УДК 004.056.5

DOI: 10.18372/2073-4751.80.19767

**Верба О.А.**, к.т.н.,
orcid.org/0000-0001-5752-5121,
e-mail: olverba@gmail.com,
**Нікольський С.С.**,
orcid.org/0000-0003-4893-3339,
e-mail: sergiy.nikolskiy@gmail.com

# МЕТОД ЕКСПОНЕНЦІЮВАННЯ НА ПОЛЯХ ГАЛУА ДЛЯ ШВИДКОЇ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНОГО ЗАХИСТУ В IoT

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

## *Вступ*

Інтернет Речей (*Internet of Things – IoT*) створювався як концепція дешевої системи віддаленого управління побутовими пристроями з використанням існуючого середовища для обміну даними – Інтернету [1]. Досягнутий за останні десятиліття динамічний прогрес якісних характеристик Інтернету педалює розширення сфери застосування цієї концепції за рамки управління побутовими пристроями. Рушійною силою цього процесу стали незаперечні переваги використання Інтернету в якості середовища передачі даних: низька вартість за рахунок можливості використання готових інфраструктурних рішень, гнучкість реконфігурації та відсутність обмеження на відстань до об'єкта управління. Відтак, технологія Інтернету Речей нині активно використовується для віддаленого контролю та управління в важливих для забезпечення життєдіяльності галузях таких, зокрема, як медицина, військова справа, транспорт, технологічні процеси в промисловості [1]. З іншого боку, розширення сфер використання *IoT* за рамки побутових пристроїв породило проблему захисту систем управління від зовнішнього втручання з огляду на вразливість Інтернету, як відкритого середовища для обміну інформацією. Іншими словами, якщо при управлінні побутовими пристроями вартість атаки на середовище обміну даними значно перевищувала потенційну вигоду від неї, то при контролі критичними об'єктами в режимі реального часу небезпека стороннього втручання становить реальну загрозу і потребує спеціальних криптографічних методів захисту [2]. Базовим механізмом захисту від зовнішнього втручання в роботу систем управління шляхом фальсифікації даних від віддалених об'єктів або керуючих команд для них виступає цифровий підпис. Існуючий стандарт цифрового підпису (*Digital Signature Standard – DSS*) базується на криптографії з відкритим ключем в основі якої лежить операція модулярного експоненціювання над числами, довжина яких значно перевищує розрядність процесора [2]. Використання довгих чисел зумовлене вимогами до рівня захищеності; для більшості застосувань регламентується використовувати числа довжиною в 4096 біт. Обрахунок модулярної експоненти над числами такої розрядності потребує близько ста мільйонів процесорних операцій. Це створює проблему контролю автентичності команд управління при їх обробці на малопотужних пристроях в режимі реального часу. Існує два основних підходи до вирішення вказаної проблеми. Перший полягає в залученні для обрахунку модулярної експоненти в реальному часі на термінальному мікроконтролері хмарних обчислень [3]. Квінтесенція другого полягає в переході до альтернативних алгебр, в яких обчислення модулярної експоненти над довгими числами здійснюється значно швидше. Зокрема, такі властивості має алгебра кінцевих полів Галуа $GF(2^n)$. Використання першого із зазначених підходів при віддаленому управлінні критично-важливими об'єктами несе в собі

потенційну загрозу затримок та ризику втручання через віддалені хмарні обчислення. Тому більш перспективним, для таких застосувань, вбачається використання другого підходу, тобто алгебри кінцевих полів Галуа. Для реалізації цього потрібно дослідити обчислювальні процеси експоненціювання на кінцевих полях Галуа та розробити ефективні методи їх реалізації.

Таким чином, наукова задача підвищення швидкості обчислення експоненти на кінцевих полях Галуа являє собою актуальну та значиму для практики задачу з огляду на особливості сучасного стану розвитку інформаційних технологій.

### *Аналіз існуючих методів експоненціювання на полях Галуа*

На практиці операція експоненціювання на полях Галуа здійснюється над числами, довжина $n$ яких значно перевищує розрядність $r$ процесора: $n>>r$ і визначається вимогами до рівня захищеності. На сьогодні, з огляду на вимоги безпеки, для більшості застосувань прийнятною є довжина чисел, рівна 4096 [4].

В основі всіх методів обчислення експоненти на кінцевих полях Галуа лежить відповідним чином видозмінений класичний алгоритм модулярного експоненціювання. При цьому традиційна операція додавання змінюється на відповідний аналог поліноміальної алгебри – логічне додавання, або *XOR*, позначене символом «⊕»; аналогічно традиційне множення змінюється на поліноміальне, яке позначається як «⊗» . Змін зазнає й операція модулярної редукції, яка на кінцевих полях змінюється на знаходження остачі від поліноміального ділення на утворюючий поліном поля Галуа та позначається як *rem*. Аналогічно класичному алгоритму модулярного експоненціювання $A^E \ mod \ M$ на полях Галуа застосовують два різновиди: з аналізом розрядів коду експоненти $E = e_{n-1}\cdot 2^{n-1} + e_{n-2}\cdot 2^{n-2} + \ldots + e_1\cdot 2 + e_0$ в напрямку від молодших до старших, та навпаки: від старших до молодших [5].

Перший з них організовано в вигляді $n$ циклів, в яких лічильник $j$ поточного розряду експоненти змінюється від нуля до $n$. Перед початком виконання змінній $D$ присвоюється значення числа $A$: $D = A$. Змінна $R$ , що використовується для формування в ній результату, на початку встановлюється в одиницю: $R = 1$. В кожному з $n$ циклів, поточний результат $R$ множиться на змінну $D$ на полі Галуа: $R = R \otimes D \ rem \ P$, якщо значення поточного розряду експоненти $e_j$ дорівнює одиниці ($e_j = 1$), крім того, змінна $D$ поліноміально підноситься до квадрату: $D = D^2 \ rem \ P$. Після виконання $n$ циклів, в змінній $R$ зберігається результат операції $A^E \ rem \ P$.

Відмінність другого різновиду класичного алгоритму експоненціювання полягає в тому, що в рамках $n$ циклів значення $j$ – поточного розряду експоненти змінюється від $n$ до нуля. Використовується одна змінна – $R$, значення якої перед початком виконання циклів встановлюється в одиницю: $R = 1$. В кожному з $n$ циклів змінна $R$ підноситься до квадрату на кінцевих полях Галуа: $R = R^2 \ rem \ P$, після чого, за умови, що поточний біт експоненти дорівнює одиниці($e_j = 1$), $R$ множиться на число $A$ на полі Галуа: $R = R \otimes A \ rem \ P$. Після виконання $n$ циклів в змінній $R$ фіксується результат: $R = A^E \ rem \ P$.

Оскільки обидва різновиди класичного алгоритму експоненціювання мають строго послідовний характер, зусилля дослідників концентруються на прискоренні виконання його базових операцій: піднесення до квадрату та множення на полях Галуа.

Вказані операції включають в себе мультиплікативну складову: поліноміальне множення або піднесення до квадрату та редукцію, тобто віднаходження залишку від поліноміального ділення добутку чи квадрату на утворюючий поліном $P(x)$ поля Галуа. Ці дві складових можуть виконуватися як послідовно, так і з суміщенням в часі. В силу того, що архітектура сучасних процесорів не підтримує на апаратному рівні операцію поліноміального множення, для її реалізації застосовується побітова обробка розрядів множника [6]. Тобто поліноміальне множення організується у вигляді $n$ циклів, в кожному із яких

здійснюється зсув множимого та його логічне додавання до суми часткових добутків. Оскільки остання операція виконується за умови одиничного значення поточного біту множника, тобто зі ймовірністю 0.5, для поліноміального множення використовується $1.5 \cdot n$ операцій над довгими, $n$-розрядними числами, або $1.5 \cdot n^2/r$ процесорних операцій. Редукція $2 \cdot n$-розрядного поліноміального добутку виконується у вигляді циклу, що повторюється $n$ раз. В кожному такому циклі здійснюється зсув та, зі ймовірністю 0.5, логічне додавання коду утворюючого поліному. З цього випливає, що редукція поліноміального добутку потребує також $1.5 \cdot n^2/r$ процесорних операцій. Це означає, що операція множення на полі Галуа займає в класичному варіанті $3 \cdot n^2/r$ процесорних операцій.

Для прискорення комп'ютерної реалізації операцій множення та піднесення до квадрату довгих чисел на полях Галуа засовуються такі підходи:

Використання властивості операції поліноміального піднесення до квадрату числа $A=a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \ldots + a_1 \cdot 2 + a_0$, $\forall j=0,1,\ldots,n-1$: $a_j \in \{0,1\}$, яка полягає в тому, що $A \otimes A = a_{n-1} \cdot 2^{2 \cdot (n-1)} + a_{n-2} \cdot 2^{2 \cdot (n-2)} + \ldots + a_1 \cdot 2^2 + a_0$. Іншими словами, поліноміальне піднесення до квадрату зводиться до вставки нулів між двійковими розрядами числа A [4]. Ця властивість дозволяє потенційно прискорити виконання фази множення операції піднесення до квадрату на полях Галуа. Проте теоретично ця властивість не впливає на час редукції отриманого поліноміального квадрату.

Використання особливості застосування операції експоненціювання на полях Галуа в реальних системах криптографічного захисту інформації. Ця особливість полягає в тому, що вказана операція є базовою для криптографії з відкритим ключем, в яких ключі можна вважати практично незмінними [7]. Це означає, що код експоненти E та утворюючий поліном $P(x)$ можна вважати практично незмінними, що відкриває широкі можливості для виділення операцій, що залежать лише від вказаних складових і організації їх обчислення лише один раз зі збереженням результату, який може використовуватися багатократно.

Використання властивості описаного вище алгоритму експонентціювання на полях Галуа з аналізом розрядів коду експоненти від старших до молодших. Цей різновид класичного алгоритму передбачає виконання на кожному із $n$ циклів множення постійного числа A на полях Галуа за умови, що поточний біт код*е* експоненти дорівнює нулю. Це створює передумови для ефективного використання передобчислень, які залежать лише від A для прискорення реалізації зазначеної вище операції множення на полі Галуа [8].

Конкретну схему використання першої з наведений вище можливості реалізовано в рамках методу [9]. Прискорення експоненціювання на полях Галуа досягається за рахунок з рахунок організації суміщення операції поліноміального піднесення до квадрата з редукцією Монтгомері, що дозволило організувати одночасну обробку двох розрядів числа, до підноситься до квадрату і, таким чином, вдвічі зменшити час виконання цієї операції.

Ефективну схему використання незмінности утворюючого поліному поля Галуа засновано в рамках методу [7]. Для прискорення експоненціювання на полях Галуа запропоновано групову редукцію Монтгомері, яка за рахунок використання передобчислень, що залежать лише від утворюючого поліному поля дозволяє значно прискорити реалізацію фази редукції поліноміального добутку.

Наведені рішення використовують редукцію Монгомері [10], яка більш ефективна в традиційній модулярній арифметиці, і, фактично, може розгляддатися як редукція з молодших розрядів. За відсутності переносів, що характерно для операцій на алгебрі полів Галуа, значимість переваг редукції Монтгомері значною мірою нівелюються. Натомість, більш значимим стає такий недолік технології Монтгомері, як необхідність в корекції отриманого результату [11]. По аналогії з груповою редукцією Монтгомері, цілком можливим є

застосування технологій обнуління групи старших розрядів при використанні групової редукції зі старших розрядів.

Ще одна альтернативна технологія редукції на полях Галуа полягає в використанні таблиць передобчислень [12]. Це рішення дозволяє виконати обчислення квадрату з використанням теоретично найменшої кількості операцій логічного додавання довгих чисел: $n/2$. Недолік такого рішення полягає в необхідності використання значного за обсягом об'єму пам'яті для зберігання таблиці передобчислень, що може бути критичним для малопотужних термінальних мікроконтролерах.

В роботі [12] запропоновано підхід до прискорення виконання мультиплікативних операцій на полях Галуа шляхом використання універсального модуля, який обчислює $(a \otimes b)$ *rem P*. Швидка реалізація модуля базується на застосуванні властивостей базису Гребнера стосовно комунікативної алгебри. Запропоноване рішення орієнтоване на апаратну реалізацію експоненціювання на кінцевих полях з використанням програмованих матриць. В цілому, апаратні реалізації мультиплікативних операцій на полях Галуа на 1-2 порядки ефективніші в порівнянні з тими, що базуються на традиційній модулярній арифметиці [13].

Проведений аналіз показав наявність певних резервів та невикористаних можливостей для подальшого прискорення важливої для криптографічних застосувань операції експоненціювання на кінцевих полях Галуа. Мова, зокрема йде про можливість суміщення операцій піднесення до квадрату та модулярного множення, а також про застосування групової редукції зі старших розрядів, що дозволяє обходитися без корекції результату.

### *Мета досліджень*

Мета досліджень полягає в прискоренні комп'ютерної реалізації на термінальних мікроконтролерах експоненціювання на полях Галуа, яке виконується над числами, довжина яких значно перевищує розрядність процесора, за рахунок суміщення в часі виконання операцій піднесення до квадрату та множення на постійне число, а також застосування групової редукції на полях Галуа з використанням таблиць передобчислень.

### *Організація суміщення операцій піднесення до квадрату та множення на постійне число на кінцевих полях Галуа $GF(2^n)$*

Для досягнення поставленої мети пропонуєтьмя метод експоненціювання на кінцевих полях Галуа $GF(2^n)$, який базується на аналізі розрядів коду експоненти в напрямку від старших до молодших.

Використання особливості операції поліноміального піднесення до квадрату в алгоритмі експоненціювання з старших розрядів дозволяє ефективно поєднати цю операцію з множенням на постійне число. Для реалізації такої можливості пропонується використання таблиці $T$ передобчислень, що залежить лише від утворюючого поліному $P$ та числа $A$ над яким виконується операція експоненціювання. Створення такої таблиці здійснюється перед початком обрахунку поліноміальної експоненти. Для побудови таблиці $T$ передобчислень запропонована процедура, яка зводиться до наступної послідовності дій:

1. Лічильник $j$ номеру рядка таблиці $T$ встановлюється в нуль: $j = 0$. В відповідну $j$ комірку $T[0]$ записується значення числа $A$: $T[0] = A$.
2. Здійснюється інкремент лічильника $j$: $j = j + 1$. Якщо значення $j$ дорівнює $n$: $j = n$, виконується перехід на п.4
3. В поточний рядок таблиці $T[j]$ записується значення залишку від поліноміального ділення зсунутого ліворуч на 2 розряди коду попереднього рядка $T[j\text{-}1]$ на утворюючий поліном $P$ кінцевого поля Галуа $GF(2^n)$: $T[j] = (T[j\text{-}1] \ll 2)$ *rem P*. Здійснюється повернення на повторне виконання п.2.
4. Кінець процедури.

Робота розробленої процедури побудови таблиці передобчислень ілюструється наступним прикладом. Нехай утворюючий поліном $P(x)$ поля Галуа $GF(2^n)$ має наступний вигляд: $P(x) = x^6+x^4+x^3+x+1$. Цьому поліному відповідає число $P =$

$1011011_2 = 91$. Відповідно $n = 6$. Нехай обчислюється експонента $47^{25}$ *rem* 91, тобто значення $A = 47$.

У відповідності до п.1, описаної вище процедури, значення лічильника $j$ встановлюється в нуль: $j = 0$, а в нульову комірку таблиці $T[0]$ записується значення числа $A$: $T[0] = 47$. Наступним пунктом п.2 збільшується значення $j$: $j = j+1 = 0+1 = 1$. В межах наступного п.3 обчислюється значення: $T[1] = (T[0]<<2)$ *rem* $P = 47<<2$ *rem* $91 = 10$ та здійснюється повернення на повторне виконання п.2. В подальшому, при наступних значеннях $j$ процедура функціонує аналогічним чином. Сформовані результати передобчислень для розглянутого прикладу представлені в табл. 1.

*Таблиця 1*. Результати передобчислень для P = 91 і A=46.

| $j$ | $T[j]$ | $j$ | $T[j]$ |
|---|---|---|---|
| 0 | 47 | 3 | 22 |
| 1 | 10 | 4 | 3 |
| 2 | 40 | 5 | 12 |

Для суміщення виконання операцій піднесення до квадрату та множення на постійне число на кінцевих полях Галуа з використанням побудованої таблиці передобчислень запропоновано спеціальну процедуру. Тобто процедура обчислює значення $R = R^2 \cdot A$ *rem* $P$, таким чином в якості вхідних параметрів виступають значення $R$, $A$ та $P$, а результатом операції стає нове значення $R$. В формалізованому вигляді описана процедура зводиться до наступної послідовності дій:

1. Поточне значення $R$ записується в допоміжну змінну $Q$ де $Q = q_0 + q_1 \cdot 2 + q_2 \cdot 2^2 + \ldots + q_{n-1} \cdot 2^{n-1}$, $\forall\ j \in \{0,1,..,n-1\}$: $q_n \in \{0,1\}$: $Q = R$. Змінна $R$ обнуляється: $R = 0$. Лічильник $j$ поточного розряду $Q$ встановлюється в нуль: $j = 0$.

2. Якщо поточний розряд $j$ змінної $Q$ дорівнює одиниці: $q_j = 1$, то до результату $R$ логічно додається значення таблиці $T[j]$: $R = R \oplus T[j]$.

3. Здійснюється інкремент лічильника: $j = j + 1$. Якщо значення $j$ менше $n$, виконується перехід на п.2.

4. Кінець. Результат $R = R^2 \cdot A$ *rem* $P$.

Робота запропонованої процедури суміщених операцій множення на постійне число та піднесення до квадрату можна проілюструвати наступним прикладом. Нехай вхідний параметр $R = 11101_2 = 29$, а значення $A = 47$ та $P = 91$ відповідно залишаються незмінними.

Згідно з п.1 запропонованої процедури значення допоміжної змінної $Q$ дорівнює $R$: $Q=R=13$. Змінна $R$, а також лічильник $j$ поточного розряду $Q$ встановлюються в нуль: $R = 0$, $j = 0$. В рамках п.2 поточний розряд $Q$ дорівнює одиниці: $q_j = 1$, тому до результату $R$ логічно додається значення таблиці $T[0]$: $R = R \oplus T[0] = 0 \oplus 47 = 47$. Відповідно до п.3 до лічильника поточних розрядів додається одиниця: $j = j+1 = 0+1=1$. Оскільки значення $j$ менше за значення $n$: $1 < 6$, то здійснюється перехід на п.2. Згідно з п.2 поточний розряд $Q$ дорівнює нулю, тому відразу здійснюється перехід на п.3, в рамках якого до значення $j$ додається одиниця: $j = j+1= 1+1=2$. Оскільки змінна $j$ менша за $n$: $2 < 6$, то здійснюється перехід на п.2. Відповідно до п.2 значення $q_j$ дорівнює одиниці, відповідно до поточного результату $R$ логічно додається значення таблиці $T[2]$: $R = R \oplus T[2] = 47 \oplus 40 = 7$. В рамках п.3 обчислюється інкремент змінної $j$: $j = j+1=2+1 =3$. Оскільки значення $j$ менше $n$: $3 < 6$ відбувається перехід на п.2. В рамках п.2 значення поточного розряду $Q$ дорівнює одиниці: $q_j = 1$, тому до результату $R$ логічно додається значення таблиці $T[3]$: $R = R \oplus T[0] = 7 \oplus 22 = 17$. Згідно п.3 до значення $j$ додається одиниця: $j = j+1= 3+1= 4$ та здійснюється перехід на п.2, оскільки $j$ менше $n$: $4 < 6$. В рамках п.2 значення поточного розряду дорівнює одиниці: $q_j = 1$, тому до результату $R$ логічно додається значення таблиці $T[4]$: $R = R \oplus T[4] = 17 \oplus 3 = 18$. Відповідно до п.3 значення $j$ інкрементується: $j = j+1 = 4+1 = 5$, та, оскільки $j$ менше $n$: $5 < 6$ здійснюється перехід на п.2. В рамках п.2 поточний розряд $q_j$ дорівнює нулю, тому відразу перехід на п.3, в рамках якого значення змінної $j$ збільшується на одиницю: $j = j +1= 5+1 =6$. Оскільки значення

$j$ дорівнює $n$: $j = n = 6$ процедура суміщеного множення та піднесення до квадрату на кінцевих полях Галуа закінчується. В змінній $R$ зберігається результат: $R = R^2 \cdot A \; rem \; P = 29^2 \cdot 47 \; rem \; 91 = 18$.

### *Організація швидкого обчислення квадрату на полях Галуа*

Базова ідея методу пролягає в тому, що поточний результат в кожному циклі зсувається на 2 позиції праворуч. При цьому, для того, щоб не була втрачена інформація, до поточного результату додається код, який обнуляє два старших розряди.

До початку піднесення до квадрату на полі Галуа створюються дві допоміжні змінні: $G = (P<<1) \oplus p_{n-1}P$ та $U = (P<<1) \oplus (p_{n-1} \oplus 1)P$. Наприклад, якщо P = 91 = 1011011, то $G = (P<<1) \oplus p_{n-1}P$ = 10110110, а $U = (P<<1) \oplus (p_{n-1} \oplus 1)P$ = 11101101.

Цілком очевидно, що зазначені змінні залежать лише від утворюючого поліному P поля Галуа, а значить, обчислюватися лише один раз і зберігатися в постійній пам'яті термінального мікроконтролера.

Розроблена процедура швидкого піднесення до квадрату на полях Галуа в формалізованому вигляді зводиться до виконання наступної послідовності операцій:

1. Індексу $j$ поточного біту числа A, що підноситься до квадрату на полі Галуа присвоюється значення $n$: $j=n-1$; $(n+2)$-розрядний код $R$ результату встановлюється в нуль: $R=0$.

2. До старшого біту поточного результату $R$ логічно додається поточний біт $a_j$ числа, що підноситься до квадрату $A$: $R=R \oplus a_j \cdot 2^{n+2}$.

3. Якщо два старші розряди коду результату $R$ дорівнюють 01, тобто $r_{n+2}=0$ і $r_{n+1}=1$, то до коду $R$ логічно додається $P$ : $R = R \oplus P$ . Перехід на п. 6.

4. Якщо два старші розряди результату утворюють код 10, тобто $r_{n+2}=1$ та $r_{n+1}=0$, то код $R$ змінюється логічним додаванням до нього коду $G$: $R = R \oplus G$. Перехід на п. 6.

5. Якщо два старші розряди результату $R$ утворюють код 11, тобто $r_{n+2}=1$ і $r_{n+1}=1$, то до коду $R$ логічно додається код $U$: $R = R \oplus U$.

6. Виконується зсув $R$ на два розряди ліворуч: $R = R<<2$; індекс $j$ зменшується на одиницю: $j = j-1$, якщо $j>n/2$ здійснюється повернення на повторне виконання п. 2.

7. Здійснюється зсув коду $R$ на дві позиції праворуч: $R = R>>2$. Індекс $q$ встановлюється рівним $n$-1: $q = n-1$.

8. До $q$-го біту поточного результату $R$ логічно додається $j$-тий розряд числа $A$: $R=R \oplus a_j \cdot 2^q$.

9. Здійснюється інкремент індексу $j$: $j=j+1$, а індекс $q$ зменшується на два: $q=q-2$. Якщо $j>0$ виконується перехід на повторне виконання п.8.

10. Кінець. Результат в $R$: $R=A^2 \; rem \; P$.

Функціонування запропонованої процедури швидкого піднесення до квадрату на полях Галуа може бути ілюстроване прикладом обчислення $53^2 \; rem \; 91 = 23 = 10111_2$. В цьому прикладі $A = 53 = 110101_2$, а код утворюючого поліному $P = 91 = 1011011_2$, $n = 6$.

В рамках виконання п.1 індексу $j$ присвоюється значення $n$: $j= 6$. Код $R$ поточного результату довжиною $n+2 = 8$ бітів встановлюється в нуль: $R=0$.

У відповідності з п.2, оскільки $a_j = 1$ здійснюється інвертування старшого, 8-го розряду коду $R$, відповідно цей код стає рівним $R=10000000_2$. Оскільки два старших розряди $R$ утворюють код 10, то відповідно до п. 4 процедури, нове значення поточного результату формується в результаті логічного додавання до нього коду $G$=10110110: $R$ = 10000000⊕10110110 = 00110110. Наступним п.6 процедури код $R$ зсувається ліворуч на дві позиції: $R = R<<2$ = 11011000. Індекс $j$ зменшується на одиницю: $j$=6-1 = 5. Оскільки $j>n/2 = 3$, то здійснюється перехід на повторне виконання п.2 запропонованої процедури.

Згідно з п.2 одиничний 5-й біт числа A інвертує старший розряд поточного результату $R$: $R$ = 11011000⊕10000000 =

01011000. Оскільки два старших розряди $R$ утворюють код 01, то відповідно до п. 3 процедури, нове значення поточного результату формується в результаті логічного додавання до нього коду $P$ = 1011011: $R$ = 01011000⊕01011011 = 00000011. Далі, в рамках реалізації п.6 алгоритму код $R$ зсувається на дві позиції ліворуч: $R$ = 00000011<<2 = 00001100. Індекс $j$ декрементується: $j$ = 5-1 = 4. Оскільки $j$> $n$/2, то здійснюється перехід на повторне виконання п.2 процедури.

Оскільки 4-й біт числа A дорівнює нулю, виконання п.2. не змінює поточного коду $R$ результату. В силу того, що його два старших розряди дорівнюють нулю, наступним виконується п.6 процедури, яким код $R$ зсувається на два розряди ліворуч: $R$ = 00001100<<2 = 00110000.

Оскільки $j$ = $n$/2 = 3, то в рамках виконання наступного п.7 запропонованої процедури код $R$ зсувається на три розряди праворуч: $R$ = $R$>>3 = 000110, індекс $q$ встановлюється рівним $n$-1 = 5: $q$ = $n$-1 = 5. В ході виконання наступного п.8 процедури до 5-го біту поточного результату $R$ логічно додається рівний одиниці 3-й біт числа A: $R$ = 000110⊕10000 = 010110. В рамках реалізації п.9 індекс $j$ зменшується на одиницю: $j$ = 3-1 = 2, а індекс $q$ – на два: $q$ = $q$-2 = 5-2 = 3. В силу того, що поточне значення $j$>0, здійснюється повернення на повторне виконання п.8, в якому значення $R$ не зазнає змін, так як поточний, 2-й біт числа A дорівнює нулю: $a_2$ = 0.

Після цього виконується п.9, яким значення індексу $j$ декрементується: $j$ = 2-1 = 1, а значення індексу $q$ зменшується на два: $q$ = $q$-2 = 1. Так як обчислене значення $j$>0, наступним виконується п.8, в якому до молодшого біту $R$ логічно додається рівний одиниці молодший біт числа A: $R$ = $R$⊕1 = 10110⊕1 = $10111_2$=23.

Запропонована процедура складається з двох циклів, кожен з яких повторюється $n$/2 раз. При виконанні кожної ітерації першого циклу в середньому виконується 1.75 логічних операцій над довгими числами (з ймовірністю 0.75 здійснюється логічне додавання до поточного результату $R$ кодів $P$, $G$ або $U$ та завжди здійснюється операція зсуву на два розряди ліворуч). При виконанні другого циклу операцій над довгими числами не використовуються, тобто загальна кількість операцій над довгими числами при реалізації запропонованої організації піднесення до квадрату на полях Галуа оцінюється величиною $1.75 \cdot n/2 = 0.86 \cdot n$.

### *Метод швидкого експоненціювання на полях Галуа*

Для швидкого обчислення експоненти $A^E$ *rem* $P$ на кінцевих полях з використанням запропонованих вище способів піднесення до квадрату та суміщеного множення на постійне число з піднесенням до квадрату запропоновано процедуру. В якості вхідних параметрів процедури виступають значення $A$, $E$ та $P$, де $E = e_0 + e_1 \cdot 2 + e_2 \cdot 2^2 + \ldots + e_{n-1} \cdot 2^{n-1}$, $\forall\ j \in \{0,1,..,n\text{-}1\}$: $e_n \in \{0,1\}$.Запропонована процедура експоненціювання на полях Галуа зводиться до наступної послідовності дій:

1. Формується таблиця передобчислень $T$ для заданих значень $P$ та $A$.
2. Лічильнику $j$ поточного розряду експоненти $E$ присвоюється значення $n$-1. Змінна для збереження результату $R$ встановлюється в одиницю: $R$ = 1.
3. Якщо поточний розряд $j$ експоненти $E$ дорівнює одиниці: $e_j$ = 1, в змінну результату обчислюється значення $R = R^2 \cdot A$ *rem* $P$ з використанням описаної вище процедури суміщення операцій. В іншому випадку($e_j$ = 0) обчислюється значення $R = R^2$ *rem* $P$ за допомогою запропонованої процедури піднесення до квадрату на кінцевих полях Галуа.
4. Лічильник $j$ зменшується на одиницю: $j$ = $j$-1. Якщо змінна $j$ більша чи дорівнює нулю: $j$ >= 0, здійснюється перехід на п.3.
5. Кінець. $R = A^E$ *rem* $P$.

Роботу розробленого методу швидкого експоненціювання на кінцевих полях Галуа можна проілюструвати наступним прикладом. Нехай обчислюється експонента $47^{25}$ *rem* 91. В якості вхідних параметрів виступають значення $A$ = 47, $P$ = 91 та $E$ = 25 = $11001_2$.

В рамках п.1 формується таблиця передобчислення для значень $A = 47$ та $P = 91$. Відповідно до п.2 лічильнику $j$ присвоюється значення $n$-1: $j = n\text{-}1 = 6\text{-}1 = 5$. Змінна для збереження результату $R$ встановлюється в одиницю: $R = 1$. Згідно з п.3 поточний біт експоненти дорівнює нулю: $e_5 = 0$, в такому випадку змінна результату $R$ підноситься до квадрата за допомогою розробленої процедури: $R = R^2 \; rem \; P = 1$. Відповідно до п.4 лічильник $j$ зменшується на одиницю: $j = j\text{-}1 = 5\text{-}1 = 4$ після чого здійснюється перехід на п.3. Оскільки в рамках п.3 значення поточного біту експоненти дорівнює одиниці $e_4 = 1$, то в змінну результату обчислюється значення $R = R^2 \cdot A \; rem \; P$: $R = 1^2 \cdot 47 \; rem \; 91 = 47$. В рамках п.4 від значення змінної $j$ віднімається одиниця: $j = j\text{-}1 = 4\text{-}1 = 3$, та здійснюється перехід на п.3. Відповідно до п.3 поточний розряд експоненти $e_3$ дорівнює одиниці: $e_3 = 1$, тому в змінну $R$ обчислюється значення $R = R^2 \cdot A \; rem \; P$: $R = 47^2 \cdot 47 \; rem \; 91 = 23$.Згідно з п.4 від значення змінної $j$ віднімається одиниця: $j = j\text{-}1 = 3\text{-}1 = 2$ та здійснюється перехід на п.3. В рамках п.3 поточний розряд експоненти дорівнює нулю: $e_2 = 0$, тому значення $R$ підноситься до квадрату: $R = R^2 \; rem \; P = 23^2 \; rem \; P = 34$. Відповідно до п.4 змінна $j$ декрементується: $j = j\text{-}1 = 2\text{-}1 = 1$ та відбувається перехід на п.3.

В рамках п.3 поточний розряд експоненти дорівнює нулю: $e_1 = 0$, в такому випадку, значення змінної поточного результату підноситься до квадрата: $R = R^2 \; rem \; P = 34^2 \; rem \; P = 53$. Відповідно до п.4 значення змінної поточного дозряду зменшується на одиницю: $j = j\text{-}1 = 1\text{-}1 = 0$ та здійснюється перехід на п.3. В рамках п.3 поточний розряд експоненти дорівнює одиниці: $e_0 = 1$, відповідно результат $R$ обчислюється як: $R = R^2 \cdot A \; rem \; P$: $R = 53^2 \cdot 47 \; rem \; 91 = 8$. Згідно з п.4 від значення $j$ віднімається одиниця: $j = j\text{-}1 = 0\text{-}1 = -1$. Оскільки значення поточного розряду експоненти менше нуля процедура експоненціювання на кінцевих полях закінчується.

### *Оцінка ефективності*

Виходячи з поставлених цілей, оцінку ефективності запропонованого методу експоненціювання на полях Галуа доцільно проводити за такими критеріями:

1. Прискорення виконання цієї операції в порівнянні з відомими методами її реалізації. Кількісна оцінка цього може бути здійснена через коефіцієнт β прискорення, який визначається співвідношенням часу реалізації експоненціювання на кінцевих полях Галуа з використанням відомих методів до аналогічної характеристики запропонованого методу.

2. Об’ємом додаткових ресурсів, які використовуються в запропонованому методі задля прискорення експоненціювання на кінцевих полях Галуа.

Запропонований метод експонентціювання на кінцевих полях Галуа, як і всі відомі, базуються на логічних операціях, виконання яких на будь-яких процесорах займає приблизно однаковий час. З огляду на це, постає доцільним виконувати порівняння часу виконання експоненціювання через кількість задіяних для цього логічних процесорних операцій.

В розробленому методі експонентціювання на кінцевих полях Галуа організовано в вигляді $n$ циклів, в кожному з яких аналізується один розряд експоненти, починаючи з старших. З ймовірністю в 0,5 її поточний розряд дорівнює одиниці: відповідно виконується запропонована процедура суміщеного піднесення до квадрату та множення на постійне число А на полі Галуа з утворюючим поліномом $P(x)$: $R = R \otimes R \otimes A \; rem \; P$. В свою чергу, ця процедура організована в вигляді $n$ циклів, в кожному з яких аналізується один розряд поточного значення $R$: якщо він дорівнює одиниці, до результату логічно додається табличне значення, що залежить від постійного числа А. Відповідно, якщо вважати, що розряди $R$ з рівною ймовірністю набувають значення нуля чи одиниці, то середня кількість операцій над довгими, тобто $n$-розрядними числами, в рамках запропонованої процедури, становить $0{,}5 \cdot n$. Відповідно, середня кількість

процесорних операцій складає $0{,}5 \cdot n^2/r$, де $r$ – розрядність процесора.

Якщо поточний біт коду експоненти дорівнює нулю, то реалізується запропонована процедура суміщеного поліноміального піднесення до квадрату та редукції на полі Галуа: $R = R \otimes R \ rem \ P$. Вище було показано, що загальна кількість операцій над довгими числами при реалізації запропонованої організації піднесення до квадрату на полях Галуа оцінюється величиною $1.75 \cdot n/2 = 0.86 \cdot n$. В перерахунку на процесорні операції це становить $0.86 \cdot n^2/r$. Таким чином, загальна середня кількість $N$ процесорних операцій для експоненціювання на полі Галуа за запропонованим методом становить:

$$N = n \cdot \left(\frac{0.5 \cdot n^2}{2 \cdot r} + \frac{0.86 \cdot n^2}{2 \cdot r}\right) = 0.68 \cdot \frac{n^3}{r}. \quad (1)$$

Найбільш швидкий з відомих методів піднесення до квадрату на полях Галуа має практично ідентичні часові характеристики і відрізняється від запропонованого лише технологією редукції – в ньому вона виконується з молодших розрядів за технологією Монтгомері [14]. Проте множення на постійне число на полі Галуа виконується в ньому за традиційною схемою, тобто потребує $4.5 \cdot n^2/r$ процесорних операцій. Відповідно, операція експоненціювання на полі Галуа потребує, в середньому $n$ операцій піднесення до квадрату та $n/2$ операцій множення на полях Галуа. Це дає в сумі значення кількості $N_0$ процесорних операцій, яке визначається у вигляді: $N_0 = n \cdot 0.86 \cdot n^2/r + 0.5 \cdot n \cdot 4.5 \cdot n^2/r = 3.11 \cdot n^3/r$. Відповідно, значення коефіцієнту $\beta$ прискорення виконання операції експоненціювання на полях Галуа, яке забезпечує запропонований метод у порівнянні з відомим визначається формулою:

$$\beta = \frac{N_0}{N} = \frac{3.11}{0.68} = 4.57$$

Проведені експериментальні дослідження з використанням спеціально розробленої програми показали практично ідентичні наведеним теоретичним результати часової ефективності.

Якщо порівнювати часові характеристики запропонованого методу з тими, в яких прискорюється множення на постійне число за рахунок передобчислень [8], то вказана операція в них реалізується з використанням $0.5 \cdot n^2/r$ процесорних операцій. Якщо використовувати при цьому відому схему прискореного піднесення до квадрату на полі Галуа, то загальна середня кількість $N_1$ процесорних операцій для реалізації експонентціювання на полі Галуа складає: $N_1 = n \cdot 0.86 \cdot n^2/r + 0.5 \cdot n \cdot 0.5 \cdot n^2/r = 1.11 \cdot n^3/r$. В цьому варіанті значення $\beta_1 = N_1/N = 1.63$. Це означає, що запропонований метод дозволяє отримати виграш у часі реалізації експоненціювання на полях Галуа навіть в порівнянні з варіантом, коли застосовуються всі відомі методи прискорення виконання її складових: операції піднесення до квадрату та множення на постійне число.

Новим елементом, який забезпечує високу часову ефективність запропонованого методу в порівнянні з відомими є організація суміщення в часі виконання двох операцій: піднесення до квадрату та множення на постійне число на полі Галуа.

Для реалізації такого суміщення використовуються результати передобчислень, які залежать тільки від числа A та коду P утворюючого поліному поля Галуа. Для зберігання $n$ $n$-розрядних кодів $2 \cdot A \ rem \ P$, $4 \cdot A \ rem \ P$,…, $2^{n-1} \cdot A \ rem \ P$ в таблиці передобчислень потрібна пам'ять, об'ємом $n^2$ бітів. зокрема, при $n = 4096 = 2^{12}$ об'єм пам'яті, необхідний для зберігання таблиці передобчислень становить 2 Мбайти, який є критичним для більшості сучасних термінальних мікроконтролерів.

### *Висновки*

В результаті проведених теоретикних та експериментальних досліджень, направлених на підвищення швидкості програмної реалізації важливої для криптографічних застосувань операції експоненціювання на кінцевих полях Галуа отримані наступні результати.

Теоретично обґрунтовано, розроблено та досліджено метод прискореного обчислення експоненти на полях Галуа,

якій відрізняється від відомих тим, що організується суміщення в часі виконання двох мультиплікативних операцій на полі Галуа – піднесення до квадрату та множення на постійне число, за рахунок чого забезпечується прискорення виконання цієї важливої для криптографічних застосувань операції.

Теоретично доведено та експериментально підтверджено, що запропонований метод дозволяє в 4.5 рази прискорити комп'ютерну реалізацію експоненціювання на полях Галуа в порівнянні з відомими методами, які направлені на пришвидшення окремо піднесення до квадрату та множення на постійне число.

Виконана розробка орієнтована для швидкої реалізації складних протоколів криптографії з відкритим ключем і, насамперед, цифрового підпису на малопотужних термінальних комп'ютерних платформах систем моніторингу стану і управляння віддаленими об'єктами з використанням технологій IoT.

### *Література*


1. Elgazzar K. et al. Revisiting the internet of things: New trends, opportunities and grand challenges. *Frontiers in the Internet of Things*. 2022. № 1. P. 1–18. DOI: 10.3389/friot.2022.1073780.

2. Unal D., Ali-Ali A., Catak F. O. A secure and efficient Internet of Things cloud encryption scheme with forensics investigation compatibility based on identity-based encryption. *Future Generation Computer Systems*. 2021. Vol. 125. P. 433–445. DOI: 10.1016/J.FUTURE.2021.06.050.

3. Borges J. et al. A Secure Cloud Computing Method for Rapid Implementation of Cryptographic Data Protection in IoT. *2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)* : proceedings, Athens, Greece, 13–15 October 2023 / IEEE. 2023. P. 1–4. DOI: 10.1109/DESSERT61349.2023.10416477.

4. Марковский О. П., Аль-Мріят Гассан Абдель Жаліль. Метод прискореного модулярного множення для ефективної реалізації механізмів криптографічного захисту з відкритим ключем. *Адаптивні системи автоматичного управління*. 2024. Том. 1, № 44. С. 142–152. DOI: 10.20535/1560-8956.44.2024.302429.

5. Al-Mrayt Ghassan Abdel Jalil Halil, Markovskyi O., Stupak A. Organization of fast exponentiation on galois fields for cryptographic data protection systems. *Information, Computing and Intelligent systems*. 2022. № 3. P. 17–25. DOI: 10.20535/2708-4930.3.2022.265480.

6. Жолубак І. Аналіз алгоритмів множення в полях Галуа для криптографічного захисту інформації. *Bulletin of the Lviv Polytechnic National University "Information systems and networks"*. 2023. Вип. 13. С. 338–349. DOI: 10.23939/sisn2023.13.338.

7. Марковський О. П., Дайко І. В. Метод швидкого експоненціювання на полях Галуа для систем криптографічного захисту інформації. *Проблеми інформатизації та управління*. 2024. № 1 (77). С. 80–88. DOI: 10.18372/2073-4751.77.18660.

8. Daiko I., Selivanov V. Fast exponential method on Galois fields for cryptographic applications. *2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT)* : proceedings, Athens, Greece, 13–15 October 2023 / IEEE. 2023. P. 1–4. DOI: 10.1109/DESSERT 61349.2023.10416519.

9. Wu K., Wei G. Optimized Design of ECC Point Multiplication Algorithm Over GF($2^m$). *2019 International Conference on Electronic Engineering and Informatics (EEI)* : proceedings, Nanjing, China, 08–10 November 2019 / IEEE. 2019. P. 420–425. DOI: 10.1109/EEI48997.2019.00097.

10. Osadchyy V. The Order of Edwards and Montgomery Curve. *WSEAS Transactions on Mathematics*. 2020. Vol. 19, № 25. P. 253–264.

11. Li Y. A tile assembly model to calculate point-multiplication on conic curves over finite field GF($2^n$). *2020 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications,*

*Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)* : proceedings, Exeter, United Kingdom, 17–19 December 2020 / IEEE. 2020. P. 41–48. DOI: 10.1109/ISPA-BDCloud-SocialCom-SustainCom51426.2020.00032.

12. Fitzpatrick P., Popovici E. M. Algorithm and Architecture for a Galois Fiels multiplicative Arithmetic Processor. *IEEE Trans. on Information Theory*. 2003. Vol. 49, № 12. P. 3303–3307.

13. Zhang C., Chen C., Wu H. Area-Efficient Finite Field Multiplication in GF($2^n$) Using SingleElectron Transistors. *2021 IEEE Asia Pacific Conference on Circuit and Systems (APCCAS)* : proceedings, Penang, Malaysia, 22–26 November 2021 / IEEE. 2021. P. 25–28. DOI: 10.1109/ APCCAS51387.2021.9687675.

14. Elfard S. Justification of Montgomery Modular Reductions. *Advanced Computing*. 2012. № 11. P. 41–45.

**Верба О.А., Нікольський С.С.**

## МЕТОД ЕКСПОНЕНЦІЮВАННЯ НА ПОЛЯХ ГАЛУА ДЛЯ ШВИДКОЇ РЕАЛІЗАЦІЇ КРИПТОГРАФІЧНОГО ЗАХИСТУ В ІОТ

*Запропоновано нову організацію обчислення експоненти на полях Галуа для швидкої реалізації криптографічних механізмів захисту від зовнішнього втручання в роботу систем віддаленого контролю на базі технологій IoT. Зменшення часу обчислень досягнуто за рахунок суміщення двох мультиплікативних операцій на полях Галуа: піднесення до квадрату і множення на постійне число з застосуванням передобчислень. Наведено математичне обґрунтування запропонованого методу та числові приклади, які ілюструють його роботу.*

*Отримані теоретичні та експериментальні оцінки ефективності запропонованого методу прискорення реалізації експоненціювання на полях Галуа: показано, що він дозволяє в 4.5 рази зменшити час відповідних обчислень в порівнянні з відомими методами.*

***Ключові слова**: мультиплікативні операції на полях Галуа; криптографічні алгоритми на основі алгебри полів Галуа; експоненціювання на полях Галуа; редукція Монтгомері.*

**Verba O.A., Nikolskyi S.S.**

## METHOD OF EXPONENTIATION ON GALOIS FIELDS FOR FAST IMPLEMENTATION OF CRYPTOGRAPHIC PROTECTION IN IOT

*A new organization for the exponent on Galois fields computing is proposed for the rapid implementation of cryptographic mechanisms for protection against external interference in the operation of remote control systems based on IoT technologies. The contraction of computation time is achieved by combining two multiplicative operations on Galois fields: squaring and multiplication by a constant number with the use of recalculations. The mathematical justification of the proposed method and numerical examples illustrating its operation are given.*

*Theoretical and experimental estimates of the effectiveness of the proposed method for accelerating the implementation of exponentiation on Galois fields were obtained: it was shown that it allows reducing the time of the corresponding calculations by 4.5 times compared to known methods.*

***Key words**: multiplication operation on Galois fields; cryptographic algorithms based on Galois Fields algebra; Galois Fields exponentiation; Montgomery reduction.*