 1 1 0 1 0 1 0 1 0 0 1 1 1 1 0 1 1 1
1 1 1 0 1 0 1 0 1 0 1 0 0 0 0 1 0 0 1
1 1 1 0 1 0 1 0 1 0 1 0 0 0 1 1 1 0 1
1 1 1 0 1 0 1 0 1 0 1 0 0 1 1 0 0 1 1
1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 0 0 1 1
1 1 1 0 1 0 1 1 0 1 1 1 0 1 0 1 1 0 1
1 1 1 0 1 0 1 1 0 0 0 0 1 0 1 1 1 0 1
1 1 1 0 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1
1 1 1 0 1 0 1 1 0 1 0 1 1 1 0 0 0 0 1
1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1 1
1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 0 0 1 1
1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1 0 0 1
1 1 1 0 1 0 1 1 1 1 0 0 0 1 0 1 1 1 1

1 1 1 0 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1
1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1 1
1 1 1 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1
1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1
1 1 1 0 1 0 1 1 1 1 1 1 1 0 0 0 1 0 1
1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 0 0 1
1 1 1 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 1
1 1 1 0 1 1 0 0 0 0 0 1 0 0 0 1 0 1 1
1 1 1 0 1 1 0 0 0 0 0 1 0 1 0 1 1 1 1
1 1 1 0 1 1 0 0 0 0 1 0 1 0 1 0 1 1 1
1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 0 0 1 1
1 1 1 0 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1
1 1 1 0 1 1 0 0 0 1 1 0 1 1 0 1 0 0 1
1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 0 1 1 1
1 1 1 0 1 1 0 0 1 1 0 1 0 1 0 0 1 0 1
1 1 1 0 1 1 0 0 1 1 0 1 1 1 0 1 1 0 1
1 1 1 0 1 1 0 0 1 1 1 0 1 0 1 0 1 1 1
1 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 1 1 1
1 1 1 0 1 1 0 1 0 0 0 0 0 1 0 0 0 1 1
1 1 1 0 1 1 0 1 0 0 0 0 0 1 0 1 0 0 1
1 1 1 0 1 1 0 1 0 0 0 1 0 1 0 0 1 1 1
1 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1 1
1 1 1 0 1 1 0 1 0 1 1 1 0 1 1 0 0 1 1
1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1
1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 0 0 1 1
1 1 1 0 1 1 0 1 1 1 0 0 0 0 0 1 0 1 1
1 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 1 1
1 1 1 0 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1
1 1 1 0 1 1 1 0 0 0 1 1 0 1 0 1 0 1

1 1 1 0 1 1 1 0 0 0 0 1 1 1 0 0 0 1 1

1 1 1 0 1 1 1 0 0 1 0 1 1 1 0 1 0 1 1

1 1 1 0 1 1 1 0 0 1 1 0 0 1 0 1 0 0 1

1 1 1 0 1 1 1 0 0 1 1 1 0 1 0 0 0 0 1

1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 1 1

1 1 1 0 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1

1 1 1 0 1 1 1 0 1 0 1 1 1 0 0 0 0 0 1

1 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 0 1

1 1 1 0 1 1 1 0 1 1 0 0 0 0 1 0 1 0 1

1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 0 1

1 1 1 0 1 1 1 0 1 1 1 0 0 0 1 0 1 1 1

1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 0 0 0 1

1 1 1 0 1 1 1 1 0 0 0 1 0 1 1 1 0 1 1

1 1 1 0 1 1 1 1 0 0 1 1 0 1 0 1 1 0 1

1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 0 0 1

1 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 0 1

1 1 1 0 1 1 1 1 0 1 0 1 0 0 0 0 1 0 1

1 1 1 0 1 1 1 1 0 1 0 1 1 1 0 1 1 1 1

1 1 1 0 1 1 1 1 0 1 1 0 1 1 1 0 0 0 1

1 1 1 0 1 1 1 1 1 0 0 0 1 0 0 1 0 0 1

1 1 1 0 1 1 1 1 1 0 0 1 0 0 1 0 1 1 1

1 1 1 0 1 1 1 1 1 0 1 0 1 0 0 0 1 1 1

1 1 1 0 1 1 1 1 1 1 0 0 0 0 1 0 0 0 1

1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 1 1

1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1 0 1

1 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 1

1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1

1 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1 0 1

1 1 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 1

1 1 1 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 1

1 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0 1 1

1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 1 0 1

1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 1

1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 0 1 0

1 1 1 1 0 0 0 0 1 1 1 0 0 0 0 1 1 1 1

1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1

1 1 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 1 0

1 1 1 1 0 0 0 1 0 0 0 0 1 1 1 1 1 0 1

1 1 1 1 0 0 0 1 0 1 0 0 0 1 1 1 0 1 1

1 1 1 1 0 0 0 1 0 1 0 1 0 0 0 0 1 0 1

1 1 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1

1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 0 1 1 1

1 1 1 1 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1

1 1 1 1 0 0 0 1 1 1 0 0 0 0 0 0 1 0 1

1 1 1 1 0 0 0 1 1 1 0 0 0 1 0 1 1 0 1

1 1 1 1 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1

1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1 1 1

1 1 1 1 0 0 0 1 1 1 0 1 0 1 0 0 1 0 1

1 1 1 1 0 0 0 1 1 1 1 1 0 1 0 0 1 1 1

1 1 1 1 0 0 1 0 0 0 0 0 1 0 1 0 1 0 1

1 1 1 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 1

1 1 1 1 0 0 1 0 0 0 0 1 0 0 0 1 1 0 1

1 1 1 1 0 0 1 0 0 0 0 1 0 1 1 0 1 1 1

1 1 1 1 0 0 1 0 0 0 0 1 1 1 0 1 0 1 1

1 1 1 1 0 0 1 0 0 1 0 0 0 1 0 0 0 1 1 1 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 1 1
1 1 1 1 0 0 1 0 0 1 1 0 1 0 0 1 0 1 1 1 1 1 0 1 0 0 0 1 0 0 1 1 0 1 1 0 1
1 1 1 1 0 0 1 0 1 0 0 0 1 0 1 1 1 0 1 1 1 1 0 1 0 0 0 1 1 1 1 0 1 1 0 1 1
1 1 1 1 0 0 1 0 1 0 0 0 1 1 0 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 1 0 0 1
1 1 1 1 0 0 1 0 1 1 0 0 0 1 0 1 1 0 1 1 1 1 0 1 0 0 1 0 0 1 0 0 1 1 0 1 1
1 1 1 1 0 0 1 0 1 1 0 0 1 1 0 1 0 0 1 1 1 1 0 1 0 0 1 0 0 1 0 1 0 1 1 0 1
1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 1 1 1 0 0 0 1
1 1 1 1 0 0 1 0 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1 1 0 1
1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 1
1 1 1 1 0 0 1 1 0 0 0 0 1 0 0 1 0 0 1 1 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1 1 1
1 1 1 1 0 0 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 1 1 1 1 1
1 1 1 1 0 0 1 1 0 1 0 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1
1 1 1 1 0 0 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 1
1 1 1 1 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 0 1 0 1 1 0 0 1 0 1 1 1 1 0 1
1 1 1 1 0 0 1 1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 0 1 1 0 1
1 1 1 1 0 0 1 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 1 0 1 1 0 0 0 0 1
1 1 1 1 0 0 1 1 1 0 1 1 1 0 1 0 1 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 1
1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 0 1 0 1
1 1 1 1 0 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 1 1 1 1
1 1 1 1 0 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 0 1 1
1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 0 1 1 1 0 1 1 1
1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 0 1 0 1 1 1 1 0 1 0 1
1 1 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 0 0 1
1 1 1 1 0 1 0 0 0 0 0 1 0 0 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1
1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 0 1 1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 1 1 0 0 1
1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 1 1 1 0 1 1 0 0 0 0 1 0 0 0 0 0 1 1
1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 1 0 1 1 1 1 1 1 0 1 1 0 0 0 0 1 0 1 0 0 0 0 1
1 1 1 1 0 1 0 0 0 0 1 0 0 1 0 1 0 0 1 1 1 1 1 0 1 1 0 0 0 1 0 0 1 0 0 1 1 1
1 1 1 1 0 1 0 0 0 0 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 1 1 1 1 0 1 1

1 1 1 1 0 1 1 0 0 1 0 1 1 1 1 0 0 1 1

1 1 1 1 0 1 1 0 0 1 1 1 1 0 1 0 0 1 1

1 1 1 1 0 1 1 0 1 0 0 0 1 1 0 1 0 0 1

1 1 1 1 0 1 1 0 1 0 1 0 0 1 0 0 0 1 1

1 1 1 1 0 1 1 0 1 0 1 0 1 0 1 1 1 0 1

1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 0 0 1

1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 0 1 1

1 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1

1 1 1 1 0 1 1 0 1 1 0 0 1 1 0 0 1 1 1

1 1 1 1 0 1 1 0 1 1 1 0 0 1 0 0 1 1 1

1 1 1 1 0 1 1 0 1 1 1 0 0 1 0 1 0 1 1

1 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 0 1 1

1 1 1 1 0 1 1 1 0 0 1 1 0 0 1 1 1 0 1

1 1 1 1 0 1 1 1 1 0 0 0 0 0 0 0 1 1 1

1 1 1 1 0 1 1 1 1 0 0 1 1 0 1 1 1 1 1

1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 0 1

1 1 1 1 0 1 1 1 1 0 1 0 0 1 1 1 0 0 1

1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 0 1

1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 0 0 0 1

1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 0 1 1

1 1 1 1 0 1 1 1 1 1 0 0 0 0 0 1 0 0 1

1 1 1 1 0 1 1 1 1 1 0 0 1 1 0 0 0 1 1

1 1 1 1 0 1 1 1 1 1 0 1 1 0 0 0 1 0 1

1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 0 0 1

1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1

1 1 1 1 1 0 0 0 0 0 0 1 1 1 1 1 1 1 1

1 1 1 1 1 0 0 0 0 0 1 1 0 0 0 1 0 0 1

1 1 1 1 1 0 0 0 0 1 0 0 0 1 0 0 0 1 1

1 1 1 1 1 0 0 0 1 0 0 1 0 1 0 1 1 0 1

1 1 1 1 1 0 0 0 1 0 1 1 0 1 0 1 1 1 1

1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 0 1 0 1

1 1 1 1 1 0 0 0 1 1 0 1 1 0 1 1 0 1 1

1 1 1 1 1 0 0 0 1 1 1 0 0 1 0 0 0 1 1

1 1 1 1 1 0 0 0 1 1 1 1 0 1 0 0 1 1 1

1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 0 1 1

1 1 1 1 1 0 0 1 0 0 0 0 1 0 0 1 1 1 1

1 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 1

1 1 1 1 1 0 0 1 0 0 0 1 1 0 0 1 1 0 1

1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 1

1 1 1 1 1 0 0 1 0 0 1 0 0 1 0 1 1 0 1

1 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1

1 1 1 1 1 0 0 1 0 0 1 1 1 0 0 0 0 1 1

1 1 1 1 1 0 0 1 0 1 0 1 0 0 1 0 0 1 1

1 1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 0 0 1

1 1 1 1 1 0 0 1 1 0 0 0 0 1 1 1 1 1 1

1 1 1 1 1 0 0 1 1 0 0 1 0 1 0 1 1 1 1

1 1 1 1 1 0 0 1 1 0 0 1 1 1 0 1 0 1 1

1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1 0 0 1

1 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 1 1

1 1 1 1 1 0 0 1 1 1 1 1 0 0 0 0 0 0 1

1 1 1 1 1 0 0 1 1 1 1 1 0 1 0 0 1 0 1

1 1 1 1 1 0 1 0 0 1 0 0 0 0 0 1 0 0 1

1 1 1 1 1 0 1 0 0 1 0 0 0 1 1 1 0 0 1

1 1 1 1 1 0 1 0 0 1 1 1 0 1 1 1 1 1 1

1 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 1 0 1

1 1 1 1 1 0 1 0 1 0 0 1 1 1 1 0 1 0 1

1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1

1 1 1 1 1 0 1 0 1 0 1 0 1 1 1 1 1 1 1

1 1 1 1 1 0 1 0 1 0 1 1 1 0 0 0 1 1 1
1 1 1 1 1 0 1 0 1 1 0 1 0 0 0 1 0 0 1
1 1 1 1 1 0 1 0 1 1 0 1 0 0 1 1 0 1 1
1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1
1 1 1 1 1 0 1 1 0 0 1 1 1 1 0 1 1 1 1
1 1 1 1 1 0 1 1 0 1 0 0 0 0 0 1 0 1 1
1 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 1 1
1 1 1 1 1 0 1 1 0 1 0 1 1 1 0 1 0 0 1
1 1 1 1 1 0 1 1 0 1 1 0 0 0 0 1 1 1 1
1 1 1 1 1 0 1 1 1 0 0 1 0 1 0 0 0 0 1
1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1 1 1 1
1 1 1 1 1 0 1 1 1 0 1 0 0 0 1 0 1 1 1
1 1 1 1 1 0 1 1 1 0 1 0 1 1 0 0 1 0 1
1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 0 1 1
1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 1 1
1 1 1 1 1 1 0 0 0 0 0 0 1 1 0 0 1 1 1
1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 0 0 1 1
1 1 1 1 1 1 0 0 0 0 0 1 0 0 0 1 0 0 1
1 1 1 1 1 1 0 0 0 0 1 0 0 0 0 1 0 0 1
1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 0 1 1
1 1 1 1 1 1 0 0 0 0 1 1 0 1 1 1 1 0 1
1 1 1 1 1 1 0 0 0 1 0 0 0 1 0 1 1 0 1
1 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 1 1
1 1 1 1 1 1 0 0 0 1 1 0 0 0 0 0 1 1 1
1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 0 1 1 1
1 1 1 1 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1

1 1 1 1 1 1 0 0 1 0 0 1 0 0 0 0 0 0 1
1 1 1 1 1 1 0 0 1 0 1 0 1 0 1 1 1 0 1
1 1 1 1 1 1 0 0 1 0 1 1 0 0 0 1 0 0 1
1 1 1 1 1 1 0 0 1 0 1 1 1 0 1 0 1 0 1
1 1 1 1 1 1 0 0 1 0 1 1 1 1 0 1 0 0 1
1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 1 1 1
1 1 1 1 1 1 0 1 0 0 0 0 0 0 0 1 1 1 1
1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 1 1 1 1
1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1
1 1 1 1 1 1 0 1 0 0 0 1 1 1 1 0 1 0 1
1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1 1
1 1 1 1 1 1 0 1 0 1 0 0 1 0 0 1 0 0 1
1 1 1 1 1 1 0 1 0 1 0 1 0 1 0 0 1 1 1
1 1 1 1 1 1 0 1 0 1 0 1 0 0 1 1 1 0 1
1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1
1 1 1 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1
1 1 1 1 1 1 0 1 1 0 0 0 0 0 1 1 0 0 1
1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 0 0 1
1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 0 1 1
1 1 1 1 1 1 0 1 1 0 0 0 1 0 0 1 1 1 1
1 1 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 1
1 1 1 1 1 1 0 1 1 0 1 1 1 0 1 1 0 1 1
1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 0 0 1 1
1 1 1 1 1 1 1 0 0 0 1 0 0 1 0 0 0 1 1
1 1 1 1 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1
1 1 1 1 1 1 1 0 1 0 0 0 0 1 0 0 0 1 1
1 1 1 1 1 1 1 0 1 0 0 1 0 1 0 0 0 0 1

1 1 1 1 1 1 1 0 1 1 0 0 0 1 1 1 1 1 1

1 1 1 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1

1 1 1 1 1 1 1 0 1 1 0 1 1 0 0 1 0 0 1

1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 1 1

1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 1

1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 1 1 1 1

1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 1

1 1 1 1 1 1 1 1 0 0 1 0 1 1 0 1 0 0 1

1 1 1 1 1 1 1 1 0 0 1 1 0 1 0 0 1 0 1

1 1 1 1 1 1 1 1 0 0 1 1 0 1 1 0 1 1 1

1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1

1 1 1 1 1 1 1 1 0 1 0 1 1 0 0 0 0 1 1

1 1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 0 1

1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 1

1 1 1 1 1 1 1 1 1 0 1 0 0 0 1 0 1 0 1

1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 0 1

1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 1

1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Додаток І. Приклади excel-файлів з експериментальними результати виконання операцій

Результати виконання операцій додавання, множення, ділення та обчислення мультиплікативно оберненого елемента у полі $GF(2^{24})$

m	Basis	Addition	Multiplication	Division	InverseElement
24	Normal	0,002047	1,218876566	10,39418	9,262177487
24	Polynomia	0,001912	0,109251454	3,422463	3,299030767

Результати виконання операції Фробеніуса у полі $GF(2^{24})$ в для різних значень параметра k

m	Basis	1	2	3	4	5	7	9	10	11	12	13
24	Normal	0,002436	0,001679	0,000997	0,000994	0,000992	0,000982	0,000972	0,000991	0,000996	0,001	0,001014
24	Polynomial	0,11445	0,225372	0,339942	0,451851	0,557715	0,777917	0,989613	1,095348	1,205277	1,310586	1,41793

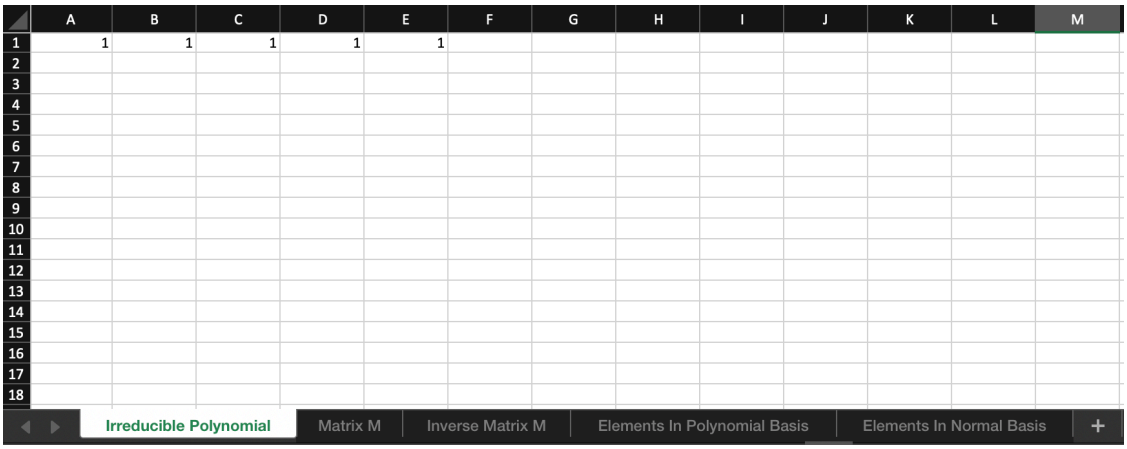
Результати виконання операції піднесення до степеня у полі $GF(2^{24})$ в для різних значень параметра k

m	Basis	1	2	3	4	5	6	8	10	11	12	13
24	Normal	0,001837	0,002277	1,320003	0,003183	1,322859	1,324961	0,004087	1,322455	2,647539	1,32894	2,652649
24	Polynomia	0,001334	0,113229	0,226636	0,225888	0,341341	0,333999	0,336103	0,443939	0,552222	0,444221	0,557409

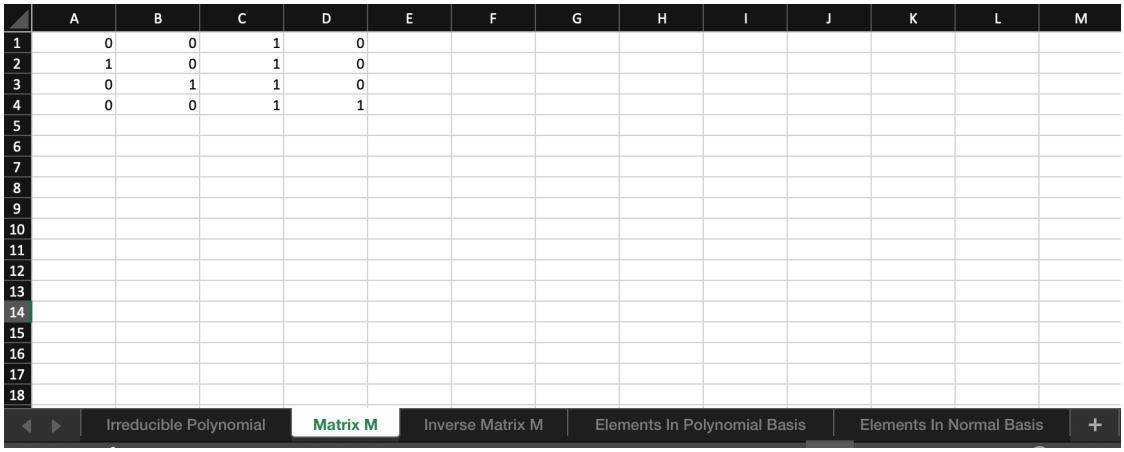
Додаток Ї. Приклад excel-файлу зі згенерованими тестовими даними для поля $GF(2^4)$

Кількість згенерованих елементів: 10000.

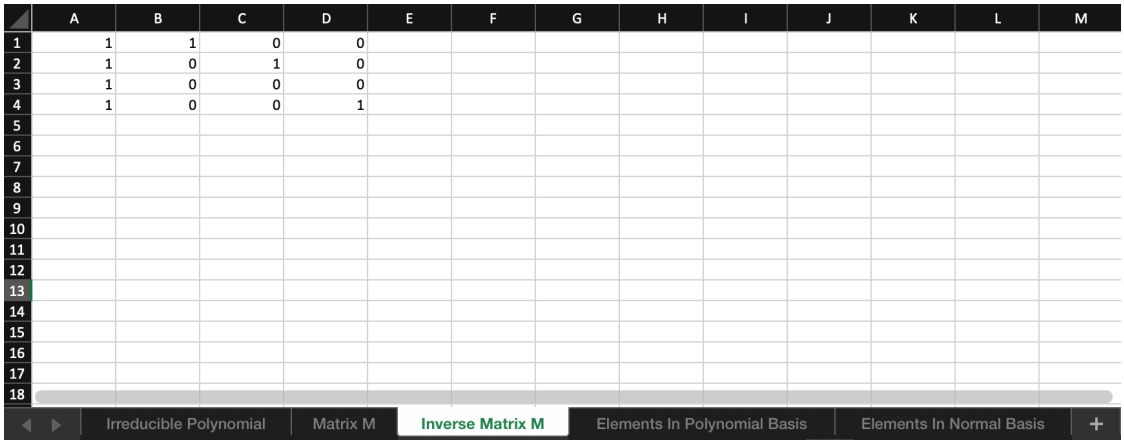
Вкладка з незвідним нормальним поліномом (Irreducible Polynomial)



Вкладка з матрицею для перетворення елементів з поліноміального базису в нормальний (Matrix M)



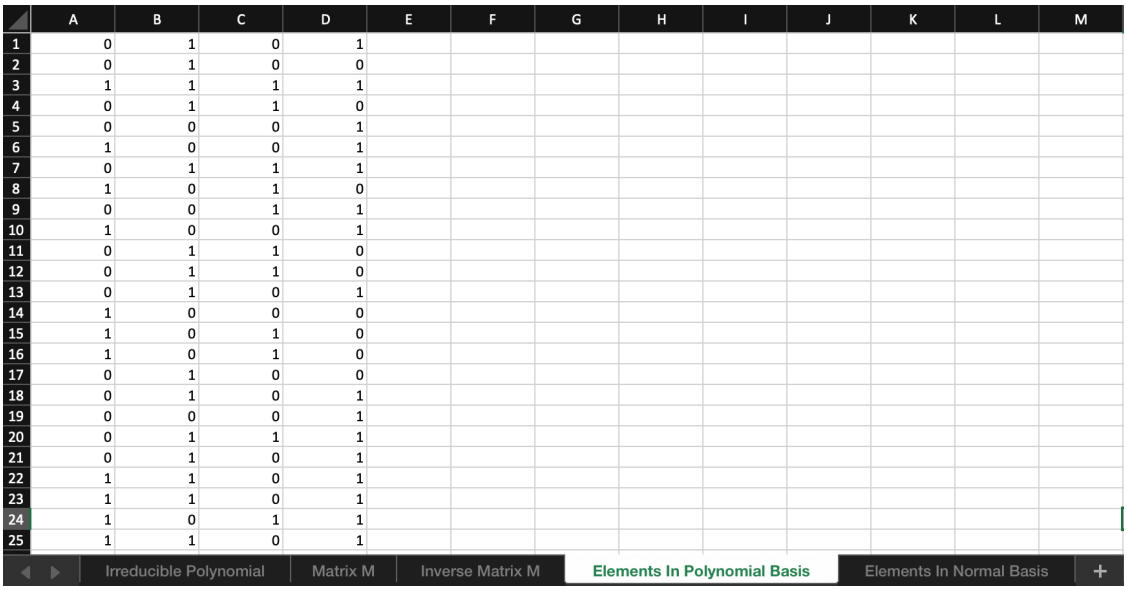
Вкладка з оберненою матрицею для перетворення елементів з поліноміального базису в нормальний (Inverse Matrix M)



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1	1	0	0									
2	1	0	1	0									
3	1	0	0	0									
4	1	0	0	1									
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													

Irreducible Polynomial | Matrix M | Inverse Matrix M | Elements In Polynomial Basis | Elements In Normal Basis

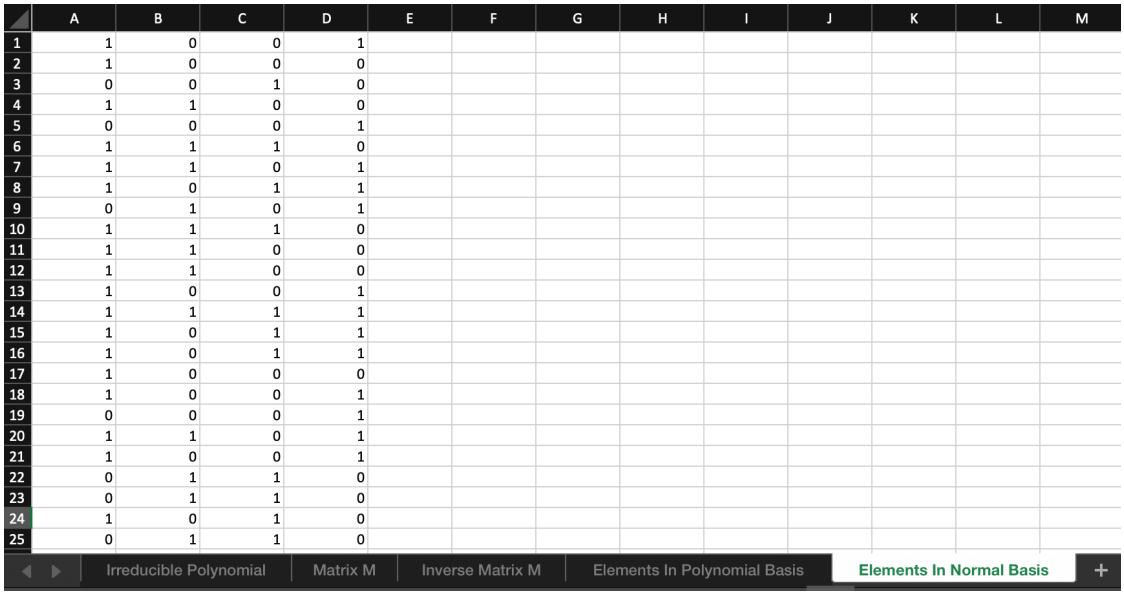
Вкладка з елементами поліноміального базису (Elements In Polynomial Basis)



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	1	0	1									
2	0	1	0	0									
3	1	1	1	1									
4	0	1	1	0									
5	0	0	0	1									
6	1	0	0	1									
7	0	1	1	1									
8	1	0	1	0									
9	0	0	1	1									
10	1	0	0	1									
11	0	1	1	0									
12	0	1	1	0									
13	0	1	0	1									
14	1	0	0	0									
15	1	0	1	0									
16	1	0	1	0									
17	0	1	0	0									
18	0	1	0	1									
19	0	0	0	1									
20	0	1	1	1									
21	0	1	0	1									
22	1	1	0	1									
23	1	1	0	1									
24	1	0	1	1									
25	1	1	0	1									

Irreducible Polynomial | Matrix M | Inverse Matrix M | Elements In Polynomial Basis | Elements In Normal Basis

Вкладка з елементами нормального базису (Elements In Normal Basis)



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1	0	0	1									
2	1	0	0	0									
3	0	0	1	0									
4	1	1	0	0									
5	0	0	0	1									
6	1	1	1	0									
7	1	1	0	1									
8	1	0	1	1									
9	0	1	0	1									
10	1	1	1	0									
11	1	1	0	0									
12	1	1	0	0									
13	1	0	0	1									
14	1	1	1	1									
15	1	0	1	1									
16	1	0	1	1									
17	1	0	0	0									
18	1	0	0	1									
19	0	0	0	1									
20	1	1	0	1									
21	1	0	0	1									
22	0	1	1	0									
23	0	1	1	0									
24	1	0	1	0									
25	0	1	1	0									

Irreducible Polynomial | Matrix M | Inverse Matrix M | Elements In Polynomial Basis | Elements In Normal Basis

Додаток Й. Характеристика набору даних MNIST

MNIST – загальнодоступний набір даних, складається з чорно-білих зразків рукописного написання арабських цифр (від 0 до 9).

Кількість класів – 10.

Набір даних MNIST є збалансованим.

Збалансованість навчального набору даних MNIST

Класи	«0»	«1»	«2»	«3»	«4»	«5»	«6»	«7»	«8»	«9»
Кількість екземплярів	5923	6742	5958	6131	5842	5421	5918	6265	5851	5949

Збалансованість тестового набору даних MNIST

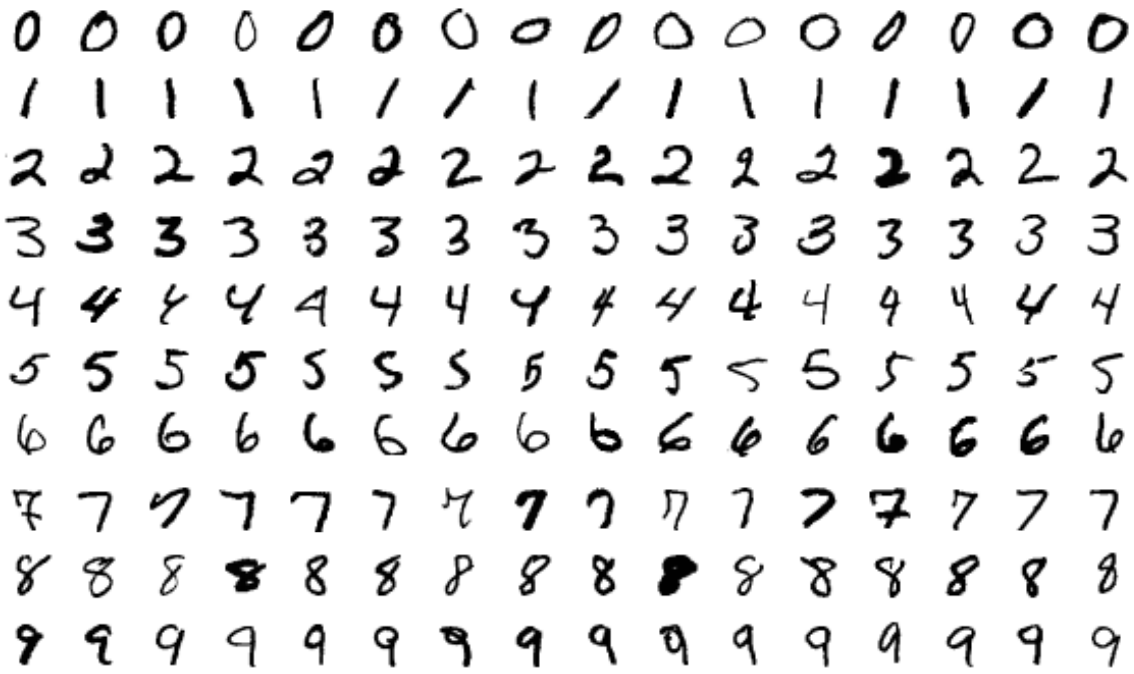
Класи	«0»	«1»	«2»	«3»	«4»	«5»	«6»	«7»	«8»	«9»
Кількість екземплярів	1001	1127	991	1032	980	863	1014	1069	945	978

Кількість екземплярів набору даних для навчання – 60000 зображень.

Кількість екземплярів набору даних для тестування – 10000 зображень.

Розмірність зображень з набору даних є однаковою й складає 28 на 28 пікселів.

Приклади екземплярів з набору даних MNIST



Додаток К. Фрагменти програмного коду системи вирішення задачі класифікації на приватних наборах даних

main.py

```
import tensorflow as tf
import matplotlib.pyplot as plt

import ImageClassifier
from EncryptionModule import ClassicCryptoNet, ModifiedCryptoNet


def check_dataset_balance(dataset):
    x = dataset[0]
    y = dataset[1]

    dictionary = dict()

    for i in range(y.shape[0]):
        class_element = y[i - 1]
        if class_element in dictionary:
            dictionary[class_element] += 1
        else:
            dictionary.update({class_element: 1})

    print(dictionary)


train_dataset, test_dataset = tf.keras.datasets.mnist.load_data()
print('train_dataset')
check_dataset_balance(train_dataset)
print('test_dataset')
check_dataset_balance(test_dataset)
img_rows = 28
img_cols = 28



image_classifier = ImageClassifier.ImageClassifier()
image_classifier.train(train_dataset)
image_classifier.evaluate(test_dataset)

image_index = 4444
image_to_predict = test_dataset[0][image_index]
plt.imshow(image_to_predict.reshape(28, 28), cmap='Greys')
plt.show()
pred = image_classifier.predict(image_to_predict.reshape(1, 28, 28, 1))
print(pred.argmax())


print('Classic Model')
image_classifier = ImageClassifier.ImageClassifier()
image_classifier.set_preprocessing_model(ClassicCryptoNet)
image_classifier.train(train_dataset)
image_classifier.evaluate(test_dataset)

image_index = 4444
image_to_predict = test_dataset[0][image_index]
```

```
plt.imshow(image_to_predict.reshape(28, 28), cmap='Greys')
plt.show()
pred = image_classifier.predict(image_to_predict.reshape(1, 28, 28, 1))
print(pred.argmax())


print('Modified Model')
image_classifier = ImageClassifier.ImageClassifier()
image_classifier.set_preprocessing_model(ModifiedCryptoNet)
image_classifier.train(train_dataset)
image_classifier.evaluate(test_dataset)

image_index = 4444
image_to_predict = test_dataset[0][image_index]
plt.imshow(image_to_predict.reshape(28, 28), cmap='Greys')
plt.show()
pred = image_classifier.predict(image_to_predict.reshape(1, 28, 28, 1))
print(pred.argmax())
```

ImageClassifier.py

```
import tensorflow as tf
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D


# from EncryptionModule import DatasetPreprocessor


class ImageClassifier:
    model = None
    learning_rate = 0.015
    metrics = ['accuracy']
    preprocessing_model = None  # DatasetPreprocessor()

    input_shape = None

    def __init__(self, input_shape=(28, 28, 1)):
        self.input_shape = input_shape
        self.model = self.__build_model()

    def train(self, dataset, epochs=10):
        x_train = dataset[0]
        y_train = dataset[1]

        # Making sure that the values are float
        x_train = x_train.reshape(x_train.shape[0],
                                  self.input_shape[0],
                                  self.input_shape[1],
                                  self.input_shape[2])
        x_train = x_train.astype('float32')
        # Normalizing the RGB codes
        x_train /= 255

        self.model.fit(x=x_train, y=y_train, epochs=epochs)

    def evaluate(self, dataset):
        x_test = dataset[0]
        y_test = dataset[1]
```

```
        # Making sure that the values are float
        x_test = x_test.reshape(x_test.shape[0],
                                self.input_shape[0],
                                self.input_shape[1],
                                self.input_shape[2])
        x_test = x_test.astype('float32')
        # Normalizing the RGB codes
        x_test /= 255

        self.model.evaluate(x_test, y_test)

    def predict(self, image):
        return self.model.predict(image.reshape(1,
                                                self.input_shape[0],
                                                self.input_shape[1],
                                                self.input_shape[2]))

    def preprocess_dataset(self, dataset):
        self.preprocessing_model.process_dataset(dataset)

    def set_preprocessing_model(self, preprocessing_model):
        self.preprocessing_model = preprocessing_model

    def __build_model(self):
        # Creating a Sequential Model and adding the layers
        model = Sequential()
        model.add(Conv2D(28, kernel_size=(3, 3),
input_shape=self.input_shape))
        model.add(MaxPooling2D(pool_size=(2, 2)))
        model.add(Flatten())  # Flattening the 2D arrays for fully
connected layers
        model.add(Dense(128, activation=tf.nn.relu))
        model.add(Dropout(0.2))
        model.add(Dense(10, activation=tf.nn.softmax))
        model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=self.metrics)
        return model
```

EncryptionModule.py

```
import tensorflow as tf
import numpy as np


class DatasetPreprocessor:
    def __init__(self):
        pass

    def process_dataset(self, dataset):
        pass


class ClassicCryptoNet(DatasetPreprocessor):
    key_size = 16
    message_size = 16
    learning_rate = 0.0015
    batch_size = 5

    alice_network = None
```

```
    bob_network = None
    eve_network = None

    key = None

    sess = None
    alice_saver = None
    bob_saver = None
    eve_saver = None

    eve_opt = None
    bob_opt = None

    def __init__(self):
        DatasetPreprocessor.__init__(self)

        self.alice_network = self.__build_model('Alice',
alice_input_message, self.key)
        self.bob_network = self.__build_model('Bob', self.alice_network,
self.key)
        self.eve_network = self.__build_model('Eve', self.alice_network)


        alice_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES,
scope='Alice')
        bob_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES,
scope='Bob')
        eve_vars = tf.get_collection(tf.GraphKeys.GLOBAL_VARIABLES,
scope='Eve')
        self.alice_saver = tf.train.Saver(alice_vars)
        self.bob_saver = tf.train.Saver(bob_vars)
        self.eve_saver = tf.train.Saver(eve_vars)

        self.eve_opt =
tf.train.AdamOptimizer(learning_rate=self.learning_rate, beta1=0.9,
epsilon=1e-08)
            .minimize(self.eve_loss_function, var_list=eve_vars])
        self.bob_opt =
tf.train.AdamOptimizer(learning_rate=self.learning_rate, beta1=0.9,
epsilon=1e-08)
            .minimize(self.bob_loss_function, var_list=[alice_vars +
bob_vars])
        self.sess = tf.Session()
        init = tf.global_variables_initializer()
        self.sess.run(init)


    def bob_loss_function(self, plain_text, prediction, eve_prediction):
        return self.eve_loss_function(plain_text,prediction) +\
               tf.reduce_sum(tf.square(float(self.message_size) / 2.0 -

self.eve_loss_function(plain_text,eve_prediction)) /
                             ((self.message_size / 2) ** 2))

    def eve_loss_function(self, plain_text, prediction):
        return (1 / self.batch_size) * tf.reduce_sum(tf.abs(prediction -
plain_text))

    def train(self, dataset, epochs, batch_size):
        messages = dataset[0]
        keys = build_key_matrix(self.message_size, self.key_size)
        steps_per_epoch = 15
```

```
        ITERS_PER_ACTOR = 2
        EVE_MULTIPLIER = 1

        # Training begins
        for i in range(epochs):

            for j in range(steps_per_epoch):

                # get batch dataset to train
                batch_messages = messages[j * batch_size: (j + 1) *
batch_size]
                batch_keys = keys[j * batch_size: (j + 1) * batch_size]

                # Train Alice and Bob
                for _ in range(ITERS_PER_ACTOR):
                    temp = self.sess.run(
                        [self.bob_opt, self.alice_loss_function(),
self.bob_loss_function, self.bob_network],
                                         feed_dict={alice_input_message:
batch_messages, key: batch_keys})

                    temp_alice_bob_loss = temp[1]
                    temp_eve_evs_loss = temp[2]
                    temp_bob_msg = temp[3]

                # train Eve
                for _ in range(ITERS_PER_ACTOR + EVE_MULTIPLIER):
                    temp = self.sess.run([eve_opt, self.eve_loss_function,
self.eve_network],
                                         feed_dict={alice_input_message:
batch_messages, key: batch_keys})

                    temp_eve_loss = temp[1]
                    temp_eve_msg = temp[2]

            # save after every 5 epochs
            if i % 5 == 0 and i != 0:
                self.save_model()

            # output bit error and loss after every epoch
            if i % 1 == 0:
                print('  epochs: ', i, '  bob bit error: ',
temp_alice_bob_loss, ' + ', temp_eve_evs_loss,
                      '   & eve bit error:', temp_eve_loss)

    def evaluate(self, dataset):
        pass

    def process_dataset(self, dataset):
        encrypted_dataset = []
        for element in dataset:
            encrypted_dataset.append(self.alice_network(element))
        pass

    def encrypt(self, plain_text):
        return self.alice_network(plain_text, self.key)

    def load_model(self, filepath="./weights/"):
        self.alice_saver.restore(self.sess, filepath +
"alice_weights/model.ckpt")
        self.bob_saver.restore(self.sess, filepath +
"bob_weights/model.ckpt")
```

```
        self.eve_saver.restore(self.sess, filepath +
"eve_weights/model.ckpt")

    def save_model(self, filepath="./weights/"):
        self.alice_saver.save(self.sess, filepath +
"alice_weights/model.ckpt")
        self.bob_saver.save(self.sess, filepath + "bob_weights/model.ckpt")
        self.eve_saver.save(self.sess, filepath + "eve_weights/model.ckpt")

    def __build_model(self, collection, message, key=None):
        if key is not None:
            combined_message = tf.concat(axis=1,
                                         values=[message, key])
        else:
            combined_message = message

        with tf.variable_scope(collection):
            fc = tf.layers.dense(combined_message, self.message_size +
self.key_size, activation=tf.nn.relu)
            fc = tf.expand_dims(fc, 2)

            conv1 = tf.layers.conv1d(fc, filters=2, kernel_size=4,
strides=1, padding='SAME', activation=tf.nn.sigmoid)
            conv2 = tf.layers.conv1d(conv1, filters=4, kernel_size=2,
strides=2, padding='VALID',
                                     activation=tf.nn.sigmoid)
            conv3 = tf.layers.conv1d(conv2, filters=4, kernel_size=1,
strides=1, padding='SAME',
                                     activation=tf.nn.sigmoid)

            # output
            conv4 = tf.layers.conv1d(conv3, filters=1, kernel_size=1,
strides=1, padding='SAME', activation=tf.nn.tanh)

            out = tf.squeeze(conv4, 2)
        return out
```