УДК 004.4; 681

Грицик В.В., Пелих Н.І., Януш Д.А.

RSA ТА ЙОГО ОПТИМІЗАЦІЯ

У статті проведено порівняння кількох можливих варіантів реалізації RSA. Алгоритм оптимізовано до середніх характеристик сучасного ПК – Процесор Intel Core 2 DUO 2.4 Ghz, ОП 2048 Мб. FSB 1066 Mhz.

***Ключові слова**: RSA, алгоритм кодування з відкритим ключем, шифрування, дешифрування, Решето Ератосфена.*

В статье проведено сравнение нескольких возможных вариантов реализации RSA. Алгоритм оптимизирован к средним характеристикам современного ПК – Процессор Intel Core 2 DUO 2.4 Ghz, ОП 2048 Мб. FSB 1066 Mhz.

***Ключевые слова:** RSA, алгоритм кодировки с открытым ключом, шифровка, дешифрация, Решето Эратосфена.*

RSA algorithm program realization procedure is showed in the paper. Analysis ways of realization is showed. Program algorithm is optimized for процессор Intel Core 2 DUO 2.4 Ghz, RAM 2048 Mb. FSB 1066 Mhz.

***Key words:** RSA, code algorithm with the opened key, enciphering, decoding, Sieve of Eratosfena.*

ВСТУП

Перший алгоритм кодування з відкритим ключем (Public Key Encryption, далі PKE) було запропоновано Вітфілдом Діффі та Мартіном Хелманом у Стендфордському університеті. Вони, а також незалежно від них Ральф Меркл, розробили основні його поняття у 1976 році. Перевага PKE полягає у відсутності потреби секретної передачі ключа.

PKE базується на нерозв'язності проблеми розкладу натурального числа на прості множники. RSA схему шифрування було запропоновано у 1978 році та названо іменами трьох його винахідників: Роном Рівестом (Ron Rivest), Аді Шаміром (Adi Shamir) та Леонардом Адлеманом (Leonard Adleman). RSA належить до класу алгоритмів кодування з відкритим ключем.

У 80-х роках криптосистема переважно використовувалася для забезпечення секретності та достовірності цифрових даних. У сучасному світі RSA використовується в web-серверах та браузерах для зберігання таємності даних що передаються по мережі.

Стійкість алгоритму можна побачити на прикладі, що подали розробники алгоритму. В 1977 році вони зашифрували повідомлення, і опублікували 129-значне десяткове число (відкритий ключ). Тільки в 1994 році це повідомлення було розшифроване. Для цього було використане розподілення задачі для обчислювальних ресурсів багатьма комп'ютерами зв'язаними через Інтернет.

В апаратному виконанні алгоритм RSA застосовується в захищених телефонах, на мережевих платах Ethernet, на смарт-картах, широко використовується в криптографічному обладнанні. Окрім цього, алгоритм входить у склад всіх основних протоколів для захищених комунікацій Internet, в тому числі S/MIME, SSL і S/WAN, а також використовується в багатьох установах, наприклад, в державних службах та в багатьох корпораціях. На осінь 2000 року технології з використанням алгоритму RSA були ліцензовані більш ніж 700 компаніями.

На початку 2001 року криптосистема RSA стала найбільш поширеною серед асиметричних криптосистем (криптосистем з відкритим(public) ключем). Незалежно від офіційних стандартів, існування такого стандарту є надзвичайно важливо для розвитку електронної комерції і

економіки загалом. Єдина система відкритого (public) ключа допускає обмін документами з електронно-цифровими підписами між користувачами різних країн, що використовують програмне забезпечення на різних платформах; така можливість просто необхідна для розвитку електронної комерції. Розповсюдження системи RSA дійшло до такого рівня, що її враховують при створенні нових стандартів. При розробці стандартів цифрових підписів, в першу чергу в 1997 був створений стандарт ANSI X9.30, що підтримував Digital Signature Standard (стандарт Цифрового підпису). На рік пізніше був введений ANSI X9.31, в якому є присутнім акцент на цифрових підписах RSA, що був відповідний економічній ситуації для фінансових установ на той час.

Фактично криптосистема RSA є частиною багатьох стандартів. Зокрема, стандарт ISO 9796 описує RSA як спільний з ним криптографічний алгоритм, що відповідає стандарту безпеки ITU-T X.509. Окрім цього, криптосистема RSA є частиною стандартів SWIFT, ANSI X9.31 rDSA, а також проекту стандарту X9.44 для американських банків. Австралійський стандарт керування ключами AS2805.6.5.3 теж включає систему RSA.

Алгоритм RSA використовується в Internet, а саме він входить в такі протоколи як S/MIME, IPSEC (Internet Protocol Security) і TLS (який повинен замінити SSL), а також в стандарт PKCS. Для розробників програм із застосуванням PKCS організація OSI Implementers' Workshop (OIW) випустила маніфест, який частково присвячений алгоритму RSA.

Велика кількість проектів та стандартів, що розробляються в наш час містять в собі або сам алгоритм RSA, або його підтримку, або ж рекомендують криптосистему RSA для забезпечення секретності. Як приклад, рекомендації IEEE P1363 і WAP WTLS включають в себе систему RSA [1].

1. ПОСТАНОВКА ЗАДАЧІ

Захищеність і складність обрахунків алгоритму є одночасно перевагою і недоліком. Недолік полягає у часі реалізації програмою шифрування, дешифрування, перевірки числа в приналежності до множини простих чисел.

Процес дешифрування відбувається набагато швидше, оскільки для цього ми маємо всі необхідні дані і ми не тратимо час на генерацію простих чисел, як при шифруванні.

Таким чином, важливою задачею є вибір алгоритму програмної реалізації RSA.

2. АНАЛІЗ ВАРІАНТІВ РЕАЛІЗАЦІЇ

Першим кроком є генерація ключа. Отже, нам потрібно знайти два простих числа q і p. Число називають простим, якщо воно має тільки два дільника 1, і саме число. Наприклад, числа 22 і 14 не є простими, тоді як 7, 11, 29 – прості. Існують різноманітні тести простоти. AKS тест простоти (також відомий як Агравал-Кайал-Саксена тест простоти та циклотомічний AKS тест) є детермінованим алгоритмом доведення простоти, розробленим та опублікованим трьома індійськими науковцями Маніндрою Агравалом, Ніраєм Кайалом та Нітоном Саксеною 6 серпня 2002 р. в статті «PRIMES is in P» («Перевірка простоти належить до класу P»). Автори отримали за цю роботу у 2006 премію Геделя [2].

Існують експоненціальні формули за допомогою яких можна вирахувати прості числа, $M(n) = 2^n - 1$ – це формула Мерсена, або $F(n) = 2^{2^n} + 1$ – формула Ферма. Числа із цих формул є простими.

Решето Ератосфена – це один із найстарших з відомих способів виписування простих чисел. На відміну від інших методів, він не використовує ніякої спеціальної функції. Ератосфен (Erathostenes) був грецьким математиком, що народився близько 284 року до н.е. Він володів різними галузями знань, але сучасники не вважали його високим спеціалістом ні в одній з галузей. Але вже 2300 років ми використовуємо його роботи.

Принцип роботи Решета Ератосфена такий: ми виписуємо на листку всі непарні числа, що починаються з «3» і закінчуються n, де n – максимальне число до якого ми вишукуємо прості

числа. Всі числа непарні, оскільки серед парних чисел немає простих, окрім числа «2». Отож у нас перше число у списку «3», ми починаємо викреслювати кожне третє число, починаючи від наступного після «3», і так до кінця. Потім ми беремо наступне число що не є закресленим, в даному випадку це буде число «5», і викреслюємо кожне п'яте число, аналогічно, як ми це зробили в попередній раз. Таким чином ми викреслили всі числа, які були кратні спочатку «3», а потім «5». Отже, продовжуючи таким чином викреслювати цифри, в нас залишаться тільки числа, які діляться на 1, і само на себе – прості.

Наступним етапом буде знаходження добутку цих двох простих чисел: $n = p*q$. Де n називається модулем. Знаходимо значення функції Ейлера $\varphi(n) = (p-1)(q-1)$, вибираємо таке e що задовольняє умову 1< e < $\varphi(n)$ а також e та $\varphi(n)$ є взаємно простими.

Взаємно прості числа – натуральні або цілі числа, які не мають спільних дільників більших за 1, або, інакше кажучи, якщо їх найбільший спільний дільник дорівнює 1. Таким чином, 3 і 7 – взаємно прості, а 3 і 6 – ні (діляться на 3). Будь-яке натуральне число взаємно просте з 1. Якщо p – просте, а n – довільне ціле число, то вони взаємно прості тоді і тільки тоді, коли n не ділиться на p. Знаходження найбільшого спільного дільника зручно робити, використовуючи алгоритм Евкліда: маючи два натуральні числа a та b:

якщо $b = 0$, то a є шуканим НСД (Найбільший спільний дільник),

інакше повторюємо обчислення для пари b та залишку від ділення a на b (тобто $a \bmod b$).

Для наочної демонстрації представлено ітераційну версію, де $a > b$:

НСД(a, b)

поки $b \neq 0$

c := залишок від ділення a на b

$a := b$

$b := c$

поверни a

Після наведених вище дій ми вираховуємо d, так щоб ($e*d$) ділилося на $\varphi(n)$, або це виглядає так: $ed \equiv 1(\bmod \varphi(n))$ [3, 6]

Тепер числа e і d називаються відкритою і секретною експонентами, відповідно. Отже, n та e це відкрита частина ключа, а d – секретна. Обов'язковою умовою є знищення пари чисел p та q з метою утримання секретності ключів.

Тепер фактично буде відбуватись саме шифрування. А саме: для того щоб зашифрувати повідомлення m, що обов'язково повинно бути менше за n, потрібно обрахувати формулу [3]

$c = m^e \bmod n$,

де c – зашифроване число, m – текст що потрібно зашифрувати, e – частина відкритого ключа та n-модуль. Тобто потрібно цілочисельно поділити число m в степені e на модуль n. Тоді, як для розшифрування, відповідно, потрібно [3] виконати таку операцію $m = c^d \bmod n$.

Таким чином, для того щоб закодувати текстове повідомлення, кожній букві алфавіту присвоюється двозначне число. Так наприклад:

А	**Б**	**В**	**Г**	**Д**	**Е**	**Є**	**Ж**	**З**	**.....**
01	02	03	04	05	06	07	08	09	.....

Пробіл можна позначити як 00, таким чином, будь-яке повідомлення можна зашифрувати числами.

Для програмної реалізації зручно використовувати блок-схему (1).

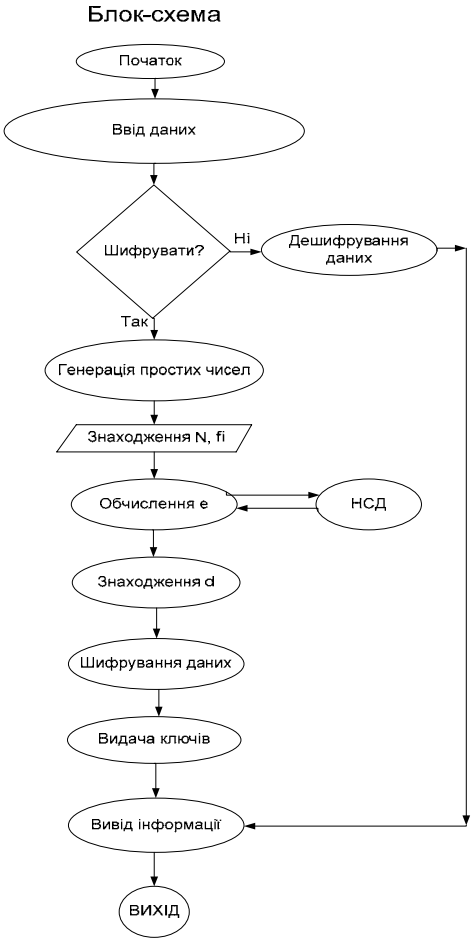

Рис. 1. Блок-схема

3. РОЗВ'ЯЗОК

Для втілення алгоритму в життя було використане середовище Borland Developer Studio 2006. Мова програмування Delphi for Win32.

Розглянемо процедуру генерації простих чисел. Тут постає вибір між кількома методами простих чисел, наприклад, ми можемо генерувати випадкове число і перевіряти його чи воно є простим, у зв'язку з тим що всі операції працюють із числами великого розміру (алгоритм перевірки описано нижче). У цьому випадку перевірка чи число є простим або займає недоцільно багато часу (для того щоб точно стверджувати що число є простим нам потрібно перевірити всі можливі варіанти ділення), або є імовірнісною; або ж можна вираховувати числа за приведеними вище формулами Мерсена чи Ферма, що теж вимагає обчислень. Таким чином, оптимальним варіантом є обрахунок простих чисел потрібного діапазону завчасно і занесення їх у базу даних. На лобову атаку такий підхід не впливає, оскільки, по-перше: отримати повний перелік таких чисел не є проблемою, по-друге: перелік цих чисел, як і час їх перевірки є достатньо великий, щоб затратити не один рік.

Тепер при виконанні даного етапу нам потрібно просто випадково вибрати прості числа потрібного діапазону з бази даних. Під діапазоном розуміється довжина чисел. Вся справа а тому, що рівень захисту алгоритму RSA залежить від довжини добутку простих чисел, тобто від модуля. Станом на 2007 рік нормальною довжиною модуля n – 1024 біт, тобто два простих числа повинні бути в приблизному діапазоні 512 біт. Швидкість обчислювальних машин відчутно зростає з кожним роком, саме тому зараз надійним захистом вважається довжина модуля 2048 біт.

На наступному етапі ми повинні порахувати модуль *n*, а також функцію Ейлера $\varphi(n)$. Тепер повинні вибрати число e і перевірити чи воно є взаємно простим із значенням функції Ейлера. Тепер знаходимо *d*, нашу секретну частину ключа із

$$ed \equiv 1(\mathrm{mod}\,\varphi(n)).$$

Нам залишилось тільки зашифрувати вхідну інформацію, видати ключі та закодовану інформацію.

Ще однією задачею при програмуванні даного алгоритму є представлення цілих чисел великого розміру в пам'яті, та виконання з ними різноманітних операцій, таких як піднесення до степеня та ділення. Максимальна цілочисельна змінна у Delphi може містити 64 біта. Тут теж існує багато теорій, такі як зберігання числа у стрічковому форматі, така ідея відкидається через досить великий час виконання. Також можна представляти число як масив бітів, і всі операції з числами виконувати побітово у двійковій системі – цей варіант є відносно швидкодіючим. Однак найкращим підходом є варіант, який включає ассемблерні вставки, оскільки асемблер вміє працювати із числами будь-якого розміру, а командам потрібно задавати не кожен біт, а по 32 біта (що дозволить оптимально використовувати архітектуру процесора [4]). Звісно асемблер теж буде виконувати операції побітово, але він це зробить набагато швидше ніж мова високого рівня така як Delphi. Ось приклад асемблерного коду що оперує числами великого розміру [5]:

```
; Без знакове множення двох 64 бітних чисел X та У і зберігання
;результату у 128 бітне число Z
  {
mov  eax,     dword ptr x
mov  ebx,eax
mul  dword ptr y          ;помножити молодші двойні слова
mov  dword ptr z,  eax       ;зберегти молодше слово добутку
mov  ecx,     edx       ;зберегти старше слово добутку
mov  eax,     ebx       ;молодше X в eax
mul  dword ptr y[4]          ;помножити молодше слово на старше
add  eax,     ecx       ;
adc  edx,     0       ;добавати перенос
mov  ebx,     eax       ;зберегти частинний добуток
mov  ecx,     edx
mov  eax,     dword ptr x[4]
mul  dword ptr y          ;помножити старше слово на молодше
add  eax,     ebx       ;додати до частинного добутку
mov  dword ptr z[4], eax
adc  ecx,     edx
mov  eax,     dword ptr x[4]
mul  dword ptr y[4]      ;помножити старші слова
add  eax,  ecx     ;додати до частинного добутку
adc  edx,     0       ;додати перенос
mov  word ptr z[8], eax
mov  word ptr z[12], edx
                }
```

ВИСНОВОК

Основною перевагою даної реалізації є її пристосованість до архітектури сучасних ЕОМ, що дозволяє оптимально використовувати їх потужність.

При подальшому розвитку електронно-обчислювальної техніки для пришвидшення роботи алгоритму його реалізацію потрібно буде коректувати. Однак доцільно буде зауважити, що при

збільшенні обчислювальних можливостей комп'ютерів, автоматично буде зростати і довжина ключа.

На сьогоднішній день є кілька способів математичного розкладу числа на множники, чим і пробують скористатись зловмисники. Але повідомлення, що закодоване за допомогою ключа в 2048 біт, якщо злом почати саме тепер, буде, можливо, розкодоване через добрий десяток років.

ЛІТЕРАТУРА

1. Сайт: http://www.mpgu.ru
2. Сайт: http://wikipedia.org
3. Коутинхо С. Введение в теорию чисел. Алгоритм RSA // И-во: Постмаркет ISBN: 5-901095-09-X.– 2001.– С. 328.
4. http://www.intel.com/products/processor/manuals/index.htm
5. Зубков С. В. «Assembler для DOS, Windows и UNIX»
6. В.Ємець, А.М. Мельник, Р.Попович. «Сучасна криптографія. Основні поняття» // Львів. – 2003. – 144 с.

Рецензенти: д.т.н., проф. Коваленко І.І.
д.т.н., проф. Фісун М.Т.

© Грицик В.В., Пелих Н.І., Януш Д.А., 2009

Стаття надійшла до редколегії 10.03.09