УДК. 519.41

Я.Цепенюк; М.Цепенюк, канд.техн.наук
Тернопільський державний технічний університет імені Івана Пулюя

ЗАДАЧА ФАКТОРИЗАЦІЇ ДОВГИХ ЦІЛИХ В КОНТЕКСТІ КРИПТОГРАФІЇ ІЗ ВІДКРИТИМ КЛЮЧЕМ

Розглянуто зв'язок стійкості криптосистеми RSA із проблемою факторизації довгих цілих. Результатом роботи є побудова ефективного розподіленого програмного забезпечення для факторизації довгих цілих на основі алгоритму Полларда. Система працює в середовищі Windows NT і використовує відповідні мережні можливості цієї ОС.

Перший конкретний алгоритм для відкритого шифрування запропонували в 1978 р. Райвест, Шамір та Ейдлман [1]. Ця криптографічна система називається RSA (Rivest-Shamir-Adleman), вона широко використовується і сьогодні.

Відкритим ключем в RSA є число n>0, $n \in Z$ (взагалі, величезне, не менше 512 біт) та число $e \in Z$, 1<e<n. Число n є добутком двох простих чисел: n=pq, але числа p і q тримаються в таємниці. Таємним є і розшифровуючий ключ d. Ключ d та число e задовольняють конгруенції $de \equiv 1 \bmod (p-1)(q-1)$. Тепер зрозуміло, чому потрібно тримати у таємниці (або й просто забути) p і q. Причина в тому, що, знаючи e, p і q, можна із останньої рівності за допомогою розширеного алгоритму Евкліда відновити d.

Текст, який підлягає шифруванню, є послідовністю алфавітно-цифрових символів, скажімо, у коді ASCII - символам відповідають числа. Таким чином, текст є послідовністю тризначних чисел, які, будучи виписаними у ряд, утворюють деяке довге число m. Нехай m таке, що $1 \le m < n$, якщо m>n, то розіб'ємо текст на менші блоки. Схема шифрування/дешифрування повідомлення в системі RSA має наступний вигляд:

Етап	Відправник	"Ворог"	Приймач
1.	Зашифровує повідомлення m: $c = m^e \bmod n$, відправляє c ⋯⋯▶	ПЕРЕХОПЛЮЄ	
2.			Отримує c і розшифровує його: $m = c^d \bmod n$

"Ворог" знає n та e, оскільки вони загальнодоступні, але, перехопивши $c = m^e \bmod n$, він не може відновити повідомлення m, тому що ця проблема не розв'язується достатньо ефективним алгоритмом [1]. З іншого боку, отримувач, знаючи d (приватний ключ), розшифровує m. В загальному випадку справедлива наступна

ТЕОРЕМА. Якщо n=pq, ed≡1 mod (p-1)(q-1), 1<m<n, m,n-взаємно прості, то має місце рівність $(m^e)^d = m^{ed} \equiv m \bmod n$.

Щоб розкрити шифр RSA, потрібно вміти ефективно знаходити d, для цього достатньо знайти співмножники p та q, що утворюють n або, як говорять, факторизувати n.

ЗАДАЧА: Дано $n \in Z$, знайти $p,q \in Z | pq = n$.

ρ-алгоритм Полларда

В класичних працях П. Л. Чебишева, Сільвестра, Ерміта та ін. була отримана відома оцінка для $\pi(x)$ (див. [2]) - кількості простих чисел, що передують числу x:

$$\lim_{x \to \infty} \frac{\pi(x) \ln x}{x} = 1.$$

Оскільки найменший дільник числа n не перевищує

$$L = \lfloor \sqrt{n} \rfloor,$$

то при факторизації методом прямого перебору потрібно буде ділити n в граничному випадку на $\pi(L) = \frac{L}{\ln L}$ перших простих чисел. Дана ситуація матиме місце, якщо ми послідовно будемо проводити ділення на прості дільники. На практиці, як правило, спочатку складається таблиця простих чисел, скажімо до 1000, а тоді, діючи методом сита, відсіваємо всі пробні дільники, кратні до знайдених простих чисел.

Різного роду оптимізації можуть покращити роботу переборного алгоритму на константу порядку навіть до 1000. Проте, для факторизації чисел довжиною 60 десяткових знаків і більше такий підхід неприйнятний. Більше того, алгоритм повинен задовільно працювати для складного загального випадку – число n є добутком лише двох простих чисел, причому кожне із них є порядку $L = \lfloor \sqrt{n} \rfloor$, при цьому різниця між ними є теж достатньо велике число. Застосування окремих прийомів факторизації для чисел спеціального виду [2], наприклад, факторизація n, для якого відомо розклад n-1, не приведе до успіху, оскільки в криптографії ці відомі окремі випадки ретельно уникаються. Покращити ситуацію тут можна лише за допомогою побудови розподілених систем факторизації [1] із використанням спеціалізованих алгоритмів. Варіант такої системи ми побудували на основі Pollard-Rho алгоритму [2].

Нехай p - найменший простий співмножник числа n. Також нехай в результаті випадкового генерування чисел вдалося отримати такі цілі a та b (a>b>0), що

$$a \equiv b \mod p. \qquad (1)$$

Тоді, очевидно, число p є дільником a-b і може бути отримане обчисленням НСД(a-b,n).

Таким чином, факторизацію n можна звести до знаходження двох чисел a та b, що дають при діленні на p однакові остачі. За принципом Діріхле, це завжди можна зробити, породивши p+1 різних чисел.

На щастя, отримати такі a та b з великою ймовірністю можна значно раніше. Задамося числом r, r<p, і оцінимо ймовірність того, що, породивши випадковим чином r різних чисел, ми отримаємо їх усіх із різними остачами при діленні на p (невідоме p). Ця ймовірність, як показав Феллер (див. [2]), V(r, p), залежна від r та p, дорівнює

$$V(r,p) = \frac{p(p-1)(p-2)\ldots(p-r+1)}{p^r} = \frac{p}{p}\frac{p-1}{p}\frac{p-2}{p}\ldots\frac{p-r+1}{p} =$$

$$= \left(1-\frac{1}{p}\right)\left(1-\frac{2}{p}\right)\ldots\left(1-\frac{r-1}{p}\right).$$

Оцінимо логарифм цієї ймовірності:

$$\ln V(r,p) = \sum_{j=1}^{r-1} \ln\left(1-\frac{j}{p}\right) < -\sum_{j=1}^{r-1} \frac{j}{p} = -\frac{r(r-1)}{2p} \approx -\frac{r^2}{2p},$$ в останній викладці використовується нерівність $\forall x \in (0,1): \ln(1-x) < -x$.

При рості r ймовірність V(r,p) падає швидко: при $r=\sqrt{p}$ маємо ln V=-1/2, $V=\frac{1}{\sqrt{e}} \approx 0{,}61$; а при $r=\sqrt{2p}$ $V=\frac{1}{e} \approx 0{,}37$.

Відповідно, ймовірності знайти a та b, які задовольняють (1), дорівнюють 0,39 та 0,63. Оскільки $p \approx \sqrt{n}$, є значні шанси факторизувати n за $\sqrt[4]{n}$ спроб. Враховуючи те, що сучасний ПК за добу може виконати порядку 10^{10} ітерацій, реально цим методом за ту ж добу факторизувати число довжиною 40 десяткових розрядів (використовуючи

тільки один ПК). Це значно кращий результат, ніж для методу прямого перебору, оскільки для нього за ту ж кількість дій верхньою межею є факторизація чисел довжиною 20 десяткових знаків.

Розглянемо, як реалізувати цей алгоритм. Спочатку необхідно утворити послідовність випадкових чисел за деяким модулем m>n. Сформульовано ряд критеріїв (див. [2]), які забезпечують гарантію якості псевдовипадкового генератора чисел. Наведений нижче датчик породжує необхідну нам послідовність псевдовипадкових чисел.

Алгоритм генерації псевдовипадкових чисел

1. [Ініціалізація.] В датчик вводиться довільне натуральне число $x_0 \in [1, m-1]$.
2. [Загальний крок.] На k-му кроці, k=1,2,…, породжується число $x_k := P(x_{k-1}) \bmod m$, де P – деякий поліном.

В якості P можна вибирати різні функції. Д. Кнут радить вибирати в якості P функцію P(x)=ax+c, де a та c-фіксовані випадкові нетривіальні натуральні числа. При цьому повинні виконуватися дві умови:

1. (a mod 8)=5,
2. c-непарне.

Використовуючи описаний датчик, породжуємо випадкову послідовність

$$x_1, x_2, ..., x_k .$$

Кожне отримане число x_k необхідно перевірити на виконання умови $x_k \equiv x_j \bmod p$ для всіх раніше отриманих x_j, j<k.

Для виконання цієї перевірки необхідно здійснити квадратичний перебір. Обійти його можливо, користуючись результатами наступної теореми.

ТЕОРЕМА. Нехай x_k, x_j породжені поліноміальним датчиком $P(x)= \sum_{i=0}^{s} a_i x^{s-i}$, і, крім цього, $x_k \equiv x_j \bmod p$, тоді для довільного цілого a>0 $x_{k+a} \equiv x_{j+a} \bmod p$.

Із наведеного випливає, що для факторизації числа достатньо знайти пару x_k, x_j із потрібною різницею в індексах. Тепер можна сформулювати

ρ - алгоритм Полларда

1. [Ініціалізація.] Випадковим чином породити x_k; $x_{2k} := x_k$.
2. [Загальний крок.] $x_k := P(x_k) \bmod m$; $x_{2k} := P(P(x_{2k})) \bmod m$.
3. Знайти НСД(n, $|x_k - x_{2k}|$) алгоритмом Евкліда.
4. Якщо НСД=1, іти в 2.
5. Якщо НСД<n, то p:= НСД, кінець.
6. [НСД ділиться не тільки на p, а й на n.] Йти в 1.

Побудова розподіленої системи

Загальне уявлення про розроблену систему дає рис. 1.

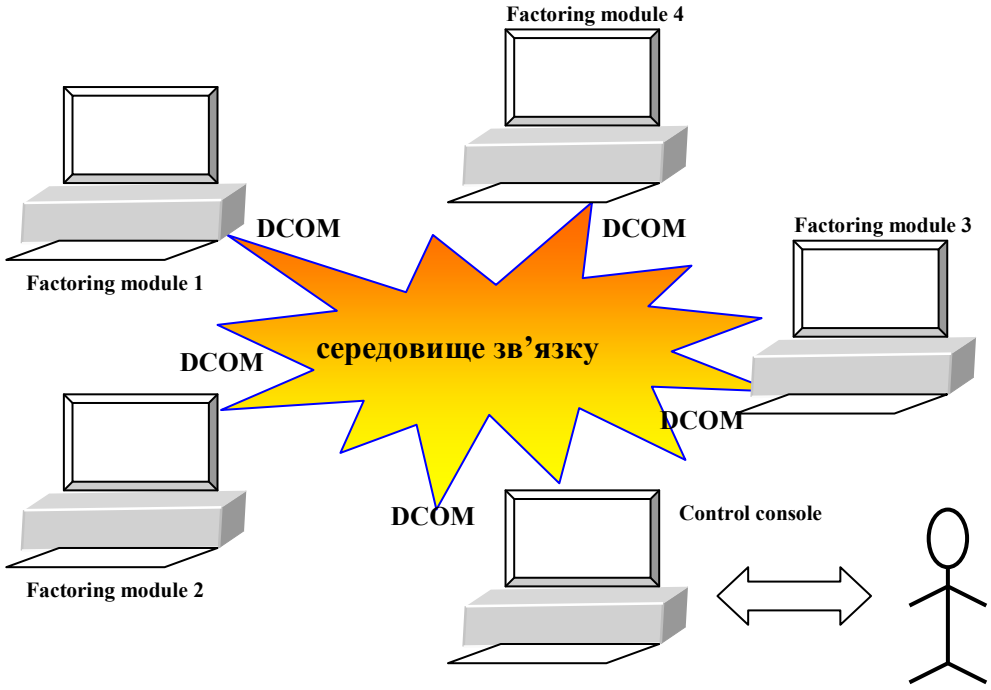

Рис. 1. Архітектура розподіленої системи

Середовище зв'язку в даному випадку являє собою швидкісну локальну мережу (LAN), яка сполучає кілька десятків(сотень) робочих станцій. На кожній із цих станцій запускається свій процес(модуль факторизації), що реалізує алгоритм Полларда. Одна з робочих станцій вибирається в якості керуючої консолі, призначеної для узгодженого функціонування інших та збору результатів факторизації за допомогою певного зв'язуючого протоколу (у нашому випадку DCOM [3]).

Таким чином, досягається лінійне прискорення факторизуючого алгоритму, яке вкладає часові обмеження процесу в допустимі рамки. Відзначимо, що розподілена система є єдиним можливим механізмом факторизації великих чисел, оскільки при субекспоненційній складності найкращих сучасних алгоритмів (наприклад, PPMPQS, див. [1]), факторизація довгого числа довжиною від 80 десяткових знаків на одному ПК не здійснюється за прийнятний відрізок часу.

Система працює в середовищі Windows NT і використовує відповідні мережні можливості цієї ОС. В якості протоколу зв'язку використовується протокол DCOM [3]. Керуюча консоль(клієнт, з якого здійснюється керування) та COM сервер модуля факторизації створені за допомогою середовища розробки Visual Basic від Microsoft [4]. Ядро модуля факторизації написане на Visual C++ із використанням високоефективної бібліотеки MIRACL (Multiprecision Integer Rational Arithmetic C/C++ Library) від Shamus Ltd. Ядро запускається як окремий процес і взаємодіє із модулем факторизації через COM інтерфейс. У свою чергу, модуль факторизації підтримує зв'язок із керуючою консоллю за допомогою DCOM (див. рис. 2).

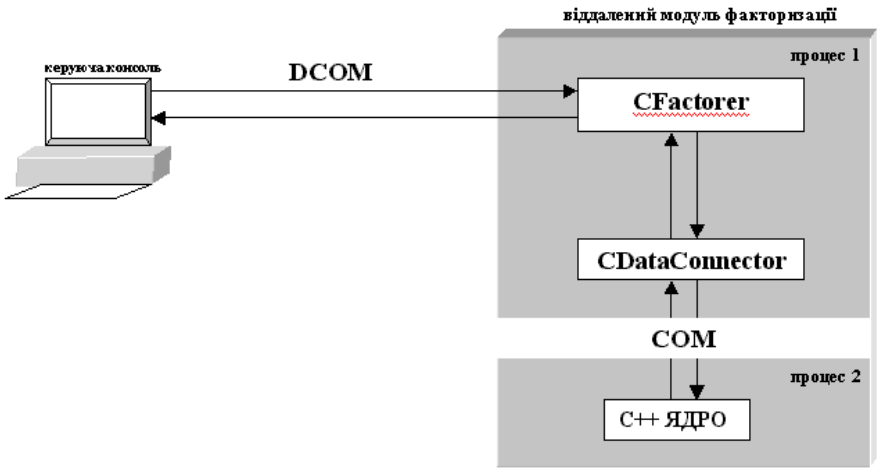

Рис. 2. Взаємодія консолі із віддаленим модулем факторизації

Керуюча консоль (рис. 3) є простим клієнтом, написаним на VB для керування віддаленими модулями факторизації. Керуюча консоль складається із двох панелей відображення статусної інформації та поля для введення числа, яке буде факторизуватися. Спочатку необхідно ввести число і додати віддалені сервери факторизації (факторизуючі модулі). Всі додані сервери повинні з'явитися у правій панелі і відразу підключитися до роботи. Також там відображається кількість ітерацій ядра на відповідному модулі факторизації. Ліва панель є журналом роботи всіх серверів, який включає також знайдені кофактори. Консоль може залучати до процесу факторизації тільки ті комп'ютери, на яких проінстальовано сервер підтримки віддаленого модуля факторизації.

Приклади факторизації довгих цілих системою

Як видно з рис.3, факторизація числа n = 519126808971966443089280 9 = 77158673929*667280421310721 тривала 28451 кроків. Факторизація проводилася із використанням *одного* факторизуючого модуля і зайняла 20 секунд на комп'ютері Pentium™ 450MHz. Інші отримані результати приведені в таблиці:

N	p	q	К-сть ітерацій
519126808971966443089280 9	67280421310721	77158673929	28451
18446744073709551616	274177	67280421310721	6120
2286227974369	18837001	121369	1457
439125228929	65537	6700417	679

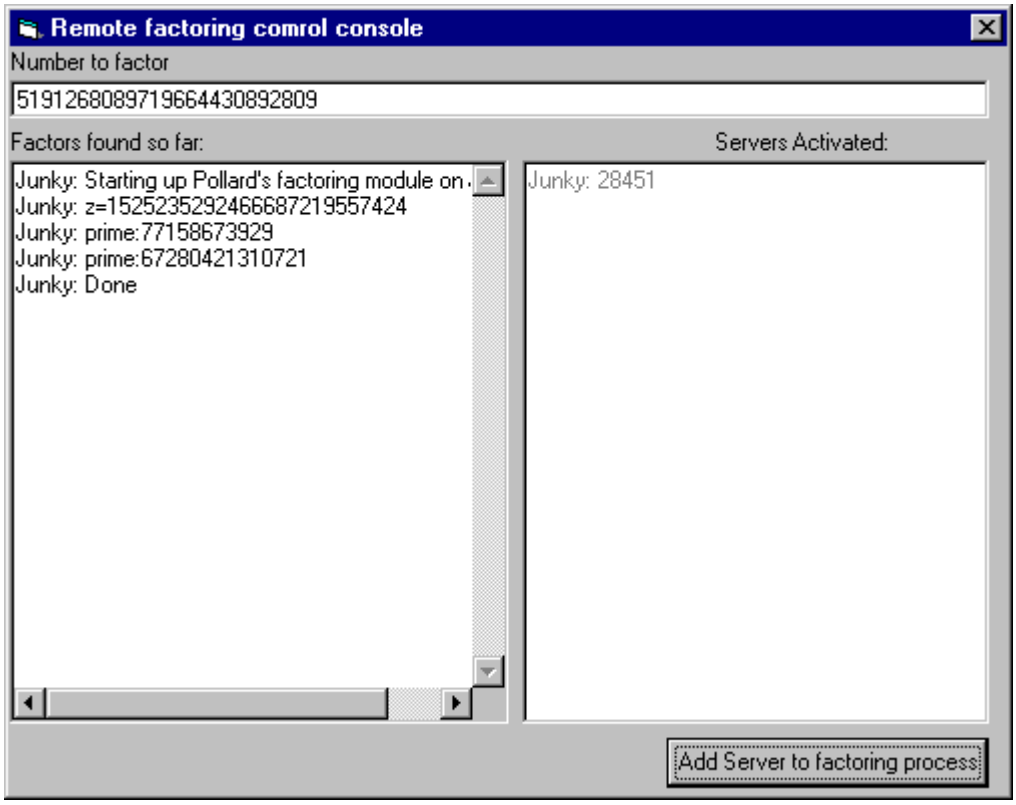

Рис. 3. Графічний інтерфейс керуючої консолі

Висновки

Отримані числа добре узгоджуються із теоретичними розрахунками. А саме, кількість ітерацій, витрачених на знаходження кофакторів, співрозмірна із $\sqrt[4]{n}$, що значно краще, ніж у випадку переборного алгоритму.

Практичною цінністю розробленої системи є застосування компонентного підходу в паралельних обчисленнях. Ядро факторизуючого модуля є незалежною компонентою , яка слідує фіксованому інтерфейсу та інкапсулює в собі факторизуючий алгоритм. Таким чином, дослідник може легко змінювати використовуваний алгоритм, не вникаючи в тонкощі реалізації розподілених обчислень.

Постійне зростання апаратних потужностей і вдосконалення алгоритмічної бази змушує поступово збільшувати довжину ключів у RSA подібних шифрах. Практичний аналіз складності факторизації великих чисел на системах, подібних нашій, допомагає оптимізувати довжину ключів з точки зору надійності та швидкодії.

The relationship between the RSA cryptosystem security and the problem of long integers factoring was considered. As a result, the effective distributed software based on the Pollard Rho method for factoring integers was built. This system works in the Windows NT environment and handles NT networking facilities.

Література

1. О.В.Вербіцький. Вступ до криптології// ВНТЛ.- Львів, 1998.-357 с.
2. Д.Кнут. Искусство программирования для ЭВМ. - Т. 2. - М.: Мир, 1977.- 724 с.
3. Э.Рофейл, Я.Шохауд. COM и COM+. Полное руководство. ТОО «Век», 2000.- 584 с.
4. К.Гетц, М.Гилберт. Программирование на Visual Basic 6 и VBA. Руководство разработчика. - Ирина, BHV. - Киев, 2001.- 980 с.

Одержано 02.12.2002 р.