УДК 004.056.5

DOI: 10.18372/2073-4751.81.20135

Селіванов В.Л., к.т.н.,
orcid.org/0000-0001-8519-6038,
e-mail: v.selivanov2013@gmail.com,
Гуцуляк Н.А.,
orcid.org/0009-0009-0472-9695,
e-mail: nyancatandme0@gmail.com,
Володін В.В.,
orcid.org/0009-0002-6995-0613,
e-mail: valeradom2004@gmail.com

МОДУЛЯРНЕ МНОЖЕННЯ НА ПОСТІЙНЕ ЧИСЛО З СУМІЩЕННЯМ ГРУПОВОЇ ОБРОБКИ РОЗРЯДІВ МНОЖНИКА ТА РЕДУКЦІЇ МОНТҐОМЕРІ

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

Вступ

Криптографія з відкритим ключем лежить в основі багатьох фундаментальних механізмів захисту інформації. В їх число входять: цифровий підпис, обмін ключами, криптографічно строга ідентифікація та несиметричне шифрування.

Базовою обчислювальною операцією криптографії з відкритим ключем являється модулярне експоненціювання $A^E \bmod M$ над числами довжина яких значно перевищує розрядність процесора. Використання чисел такої довжини зумовлене необхідністю забезпечення практичного рівня захищеності, який визначається складністю реалізації задачі факторизації модуля. Тобто, складність цієї задачі повинна бути такою щоб практично унеможливити її комп'ютерну реалізацію, або зробити її недоцільною з огляду на об'єм витрачених ресурсів.

Виходячи з цього, для більшості практичний застосувань рекомендована *NIST* мінімальна n розрядність складає 4096 біт [1]. Враховуючи факт того, що об′єм обчислень необхідний для проведення модулярного експоненціювання має кубічну залежність від розрядності операнд, то при рекомендованій довжині чисел об′єм складає сотні мільйонів процесорних операцій.

Для комп′ютерів великої та середньої потужності проблема швидкого обчислення модулярної експоненти вирішується шляхом застосування криптопроцесорів. Втім, їх використання не вирішує проблему швидкого обчислення модулярної експоненти на термінальних мікроконтролерах(ТМК). Пов'язано це з специфічними вимогами концепції обчислювальних платформ цього класу: дешевизна, компактність та низьке споживання потужності. Тому, практична реалізація криптографії з відкритим ключем на ТМК потребує застосування специфічних підходів до прискорення модулярного експоненціювання.

Серед відомих підходів можна виділити залучення для обчислення модулярної експоненти віддалених потужностей на базі хмарних технологій [2]. Хоч цей підхід дозволяє значно прискорити обчислення експоненти, проте потребує додаткових ресурсів на захист від незаконного доступу до інформації, при її віддаленій обробці.

Ще один підхід до прискорення модулярного експоненціювання полягає в переході до альтернативних алгебр, обчислення в яких виконується простіше відносно традиційної алгебри; на практиці, прикладом такого підходу є перехід до використання кінцевих полів Галуа [3] $GF(2^n)$. На відміну від залучення хмарних технологій, всі обчислення при такому підході проводяться безпосередньо на ТМК, проте

архітектура сучасних процесорів не пристосована до алгебри кінцевих полів Галуа $GF(2^n)$, що вимагає специфічних та складних програм.

Виходячи з цього, найбільш перспективним видається підхід до прискорення модулярного експоненціювання, що полягає в використанні специфічних особливостей цієї операції при її застосуванні в сфері захисту інформації.

Оскільки процедуру модулярного експоненціювання неможливо розпаралелити, зусилля дослідників зосередженні на прискоренні її базових складових: модулярних операціях множення та піднесення до квадрату. Зокрема, в одному з різновидів класичного алгоритму модулярного експоненціювання [4], використовується множення на постійне число, що створює потенційні можливості прискорити цю операцію.

Таким чином, наукова задача прискорення модулярного множення на постійне число, як складової модулярного експоненціювання – базової операції криптографії з відкритим ключем, являється актуальною з огляду на особливості сучасного етапу розвитку інформаційних технологій.

Огляд існуючих підходів до прискорення мультиплікативних операцій модулярної алгебри

Проблема швидкого обчислення модулярної експоненти $A^E \bmod M$, як основної операції криптографії з відкритим ключем, стимулювала інтенсивні дослідження направлені на пошук шляхів прискорення [5] цієї операції.

На практиці, обчислення модулярної експоненти $A^E \bmod M$ проводиться за одним з двох варіантів класичного алгоритму [4] модулярного експоненціювання. Обидва алгоритми передбачають аналіз поточного біту експоненти $E=e_0+e_1\cdot 2+\ldots+e_{n-1}\cdot 2^{n-1}$, $\forall i \in \{0,1,\ldots,n-1\}$: $e_i \in \{0,1\}$,проте в першому варіанті аналіз проводиться з старших розрядів, а в другому з молодших. Ще однією відмінністю є те, що в першому різновиді алгоритму використовується одна змінна R для збереження результату, яка на початку ставиться в одиницю: $R = 1$, в той час, як в другому варіанті використовується дві змінних R та F, стартові значення яких дорівнюють одиниці та A відповідно: $R = 1$, $F = A$. Обидва алгоритми організовані у вигляді n циклів. В варіанті експоненціювання з старших бітів експоненти E, в кожному циклі виконується дві дії: піднесення поточного результату R до квадрату по модулю M: $R = R^2 \bmod M$, та аналіз поточного біту експоненти: якщо він дорівнює одиниці, то значення змінної R множиться на значення A по модулю M: $R = R\cdot A \bmod M$. В варіанті експоненціювання з молодших розрядів також виконується дві дії: аналіз поточного розряду експоненти E, в випадку, якщо він дорівнює одиниці значення поточного результату R множиться на змінну F по модулю M: $R = R\cdot F \bmod M$, після цього значення змінної F підноситься до квадрату по модулю M: $F = F^2 \bmod M$.

Обидва різновиди алгоритму модулярного експоненціювання передбачають виконання $1.5\cdot n$ операцій модулярного множення над довгими числами. Цілком очевидно, що обидва різновиди мають строго послідовний характер, відповідно, не можуть бути розпаралелені [6]. В зв'язку з цим, зусилля науковців, в своїй більшості, покладені на прискорення складових модулярного експоненціювання, а саме модулярних операцій множення [7] та піднесення до квадрату [6]. Обидві операції складаються з двох фаз: власне множення чи піднесення до квадрату та модулярної редукції, яка полягає в віднаходженні остачі від ділення результату операції на модуль M.

Ці дві фази можуть реалізовуватись як послідовно, так і суміщено.

Для прискорення фази реалізації множення застосовуються такі підходи:

- паралельне обчислення операції множення [8].
- використання схем скороченого множення А.Карацуби [7], М.Фюрера [8] та Аль-Мраят [9].

- Одночасна обробка декількох розрядів множника [5].
- Використання передобчислень при множенні на постійне число [10].

Технологічно, фаза множення може виконуватися побітно чи посекційно. При посекційному множенні довжина однієї секції дорівнює розрядності r процесора. Тобто, кількість операцій процесорного множення в середньому складає S^2 , де $S = n/r$ – кількість секцій. При цьому, для формування результату виконується така ж кількість (S^2) процесорних операцій додавання. Основна перевага секційного обчислення добутку полягає в значному скорочені часу виконання цієї фази за рахунок використання вбудованих операцій множення процесору.

При секційному множенні використовуються перші два підходи до прискорення фази операції множення. Зокрема, метод Аль-Мраят [9] дозволяє скоротити кількість операцій процесорного множення вдвічі, тобто до рівня $S^2/2$. Такого результату автору вдалося досягти за рахунок конкретизації методів А.Карацуби [11] та М.Фюрера [12] для криптографічних застосувань.

З іншого боку, посекційне знаходження добутку, відокремлене від редукції, тобто, для реалізації другої фази модулярного множення, потрібно знаходити остачу від ділення на модуль M чисел розрядністю $2 \cdot n$, що вдвічі збільшує витрати часу на редукцію, в порівнянні з побітовою обробкою множника.

За умови використання секційного множення здебільшого застосовують редукцію П. Барретта [13], суть якої полягає в замінні операції ділення на два множення двох довгих чисел.

При побітовій обробці множника найбільш ефективним варіантом є алгоритм модулярного множення Монтґомері[14]. На початку виконання алгоритму змінна R для збереження результату $A \cdot B \bmod M$ встановлюється в нуль: $R = 0$. Виконання модулярного множення передбачає n циклів, в кожному з яких аналізується розряд множника $B=b_0+b_1 \cdot 2+\ldots+b_{n-1} \cdot 2^{n-1}$, $\forall i \in \{0,1,\ldots,n-1\}$: $b_i \in \{0,1\}$ починаючи з молодшого b_0. Якщо поточний біт множника B дорівнює одиниці, то до змінної R, додається значення A: $R = R+A$. Якщо отриманий результат R непарний, то до R додається модуль M: $R = R+M$. Після чого результат R зсувається праворуч на один біт: $R = R>>1$. Після виконання n циклів, якщо поточний результат R більше за значення модуля M, то від R віднімається M: $R = R-M$. В кінці роботи алгоритму в змінній R фіксується значення: $R = A \cdot B \cdot Q \bmod M$, де Q – мультиплікативна інверсія 2^n.

Середня кількість операцій над довгими числами для реалізації модулярного множення Монтґомері складає $2 \cdot n$. Проте, їх кількість може бути скорочена за рахунок одночасної обробки декількох розрядів множника, що досягається застосуванням перед обчислень [5].

Обробка декількох розрядів множника ефективна при організації групової редукції Монтґомері з використанням передобчислень.

Ще один підхід до прискорення обчислення модулярного множення полягає в використання передобчислень залежних від модуля M, оскільки в криптографічних застосуваннях він є частиною відкритого ключа, тому може вважатися практично незмінним [5]. Це відкриває можливість до виокремлення залежних від модуля обчислень, їх попередній обрахунок та подальше використання при кожному обчисленні модулярної експоненти.

Проведений аналіз засвідчив, що відомі методи прискорення операції модулярного множення не повною мірою задовольняють сучасним вимогам що до оперативності реалізації механізмів захисту на основі криптографії з відкритим ключем. Разом з тим, аналіз показав, що існують не використані резерви прискорення операції модулярного множення. Зокрема, теоретично можлива інкапсуляція передобчислення не тільки на основі модуля M, а й на базі постійного числа A при використанні різновиду класичного алгоритму

модулярного експоненціювання з аналізом зі старших розрядів експоненти.

Мета досліджень

Мета дослідження полягає в прискоренні обчислюваної реалізації модулярного множення на постійне число – складової модулярного експоненціювання з старших розрядів, за рахунок використання передобчислень для суміщення групової обробки розрядів множника з редукцією Монтґомері.

Процедура побудови таблиці передобчислень

Для досягнення поставленої мети пропонується метод прискореного модулярного множення $A \cdot B \bmod M$, де A – постійне число. Метод орієнтовано на використання в криптографічних застосуваннях при реалізації модулярного експоненціювання з старших розрядів коду експоненти.

В якості чинника прискорення модулярного множення на постійне число пропонується організація суміщеної обробки k розрядів множника B з використанням передобчислень залежних від числа A, що не змінюється в рамках однієї операції модулярного експоненціювання, та модуля $M = m_0+m_1\cdot 2+\ldots+m_{n-1}\cdot 2^{n-1}$, $\forall i\in\{0,1,\ldots,n-1\}$: $m_i\in\{0,1\}$ та.

В запропонованому методі звернення до таблиці передобчислень здійснюється за k-розрядним кодом $r_0+r_1\cdot 2+\ldots+r_{k-1}\cdot 2^{k-1}$ молодших розрядів поточного результату $R= r_0 + r_1\cdot 2 +\ldots+r_{k+n}\cdot 2^{k+n}$, $\forall l\in\{0,1,\ldots,n+k\}$: $r_l\in\{0,1\}$ та k-розрядним кодом $b_0+b_1\cdot 2+\ldots+b_{k-1}\cdot 2^{k-1}$ молодших розрядів поточного результату $B=b_0+b_1\cdot 2+\ldots+b_{n-1}\cdot 2^{n-1}$, $\forall i\in\{0,1,\ldots,n-1\}$: $b_i\in\{0,1\}$. При цьому, k-розрядний код молодших розрядів R використовується в якості молодших розрядів $2\cdot k$–розрядної адреси рядка таблиці передобчислень, а старші розряди адреси утворюються з k молодших розрядів коду множника B.

Розроблена процедура формування таблиці передобчислень передбачає виконання наступної послідовності дій:

1. Значення допоміжної змінної D встановлюється рівним нулю: D=0. Значення індексу j номеру поточного рядка таблиці встановлюється рівним нулю: j=0. Обчислюється W – значення додаткового коду від $m_0+m_1\cdot 2+\ldots+m_{k-1}\cdot 2^{k-1}$ – k-бітового коду молодших розрядів модуля M : $W= 1+(m_0\oplus 1) + (m_1\oplus 1)\cdot 2+\ldots+(m_{k-1}\oplus 1)\cdot 2^{k-1}$. Перехід на п.3.

2. Значення допоміжної змінної D збільшується на A: $D=D+A$. Якщо обчислене таким чином значення D перевищує модуль, тобто $D \ge M$, то від D віднімається значення модуля M: $D=D-M$.

3. В поточний рядок таблиці передобчислень $T[j]$ записується значення D: $T[j]=D$. Значення індексу l номеру розряду встановлюється рівним нулю: l=0; Значення H встановлюється рівним нулю: $H = h_0+h_1\cdot 2 +\ldots+h_{k-1}\cdot 2^{k-1} = 0$.

4. Якщо l-тий біт n-розрядного коду $D=d_0+d_1\cdot 2+\ldots+d_{n-1}\cdot 2^{n-1}$, $\forall i\in\{0,1,\ldots,n-1\}$: $d_i\in\{0,1\}$, дорівнює одиниці, тобто d_l=1, то до коду $T[j]$ додається значення модуля M, зсунутого на l розрядів ліворуч $T[j] = T[j] + M<<l$. Значення l–того двійкового розряду коду H встановлюється в одиницю: $h_l = 1$.

5. Значення індексу l збільшується на одиницю: $l=l+1$. Якщо l менше за k, тобто $l<k$, то здійснюється перехід на повторне виконання п. 4.

6. Значення номеру j поточного рядка таблиці передобчислень збільшується на одиницю: $j=j+1$. Якщо залишок від ділення j на 2^k дорівнює нулю, тобто $j \bmod 2^k=0$, то за умови, що j менше за $2^{2\cdot k}$, тобто $j<2^{2\cdot k}$, здійснюється перехід на повторне виконання п.2; якщо $j=2^{2\cdot k}$ виконується перехід на п.9.

7. Значення змінної Y обчислюється як сума H та W по модулю 2^k мінус H: $Y = (H+W) \bmod 2^k - H$.

8. Значення j-го рядка $T[j]$ таблиці передобчислень формується як сума значення $T[j-1]$ та добутку Y на модуль M: $T[j]=T[j-1]+Y\cdot M$. Значенню змінної H присвоюється значення змінної Y: $H = Y$. Здійснюється перехід на повторне виконання п.6.

9. Кінець формування таблиці передобчислень.

Розроблена процедура формування таблиці може бути проілюстрована наступним прикладом:

Нехай модуль M утворюється шляхом множення двох простих чисел p =61 та q = 71: $M = p \cdot q$= 4331 = 0001 0000 1110 1011_2. Розрядність модуля дорівнює номеру його старшого одиничного розряду, отже n = 12. Нехай постійне число A дорівнює 1589: A=1589=0110 0011 0101_2, а кількість одночасно оброблюваних розрядів k дорівнює трьом: k = 3.

Згідно п.1 процедури побудови таблиці передобчислень D встановлюється в нуль: D=0, індексу j, що використовується для позначення поточного рядка таблиці T теж присвоюється значення нуля: j=0. В рамках того ж пункту обчислюється значення змінної W як додатковий код від молодших k розрядів модуляM:W=1+(1⊕1)+(1⊕1)·2+…+ (0⊕1)·2^2= 101_2. Здійснюється перехід на п.3, в рамках якого поточному рядку таблиці передобчислень присвоюється значення змінної D: T[0] = 0. В рамках того ж пункту: l = 0, H = 0.

Оскільки поточне значення D дорівнює нулю доцільно одразу перейти на п.6, в рамках якого до поточного значення j додається одиниця: j = 0+1 =1. Згідно з п.7 формується значення змінної Y: $Y = (0+5) \ mod \ 2^3 - 5 = 5$. В відповідності з п.8 номер рядка таблиці T, який відповідає поточному значенню j обчислюється наступним чином: $T[1] = T[0] + 5 \cdot M$ = 0+21655. В рамках цього ж пункту значенню змінної H присвоюється поточне значення Y: $H = Y$ = 5, та здійснюється перехід на п.6.

За таким принципом обчислюються ще шість значень таблиці T передобчислень, після чого на п.6 залишок від ділення j на 2^k дорівнює нулю ($8 mod 2^3 = 0$), та оскільки виконується умова $j<2^{2 \cdot k}$ здійснюється перехід на п.2.

В рамках цього пункту до змінної D додається значення A: D=0+1589= 1589. Згідно п.3 в поточний рядок таблиці записується значення D:T[8] = 1589. Значення l та H встановлюється в нуль: l = 0, H = 0.

В рамках п.4, оскільки поточне значення d_l дорівнює одиниці(d_0=1), то до поточного рядка таблиці T додається значення модуля M: T[8] = T[8] + M = 1589 + 4331 = 5920. В рамках того ж пункту молодший розряд H встановлюється в одиницю: H = 1. Згідно п.5 значення індексу l інкрементується: l = 0+1=1, та оскільки поточне значення l менше k здійснюється перехід на п.4. Оскільки поточний розряд d_l рівний нулю, то доцільно одразу перейти на п.5. В рамках п.5 до індексу l додається одиниця: l= 1+1=2 та здійснюється перехід на п.4. Згідно з п.4 d_2 =1, таким чином до коду $T[j]$ додається значення модуля M, зсунутого на 2 розряди ліворуч:T[8]= 5920+ 4331<<2 =5920+4331·4=23244, значення другого розряду змінної H встановлюється в одиницю: h_2 = 1, тобто H = 5. Згідно п.5 поточне значення індексу l інкрементується: l = 2+1=3.

В результаті послідовного виконання запропонованої процедури результати передобчислень набувають вигляду представлених в таблиці 1.

Об'єм пам'яті необхідний для зберігання таблиці передобчислень визначається як добуток кількості рядків на кількість стовбців цієї таблиці. Кількість рядків складає 2^{2k} біт. Оскільки при множенні розрядність добутку складається з суми розрядності n оброблюваних чисел та значення k, то кількість стовбців дорівнює $n+k$. Таким чином об'єм пам'яті складає $2^{2k} \cdot (n+k)$ біт, або ж $2^{2k-3} \cdot (n+k)$ байт.

Процедура модулярного множення на постійне число

Розроблена процедура модулярного множення Монтґомері $A \cdot B \ mod \ M$, де A – постійне число, з суміщеною обробкою k розрядів множника та редукцією Монтґомері над k розрядами з використанням об'єднаної таблиці T передобчислень, зводиться до виконання наступної послідовності дій.

1. Множник $B=b_0+b_1 \cdot 2+\ldots+b_{n-1} \cdot 2^{n-1}$, $\forall i \in \{0,1,\ldots,n-1\}$: $b_i \in \{0,1\}$ розбивається на s кластерів по k розрядів в кожному: $s = n/k$.

2. Лічильник v поточного номеру кластеру встановлюється в нуль: $v = 0$. Значення поточного результату $R=r_0+r_1\cdot 2+\ldots+r_{n-1}\cdot 2^{n-1}$, $\forall i\in\{0,1,\ldots,n\text{-}1\}$: $r_i\in\{0,1\}$ також встановлюється в нуль: $R=0$.

3. До поточного результату R додається значення рядка таблиці T номер якого відповідає сумі k молодших розрядів множника, зсунутого на k біт ліворуч та k молодших розрядів поточного результату R: $R = R + T[(2^{k-1}\cdot b_{k\cdot v+k-1} +2\cdot b_{k\cdot v+1} + b_{k\cdot v})\cdot 2^k +2^{k-1}\cdot r_{k-1} +2\cdot r_1+r_0]$. Результат R зсувається праворуч на k біт: $R = R>>k$.

4. Лічильник v збільшується на одиницю: $v = v+1$. Якщо лічильник v поточних кластерів не дорівнює s – номеру останнього кластеру: $v \neq s$, то здійснюється перехід на п.3. В іншому випадку – перехід на п.5.

5. Якщо змінна R більша модуля M: $R>M$, то від неї віднімається модуль M: $R = R\text{-}M$.

6. Кінець алгоритму.

Робота запропонованого методу модулярного множення на постійне число з суміщенням обробки розрядів множника з редукцією Монтґомері може бути ілюстрована наступним прикладом.

Нехай обчислюється модулярний добуток $1589\cdot 2222 \ mod\ 4331 = 993$, де постійне число $A = 1589$, множник $B = 2222 = 100010101110_2$, а модуль $M = 4331 = 0001\ 0000\ 1110\ 1011_2$. Розрядність n модуля M дорівнює номеру його старшого одиничного біту, в цьому випадку дванадцяти: n=12. Нехай, кількість розрядів для одночасної обробки дорівнює три: $k = 3$.

Згідно з п.1 процедури множення, кількість s кластерів дорівнює: $s = n/k = 12/3= 4$. У відповідності до п.2 лічильник v номеру поточного кластеру та значення поточного результату R встановлюються в нуль: $v = 0$, $R = 0$.

В рамках п.3, описаної вище процедури модулярного множення, до поточного результату (R=0) додається значення рядка таблиці T, номер якого відповідає сумі трьох молодших розрядів множника $B = 100010101\mathbf{110}_2$, зсунутого на три розряди ліворуч (110000), та k молодших розрядів поточного результату R: R =0+ $T[110000_2] = 35520$; Результат R логічно зсувається праворуч на три розряди: $R = 35520>>3 = 4440$. Відповідно з п.4 значення змінної v інкрементується: $v = v+1=0+1=1$. В рамках цього ж пункту відбувається повернення на п.3, оскільки $v \neq s$, $1 \neq 3$.

На цьому обробка нульового кластеру закінчується. Обробка всіх наступних кластерів, від одного до трьох, здійснюється аналогічним чином.

Динаміка зміни результату R, при обробці кожного з кластерів, представлена в табл. 2.

Таблиця 1. Приклад формування передобчислень T для $A = 1589$ та $M = 4331$

i	$T[i]$	i	$T[i]$	i	$T[i]$	i	$T[i]$
000000	0	010000	$2A+2M$	100000	$4A+4M$	110000	$6A+6M$
000001	$5M$	010001	$2A+7M$	100001	$4A+M$	110001	$6A+3M$
000010	$2M$	010010	$2A+4M$	100010	$4A+6M$	110010	$6A$
000011	$7M$	010011	$2A+M$	100011	$4A+3M$	110011	$6A+5M$
000100	$4M$	010100	$2A+6M$	100100	$4A$	110100	$6A+2M$
000101	M	010101	$2A+3M$	100101	$4A+5M$	110101	$6A+7M$
000110	$6M$	010110	$2A$	100110	$4A+2M$	110110	$6A+4M$
000111	$3M$	010111	$2A+5M$	100111	$4A+7M$	110111	$6A+M$
001000	$A+5M$	011000	$3A+7M$	101000	$5A+M$	111000	$7A+3M$
001001	$A+2M$	011001	$3A+4M$	101001	$5A+6M$	111001	$7A$
001010	$A+7M$	011010	$3A+M$	101010	$5A+3M$	111010	$7A+5M$
001011	$A+4M$	011011	$3A+6M$	101011	$5A$	111011	$7A+2M$
001100	$A+M$	011100	$3A+3M$	101100	$5A+5M$	111100	$7A+7M$
001101	$A+6M$	011101	$3A$	101101	$5A+2M$	111101	$7A+4M$
001110	$A+3M$	011110	$3A+5M$	101110	$5A+7M$	111110	$7A+M$
001111	A	011111	$3A+2M$	101111	$5A+4M$	111111	$7A+6M$

Таблиця 2. Динаміка зміни результату R при обробці кластерів множника

v	$b_2b_1b_0$	$r_2r_1r_0$	R	
			додавання	зсув
0	110	0	$R=T[110000_2]=35520$	4440
1	101	0	$R=4440+T[101000_2]=34040$	4255
2	010	111	$R=4255+T[010111_2]=29088$	3636
3	100	100	$R=3636+T[100100_2]=9992$	1249

Оцінка ефективності

З огляду на поставлену ціль, в якості оцінки ефективності запропонованого методу доцільно розглядати коефіцієнт прискорення β. Цей коефіцієнт визначається як співвідношення часу виконання модулярного множення на постійне число з використанням відомих методів до часу виконання цієї операції запропонованим методом:

$$\beta = \frac{Tв}{Tз}, \qquad (1)$$

де Тв – час виконання модулярного множення з використанням відомих методів та Tз – час виконання цієї операції з використанням запропонованого методу

Зокрема, при використанні класичного модулярного множення Монтґомері [4] час виконання цієї операції над довгими числами складає: Тв= $2n \cdot T_A$, де T_A – тривалість виконання операцій додавання та логічного зсуву над n розрядними числами. В запропонованому методі здійснюється n/k циклів, в кожному з яких виконується дві операції над довгими числами; таким чином час виконання множення складає:

$$Tз = \frac{2 \cdot n}{k} \cdot T_A . \qquad (2)$$

Відповідно коефіцієнт прискорення β обчислюється в наступному вигляді:

$$\beta = \frac{Tв}{Tз} = \frac{2 \cdot n \cdot T_A}{\frac{2 \cdot n \cdot T_A}{k}} = k . \qquad (3)$$

Таким чином, коефіцієнт прискорення β дорівнює k: це означає, що запропонований метод дозволяє досягти прискорення модулярного множення на постійне число в k разів в порівнянні з класичним методом Монтґомері.

При практичному застосуванні, значення k може бути обмежено двома параметрами: часом побудови таблиці передобчислень та необхідним для її зберігання об'ємом пам'яті.

Визначення часу T_{FT} формування таблиці передобчислень може бути здійснене наступним чином. Таблиця заповнюється фрагментами по 2^k рядків. Кількість таких фрагментів становить також 2^k. Для формування значення таблиці першого рядка кожного з фрагментів потрібно виконати пп. 2-4 процедури формування таблиці, тобто реалізувати, в середньому, $1.5+0.5 \cdot k$ операцій додавання n-розрядних чисел. Заповнення всіх наступних 2^k-1 рядків кожного фрагменту зводиться до однієї операції додавання n-розрядних чисел. Тобто, для формування одного фрагменту таблиці потрібно виконати, в середньому $1.5+0.5 \cdot k + 2^k\text{-}1 \approx 2^k + 0.5 \cdot k$ операцій додавання n-розрядних чисел. Таким чином, загальний час T_{FT} формування таблиці передобчислень визначається наступною формулою:

$$T_{FT} = (2^{2 \cdot k} + 2^{k-1} \cdot k) \cdot T_A . \qquad (4)$$

Другий параметр, що обмежує значення k, це об'єм пам'яті таблиці передобчислень. На практиці його значення залежить від об'єму пам'яті ТМК, що на сьогоднішній момент складає близько 2Мб, або ж 2^{21} байтів. Якщо вважати, що максимальне допустиме значення k дорівнює шести ($k = 6$), то при об'ємі пам'яті в 2^{21} байтів об'єм необхідний для зберігання таблиці передобчислень складає $2^{2 \cdot 6\text{-}3} \cdot 2^{12} = 2^{21}$ байт, тобто дорівнює гранично допустимому значенню. Таким чином, в сучасних умовах реальне значення k дорівнює

шести та, враховуючи швидкість росту об'єму пам'яті ТМК, має тенденцію до збільшення.

Час побудови таблиці при значенні $k = 6$, згідно формули(4) складає: $T_{FT} = (2^{2 \cdot k} + 2^{k-1} \cdot k) \cdot T_A = (2^{2 \cdot 6} + 2^{6-1} \cdot 6) \cdot T_A = 4156 \cdot T_A$. Оскільки таблиця використовується для $n/2$ множень, то доцільно розділити час побудови таблиці на всі операції множення. Виходячи з цього, частка часу побудови таблиці, яка приходиться на одну операцію модулярного множення становить: $2 \cdot 4156 \cdot T_A / 4096 = 2.02 \cdot T_A$. Приймаючи до уваги, що кількість операцій додавання та зсувів модулярного множення Монтґомері дорівнює $2 \cdot n = 8192$, приведене число таких операцій для побудови таблиці: 2.02, час формування таблиці практично не впливає на час виконання модулярного множення з використанням таблиці.

В результаті проведених експериментальних досліджень встановлено, що запропонований метод для реальних умов застосування на термінальних обчислюваних платформах систем дистанційного управління на базі *IoT* дозволяє досягти прискорення комп'ютерної реалізації модулярного множення на постійне число в шість раз в порівнянні з класичним множенням Монтґомері.

Висновки

В результаті проведених досліджень, спрямованих на прискорення модулярного експоненціювання – базової операції криптографії з відкритим ключем, теоретично обґрунтовано та розроблено метод прискорення складової цієї операції – модулярного множення на постійне число з використанням передобчислень.

Відмінність запропонованого методу полягає в використанні передобчислень, залежних від постійного множеного та модуля, що забезпечує суміщену обробку групи k розрядів множника з відповідною редукцією Монтґомері, за рахунок чого досягається скорочення часу виконання модулярного множення.

Обрахунок передобчислень здійснюється перед кожною операцією модулярного експоненціювання, що дозволяє зменшити об'єм обчислень для її реалізації.

Теоретично показано та експериментально підтверджено, що використання розробленого методу дозволяє зменшити час виконання модулярного множення на постійне число в k раз в порівнянні з відомими методами.

При чому значення k обмежується об'ємом пам'яті обчислюваної платформи на якій реалізується розроблений метод. Зокрема, при експериментальному тестуванні методу на ТМК з об'ємом пам'яті в 2 Мбайти реально досягнуто прискорення модулярного множення 4096-розрядних чисел в шість раз.

Запропонований метод може бути використаний для швидкої реалізації протоколів захисту даних на базі криптографії з відкритим ключем на термінальних мікроконтролерах систем управляння, які працюють на технології *IoT*.

Література

1. Русанова О. В., Гайдукевич М. А. Метод розподіленого модулярного експоненціювання на термінальних мікроконтролерах IoT із захищеним залученням хмарних технологій. *Проблеми автоматизації та управління*. 2024. Т. 2, № 76. С. 91–103. DOI: 10.18372/2073-4751.78.18966

2. Haidukevych O. A. et al. 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.

3. Daiko I., Selivanov V. Fast exponential method on Galois field 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/DESSERT61349.2023.10416519.

4. Menezes A., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. CRC Press, 2001. 780 p.

5. Селіванов В. Л., Аль-Мріят Гассан Абдель Жаліль. Метод модулярного множення на постійне число для швидкої реалізації криптографії з відкритим ключем в IoT. *Проблеми автоматизації та управління*. 2024. Вип. 3(79). С. 50–58. DOI: 10.18372/2073-4751.79.19371.

6. Markovskyi O. et al. An Accelerate Approach for Public Key Cryptography Implementation on IoT Terminal Platforms. *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. 10416516.

7. Buhrow B., Gilbert B., Haider C. Parallel modular multiplication using 512-bit advanced vector instructions: RSA fault-injection countermeasure via interleaved parallel multiplication. *Journal of Cryptographic Engineering*. 2021. No. 2. P. 46–55. DOI: /10.1007/s13389-021-00256-9.

8. Марковський О. П., Аль-Мріят Гассан Абдель Жаліль. Метод прискорення модулярного множення для механізмів криптографічного захисту з відкритим ключем. *Проблеми управління та інформатизації*. 2023. Вип. 4(76). С. 48–58. DOI: 10.18372/2073-4751.76.18240.

9. 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. Т. 1, № 44. С. 142–152. DOI: 10.20535/1560-8956.44.2024.302429.

10. Boiarshyn I., Markovskyi O., Ostrovska B. Organization of parallel execution of modular multiplication to speed up the computational implementation of public-key cryptography. *Information, Computing and Intelligent systems*. 2022. № 3. P. 26–32. DOI: 10.20535/2708-4930.3.2022.265418.

11. Карацуба А. О., Офман Ю. П. Множення багатоцифрових чисел на автоматах. *Доповіді Академії наук СРСР*. 1962. № 2. С. 293–294.

12. Fürrer M. Faste Integer Multiplication. *SIAM Journal on Computing*. 2009.
Vol. 39, no. 3. P. 979–1005. DOI: 10.1137/070711716.

13. Barrett P. Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor. Lecture Notes in Computer Science. Vol. 263. Advances in Cryptology - CRYPTO '86. Proceedings / ed. by A. M. Odlyzko. Berlin, 1987. P. 311–323.

14. Montgomery P. Modular multiplication without trial division. *Mathematics of Computation*. 1985. Vol. 44, no. 170. P. 519–521.

Селіванов В.Л., Гуцуляк Н.А., Володін В.В.

МЕТОД МОДУЛЯРНОГО МНОЖЕННЯ НА ПОСТІЙНЕ ЧИСЛО З СУМІЩЕННЯМ ГРУПОВОЇ ОБРОБКИ РОЗРЯДІВ МНОЖНИКА ТА РЕДУКЦІЇ МОНТҐОМЕРІ

Розроблено та досліджено метод прискорення обчислювальної реалізації модулярного множення на постійне число, як складової модулярного експоненціювання з старших розрядів – базової обчислювальної операції несиметричної криптографії. Прискорення досягнуто за рахунок використання передобчислень для суміщення групової обробки k розрядів множника та відповідної редукції Монтґомері. Наведено числові приклади, які ілюструють роботу запропонованого методу.

Теоретично доведено та експериментально підтверджено, що розроблений метод дозволяє в k разів прискорити модулярне множення на постійне число в порівнянні з відомими методами.

***Ключові слова**: модулярне експоненціювання; модулярне множення; редукція Монтомері; криптографія з відкритим ключем.*

Selivanov V.L., Hutsuliak N.A., Volodin V.V.

MODULAR MULTIPLICATION ON INVARIABLE NUMBER WITH GROUP PROCESSING OF MULTIPLIER BITS AND MONTGOMERY REDUCTION

Developed and studied a method for accelerating the computational implementation of modular multiplication by a constant number as a component of modular exponentiation from high bits – the basic computational operation of asymmetric cryptography. The acceleration is achieved by using precomputations to combine group processing of k bits of the multiplier and the corresponding Montgomery reduction. Numerical examples are given to illustrate the work of the proposed method.

It has been theoretically proved and experimentally confirmed that the developed method allows accelerating modular multiplication by a constant number by a factor of k in comparison with known methods.

***Keywords**: modular exponentiation; modular multiplication; Montgomery reduction; public key cryptography.*