Issue 13, 2023

https://doi.org/10.23939/sisn2023.13.338

УДК 004.382

АНАЛІЗ АЛГОРИТМІВ МНОЖЕННЯ В ПОЛЯХ ГАЛУА ДЛЯ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ

Іван Жолубак

Національний університет "Львівська політехніка",
кафедра електронних обчислювальних машин, Львів, Україна
E-mail: Ivan.M.Zholubak@lpnu.ua, ORCID: 0000-0001-8871-7222

Математичною основою опрацювання цифрового підпису є еліптичні криві. Опрацювання точок еліптичної кривої ґрунтується на виконанні операцій у полях Галуа $GF(p^m)$. Поля з простою основою недостатньо вивчені та дуже цікаві для дослідження. У роботі здійснено порівняння складності алгоритмів реалізації операції множення у полях Галуа $GF(p^m)$ з різними основами. Виконано порівняння трьох найпоширеніших алгоритмів множення. Встановлено, що для полів з основою, більшою за 2, алгоритм буде складнішим.

Ключові слова: поля Галуа $GF(d^m)$; помножувач; модифікована комірка Гілда; LUT; генератор ядер.

Вступ

Нині на практиці використовують поля Галуа з основою 2 – $GF(2^m)$ та прості поля – $GF(p)$. Поля з основою простого числа $GF(p^m)$ недостатньо вивчені. Джерела [1– 3] містять порівняння операційних пристроїв полів Галуа з основою простого числа, проте не дають уявлення про апаратні затрати на реалізацію операційних пристроїв для полів з основою простого числа на всьому проміжку основ поля. Таке порівняння уможливлюється у разі створення автоматизованої системи конфігурування вузлів криптографічного захисту інформації. Не здійснено порівняння апаратних та програмних моделей вузлів криптографічного захисту інформації. Не виконано порівняння апаратної та програмної складності, не проаналізовано варіанти злому шифрів методом простого перебору та переваги і недоліки полів з різними основами.

Аналіз літературних джерел

Математичною основою цифрового підпису є операції у полях Галуа. Найскладніша операція – операція множення. У [1–3] наведено теоретичне порівняння апаратної складності помножувачів полів Галуа з різною основою. В результаті порівняння встановлено, що за кількістю логічних блоків (елементів, які мають шість входів та один вихід) 3-ві, 5-ві та 7-ві поля матимуть найменшу апаратну складність. У [1] автори здійснили порівняння помножувачів полів Галуа, вважаючи МКГ цілісним елементом, а у [2] – вважаючи, що МКГ складається із помножувача та суматора. В [3] виконано порівняння апаратних затрат помножувачів полів Галуа, коли МКГ складається із дрібніших побітових елементів, які виконують операцію множення, ділення за модулем основи поля, додавання та знову ділення за модулем основи поля. У [4] наведено порівняння часової складності алгоритмів множення у полях Галуа.

Здійснено також порівняння структурної [5, 6, 8] складності полів Галуа з різними основами. Метою роботи є оцінювання апаратних затрат на створення помножувальної матриці [7] помножувача елементів полів Галуа у поліноміальному базисі та вибору поля із найменшими апаратними затратами. Під час виконання цієї роботи на основі запропонованої у [1] та [2] моделі помножувача виконано його імплементацію та здійснено перевірку отриманих теоретично у [1] та [2] результатів апаратної складності.

Мета роботи

Метою роботи є порівняння складності алгоритмів множення у полях Галуа з різними основами $GF(p^m)$. Виконано порівняння трьох алгоритмів реалізації операції множення. Коди елементів полів Галуа подано в поліноміальному базисі.

Злом криптографічних алгоритмів

Будь-який алгоритм, оснований на ключі, можна зламати методом простого перебору. Обчислювальна потужність комп'ютера, в цьому випадку, зростає експоненціально у разі збільшення довжини ключа. Багатьох цікавить питання: "Чи можна розв'язати таку задачу на домашньому комп'ютері?". Якщо довжина ключа – 32 біти, то потрібно 109 років, а якщо більша, то необхідні розподілені обчислення. Злом криптографічних алгоритмів – справа нетривіальна. Тому атаки можуть бути результативними лише в разі використання якихось спеціальних засобів або способів. Існують такі методи злому криптографічних шифрів:

1) використання суперкомп'ютерів;
2) розподілені обчислення;
3) використання квантових комп'ютерів;
4) криптоаналіз;
5) атаки з підміною ключа.

Суперкомп'ютери для вирішення завдань криптоаналізу дуже потужні. Щоб це зрозуміти, часто виконують такий експеримент. Припустимо, що розглядаються ідеальні алгоритми шифрування, для яких оптимальним методом є прямий перебір всіх можливих ключів. У цьому випадку стійкість криптосистем визначатиметься довжиною ключа. Суперкомп'ютери в тисячі разів швидші за персональні комп'ютери, і продуктивність суперкомп'ютерів неухильно зростає.

Останнім часом, завдяки розвитку мереж, зокрема інтернету, можна ефективно використовувати метод "грубої сили" (перебору) із розподілом операцій між багатьма обчислювальними пристроями. Такий підхід зазвичай реалізують, встановлюючи сервер з доступом до інтернету, з якого будь-хто може завантажити програму для розшифрування тестового повідомлення за допомогою перебору ключів. Зазвичай програма поставляється у вигляді вихідних текстів або скомпільована для найпоширеніших операційних систем. Після запуску програма встановлює з'єднання з сервером, отримує набір ключів для перебору, а після закінчення роботи повертає результат.

Квантові комп'ютери. У серпні 2000 р. корпорація IBM оголосила про розроблення першого в світі квантового комп'ютера. Експериментальна модель, побудована на п'яти атомах, продемонструвала потенціал таких систем, вирішуючи певні завдання зі швидкістю, що значно перевищує швидкодію звичайних комп'ютерів. Якщо збільшити кількість атомів, які працюють, до тисячі або мільйона, то потенціал квантового комп'ютера істотно зросте. Керівник групи вчених з компанії IBM, Стенфордського університету й Університету Калгарі Ісаак Чуанг (Isaac Chuang), стверджує, що квантовий комп'ютер можна використовувати, зокрема, і для злому криптографічних шифрів.

Команда Чуанг застосовує експериментальний квантовий комп'ютер для розв'язання традиційних математичних задач, що трапляються у криптографії, зокрема, пошуку періоду функції. Квантовий комп'ютер здатний вирішити будь-яке завдання цього типу за один крок, тоді як звичайному комп'ютеру потрібно багато циклів. На відміну від звичайного комп'ютера, який для вирішення

криптографічних завдань виконує послідовні додавання кількох чисел, квантова машина, використовуючи спін електрона або частинок атомного ядра, може додавати числа одночасно. За певних типів обчислень, таких як криптографічні алгоритми, квантовий комп'ютер, що складається з декількох сотень атомів, може виконувати мільярди операцій одночасно.

Криптоаналіз – це область діяльності, спрямована на розкриття зашифрованих повідомлень, ключ шифрування і/або алгоритм яких невідомий. У криптоаналізі є елементи не тільки науки, але і мистецтва, і просто щасливого випадку.

У сучасній криптографії під час проєктування криптографічної системи передбачають, що алгоритм є відкритим, а ключ надійно захищений. Стійкість алгоритму шифрування зазвичай оцінюється за кількістю операцій, необхідних для розшифрування зашифрованого повідомлення або підбору ключа шифрування найкращим алгоритмом.

Криптоаналіз використовують для таких завдань:

1) розкриття вихідної інформації з криптограми;
2) обчислення закритого ключа із відомого відкритого ключа;
3) формування електронного цифрового підпису повідомлення без знання закритого ключа;
4) створення фальшивого електронного документа, що відповідає відомому підпису.

Атака з підміною ключа – це вид нападу, який часто використовують для атак на протокол обміну ключами. Основна ідея полягає у тому, що зловмисник проникає на лінію обміну повідомленнями між двома сторонами, які обмінюються ключами для забезпечення секретного зв'язку. Після цього зловмисник видає кожній стороні свої ключі, що призводить до того, що обидві сторони отримують різні ключі, відомі зловмисникові. Як результат, зловмисник може розшифрувати кожне повідомлення за допомогою свого ключа, а потім зашифрувати його за допомогою іншого ключа, щоб відправити його адресату. Сторони вважають, що їхні повідомлення залишаються секретними, хоча насправді зловмисник читає кожне повідомлення.

Якщо ключі не відомі зловмисникові, то єдиним ефективним методом злому шифру є повний перебір усіх можливих ключів. Це означає, що чим складніший алгоритм шифрування, тим важче дешифрувати його.

Реалізація вузлів криптографічного захисту інформації на ПЛІС

У статтях [2, 3] здійснено теоретичне порівняння складності алгоритмів множення у полях Галуа. Виконано практичне порівняння алгоритмів множення у полях Галуа та наведено основні алгоритми множення. У роботі [8] наведено два поширені методи виконання ділення в $GF(2^m)$. Метод 1 – розширений алгоритм Евкліда, який використовує поліноміальне базове представлення для поля $GF(2^m)$. Спосіб 2 – піднесення до степеня. Підвищення ефективності методу особливо істотне, коли піднесення до квадрата можна виконати швидко (наприклад, у нормальному базовому поданні). Недоліком розширеного алгоритму Евкліда є залежність часу мультиплікативного зворотного обчислення полів Галуа від значення операндів. У роботі [9] запропоновано метод реалізації функцій ймовірностей Маркова, визначених на основі стохастичної матриці заданого розміру, із використанням набору багатовимірних поліномів над полем Галуа та масиву рівномірно розподілених некорельованих (псевдо) випадкових чисел заздалегідь визначеної ємності. Визначено похибку представлення елементів стохастичної матриці залежно від кількості змінних і степеня поля, над яким задано поліноми. Також визначено складність обчислення набору зазначених поліномів у архітектурі PLD класу FPGA. У роботі [10] запропоновано модель мозаїки для обчислення множення точок на конічних кривих над кінцевим полем $GF(2^n)$. Вся модель складається з двох типів підмоделей, які виконують різні функції. Обидві підмоделі містять три частини, і вони мають спільні другу та третю частини, розроблені на основі моделі поділу мозаїки. Основні відмінності між двома підмоделями полягають у першій частині, яка обчислює подвоєння точки та генерує різні параметри для інших частин. У роботі [11] запропоновано новий високоефективний алгоритм і архітектуру множення точок ECC. По-перше, проаналізовано та оптимізовано конструкції модульного додавання,

модульного множення, модульного квадрата та модульної інверсії. Потім запропоновано інтегральну послідовну та частково паралельну структуру множення точок над $GF(2^m)$ і спроєктовано загальну апаратну схему з використанням двох модульних блоків множення. У статті [12] наведено ефективні паралельні та послідовні систолічні структури для комбінованого множення та піднесення до квадрата над $GF(2^m)$. У роботі [13] розглянуто проблему зниження апаратної ефективності універсальних помножувачів $GF(2^m)$ РВ у результаті збільшення затримки завантаження входів і незвідного полінома. У роботі [14] запропоновано ефективний підхід за площею до реалізації множення кінцевих полів на основі SET, яка потребує великої кількості операцій XOR і демонструє значний потенціал для дослідження архітектур множення (таких як множення на основі алгоритму Карацуби) для подальшої економії площі.

Алгоритм на основі бінарних операцій

Алгоритм працює подібно до апаратної реалізації такого помножувача. Формуються булеві функції, які обчислюють результат кожної модифікованої комірки Гілда. Алгоритм доволі складний для реалізації, а для полів з великою основою булеві функції будуть дуже складними. Перевага цього алгоритму в тому, що для доволі малих основ поля ми обробляємо одразу багато розрядів.

```
    _int64 A[ceil(log(p) / log(2))][2 * m], A_prom[ceil(log(p) / log(2))][2 *
m], B[ceil(log(p) / log(2))][2 * m], S[ceil(log(p) / log(2))][2 * m],
F[ceil(log(p) / log(2))][2 * m], P[ceil(log(p) / log(2))][2 * m];
    //ініціалізація кожної змінної вхідними значеннями
    //прямий хід обчислень
    for (int i = 0; i < m; i++)
    {   //підготовка даних
     for (int j = 0; j < ceil(log(p) / log(2)); j++)
     {
           bool mask = maska(A[j]);//накладання маски для виявлення молодшого
біта
           shift_right(A[j], 1);//Зсуває кожний A[j] на 1 біт
           for (int k = 0; k < 2 * m / 64; k++)
           {
                 initial_number_1_bit(A_prom,   mask);//ініціалізація   усього
елементу масива 1 бітом mas;
           }
           shift_left(B[j], 1);//Зсуває кожний B[j] на 1 біт
           shift_left(A_prom[j], i);
     }
     //виконання бінарних операцій
     for (int j = 0; j < ceil(log(p) / log(2)); j++)
     {
           for (int k = 0; k < 2 * m / 64; k++)
           {
                 S[j][k] = bulian_function_j(A_prom[j][k], B[j][k], S[j][k]);
           }
     }
    }
    //зворотній хід обчислень
    for (int i = 0; i < m - 1; i++)
```

```
{   //підготовка даних
 for (int j = 0; j < ceil(log(p) / log(2)); j++)
 {
       bool f_elem = f_function(S, j);//обчислення біта S[j]
       for (int k = 0; k < 2 * m / 64; k++)
       {
             initial_number_1_bit(F[j][k], f_elem);//ініціалізація усього
елементу масива 1 бітом f_elem;
       }
       shift_right(P[j], 1);//Зсуває кожний P[j] на 1 біт
       shift_left(F[j], m - 2 - i);
 }
 //виконання бінарних операцій
 for (int j = 0; j < ceil(log(p) / log(2)); j++)
 {
       for (int k = 0; k < 2 * m / 64; k++)
       {
             S[j][k] = bulian_function_j(A_prom[j][k], B[j][k], S[j][k]);
       }
 }
}
```

Таблиця 1

Кількість логічних елементів для створення МКГ

Поле Галуа $GF(p^m)$	МКГ є цілісним елементом	МКГ складається із помножувача та суматора	Найбільша довжина ланцюжка МКГ по вертикалі
$GF(2^{100})$	2	2	199
$GF(3^{63})$	28	30	125
$GF(5^{43})$	380	137	85
$GF(7^{35})$	440	129	69
$GF(13^{27})$	3124	870	53

Складність алгоритму обчислення значення МКГ можна виразити формулою: $O_{МКГ}(p)=O(2^{2\lceil\log_2 p\rceil+1})$. Складність алгоритму множення одного вертикального ланцюжка елементів полів Галуа $GF(p^m)$ можна відобразити формулою: $O_{MUL}(p,m)=O((2^{2\lceil\log_2 p\rceil+1})2m)$. Повну складність алгоритму множення елементів полів Галуа $GF(p^m)$ можна обчислити за формулою: $C_{ариф.} = 2^{2\lceil\log_2 p\rceil+1} 4m^2 / 64$, це зумовлено розрядністю ЦП – 64 біти. У табл. 2 подано кількість логічних операцій, які необхідно виконати для множення двох елементів поля $GF(p^m)$. Варіант, коли МКГ є цілісним елементом, не розглядається, оскільки, якщо подати МКГ як сукупність із двох елементів, потрібно виконувати набагато менше бітових операцій.

Таблиця 2

Кількість бітових операцій для виконання операції множення на ЦП за 1 алгоритмом

Поле Галуа $GF(p^m)$	МКГ складається із помножувача та суматора	Найбільша довжина ланцюжка МКГ по вертикалі	Сумарна кількість бітових операцій, які необхідно виконати для 1 розряду (64 біти – розрядність ЦП)	Повна кількість бітових операцій, які необхідно виконати для обчислення результату	Відношення кількості бітових операцій у разі обчислення у полі з основою p до кількості операцій за обчислення у полі з основою 2
$GF(2^{100})$	2	199	398	796	1
$GF(3^{63})$	30	125	3750	7500	9,42
$GF(5^{43})$	137	85	11645	23290	29,25
$GF(7^{35})$	129	69	8901	17802	22,36
$GF(13^{27})$	870	53	46110	92220	115,85

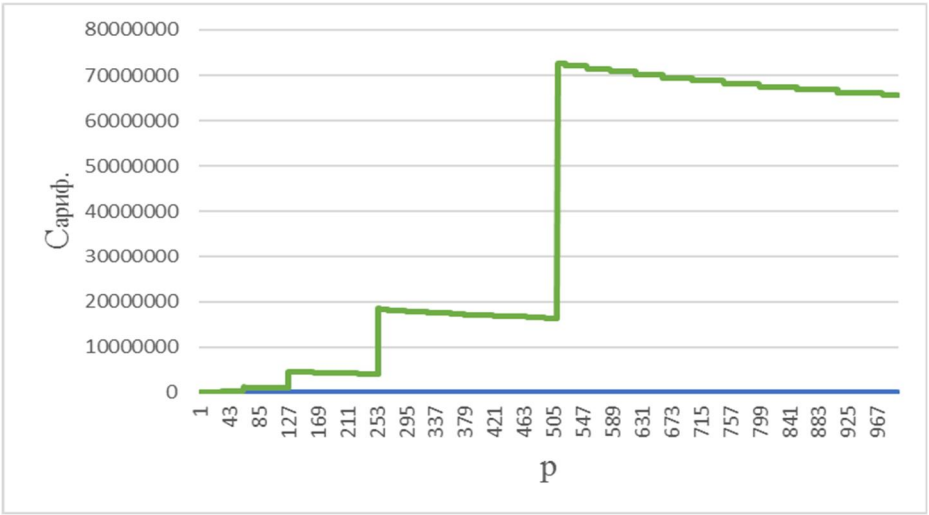

Рис. 1. Складність алгоритму множення на основі логічних функцій

Алгоритм множення можна також реалізувати на основі арифметичних операцій, які виконують у кожній МКГ. В одній МКГ потрібно виконати операцію множення, ділення (зведення за модулем), додавання і знову ділення (зведення за модулем). Таким методом можна реалізувати МКГ, для поля Галуа з основою, не більшою за 32. Це зумовлено розрядністю ЦП (може бути 64-бітний результат множення). Кожна МКГ виконуватиме чотири арифметичні операції, які матимуть вагу – відношення часу виконання складної операції до простої, відмінної від побітових операцій. Операція множення та ділення – 8, додавання – 4. Отже, для моделювання роботи МКГ потрібно виконати чотири арифметичні операції, які рівносильні 28 бінарним операціям.

```
_int64 A[m], B[m], S[2 * m - 1], F, P[m];
//ініціалізація кожної змінної вхідними значеннями
//прямий хід обчислень
int w1 = 0;
int w2 = m;
int y;
```

```
for (int i = 0; i < m; i++)
{
  y = 0;
  for (int j = w1; j < w2; j++)
  {
          S[j] = ((A[i] * B[y] % p) + S[j]) % p;// виконання операцій у МКГ
          y++;
  }
  w1++;
  w2++;
}
//зворотній хід обчислень
int r = 2 * m - 1;
for (int i = 0; i < m - 1; i++)
{   //підготовка даних
  y = 0;
  w1--;
  w2--;
  F = m - S[r];
  for (int j = w1; j < w2; j++)
  {
          S[j] = (((P[y] * F) % m) + S[j]) % m;
  }
}
```

Таблиця 3

Кількість бітових операцій для виконання операції множення на ЦП за другим алгоритмом

Поле Галуа $GF(p^m)$	Кількість елементарних операцій, які необхідно виконати для моделювання роботи 1 МКГ	Кількість МКГ у помножувачі	Сумарна кількість елементарних операцій, які необхідно виконати для множення елементів поля	Відношення кількості елементарних операцій у разі виконання моноження елементів поля з основою p до поля з основою 2
$GF(2^{100})$	28	19900	557200	1
$GF(3^{63})$	28	7875	220500	0,39
$GF(5^{43})$	28	3655	102340	0,18
$GF(7^{35})$	28	2415	67620	0,12
$GF(13^{27})$	28	1431	40068	0,07

Складність МКГ є лінійною та має асимптотне значення 28. Складність алгоритму множення визначається за формулою $O(2m^2 - m)$, тобто залежить від кількості МКГ у помножувачі, а кількість усіх арифметичних операцій оцінюється як $C_{ариф.} = 28 \times 2m^2 - m$.

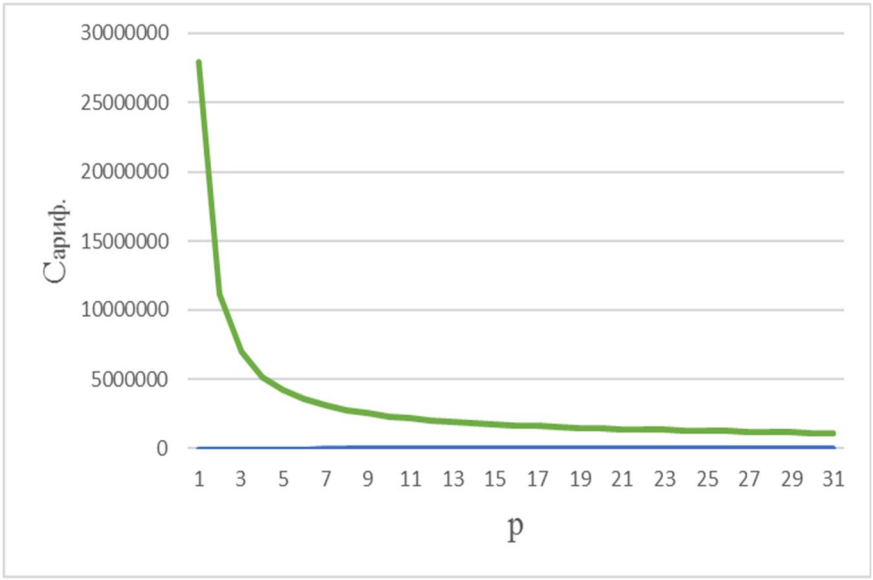

Рис. 2. Графік складності алгоритму множення на основі арифметичних операцій

Аналіз помножувача у разі реалізації МКГ на основі елементарних побітових операцій під час виконання множення, ділення, додавання і знову ділення викладено нижче. Використовуючи елемент, який виконує *2p*-розрядні побітові операції, потрібно виконати для реалізації операції множення $\lceil \log_2 p \rceil$ елементарних операцій, ділення – $\lceil \log_2 p \rceil$ операцій, додавання – одну операцію та знову ділення – $\lceil \log_2 p \rceil$ операцій. Загалом для реалізації МКГ потрібно виконати $4(3\lceil \log_2 p \rceil+1)$ бінарних операцій, тому що кожен елемент у матричному помножувачі та подільнику повинен виконувати приблизно чотири бінарні операції, для реалізації поля з основою, не більшою за 32 та $4(3\lceil \log_2 p \rceil+1)\lceil \log_2 p \rceil/32\rceil$, для поля з будь-якою основою. Побітові операції виконують на ЦП з розрядністю 64, але для точності обчислення проміжні результати множення не повинні перевищувати половину розрядів процесора. Формула для оцінювання кількості арифметичних операцій, які виконано для множення двох елементів поля: $C_{ариф.} = (4(3\lceil \log_2 p \rceil+1)\lceil\lceil \log_2 p \rceil/32\rceil)(2m^2 - m)$.

Таблиця 4

Кількість бітових операцій для виконання операції множення на ЦП за третім алгоритмом

Поле Галуа $GF(p^m)$	Кількість елементарних операцій, які виконуються 1 МКГ	Кількість МКГ у помножувачі	Сумарна кількість елементарних операцій, які необхідно виконати для множення елементів поля	Відношення кількості бітових операцій у разі множенняу полі з основою p до аналогічної кількості у полі з основою 2
$GF(2^{100})$	16	19900	318400	1
$GF(3^{63})$	28	7875	220500	0,69
$GF(5^{43})$	40	3655	146200	0,45
$GF(7^{35})$	40	2415	96600	0,3
$GF(13^{27})$	52	1431	74412	0,23

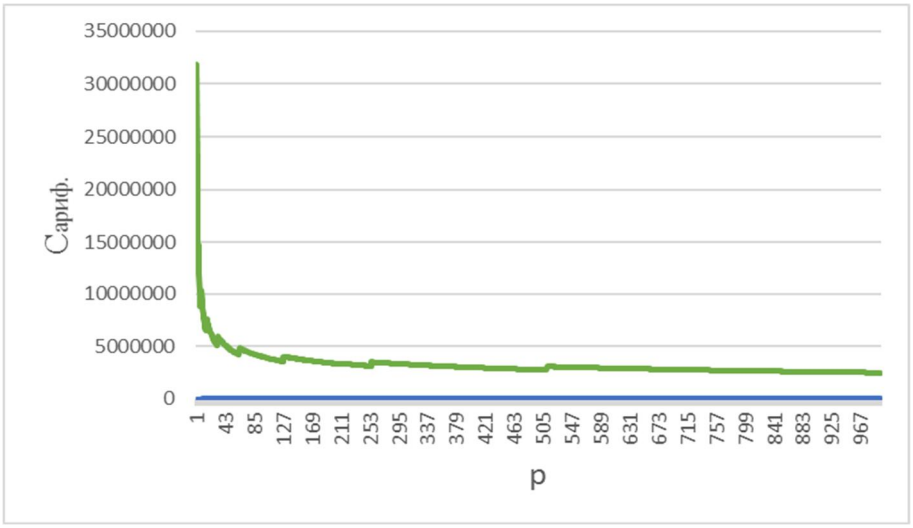

Рис. 3. Складність алгоритму множення за кількістю логічних опреацій, необхідних для виконання арифметичних операцій

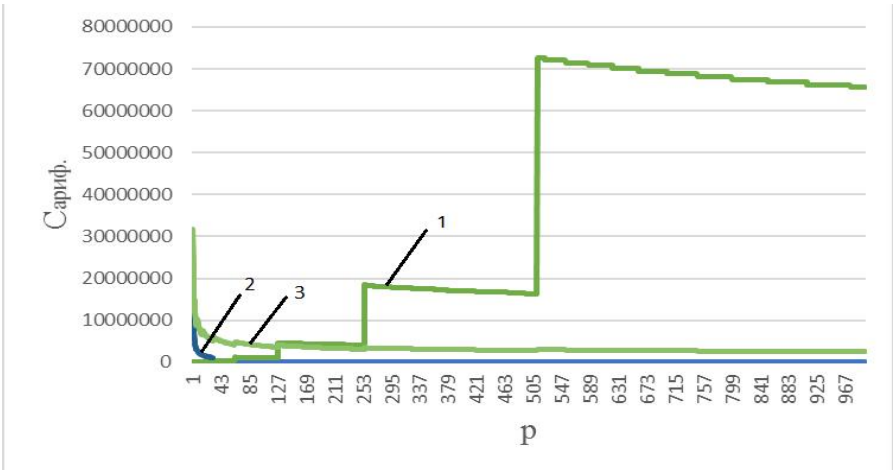

Рис. 4. Порівняння складності алгоритмів множення елементів полів Галуа на ЦП у разі реалізації операції множення трьома різними алгоритмами

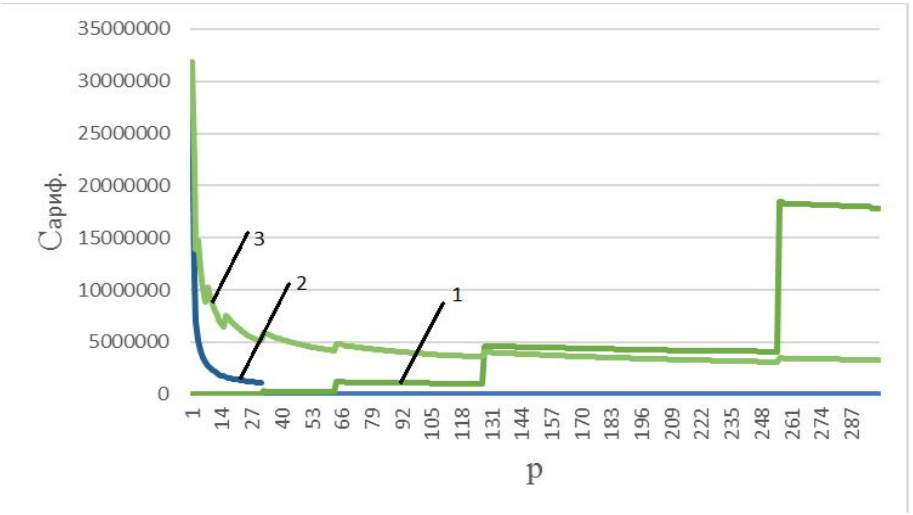

Рис. 5. Порівняння складності алгоритмів множення елементів полів Галуа на ЦП трьома різними алгоритмами. Уточнено початкову ділянку функцій

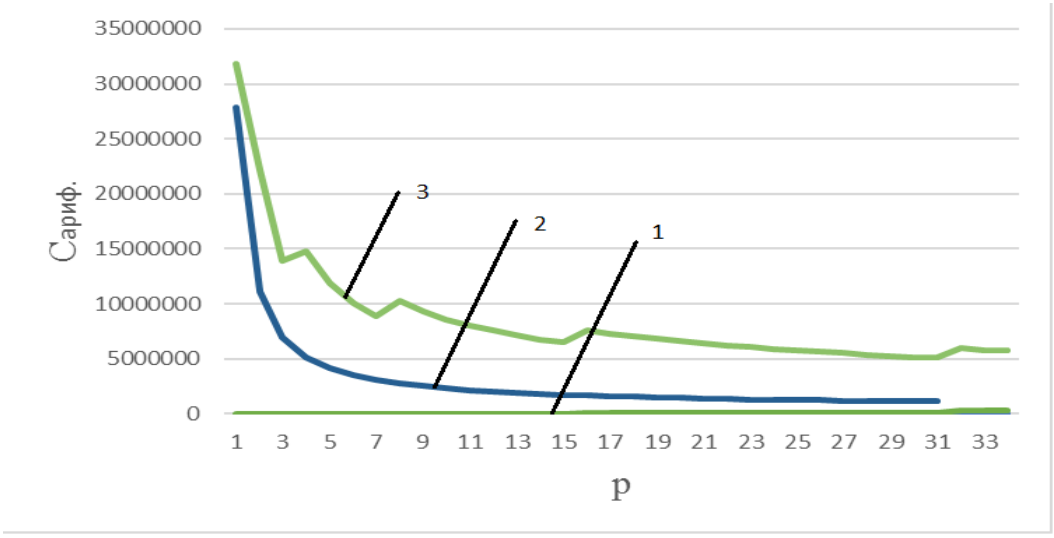

Рис. 6. Порівняння складності алгоритмів множення елементів полів Галуа на ЦП трьома різними алгоритмами. Ще уточненіша початкова ділянка функцій

Результати показують, що моделювати роботу помножувачів елементів полів Галуа краще методом елементарних побітових операцій. Складність цього алгоритму для невеликих основ поля доволі мала.

Для прикладу, складність моделювання помножувача для поля з основою 3 в 15 разів більша, ніж для поля з основою 2, тобто його у 15 разів важче зламати.

Водночас аналіз апаратної реалізації помножувачів показує, що найкращими є поля з основою 2, 3 та 7.

Отже, використання апаратних помножувачів у полях Галуа забезпечує найкращі апаратні, структурні та часові параметри і водночас ускладнює завдання моделювання їхньої роботи (злому системи зловмисниками).

Висновки

Розроблено автоматизовану систему конфігурування вузлів криптографічного захисту інформації. На її основі виконано порівняння апаратних затрат помножувачів полів Галуа з основами 2, 3, 5 7, 13 на ПЛІС фірми Xilinx Virtex-7. У результаті порівняння результатів імплементації помножувачів визначено, що найменші апаратні затрати будуть у помножувачів полів Галуа з основою 2. Також показано, що для полів з основою, більшою за 2 та меншою, ніж велике просте поле, стійкість алгоритму до злому буде вищою.

Список літератури

1. Жолубак І. М., Костик А. Т., Глухов В. С. Особливості опрацювання елементів трійкових полів Галуа на сучасній елементній базі. *Вісник Національного університету "Львівська політехніка" "Комп'ютерні системи та мережі"*. 2015. Вип. 830. С. 27–33.

2. Жолубак І. М., Глухов В. С. Визначення розширеного поля Галуа GF(d^m) з найменшою апаратною складністю помножувача. *Вісник Національного університету "Львівська політехніка" "Інформаційні системи та мережі"*. 2016. Вип. 835. С. 50–58.

3. Жолубак І. М., Глухов В. С. Апаратні витрати помножувачів полів Галуа GF(d^m) з великою основою. *Вісник Національного університету "Львівська політехніка" "Комп'ютерні науки та інформаційні технології"*. Львів, 2017.

4. Hlukhov V., Zholubak I., Kostyk A., Rahma M. (2017). Galois Fields Elements Processing Units for Cryptographic Data Protection in Cyber-Physical Systems. *Advances in Cyber-Physical Systems*, Vol. 2, No. 2, Lviv Polytechnic National University, 44–53.

5. Глухов В. С., Еліас Р. М. Зменшення структурної складності багатосекційних помножувачів елементів полів Галуа. *Електротехнічні та комп`ютерні системи*. 2015. № 19 (95). С. 222–226.

6. Черкаський М. В., Ткачук Т. І. Характеристики складності пристроїв множення. *Радіоелектронні і комп'ютерні системи*. 2012. № 5. С. 142–147.

7. Hlukhov V., Hlukhova A. (2014). Galois field elements multipliers structural complexity evaluation,/ Proceedings of the 6th International Conference ACSN-2013, Lviv, 18–19.

8. Глухов В. С., Тріщ Г. М. Оцінка структурної складності багатосекційних помножувачів елементів полів Галуа. *Вісник Національного університету "Львівська політехніка" "Комп'ютерні системи та мережі.* Вип. 806. С. 27–33.

8a. Rahma M., Zholubak I. and Hlukhov V. (2018). Devices for multiplicative inverse calculation in the binary Galois fields, 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kyiv, UKraine, 261–264. DOI: 10.1109/DESSERT.2018.8409141.

9. Shalagin S. and Zakharov V. (2021). Implementing the Markov Probability Functions Based on a Set of Polynomials over Galois Field, 2021 International Conference on Information Technology and Nanotechnology (ITNT), Samara, Russian Federation, 1–3. DOI: 10.1109/ITNT52450.2021.9649171.

10. Li Y. (2020). A tile assembly model to calculate point-multiplication on conic curves over finite field GF(2n), 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Exeter, United Kingdom, 41–48. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00032.

11. Wu K. and Wei G. (2019). Optimized Design of ECC Point Multiplication Algorithm Over GF(2m), 2019 International Conference on Electronic Engineering and Informatics (EEI), Nanjing, China, 420–425. DOI: 10.1109/EEI48997.2019.00097.

12. Ibrahim A. (2019). Efficient Parallel and Serial Systolic Structures for Multiplication and Squaring Over GF(2m), in *Canadian Journal of Electrical and Computer Engineering*, Vol. 42, No. 2, 114–120, Spring 2019. DOI: 10.1109/CJECE.2019.2900087.

13. El-Razouk H. (2022). Input-Latency Free Versatile Bit-Serial GF(2m) Polynomial Basis Multiplication in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 30, No. 5, 589–602, May 2022, DOI: 10.1109/TVLSI.2022.3155611.

14. Zhang C., Chen C. and Wu H. (2021). Area-Efficient Finite Field Multiplication in GF(2n) Using Single-Electron Transistors, 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS), Penang, Malaysia, 25–28. DOI: 10.1109/APCCAS51387.2021.9687675.

References

1. Zholubak I. M., Kostyk A. T., Glukhov V. S. (2015). Peculiarities of processing the elements of triple Galois fields on the modern element base, *Bulletin of the Lviv Polytechnic National University "Computer systems and networks"*, Is. 830, 27–33.

2. Zholubak I. M., Glukhov V. S. (2016). Determination of the extended Galois field GF(dm) with the least hardware complexity of the multiplier, *Bulletin of the Lviv Polytechnic National University "Information Systems and Networks"*, Is. 835, 50–58.

3. Zholubak I. M., Glukhov V. S. (2017). Hardware costs of Galois field multipliers GF(dm) with a large basis, *Bulletin of the Lviv Polytechnic National University "Computer Sciences and Information Technologies"*.

4. Hlukhov V., Zholubak I., Kostyk A., Rahma M. (2017). Galois Fields Elements Processing Units for Cryptographic Data Protection in Cyber-Physical Systems. *Advances in Cyber-Physical Systems, Vol*. 2, No. 2, Lviv Polytechnic National University, 44–53.

5. Glukhov V. S., Elias R. M. (2015). Reducing the structural complexity of multi-section multipliers of elements of Galois fields, *Electrical and computer systems,* No. 19 (95), 222–226.

6. Cherkaskyi M. V., Tkachuk T. I. (2012). Complexity characteristics of multiplication devices, *Radioelectronic and computer systems,* No. 5, 142, 147 p.

7. Hlukhov V., Hlukhova A. Galois field elements multipliers structural complexity evaluation, Proceedings of the 6th International Conference ACSN–2013. Lviv, Ukraine, 2013, 18–19.

8. Glukhov V. S., Trish G. M. Estimation of the structural complexity of multi-section multipliers of elements of Galois fields, *Bulletin of the Lviv Polytechnic National University "Computer systems and networks"*, 2014, Vol. 806, 27–33.

8a. Rahma M., Zholubak I. and Hlukhov V. (2018). Devices for multiplicative inverse calculation in the binary Galois fields, 2018 IEEE 9th International Conference on Dependable Systems, Services and Technologies (DESSERT), Kyiv, 261–264. DOI: 10.1109/DESSERT.2018.8409141.

9. Shalagin S. and Zakharov V. (2021). Implementing the Markov Probability Functions Based on a Set of Polynomials over Galois Field, 2021 International Conference on Information Technology and Nanotechnology (ITNT), Samara, Russian Federation, 1–3. DOI: 10.1109/ITNT52450.2021.9649171.

10. Li Y. (2020). A tile assembly model to calculate point-multiplication on conic curves over finite field GF(2n), 2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Exeter, United Kingdom, 41–48. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00032.

11. Wu K. and Wei G. (2019). Optimized Design of ECC Point Multiplication Algorithm Over GF(2m), 2019 International Conference on Electronic Engineering and Informatics (EEI), Nanjing, China, 420–425. DOI: 10.1109/EEI48997.2019.00097.

12. Ibrahim A. (2019). "Efficient Parallel and Serial Systolic Structures for Multiplication and Squaring Over GF(2m)", in Canadian Journal of Electrical and Computer Engineering, Vol. 42, No. 2, 114–120, Spring 2019. DOI: 10.1109/CJECE.2019.2900087.

13. El-Razouk H. (2022). "Input-Latency Free Versatile Bit-Serial GF(2m) Polynomial Basis Multiplication", in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 30, No. 5, 589–602, May 2022. DOI: 10.1109/TVLSI.2022.3155611.

14. Zhang C., Chen C. and Wu H. (2021). "Area-Efficient Finite Field Multiplication in GF(2n) Using Single-Electron Transistors", 2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS), Penang, Malaysia, 25–28. DOI: 10.1109/APCCAS51387.2021.9687675.

ANALYSIS OF MULTIPLICATION ALGORITHMS IN GALUIS FIELDS FOR THE CRYPTOGRAPHIC PROTECTION OF INFORMATION

Ivan Zholubak[1]

Lviv Polytechnic National University,
Department of Electronic Computing Machines, Lviv, Ukraine
E-mail: Ivan.M.Zholubak@lpnu.ua, ORCID: 0000-0001-8871-7222

The mathematical basis for processing a digital signature is elliptic curves. The processing of the points of an elliptic curve is based on the operations performed in the Galois fields $GF(p^m)$. Fields with a simple foundation are not well-studied and very interesting for research. In this paper, a comparison of the complexity of algorithms for the realization of the multiplication operation in Galois fields $GF(p^m)$ with different bases is carried out. Conducts a comparison of the 3 most common multiplication algorithms. Found that fields with a base greater than 2 will have greater complexity of the algorithm.

Key words: Galois fields $GF(d^m)$; multiplier; modified Guild cell; LUT; cell generator.