УДК 004.056.55; 004.051

В. Яковина, Д. Федасюк, М. Сенів, О. Білас
Національний університет “Львівська політехніка”

ПОРІВНЯННЯ ШВИДКОДІЇ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМІВ СИМЕТРИЧНОГО (DES) ТА АСИМЕТРИЧНОГО (RSA) ШИФРУВАННЯ

Виконано дослідження швидкодії програмної реалізації алгоритму шифрування з відкритим ключем RSA та алгоритму симетричного шифрування DES. Швидкість шифрування на процесорі Intel Celeron D 351 становить 306,4±0,6 кбайт/с та 11,08±0,06 Мбайт/с для алгоритмів RSA та DES відповідно. Подано рекомендації щодо ефективності застосування цих алгоритмів залежно від обсягу даних, що підлягають шифруванню. Показано, що засоби CryptoAPI в поєднанні з платформою Microsoft .NET є гнучким архітектурно незалежним інструментом для створення ефективних та безпечних модулів криптографічного захисту інформації.

The studies of software performance of public key RSA algorithm as well as symmetric DES algorithm have been performed. The encryption velocity at Intel Celeron D 351 processor is 306.4±0.6 kbytes/s and 11.08±0.06 Mbytes/s for RSA and DES correspondingly. The recommendations concerning effective usage of these algorithms depending of input data amount are given. It is shown that CryptoAPI methods in connection with Microsoft .NET are flexible, device transparent tools for building fast and secure cryptographic software modules.

Вступ

Відповідно до призначення сьогодні склались дві області криптографії: класична, або криптографія з таємним ключем, і сучасна, або криптографія з відкритим ключем. Історія першої нараховує тисячоліття, тоді як офіційний вік другої ще не перевищує трьох десятиліть.

Алгоритми шифрування з відкритим ключем розроблялися для того, щоб розв’язати дві найважчі задачі, що виникли при використанні симетричного шифрування [1, 2].

Першою задачею є розподіл ключа. При симетричному шифруванні потрібно, щоб обидві сторони вже мали спільний ключ, що у якийсь спосіб повинний бути їм заздалегідь переданий. Ця вимога заперечує всю суть криптографії, а саме можливість підтримувати загальну таємність при комунікаціях.

Другою задачею є необхідність створення таких механізмів, при використанні яких неможливо було б підмінити кого-небудь з учасників, тобто потрібен цифровий підпис.

Сучасна криптографія дає змогу розв’язати набагато ширше коло задач, ніж криптографія класична. Однак існує ряд причин, з яких асиметричні алгоритми шифрування не можуть повноцінно замінити симетричні алгоритми [2]:

- По-перше, алгоритми з таємним ключем набагато простіше реалізовуються як програмно, так і апаратно. Через це за однакових характеристик продуктивності та стійкості складність, а значить і ціна апаратних засобів, що реалізують шифр з відкритим ключем, помітно вища за ціну апаратури, що реалізує класичний шифр, а при програмній реалізації на одному й тому самому типі процесора симетричні шифри працюють швидше від асиметричних.
- По-друге, надійність алгоритмів з відкритим ключем сьогодні обґрунтована набагато гірше, ніж надійність алгоритмів з таємним ключем і немає гарантії, що через деякий час вони не будуть розкриті, як це сталося з криптосистемою, основаною на задачі про вкладання ранця.

Тому для організації шифрованого зв’язку нині застосовуються виняткова класичні шифри, а методи сучасної криптографії використовуються тільки там, де вони не працюють, тобто для організації різних протоколів типу цифрового підпису, відкритого розподілу ключів тощо.

Одним з найпоширеніших і найвідоміших алгоритмів симетричного шифрування є DES (Data Encryption Standard). Алгоритм був розроблений у 1977 році, у 1980 році був прийнятий NIST (National Institute of Standards and Technolody США) як стандарт (FIPS PUB 46) [1, 2].

RSA (авторами є Rivest, Shamir і Adleman) – це алгоритм з відкритим ключем (асиметричний алгоритм), призначений як для шифрування, так і для аутентифікації [3]. Був розроблений в 1977 році. Відтоді алгоритм RSA широко застосовується практично в усіх додатках, що використовують криптографію з відкритим ключем [1, 4].

Крім іншого, швидкодія програмної реалізації алгоритмів симетричного і асиметричного шифрування на сучасних апаратних платформах досліджена недостатньо. Так, в літературі можна зустріти відомості, що "RSA є надзвичайно повільним алгоритмом … на програмному рівні DES принаймні у 100 разів швидший, а на апаратному – в 1000–10000 разів, залежно від виконання." [5]

У зв'язку з цим ефективна і безпечна реалізація криптографічних алгоритмів для потреб прикладного ПЗ залишається актуальною, крім того, важливим елементом такої реалізації є її швидкодія. Тому метою цієї роботи було дослідження та порівняння швидкодії програмної реалізації алгоритмів DES і RSA для обґрунтованого вибору алгоритму залежно від обсягу вхідних даних.

Швидкодія алгоритмів

Стверджується, що найшвидшою апаратною реалізацією DES є мікросхема, розроблена в Digital Equipment Corporation [6]. Вона підтримує режими ECB і CBC та основана на вентильній матриці GaAs, що складається з 50000 транзисторів. Дані можуть шифруватись та дешифруватись зі швидкістю 1 Гбіт/с, обробляючи 16,8 мільйонів блоків на секунду. Параметри швидкодії сучасних комерційних мікросхем DES коливаються в межах від 0,64 Мбайт/с до 200 Мбайт/с [2].

У [2, 7] наведено результати та оцінки швидкості програмної реалізації DES для різних мікропроцесорів Intel і Motorola; так, для процесора Intel 80486 66 МГц швидкість шифрування становить 336 кбайт/с, а для процесорів DEC Alpha 4000/610 та HP 9000/887 – 1,17 Мбайт/с і 1,50 Мбайт/с відповідно.

Апаратно RSA приблизно в 1000 разів повільніший за DES. Швидкодіючі апаратні 512-бітові модулі забезпечують швидкість шифрування на рівні 64 кбіт/с [8]. Готуються ІС, здатні виконувати такі операції зі швидкістю 1 Мбайт/с. Існують також мікросхеми, які виконують 1024-бітове шифрування RSA.

Програмно DES приблизно у 100 разів швидший за RSA. Ці числа можуть незначно змінитись при зміні технології, однак RSA ніколи не досягне швидкості симетричних алгоритмів [2]. Швидкість програмного шифрування RSA становить 1,56–2,0 кбайт/с залежно від розміру блока повідомлення при 8-бітовому відкритому ключі (при використанні процесора SPARC II) [9].

Шифрування RSA виконується набагато швидше, якщо правильно вибрати значення e. Трьома найчастішими варіантами є 3, 17 та 65537 (2^{16}+1). (Двійкове представлення 65537 містить тільки дві одиниці, тому для піднесення до степеня треба виконати тільки 17 множень.) Стандарт X.509 радить 65537 [10], PEM рекомендує 3 [11], а PKCS#1 – 3 або 65537 [4]. Не існує жодних проблем безпеки, пов'язаних з використанням в якості e будь-якого з цих трьох значень (за умови доповнення повідомлення випадковими числами), навіть якщо одне й те саме значення e використовується цілою групою користувачів [2].

Операції із закритим ключем можна пришвидшити за допомогою китайської теореми про остачі [1], якщо зберегти значення p і q, а також додаткові значення: $d \bmod(p-1)$, $d \bmod(q-1)$ і $q^{-1} \bmod p$ [2, 12]. Ці додаткові числа можна легко обчислити за закритим і відкритим ключами.

Програмна реалізація алгоритмів

Для забезпечення гнучкості та апаратної незалежності програмної реалізації та можливості коректного порівняння результатів тестування швидкодії програмної реалізації алгоритмів, нами було використано стандартні криптопровайдери, які реалізують зазначені алгоритми та входять до складу Microsoft CryptoAPI, а сама програмна реалізація здійснена мовою C# на платформі Microsoft .NET.

Microsoft .NET — це платформа і набір технологій, програми якої не компілюються – вони перетворюються на проміжну мову Microsoft Intermediate Language (MSIL), і виконуються під управлінням віртуальної машини, що має назву Common Language Runtime (CLR). Такий підхід має ряд переваг, оскільки у такому разі всі мови мають доступ до єдиного набору сервісів [13].

Архітектура Microsoft .NET може бути описана так:

- наявність широкого набору сервісів, доступних з різних мов програмування;
- сервіси реалізовані у вигляді проміжного коду, незалежного від базової архітектури;
- сервіси виконуються під управлінням віртуальної машини CLR, яка також управляє ресурсами і стежить за виконанням програм.

Ключовою особливістю виконання програм в середовищі .NET є JIT (Just-In-Time) компіляція. JIT-компіляція полягає в тому, що проміжний код (CIL – Common Intermediate Language) знаходиться в збірці і запускається тут же й компілюється в машинний код, на який потім передається управління.

Така схема виконання програм в середньому є ефективнішою, ніж інтерпретація інструкцій CIL, оскільки втрата часу на попередню компіляцію CIL-коду компенсується високою швидкістю роботи коду, що відкомпілювався.

Важливим моментом є наявність технології Garbage Collector (збірка сміття) [13]. Її користь особливо помітна в задачах шифрування інформації, коли необхідно знищити таємні дані з оперативної пам‘яті комп'ютера після закінчення роботи програми.

Для реалізації алгоритмів DES та RSA були використані функції CryptoAPI [14]. Реалізація всіх алгоритмів (шифрування, цифрового підпису тощо) повністю винесена із складу самого CryptoAPI і реалізується в окремих незалежних динамічних бібліотеках – "криптопровайдерах" (Cryptographic Service Provider – CSP).

Криптопровайдером називають незалежний модуль, що забезпечує безпосередню роботу з криптографічними алгоритмами. Кожен криптопровайдер повинен забезпечувати:

- реалізацію стандартного інтерфейсу криптопровайдера;
- роботу з ключами шифрування, призначеними для забезпечення роботи алгоритмів, специфічних для цього криптопровайдера;
- неможливість втручання третіх осіб в схему роботи алгоритмів.

Криптопровайдери реалізуються у вигляді динамічно завантажуваних бібліотек (DLL). Втрутитись у хід виконання алгоритму, реалізованого в криптопровайдері, достатньо важко, оскільки компоненти криптосистеми Windows повинні мати цифровий підпис. У криптопровайдері відсутні можливості зміни алгоритму через встановлення його параметрів. Так розв’язується задача забезпечення цілісності алгоритмів криптопровайдера.

CryptoAPI надає методи для роботи з сеансовими (симетричними ключами) і з відкритими ключами [14]. Усі ключі управляються і використовуються за допомогою певних ідентифікаторів, і додаток не отримує відкритого доступу до них.

Сеансові ключі. Сеансові ключі змінюються для кожного сеансу, і криптопровайдер не зберігає їх на диски чи іншу пам'ять. В ситуаціях, коли додатку потрібно отримати в якомусь вигляді ключ, в системі є функції обміну ключами.

Сеансові ключі можуть генеруватися не тільки у випадковий спосіб, але і за якими-небудь даними. В останньому випадку гарантується, що один і той самий CSP буде повертати один і той самий ключ за однаковими даними. Така можливість використовується для генерування ключів на основі паролів.

Відкриті ключі. Відкриті ключі і їхні закриті частини для асиметричного шифрування, на відміну від сеансових ключів, зберігаються криптопровайдером в контейнерах ключів в зашифрованому вигляді. Реалізовані вони можуть бути різними способами: файли на дисках, ключі в реєстрі тощо. Проте цими ключами також можна обмінюватися. Для цього використовуються ті самі функції, що і для сеансових ключів.

Довжини ключів RSA і алгоритмів обчислення цифрового підпису та обміну ключами можуть коливатися від 384 до 16384 біт з інтервалом в 8 біт.

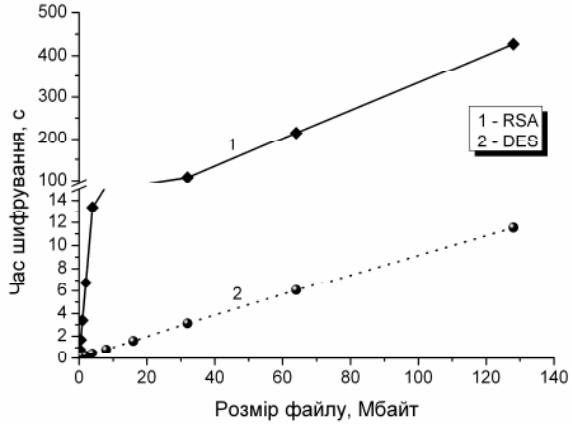
Результати тестування

За допомогою програми генерації тестових файлів було створено множину файлів випадкового вмісту розміром від 4 кбайт до 128 Мбайт. Експерименти здійснювалися тричі з метою мінімізації випадкової складової похибки вимірювання. Вимірювання часу операцій шифрування / розшифрування визначалось з точністю ±5 мс, а після отримання всіх трьох значень для кожного з тестових файлів час операцій усереднювався. Дослідження цього алгоритму виконували на комп'ютері з процесором Intel Celeron D з тактовою частотою 3.2 ГГц, та обсягом оперативної пам'яті 2 Гбайт. Результати тестових експериментів наведено в таблиці.

Усереднені значення тестових результатів

Розмір файлу, кбайт	Час операції шифрування, мс.	
	RSA	DES
4	15	46
8	31	93
16	46	109
32	109	117
64	218	125
128	421	125
256	828	140
512	1687	156
1024	3390	203
2048	6734	296
4096	13390	481
8192	26859	828
16384	56828	1593
32768	107156	3140
65536	214953	6046
131072	427312	11546

Методом найменших квадратів було апроксимовано залежність часу шифрування від розміру файла лінійною функцією. Ці залежності наведено на рис. 1 (точками позначено результати тестування, а суцільними лініями – розраховані функції). В усіх випадках дані з високою точністю апроксимуються лінійною залежністю (коефіцієнт кореляції не менший ніж 0,99903), а коефіцієнт *k*, що характеризує зростання часу шифрування, зі зростанням розміру файла становить для алгоритму DES 0,0889±0,0006, а для алгоритму RSA – 3,299±0,009. Незважаючи на те, що швидкість роботи алгоритму RSA сильно зростає зі збільшенням розміру вхідного тексту, з рисунка та таблиці видно, що при розмірі вхідних даних до 256 кбайт алгоритм поступається DES за швидкістю менше ніж у 6 разів, а час шифрування є меншим, ніж 1 с.



Залежності часу шифрування від розміру вхідних даних алгоритмів DES та RSA

Швидкодія програмної реалізації алгоритму визначалась як коефіцієнт нахилу прямої, що описує залежність розміру відкритого тексту від часу шифрування. Розрахована у такий спосіб

швидкість шифрування цієї реалізації алгоритму RSA становить 306,4±0,6 кбайт/с, а алгоритму DES – 11350±60 кбайт/с. Треба зазначити, що отримана швидкодія програмної реалізації алгоритму RSA значно перевищує кращі показники апаратної реалізації кінця 90-х, а швидкодія DES є на рівні деяких апаратних реалізацій алгоритму. Крім того, з цих результатів видно, що швидкодія програмної реалізації алгоритму RSA в 37 разів меншою порівняно з алгоритмом DES.

Висновки

У роботі виконано дослідження швидкодії програмної реалізації алгоритмів RSA та DES на процесорі Intel Celeron D 351. Швидкість шифрування алгоритму RSA становить 306,4±0,6 кбайт/с, що значно перевищує кращі показники апаратної реалізації кінця 90-х. Показано, що швидкодія програмної реалізації алгоритму є лінійною в усьому дослідженому діапазоні розмірів вхідних файлів, а коефіцієнт зростання часу шифрування зі зростанням розміру файлу становить 3,299. Показано доцільність використання алгоритму RSA для задач шифрування даних обсягом до 256 кбайт (в цьому діапазоні різниця з алгоритмом DES є меншою ніж у 6 разів, а абсолютне значення часу шифрування є меншим за 1 с). Швидкість шифрування алгоритму DES становить 11350±60 кбайт/с, що є на рівні деяких апаратних реалізацій алгоритму, а швидкість роботи реалізації алгоритму залишається лінійною в діапазоні розмірів вхідного тексту до 128 Мбайт. Показано, що порівняно з алгоритмом DES швидкість шифрування програмної реалізації RSA є в 37 разів меншою, що є на порядок кращим від літературних даних.

У цій роботі показано, що засоби CryptoAPI в поєднанні з платформою Microsoft .NET є ефективним, гнучким та архітектурно незалежним інструментом для створення ефективних та безпечних модулів криптографічного захисту інформації в середовищі Windows. Завдяки цифровим підписам криптопровайдерів, використанню CIL-коду та наявності Garbage Colector цей інструмент дає змогу реалізувати вимоги до захищеності криптографічних модулів ПЗ та забезпечити їхню достатню швидкодію.

При розробленні компонентів захисту інформації, наприклад, для Веб-орієнтованої системи теплового проектування [15], отримані результати дають змогу стверджувати таке: при шифруванні потоку даних на прикладному рівні необхідно застосовувати симетричні алгоритмі шифрування у поєднанні з алгоритмами обміну ключів або асиметричного шифрування для передавання ключів; при шифруванні сегментів транспортного рівня можна використовувати тільки асиметричні алгоритми шифрування забезпечуючи конфіденційність, аутентифікацію та цифровий підпис – втрати у швидкості шифрування будуть незначними, а час шифрування буде меншим ≈ 0,3 с (для симетричного алгоритму DES – 0,15 с).

1. Столлингс В. Криптография и защита сетей: принципы и практика. – М.: Вильямс, 2001. – 672 с. 2. Б. Шнайер Прикладная криптография: Протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2003. – 816 с. 3. RSA Laboratories, "PKCS#1: RSA Encryption Standard", version 1.5, Nov 1993. 4. W. Diffie The First Ten Years of Public Key Cryptography // Proceedings of the IEEE, Vol. 76 (1988), No 5, pp. 560–577. 5. С. Макаренко, А. Брусникин Алгоритмы шифрования для передачи данных в открытых сетях // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. – вип. 2 (2001). – С. 206–211. 6. H. Erble A high-speed DES implementation for network applications // Advances in cryptology CRYPTO'92 Proceedings, Springer-Verlag, pp. 521–539. 7. G. Garon, R. Outerbridge DES watch: an examination of the sufficiency of the Data Encryption Standard for financial institution information security in the 1990's // Cryptologia, Vol. 15, No3, (1991), pp. 177–193. 8. E.F. Brickell Survey of Hardware Implementations of RSA // Advances in Cryptology CRYPTO'89 Proceedings, Springer-Verlag, 1990, pp. 368–370. 9. J.B. Lacy, D.P. Mitchell, W.M. Schell CryptoLib: Cryptography in Software // UNIX Security Symposium Proceedings, USENIX Association, 1993, pp. 1–17. 10. CCITT, Recommendation X.509, "The Directory Authentication Framework", Consultation Committee, International Telephone and Telegraph, International Telecommunications Union, Geneva, 1989. 11. D. Balenson Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers // RFC 1423, February 1993. 12. J.-J. Quisquater, C. Couvreur Fast Decipherment Algorithm for RSA Public Key Cryptosystem // Electronic Letters, Vol. 18 (1982), pp. 155–168. 13. Рихтер Дж. Программирование на платформе Microsoft .NET Framework. – М.: Русская редакция, 2005. – 512 с. 14. Щербаков А., Домашев А. Прикладная криптография. Использование и синтез криптографических интерфейсов. – М: Русская редакция, 2003. – 406 с. 15. Fedasyuk D., Levus Y., Yaroshchuk N., Petrov D. The mechanism of the relational database architecture for the microelectronic components thermal design in the Internet/ Proceedings of International Conference MIXDES`2005, Krakow, Poland, 2005, pp.261–266.