ISSN 2707-1898 (print)

Український журнал інформаційних технологій
Ukrainian Journal of Information Technology
http://science.lpnu.ua/uk/ujit
https://doi.org/10.23939/ujit2022.01.063
Article received 24.03.2022 p.
Article accepted 19.05.20221 p.
UDK 004.421

Correspondence author
I. O. Prots'ko
protsko@polynet.lviv.ua

І. О. Процько[1], Р. В. Рикмас[2], О. В. Грищук[3]

[1]Національний університет "Львівська політехніка", м. Львів, Україна
[2]ТОВ "ЮніСервіс", м. Львів, Україна
[3]ТОВ "СофтСерв", м. Львів, Україна

ПОРІВНЯЛЬНИЙ АНАЛІЗ ФУНКЦІЙ ОБЧИСЛЕННЯ МОДУЛЬНОЇ ЕКСПОНЕНТИ

Обчислення модульної експоненти для великих чисел широко використовується для знаходження дискретного логарифму, в теоретико-числових перетвореннях та в криптографічних алгоритмах. Для ефективного обчислення модульної експоненти проводяться дослідження нових методів, алгоритмів та засобів їх реалізації. Виділяють три напрями методів модульного піднесення до степеня: загальне модульне піднесення до степеня, та обчислення модульної експоненти з фіксованим показником або з фіксованою основою. Розроблено спеціальні функції для виконання піднесення до степені за модулем у математичних і криптографічних програмних бібліотеках. У роботі проведено порівняльний аналіз вільнодоступних функцій обчислення модульної експоненти з бібліотек Crypto++, OpenSSL, Pari/GP та MPIR та розроблених трьох функцій на основі алгоритму бінарного зсуву справа на ліво. Для роботи з великими числами у розроблених функціях використовується окремий тип числових даних з бібліотеки MPIR. Розроблені функції реалізують бінарний ітераційний алгоритм в одному основному потоці, у двох потоках та одному потоці з використанням передобчислення. За основу порівняння вибрано усереднений тривалість виконання обчислення модульної експоненти для псевдовипадкових даних розрядністю 1К і 2К біт, що відповідає розрядності біля 300 і 600 десяткових знаків. Результати часу виконання, що зведені у таблицю, показують, що найшвидше обчислюється модульна експонента функцією з бібліотеки OpenSSL в універсальних комп'ютерних системах. Реалізації математичними та криптографічними програмними бібліотеками функції обчислення модульної експоненти використовує більш оптимальний алгоритм множення за модулем, так зване множення Монтгомері. У розроблених трьох функціях використовуються операції множення за модулем для множників менших за значення модуля. Окремо проаналізовано функцію з використанням передобчислення залишків для фіксованої основи та модуля, що може ефективно використовуватись для обчислення дискретного логарифму.

Ключові слова: модульне піднесення до степеня; дискретний логарифм; бібліотечні функції; бінарний алгоритм; великі числа.

Вступ / Introduction

Модульна експонента та дискретний логарифм для великих чисел вимагають значних обчислювальних засобів для своєї реалізації. Вважають дискретний логарифм однонаправленою функцією (1), тому що обчислити його за умовно прийнятний час, наприклад для зламування криптографічного коду, достатньо складно.

Задача визначення дискретного логарифму [1] формулюється так, для відомих цілих A, N, y знайти ціле число x, таке, що

$$x = log_A y,\ (0 \le x \le N-1) \qquad (1)$$

де $(A, N)=1$; $A, N, y, x \in Z$).

Число $x > 0$ називають дискретним логарифмом числа y за основою A та модулем N, обчислюють відповідно за формулою (1).

Вирішенням задачі дискретного логарифму може стати розв'язок рівняння

$$A^x mod N = y\ . \qquad (2)$$

Тобто, визначивши число x, яке є розв'язком рівняння (2) знайдемо дискретниий логарифм. Отже, задача дискретного логарифму зводиться до обчислення модульної експоненти у вигляді (2). Розроблення ефективного обчислювального алгоритму цілочисельного степеня числа за модулем для великих чисел є актуальним для вирішення проблем сучасної асиметричної криптографії, теоретико-числових перетворень та інших прикладних задач [2], [3].

Об'єкт дослідження – процес порівняльного аналізу основних обчислювальних характеристик наявних бібліотечних функцій обчислення модульної експоненти для великих чисел.

Предмет дослідження – методи, алгоритми та програмна реалізація обчислення модульної експоненти для великих чисел.

Мета роботи – порівняльний аналіз досліджених обчислювальних характеристик наявних вільнодоступних бібліотечних функцій обчислення модульної експоненти для великих чисел. Це дасть змогу оптимального вибору розроблених бібліотечних функцій обчислення модульної експоненти для конкретних прикладних застосувань.

Для досягнення зазначеної мети визначено такі основні *завдання дослідження*:

- аналіз та дослідження методів ефективного обчислення модульної експоненти;
- визначення способу проведення аналізу обчислення модульної експоненти для великих чисел;
- аналіз особливостей обчислення бібліотечними функціями модульної експоненти для великих чисел;
- порівняння розробленої програмної функції засобів обчислення модульної експоненти для великих чисел.

Матеріали і методи дослідження. Дослідження спираються на аналізі відомих вільнодоступних бібліотек в мережі Інтернет, що містять функції обчислення модульної експоненти для великих чисел. На основі ін-

формації, зібраної з різноманітних офіційних джерел, досліджено, проаналізовано та з'ясовано основні обчислювальні характеристики та особливості використання бібліотечних функцій для обчислення модульної експоненти для великих чисел.

Аналіз останніх досліджень та публікацій. Розроблено багато ефективних методів модульного піднесення до степеня, огляд яких проведено в роботах [4], [5], [6], [7]. Виділяють три види алгоритмів модульного піднесення до степеня $A^x \bmod N$ [8], до яких відносять:

1) основні алгоритми модульного піднесення до степеня;
2) алгоритми з фіксованим показником x;
3) алгоритми з фіксованою базою A.

Серед основних алгоритмів модульної експоненти виділяють такі ефективні рішення: зсув справа на ліво k-нарної експоненти (right-to-left k-ary exponentiation), зсув зліва на право k-нарної експоненти (left-to-right k-ary exponentiation), експонента з ковзаючим вікном (sliding window exponentiation), багатоланкові одночасні експоненти Монтгомері (Montgomery ladder Simultaneous multiple exponentiation) і їх модифікації. У роботах Кнута [9], Баха та Шалліта [10] описується метод бінарного піднесення зсувом справа наліво. У статті Коена [11] представлено більш повний розгляд бінарних методів справа наліво та зліва направо разом із їх узагальненнями на k-нарний метод.

Багато дослідницьких робіт були зосереджені на фіксованій основі обчислення модульної експоненти [12] Популярним застосуванням функцій модульної експоненти з фіксованою основою A є криптографія з еліптичною кривою, наприклад, ключ Діффі-Хеллмана та перевірки підпису ECDSA.

Значну увагу приділяють їхній апаратній реалізації направленій на ефективне визначення дискретного логарифму x. Одним із способів реалізації прискорення обчислення модульного піднесення до степені є розпаралелення обчислень за допомогою сучасних технологій в універсальних комп'ютерних системах [13], [14], [15], [16].

Програмна реалізація обчислення модульної експоненти входить до криптографічних програмних бібліотек Crypto++ та OpenSSL призначених для роботи з великими числами [17]. Важливість прискорення програмного обчислення модульного піднесення до степені приводить до нових алгоритмічних рішень та їх реалізацій [18], [19].

Результати дослідження та їх обговорення / Research results and their discussion

Для порівняння ефективності обчислення функцій модульної експоненти використаємо відомі математичні та криптографічні програмні бібліотеки Crypto++, OpenSSL, MPIR та Pari/GP. Також, порівняння аналізується з розробленими функціями *Single*(), що виконує обчислення в одному основному потоці, *Parallel*(), що виконує обчислення за допомогою двох потоків [20] та функцією *Period_mod*(), що виконує обчислення модульної експоненти для фіксованої основи з передобчисленням [21].

Математична бібліотека Pari/GP [22] містить великий набір програм для швидких обчислень математичних функцій. Бібліотека Pari/GP також містить обчислення функції *Mod* (*a, n*)^*m* для багаторозрядних чисел та інших спеціалізовних чисел, при цьому використовуючи невеликий обсяг пам'яті в процесі виконання обчислень. Для роботи з числами за модулем бібліотека використовує окремий тип t_INTMOD. Його особливість полягає в представленні числа в спеціальній формі (Montgomery reduction), що спрощує обрахунок ділення за модулем. Бібліотеку Pari/GP можна використовувати в середовищі Linux або Mingw.

Порівняно з бібліотекою Pari/GP більш простішою у використанні є високооптимізована модифікація відомої бібліотеки GMP (GNU Multiple Precision Arithmetic Library (GMP)) бібліотека MPIR (Multiple Precision Integers and Rationals) [23], що легко компілюється у середовищі Windows. Тому для реалізації алгоритму обчислення цілочисельного степеня числа за модулем використано бібліотеку MPIR, яка написана на мові C та ассемблер, і надає можливість компіляції її функцій в середовищі Visual Studio C++. Відповідно, у бібліотеці MPIR тип даних mpz_t представляє великі числа довільної довжини, які вибрано для степеня *Exp* числа *Base* та модуля *mod* з кількістю розрядів від 256 до 2048 біт для тестування. Функція *mpz_powm*() виконує піднесення числа у степінь за модулем з бібліотеки MPIR**,** реалізуючи алгоритм плаваючого вікна ("Sliding Window") з використанням множення Монтгомері [8]. В бібліотеці MPIR також розроблено програмне забезпечення для функцій, які оптимізовані під спеціалізовані мультиядерні процесорні архітектури.

Бібліотека криптографічних алгоритмів і схем Crypto++ реалізована на мові C++ і підтримує платформи Unix (AIX, OpenBSD, Linux, MacOS, Solaris, etc.), Win32, Win64, Android, iOS, ARM [24]. Бібліотека містить сукупність доступних примітивів для теоретико-числових операцій, таких як генерація та перевірка простих чисел, арифметика над скінченним полем, операції над поліномами. У кожен з примітивів бібліотеки Crypto++ входить набір функцій, серед яких є функція *exp_crypto*() для обчислення модульної експоненти.

Бібліотека OpenSSL (The Open Source toolkit for SSL/TLS) містить набір інструментів для криптографії, що реалізує мережеві протоколи Secure Sockets Layer (SSL v2/v3) і Transport Layer Security (TLS v1), а також відповідні стандарти криптографії [25]. OpenSSL підтримує платформи Solaris, HP-UX, Linux, Android, BSD, UnixWare, Win 32 і 64, macOS, iOS та інші. Бібліотека містить три функції обчислення модульної експоненти за допомогою використання множення Монтгомері:

- *BN_mod_exp_mont*() обчислює значення A в степені p за модулем m;
- *BN_mod_exp_mont_consttime*() обчислює значення A в степені p за модулем m. Це варіант попередньої функції, який використовує фіксовані вікна та спеціальну пам'ять для попереднього обчислення, щоб обмежити залежність від даних до мінімуму для захисту секретних експонентів;
- *BN_mod_exp_mont_consttime_x*2() обчислює два незалежних піднесень значення A_1 до степеня p_1 за модулем m_1 і значення A_2 до степеня p_2 за модулем m_2. Для деяких фіксованих і рівних значень модуля m_1 і m_2 функція використовує оптимізації, які дають змогу прискорити виконання обчислень.

Для порівняння ефективності виконання обчислення модульної експоненти розроблено функції *Single*(), *Parallel*(), *Period_mod*(), які реалізовано в IDE Visual C++

2022 і можуть бути скомпільовані для іншого середовища. Для реалізації функцій використовуються бібліотечні функції *mpz_init_set* (mul, base), *mpz_sizeinbase* (exp, 2), *mpz_tstbit* (exp, i), *mpz_mul* (r, r, mul) з бібліотеки MPIR, параметрами яких є багаторозрядні дані base, exp, mod розрядністю до 2048 біт. В основу реалізації функцій вибрано бінарний ітераційний алгоритм зсуву справа на ліво (right-to-left binary exponentiation).

Функція *Single* (mpz_t r, mpz_t base, mpz_t exp, mpz_t mod) виконує бінарний ітераційний алгоритм в одному основному потоці.

Функція *Parallel* (mpz_t base, mpz_t exp, mpz_t mod) виконує базовий бінарний алгоритм, що здійснюється двома потоками. Перший потік обчислює піднесення значення до квадрату за модулем, другий потік обчислює добутки за модулем над значеннями з черги, сформованої першим потоком.

Функція *Period_mod*() виконує базовий ітераційний алгоритм обчислення модульної експоненти за допомогою передобчислення для формування скороченої послідовності залишків фіксованої основи.

Чисельні експерименти проводилися на платформі Windows 64 в комп'ютерних системах з багатоядерним мікропроцесором із спільною пам'ятю з процесорами Intel Core i9-10980XE (18 ядер, 36 потоків, 3.03GHz) і AMD Ryzen 3600 (6 ядер, 12 потоків, 3.003GHz).

Результат виконання функції записується у змінну *expected_result*, а тривалість обчислення фіксується й усереднюється з виведенням середнього значення "*average time*" в мікросекундах.

Середній тривалість виконання обчислення модульної експоненти для бібліотечних функцій: *mpz_powm*() з бібліотеки MPIR, *crypto*++() з бібліотеки Crypto++, *BN_mod_exp_mont*() з бібліотеки OpenSSL подано в табл. 1. Для порівняльного аналізу внесені до табл. 1 результати тестування розроблених функцій *Single*(), *Parallel*(), *Period_mod*().

Для обчислення модульної експоненти із заданою кількістю випробувань значення експоненти *Exp*, числа *Base* і *mod* були сформовані псевдовипадковими числами з кількістю двійкових розрядів не більшою 1024 та 2048 біт. Результати значень усередненого часу (мікросекунди) виконання обчислення функціями модульної експоненти (табл. 1) визначені для псевдовипадкових даних *Base*1, *Exp*1, *mod*1 розрядністю 1024 біт і *Base2*, *Exp2*, *mod*2 розрядністю 2048 біт, які відповідають значенням.

Base1=1559258775283944826146106259936745880191007710663592180785545871625708812395675768044611261158879037993084198450985791808156700960355218748709089617996382691960685601037523086698181288606777194603813043975878625936079683582865706857947976367181795514428394574961576857372558029191049473542841197650787788916;

Exp1=1428352097864899971708732555079465735564482140846285246054211882240996873229846735436141768520710535474156982526225738456830754529824749225178413672786090891369885834477564794839184417957332157713350938927468516042522797303684115973975776261194473923550803446318750817487083947368197108361761563379253499951 64;

mod1=5891572246297868253412624759745406795585333619437698636102450190718985181854272934901524353228514225430991703852776048028896537550292368120372032210211051014369063473209609243694640800275752961327153630792783722354532277724001895425274132047428375298344510292277365376189309328346658848802247873952610445828 8;

Base2=2567738760497917474565011343924187031994869216998758698706421709528086295599290907230065316863162179428669144090248166533311695144588834441618096640734511107106531362356071374321507249531854461586787197202959282597811236381838305962925803769346712708347766577897129937849667886402861740861770565666978446548767497029913351286923714957597816949211708272732020400851990724183722906799368441003878461018521548890319346114385586816108217151247348288474481211605784542061549242679745890886509283127487243351737251588531055149430134861136434044363046876468018170052569298949044683219014189194447340722437694507812846021234 0;

Exp2=2069711866746029428932913192684286237126085871896162254239039412857796588346206546472647626562150058442210109032488059047889420506452685343718712576517249128576248539133195446658184539707742058059362765132351678047573306711713602293369100742027668408195239640844115544602640066683598670241993486682444189036477422814089915895901005975749449200598115305587218744249548241174513464612058480455592955418351569792242918862372206056989487126897246500808974441045354232826766208959387777042971769880327762033155857980482303238918889333926985203629914823135222855353547016859173882001780159272297308256145856605458847223736 80;

mod2=1575172313457203559599568743681259608254440573656861701382162511418186803526306451419569115886878055308740967587739361731922128494702349823709785322693518660273267146847449124775067704340487870135582678102049951096446593419054681899518334410792921307143499954265358560545893952695502234824684729370866539585264690942334837436559095193843277113108303325186274650168082850048905318634729938537417490687299729788885279263013200339077021629960904568618885515772917923280644659754459311463103183288771606668121786492047222814542774350966063675771760977395343458836197101195888587251900933188447377466402318085762388758106 8.

Результати тестування середнього часу (мікросекунди) виконання обчислення модульної експоненти на основі використання бібліотек подано у табл. 1.

Табл. 1. Середній тривалість обчислення функціями модульної експоненти (мкс) / Average time of the computation the functions of modular exponentiation (μs)

Release/x86	AMD Ryzen 3600		Intel Core i9-10980XE	
Тестові дані	*Base1, Exp1, mod1*	*Base2, Exp2, mod2*	*Base1, Exp1, mod1*	*Base2, Exp2, mod2*
біт/ спроб	1024 / 1000	2048 / 500	1024 / 1000	2048 / 500
Single()	2011	12851	1928	12951
Parallel()	1671	9066	1807	10387
mpz_powm()	1166	8265	1135	8474
crypto++()	519	3721	435	3245
BN_mod_exp()	**327**	**2317**	**302**	**2221**
Period_mod()	736	4802	706	4734

Щоб скоротити загальний тривалість обчислення при збільшенні кількості розрядів великих чисел вдвічі, відповідно вдвічі зменшується кількість спроб фіксації часу обчислення.

Обговорення результатів дослідження. Реалізація відомими математичними та криптографічними програмними бібліотеками функції обчислення модульної експоненти використовує більш оптимальний алгоритм множення за модулем, так зване множення Монтгомері (Montgomery multiplication/reduction) [26]. Для покращення часу виконання обчислення модульної експоненти розробленими функціями *Single*(), *Parallel*(), *Period_mod*(), також можна використати модифіковані алгоритми множення Монтгомері [27] замість операції множення та визначення залишка.

В основу реалізації розроблених функцій використовується бінарний ітераційний алгоритм зсуву справа на ліво (right-to-left binary exponentiation). В бінарно-бітових алгоритмах [28] використовується в бінарній формі подання експоненти *Exp,* аналіз вмісту бітів якої визначає хід процесу обчислення. Прискорення часу обчислення функціями залежить від кількості одиниць у бінарному представленні експоненти, що визначають виконання операцій множення за модулем.

Функцію *Period_mod*() обчислення модульної експоненти окремо виділено, адже інші функції виконують загальне модульне піднесення до степеня. Функція *Period_mod*() використовує передобчислення залишків для фіксованої основи та модуля, що може ефективно використовуватись для обчислення дискретного логарифму. У цій функції для ефективного обчислення модульної експоненти над великими числами використано властивість періодичності послідовності залишків для степенів основи, рівних цілій степені два (A^2^i) mod N. В програмі за допомогою передобчислення здійснюється пошук періоду у послідовності залишків і встановлення індикації знаходження при цьому вкорочується довжина послідовності залишків до кінця першої періодичності і записується зміщення початку періоду і величина періоду у відповідні поля структури. Ця функціональність була винесена в окрему функцію, щоби оптимізувати багаторазове виконання пошуку залишків (A^2^i) mod N для криптографічних алгоритмів. Тому час передобчислення для формування послідовності залишків не враховується в процесі визначення середнього часу обчислення функцією *Period_mod*() модульної експоненти. Періодичність послідовності залишків має свої особливості і залежить від конкретних значень *Base, mod* and *Exp*, адже вони на багато порядків можуть відрізнятись один від одного.

Програмна реалізація *Period_mod*() через однопотокове обчислення показує незначне скорочення часу визначення модульної експоненти зі збільшенням характеристик продуктивності мікропроцесорів (табл. 1). Тому на основі розробленого програмного забезпечення подальша реалізація обчислення з використанням багатопоточних технологій забезпечить можливість високопродуктивного обчислення модульної експоненти.

Отже, за результатами виконаної роботи можна сформулювати такі наукову новизну та практичну значущість результатів дослідження.

Наукова новизна отриманих результатів дослідження – досліджено та визначено оптимальний вибір з наявних бібліотечних функцій обчислення модульної експоненти для великих чисел на основі часових характеристик в універсальних комп'ютерних системах**.**

Практична значущість результатів дослідження – вибір і застосування функцій обчислення модульної експоненти для великих чисел на основі ефективних алгоритмів забезпечить швидше проведення теоретико-числових перетворень та реалізацію виконання алгоритмів криптографічного захисту інформації.

Висновки / Conclusions

У роботі порівнюється та аналізується розроблена програмна реалізація трьох функцій та програмна реалізація функцій бібліотек Crypto++, OpenSSL та MPIR обчислення модульної експоненти. Охарактеризовано обчислювальні алгоритми та програмні реалізації функцій модульної експоненти, результати середнього часу виконання цих функцій на багатоядерних мікропроцесорах універсальних обчислювальних систем. Внаслідок, найшвидше обчислюється модульна експонента в комп'ютерних засобах з процесорами виробників AMD та Intel функцією з криптографічної бібліотеки OpenSSL.

Розроблені функції, що використовують бінарний алгоритм зсуву справа на ліво (right-to-left binary exponentiation), можуть покращити свою тривалість обчислення за умови використання алгоритму множення Монтгомері.

Практичне значення роботи полягає в тому, що отримані результати порівняльного аналізу функцій модульної експоненти можуть бути успішно використані в сучасній асиметричній криптографії, для ефективного обчислення теоретико-числових перетворень та інших прикладних завдань.

References

[1] Studholme, C. (2002). The Discrete Log Problem. Retrieved from: http://www.cs.toronto.edu/~cvs/dlog/research_paper.pdf

[2] Satyanarayana, V. N., & Ramasubramanian, U. T. (2021). *Energy-Efficient Modular Exponential Techniques for Public-Key Cryptography*. Springer Nature Singapur Pte Ltd. 255 p. https://doi.org/10.1007/978-3-030-74524-0

[3] Tandrup, M. B., Jensen, M. H., Andersen, R. N., & Hansen, T. F. (2004). Fast Exponentiation In practice. Retrieved from: https://cs.au.dk/~ivan/FastExpproject.pdf

[4] Jakubski, A., & Perliński, R. (2011). Review of General Exponentiation Algorithms. *Scientific Research of the Institute of Mathematics and Computer Science*, *2*(10), 87–98. Retrieved from: http://amcm.pcz.pl/2011_2/art_10.pdf

[5] Rezai, A., & Keshavarzi, P. (2015). Algorithm design and theoretical analysis of a novel CMM modular exponentiation algorithm for large integers. RAIRO – *Theoretical Informatics and Applications*, *49*(3), 255–268. https://doi.org/10.1051/ita/2015007

[6] Marouf, I., Asad, M. M., & Al-Haija, Q. A. (2017). Comparative Study of Efficient Modular Exponentiation Algorithms. COMPUSOFT, *International journal of advanced computer technology*, 6(8), 2381–2392.

[7] Vollala, S., Geetha, K., & Ramasubramanian, N. (2016). Efficient modular exponential algorithms compatible with hardware implementation of public-key cryptography. *Security and Communication Networks*, *9*(16), 3105–3115.

[8] Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of applied cryptography. CRC Press, Boca Raton. https://doi.org/10.1201/9780429466335

[9] *Knuth,* D. E. (1998). *The art of computer programming*. 3 d ed. Reading (Mass): Addison-Wesley, cop. 712 p.

[10] Bach, E., & Shallit, J. (1996). *Algorithmic Number Theory*. Volume I: Efficient Algorithms. Cambridge, USA: MIT Press. 516 p.

[11] *Cohen,* H. (*1993*). A course in computational *algebraic* number theory. Berlin, Heidelberg: *Springer.* 536 p. https://doi.org/10.1007/978-3-662-02945-9

[12] Robert, J.-M., Negre, C., & Plantard, T. (2019). Efficient Fixed Base Exponentiation and Scalar Multiplication based on a Multiplicative Splitting Exponent Recoding. *Journal of Cryptographic Engineering*, Springer, 9(2), 115–136. https://doi.org/10.1007/s13389-018-0196-7

[13] Lara, P., Borges, F., Portugal, R., & Nedjah, N. (2012). Parallel modular exponentiation using load balancing without precomputation. *Journal of Computer and System Sciences*, 78(2), 575–582. https://doi.org/10.1016/j.jcss.2011.07.002

[14] Nedjah, N., & Mourelle, Ld. M. (2006). Three hardware architectures for the binary modular exponentiation: Sequential, parallel, and systolic. *Circuits and Systems I*: Regular Papers, IEEE Transactions, 53(3), 627–633. https://doi.org/10.1109/TCSI.2005.858767
[15] Emmart, N., Zheng, F., & Weems, C. (2018). Faster Modular Exponentiation using Double Precision Floating Point Arithmetic on the GPU. 25th IEEE Symposium on Computer Arithmetic, 126–133. https://doi.org/10.1109/ARITH.2018.8464792
[16] Gopal, V., Guilford, J., Ozturk, E., Feghali, W, Wolrich, G., & Dixon, M. (2009). Fast and Constant-Time Implementation of Modular Exponentiation. 28th International Symposium on Reliable Distributed Systems. Niagara Falls, New York, USA. Retrieved from: https://cse.buffalo.edu/srds2009/escs2009_submission_Gopal.pdf
[17] Comparison of cryptography libraries. Retrieved from: https://en.wikipedia.org/wiki/Comparison_of_cryptography_libraries
[18] Negre, C., & Plantard, T. (2017). Efficient Regular Modular Exponentiation Using Multiplicative Half-Size Splitting. *Journal of Cryptographic Engineering*, Springer, 7(3), 245–253. https://doi.org/10.1007/s13389-016-0134-5
[19] Gueron, S. (2012). Efficient software implementations of modular exponentiation. *Journal of Cryptographic Engineering*, 2, 31–43. https://doi.org/10.1007/s13389-012-0031-5
[20] Protsko, I., Kryvinska, N., & Gryshchuk, O. (2021). The Runtime Analysis of Computation of Modular Exponentiation. *Radio Electronics, Computer Science, Control*, 3, 42–47. https://doi.org/10.15588/1607-3274-2021-3-4
[21] Protsko, I., & Gryshchuk, O. (2022). The Modular Exponentiation with precomputation of redused set of resedues for fixed-base. *Radio Electronics, Computer Science, Control, 1*. (accepted).
[22] PARI/GP home. Retrieved from: http://pari.math.u-bordeaux.fr/
[23] MPIR: Multiple Precision Integers and Rationals. Retrieved from: http://mpir.org/
[24] Crypto++ Library 8.6. Retrieved from: https://www.cryptopp.com
[25] OpenSSL. Cryptography and SSL/TLS Toolkit. Retrieved from: http://www.openssl.org/
[26] Montgomery, P. (1985). Modular Multiplication without Trial Division. Mathematics of Computation, 44(170), 519–521.
[27] Hars, L. (2004). Long Modular Multiplication for Cryptographic Applications. Retrieved from: https://eprint.iacr.org/2004/198.pdf
[28] Protsko, I. (2020). Binarno-bitovi alhorytmy: prohramuvannya i zastosuvannya. Navchal′nyy posibnyk. Lviv: "Triada plyus". 120 p. [In Ukrainian].

I. O. Protsko[1], R. V. Rykmas[2], O. V. Gryshchuk[3]

[1] Lviv Polytechnic National University, Lviv, Ukraine
[2] LtdS "Uniservice", Lviv, Ukraine
[3] LtdS "Softserve", Lviv, Ukraine

COMPARISON ANALYSIS OF THE FUNCTIONS A COMPUTATION OF MODULAR EXPONENTIATION

The computation of the modular exponentiation for big numbers is widely used to find the discrete logarithm, in number-theoretic transforms and in cryptographic algorithms. To efficient compute the modular exponent, new methods, algorithms and means of their implementation are being developed. There are three directions of computational method of modular exponentiation: general modular exponentiation, and computation of the modular exponentiation with a fixed exponent or with a fixed base. Special functions have been developed to perform modular exponentiation in mathematical and cryptographic software libraries. The paper compares the freely available functions of computing the modular exponentiation from the Crypto ++, OpenSSL, Pari / GP and MPIR libraries and developed three functions based on the right-to-left binary shift algorithm. A separate type of numeric data from the MPIR library is used to work with big numbers in the developed functions. The developed functions implement a binary iterative algorithm in one main stream, in two streams and one stream using precomputation. The comparison is based on the average time of execution of the modular exponentiation for pseudo-random data with 1K and 2K bits, which corresponds to the size of about 300 and 600 decimal signs. The runtime results summarized in the table show that the modular exponentiation is computed the fastest by a function from the OpenSSL library, which is almost twice smaller than the function from the Crypto ++ library and three times smaller than the MPIR function in universal computer systems. The implementation of the function of computing the modular exponentiation by mathematical and cryptographic software libraries uses a more optimal modulus multiplication algorithm, the so-called Montgomery multiplication. The developed three functions use multiplication by modulo operations for factors smaller than the module value. The function using precomputation of the remainders for the fixed basis and the module is analyzed separately. After all, in the testing process, the time of precomputation and determination of the periodicity of residues for this function is not taken into account. Further parallelization of the computation of parts of a multi-bit exponent and the use of the Montgomery multiplication algorithm will allow efficient use of the developed function with precomputation for the calculation of the discrete logarithm.

Keywords: modular exponentiation; discrete logarithm; library functions; binary algorithm; big number.

Інформація про авторів:

Процько Ігор Омелянович, д-р техн. наук, доцент, професор, кафедра автоматизованих систем управління. **Email:** Ihor.O.Protsko@lpnu.ua; https://orcid.org/0000-0002-3514-9265

Рикмас Роман Володимирович, провідний розробник програмного забезпечення. **Email:** roman.rikmas@gmail.com; http://orcid.org/0000-0002-1118-2036

Грищук Олександр Васильович, розробник програмного забезпечення. **Email:** ocr@ukr.net; https://orcid.org/0000-0001-8744-4242

Цитування за ДСТУ: Процько І. О., Рикмас Р. В., Грищук О. В. Порівняльний аналіз функцій обчислення модульної експоненти. *Український журнал інформаційних технологій.* 2022, т. 4, № 1. С. 63–67.

Citation APA: Protsko, I. O., Rykmas, R. V., & Gryshchuk, O. V. (2022). Comparison analysis of the functions a computation of modular exponentiation. *Ukrainian Journal of Information Technology, 4*(1), 63–67. https://doi.org/10.23939/ujit2022.01.063