ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА КОМП'ЮТЕРНА ТЕХНІКА

УДК 681.3.06
В. В. Сокирук, асп.;
В. А. Лужецький, д. т. н., проф.

ПРОГРАМНА РЕАЛІЗАЦІЯ БЛОКОВОГО СИМЕТРИЧНОГО ШИФРУ НА ОСНОВІ АРИФМЕТИЧНИХ ОПЕРАЦІЙ ЗА МОДУЛЕМ 2^n

Розглянуто особливості програмної реалізації блокового симетричного шифру (БСШ), побудованого на основі арифметичних операцій за модулем 2^n. Наведено алгоритми реалізації використовуваних основних операцій. Проведено тестування та порівняльний аналіз продуктивності програмної реалізації БСШ мовою Ansi C на сучасних 32-х розрядних процесорах для платформи IA-32.

Вступ

Важливою вимогою, що висувається до сучасного блокового симетричного шифру (БСШ), є наявність простої та одночасно ефективної програмної реалізації за допомогою розповсюджених мов програмування. Особлива увага приділяється продуктивності програмної реалізації БСШ на 32-х розрядних процесорах, що є найрозповсюдженішими у світі.

В рамках конкурсу AES [1, 2] проводилась оцінка ефективності програмної реалізації та продуктивності на різних платформах кожного БСШ, що був проданий до розгляду, і перевага серед БСШ з рівним ступенем криптографічної стійкості віддавалась тим, продуктивність та простота реалізації яких була вищою.

В роботах [3, 4] запропоновано БСШ на основі арифметичних операцій за модулем, який відповідає сучасним тенденціям розвитку БСШ і побудований з використанням операцій, що можуть ефективно виконуватись сучасними 32-х та 64-х розрядними процесорами.

Метою даної роботи є підвищення ефективності програмної реалізації БСШ на 32-х розрядних процесорах шляхом використання цілочисельної арифметики за модулем 2^n.

Для досягнення поставленої мети розв'язуються такі **задачі**:

1. Пошук оптимальної програмної реалізації основних операцій, що використовуються у БСШ, запропонованому в роботах [3, 4].
2. Аналіз продуктивності програмної реалізації даного БСШ мовою Ansi C на сучасних 32-х розрядних процесорах.

БСШ на основі арифметичних операцій за модулем

БСШ, запропонований в роботах [3, 4], розглядає блоки даних, підключі секретного ключа та блоки шифротексту як n-розрядні цілі числа, де n — розмір блока, що має бути кратним розміру машинного слова процесора для створення ефективної програмної реалізації. Основним елементом БСШ, який забезпечує перемішування та розсіювання розрядів відкритого тексту, є операція множення за модулем 2^n. Криптографічна стійкість БСШ забезпечується поєднанням операцій з різних груп: множення за модулем 2^n, додавання за модулем 2 та перестановки блоків розрядів n-розрядного цілого числа.

В загальному випадку пропонується здійснювати зашифрування, виконуючи такі обчислення:

$$C = \left(\left(\left(\left(M \oplus K_1 \right) \times K_2 \right)_{\mathrm{mod}\, 2^n} \right)^{\leftrightarrow k} \times K_3 \right)_{\mathrm{mod}\, 2^n}, \qquad (1)$$

де M, C — блоки відкритого та зашифрованого тексту відповідно; K_1, K_2, K_3 — підключі, які отримуються з секретного ключа K шляхом використанням процедури розгортання ключа, що описана в роботі [4]; $\leftrightarrow k$ — операція перестановки k-розрядних блоків n-розрядного числа.

В роботі [4] проведено статистичний аналіз БСШ з процедурою зашифрування (1) та показано, що статистично безпечним даний БСШ є у випадку використання перестановки блоків розрядів при $k = 1$, що відповідає дзеркальній перестановці розрядів n-розрядного цілого числа. Окрім перестановки при $k = 1$ в даній роботі також розглядаються варіанти побудови БСШ з використанням перестановок при $k = 32$ та $k = 8$, оскільки програмна реалізація таких перестановок є простішою, а для перестановки при $k = 8$ виявлені в роботі [4] статистичні аномалії є незначними, що робить можливим використання БСШ з такими перестановками для вирішення певних задач, в яких важливим чинником є швидкодія БСШ при задовільному рівні криптографічної стійкості.

Для розшифрування необхідно виконати зворотні обчислення

$$M = \left(\left(\left(C \times K_3^{-1} \right)_{\mathrm{mod}\, 2^n} \right)^{\leftrightarrow k} \times K_2^{-1} \right)_{\mathrm{mod}\, 2^n} \oplus K_1, \qquad (2)$$

де K_2^{-1}, K_3^{-1} — числа, що є оберненими відповідно до K_2 та K_3 за модулем 2^n.

Необхідність обчислення K_2^{-1} та K_3^{-1} накладає додаткові вимоги на підключі K_2 та K_3, які мають бути взаємно простими з числом 2^n, тобто непарними. Значення підключів для процедури розшифрування, що описується виразом (2), обчислюються на етапі розгортання ключа для розшифрування.

Особливості програмної реалізації основних операцій

Програмна реалізація БСШ на основі арифметичних операцій за модулем полягає у реалізації таких операцій:

1) додавання n-розрядних цілих чисел за модулем 2;
2) перестановки блоків по k розрядів n-розрядних цілого числа при $k = 32$, $k = 8$ або $k = 1$;
3) множення за модулем 2^n n-розрядних цілих чисел;
4) знаходження числа, оберненого за модулем 2^n.

Реалізація операції додавання n-розрядних цілих чисел за модулем 2 є простою і полягає у використанні 4-х операцій додавання 32-х розрядних цілих чисел за модулем 2.

Операції перестановки при $k = 32$ та $k = 8$ також реалізуються просто. В першому випадку необхідно виконати дві операції обміну 32-х розрядними словами, які реалізуються або шляхом копіювання ділянок пам'яті, або звичайним присвоєнням. Для виконання такої операції необхідне використання додаткової пам'яті розміром одне машинне слово для зберігання проміжного результату.

Перестановка по 8 розрядів найшвидше виконується таким чином: спочатку виконується перестановка при $k = 32$, а далі для кожного 32-х розрядного слова — операція дзеркальної перестановки байтів, для швидкого виконання якої призначена інструкція процесора BSWAP. Коли використання машинної команди неможливе (наприклад, для реалізації перестановки при $k = 8$ на мові Java), така перестановка може бути виконана з використанням операцій зсуву, циклічного зсуву та логічних операцій.

Дзеркальна перестановка розрядів n-розрядного цілого числа (при $k = 1$) може бути виконана двома способами:

1) виконується перестановка блоків по 32, 16 або 8 розрядів, далі для кожного блока виконується дзеркальна перестановка розрядів за допомогою логічних операцій та операцій зсуву;

2) виконується перестановка блоків по 8 розрядів, далі для кожного блока виконується таблична підстановка попередньо обчисленого значення.

В другому випадку необхідно зберігати таблицю з 256-ти елементів попередньо обчислених

дзеркальних перестановок розрядів 8-розрядних цілих чисел. Такий підхід є ефективнішим, якщо немає обмежень у використанні додаткової пам'яті.

Для реалізації операції множення n-розрядних цілих чисел за модулем використовується алгоритм множення «у стовпчик». При цьому n-розрядне число представляється як масив з цілих w-розрядних чисел, над якими виконуються операції множення та додавання, де w — довжина машинного слова процесора.

Нехай A, B — n-розрядні цілі числа, A_i — i-й w-розрядний елемент числа A, B_j — j-й w-розрядний елемент числа B, $q = \frac{n}{w}$. Тоді операцію множення чисел A і B за модулем 2^n можна представити у такому вигляді:

$$\begin{array}{ccccc} & \times A_{q-1} & \cdots & A_1 & A_0 \\ & B_{q-1} & \cdots & B_1 & B_0 \\ \hline & A_{q-1}\cdot B_0 & \cdots & A_1\cdot B_0 & A_0\cdot B_0 \\ + & A_{q-2}\cdot B_1 & \cdots & A_0\cdot B_1 & \\ + & \cdots & \cdots & & \\ + & A_0\cdot B_{q-1} & & & \\ \hline & C_{q-1} & \cdots & C_1 & C_0 \end{array} \qquad (3)$$

Всі дії при виконанні множення «у стовпчик» виконуються з перенесенням залишку від молодших до старших розрядів (блоків), тобто значення C_1 не може бути обчислене перед обчисленням значення C_0 і т. д. Таким чином виключається можливість паралельного виконання даної операції декількома процесорами або спеціалізованими обчислювальними пристроями, що може негативно вплинути на продуктивність програмної реалізації БСШ. Прискорити виконання операції множення за модулем 2^n можна за рахунок використання додаткової пам'яті для попереднього обчислення і зберігання добутків $A_i \cdot B_j$. В такому випадку на етапі множення можна використати переваги сучасних процесорів, здатних виконувати декілька арифметичних операцій за один такт. Також такий підхід значно спрощує програмний код і зменшує імовірність появи помилки при реалізації БСШ.

За наявності обмежень на використання додаткової пам'яті програмна реалізація операції множення за модулем 2^n потребує додатково лише $2w$ розрядів для виконання операцій множення та додавання зі збереженням перенесення, але є більш складною для реалізації і повільнішою при виконанні, оскільки містить більшу кількість операцій.

Для знаходження оберненого числа за модулем 2^n можна використати модифіковані версії розширеного алгоритму Евкліда [5], операцію ділення за модулем 2^n, що докладно описана в роботі [3], та алгоритм, запропонований в роботі [6]. Оскільки перші два алгоритми працюють над визначенням кожного розряду невідомого числа окремо і потребують значних обчислень, їх програмна реалізація є неефективною. В роботі [6] для обчислення оберненого числа за модулем пропонується використовувати більш швидкий алгоритм, який використовує властивості арифметичних операцій за модулем 2^n.

Нехай $\varphi(a, m)$ — число, що є оберненим до a за модулем m; r — залишок від ділення цілого числа a на число m. Тоді справедливе твердження

$$\varphi\left(a, m^2\right) = \left(2 - a\varphi(r, m)\right)\varphi(r, m). \qquad (4)$$

Таким чином, знаходження оберненого числа за модулем m^2 згідно з цим алгоритмом зводиться до обчислення оберненого числа за модулем m і виконання додаткових арифметичних операцій з m-розрядними числами. Вираз (4) використовується допоки не стане можливим просте обчислення значення $\varphi(r, m)$ за допомогою табличної підстановки. Оскільки K_2 та K_3 — непарні, достатньо зберігати таблицю з 128 елементів, в якій будуть міститись попередньо обчислені обернені значення за модулем 2^8 для всіх непарних цілих чисел з множини Z_{2^8}.

Наведений алгоритм містить арифметичні операції віднімання та множення чисел з різною роз-

рядністю. Якщо $n = 128$, для реалізації множення 64-розрядних чисел із збереженням старшої частини може бути використаний алгоритм множення за модулем 2^{128}. Арифметичні дії над числами меншої розрядності реалізуються на 32-х розрядних процесорах без використання додаткових алгоритмів.

Аналіз продуктивності програмної реалізації БСШ

Методика оцінки продуктивності програмної реалізації БСШ запропонована NIST в рамках конкурсу AES і докладно описана в роботах [1, 2]. Вона полягає у визначенні середньої кількості тактів процесора (або часу), необхідних для виконання таких операцій:

1) зашифрування блока даних;
2) розгортання ключа для зашифрування;
3) розшифрування блока шифротексту;
4) розгортання ключа для розшифрування.

Згідно з цією методикою, для визначення кількості тактів використовується інструкція RDTSC, що призначена для отримання значення внутрішнього лічильника тактів процесора. Для того, щоб зменшити вплив процесів, що виконуються паралельно, на кінцевий результат, перед виконанням обчислень відключаються внутрішні переривання, а також використовується інструкція процесора CPUID для запобігання отримання значення лічильника тактів до повного завершення виконання коду [1].

В проведених дослідженнях розглядався БСШ на основі арифметичних операцій за модулем 2^n при $n = 128$ з процедурою зашифрування (1), розшифрування (2) та процедурами розгортання ключа для зашифрування та розшифрування, що описані в роботі [4].

В табл. 1 наведено результати тестування програмної реалізації БСШ мовою Ansi C з використанням компілятора Microsoft compiler 6.0, з метою визначення кількості тактів, необхідної для виконання операцій на різних процесорах. Оскільки процедури зашифрування та розшифрування містять однакову кількість операцій і відрізняються тільки порядком їх виконання, різниця в кількості тактів, необхідних для зашифрування та розшифрування, або відсутня, або не перевищує кількох тактів процесора (що було також підтверджено при проведенні експериментів), тому результати для операцій зашифрування та розшифрування подаються одним значенням.

Таблиця 1

Кількість тактів, необхідних для виконання операцій

Виконувана операція	Кількість тактів				
	Intel Pentium II	Intel Pentium M	Intel Pentium 4	AMD Duron	AMD Athlon 64
Зашифрування/розшифрування при $k = 32$	263	211	277	190	172
Зашифрування/розшифрування при $k = 8$	264	215	279	192	173
Зашифрування/розшифрування при $k = 1$	330	282	481	323	316
Розгортання ключа для зашифрування	1038	850	1433	989	952
Розгортання ключа для розшифрування	1686	1504	2287	1562	1448

Як і очікувалось, БСШ з перестановками $k = 32$ та $k = 8$ розрядів є набагато швидшим, ніж з використанням дзеркальної перестановки розрядів. Причому різниця між варіантами БСШ з перестановками $k = 32$ та $k = 8$ дуже незначна. Це пояснюється тим, що операція перестановки байтів 32-х розрядного числа виконується дуже швидко з використанням інструкції процесора BSWAP.

Якщо кількість тактів, необхідних для зашифрування та розшифрування блока даних приблизно однакова, то операція розгортання ключа для розшифрування є повільнішою ніж операція розгортання ключа для зашифрування, оскільки додатково обчислюються значення K_2^{-1} та K_3^{-1}.

Найкращі результати отримані для процесорів Athlon 64 та Pentium M, причому якщо перший є

швидшим в операціях зашифрування та розшифрування при $k = 32$ та $k = 8$, то другий ефективніше виконує дзеркальну перестановку розрядів і відповідно процедури зашифрування та розшифрування при $k = 1$. Висока ефективність зазначених процесорів в 32-х розрядному режимі роботи досягається за рахунок низької латентності виконання арифметичних та логічних операцій, короткого конвеєра та наявності ефективного алгоритму попередження переходів та декодування інструкцій.

Низькі результати, отримані для процесорів Intel Pentium 4, пов'язані насамперед з високою латентністю операції множення (14 тактів для процесорів на ядрі Northwood і 10—11 — для процесорів на основі ядра Prescott) та найдовшим серед сучасних процесорів конвеєром (від 20 для ядра Northwood до 31 для ядра Prescott), що негативно впливає на продуктивність не оптимізованого спеціальним чином коду.

В роботі [1] аналізується ефективність програмної реалізації мовою Ansi C БСШ, що є фіналістами конкурсу AES, для різних тестових платформ з використанням компіляторів Borland C++ 5.0 та Microsoft Visual Studio 6.0. Серед наведених результатів є результати для процесора Pentium II, що робить можливим порівняння наведених в роботі [1] результатів із отриманими в даній роботі для аналогічного процесора Pentium II.

Для порівняння результатів продуктивності БСШ на основі арифметичних операцій за модулем 2^n та відомих БСШ, був обраний статистично безпечний варіант БСШ на основі арифметичних операцій за модулем з процедурою перестановки розрядів при $k = 1$, який є одночасно і найповільнішим.

Зауважимо, що таке порівняння є приблизним, оскільки продуктивність БСШ може відрізнятись навіть для процесорів одного сімейства з деякими відмінностями в побудові.

В табл. 2 наведена кількість тактів, необхідна для виконання операцій на процесорі Intel Pentium II для різних БСШ.

Таблиця 2

Кількість тактів, необхідних для виконання операцій на процесорі Intel Pentium II

Виконувана операція	Кількість тактів					
	БСШ на основі арифметичних операцій за модулем 2^n	MARS	RC6	Rijndael	Serpent	Twofish
Зашифрування	330	669	330	826	1281	800
Розшифрування	330	583	320	796	1122	628
Розгортання ключа для зашифрування	1038	4937	2283	1292	6947	9266
Розгортання ключа для розшифрування	1686	4938	2284	1722	6935	9249

Навіть з урахуванням усіх відхилень, що можуть виникнути внаслідок різниці двох тестових платформ, можна стверджувати, що продуктивність програмної реалізації мовою Ansi C БСШ на основі арифметичних операцій за модулем 2^n, яка не оптимізована спеціальним чином під той чи інший тип процесора з використанням команд асемблера тощо, приблизно відповідає продуктивності БСШ RC6, що є найшвидшим серед 5-ти фіналістів конкурсу AES, і вища за продуктивність інших фіналістів конкурсу БСШ, наведених в табл. 2.

Висновки

1. БСШ на основі арифметичних операцій за модулем 2^n, запропонований в роботах [3, 4], є простим і швидким для програмної реалізації мовами високого рівня.
2. Реалізація БСШ на основі арифметичних і логічних інструкцій, що швидко виконуються сучасними процесорами, дозволяє уникнути необхідності використання специфічних для певних процесорів команд з метою прискорення виконання тих чи інших операцій, і тим самим спростити програмну реалізацію, зробивши її одночасно універсальною для різних платформ.

3. Програмна реалізація мовою Ansi C блокового шифру, що розглядається в роботі, є високоефективною для сучасних 32-х розрядних процесорів.

СПИСОК ЛІТЕРАТУРИ

1. Lawrence E. Bassham III. Efficiency Testing of ANSI C Implementations of Round 2 Candidate Algorithms for the Advanced Encryption Standard // Proceedings of 3rd AES conference. New York, 2000 — P. 136—149.

2. Kazumaro Aoki, Helger Lipmaa. Fast Implementations of AES Candidates // Proceedings of 3rd AES conference. New York, 2000. — P. 106—120.

3. Сокирук В. В., Лужецький В. А. Блоковий симетричний шифр на основі модульної арифметики // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. Вип. 10. — 2005. — С. 122—126

4. Сокирук В. В., Лужецький В. А. Побудова статистично безпечного БСШ на основі арифметичних операцій за модулем // Інформаційні технології та комп'ютерна інженерія. — 2006. — № 1. — С. 158—163.

5. Петров А. А. Компьютерная безопасность. Криптографические методы защиты. — М.: ДМК, 2000. — 448 с.

6. Machado A. W. The nimbus cipher: A proposal for NESSIE // NESSIE Proposal. — 2000. — № 7.

Матеріали статті рекомендовані до опублікування оргкомітетом XIII Міжнародної конференції з автоматичного управління (Автоматика-2006, 25—28.09.2006 р.)

Надійшла до редакції 23.11.06
Рекомендована до друку 12.12.06

Сокирук Віталій Віталійович — аспірант, ***Лужецький Володимир Андрійович*** — завідувач кафедри.
Кафедра захисту інформації, Вінницький національний технічний університет