УДК-004.056(043.2)

АНАЛІЗ ЧАСОВИХ АТАК НА АПАРАТНИЙ ШИФРАТОР ПЕРСОНАЛЬНОГО ЗАСОБУ КРИПТОГРАФІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ ШИПКА

***А. Б. Петренко**, канд. техн. наук. доц.; ***О. С. Шматок**, канд. техн. наук. доц.

****С. А. Шматок**, д-р техн. наук, проф.; ***Є. О. Агеєнко**

*Національний авіаційний університет

**Національний технічний університет України «КПІ»

lizavetaageenko@gmail.com

Обґрунтовано наявність вразливості апаратного шифратора персонального засобу криптографічного захисту інформації (ПЗКЗІ) ШИПКА до часових атак. Докладно розглянуто алгоритми, які використовуються в криптосистемі RSA, а саме: алгоритм швидкого піднесення до степеня, алгоритм Монтгомері, Китайська теорема про залишки. Описано криптоаналіз RSA за допомогою часових атак. Запропоновано використання методів Blinding для ПЗКЗІ ШИПКА як засіб протидії часовим атакам на криптосистему RSA.

Ключові слова: ПЗКЗІ ШИПКА, апаратні шифратори, часові атаки, алгоритм швидкого піднесення до степеню, Китайська теорема про залишки, алгоритм Монтгомері, методи Blinding.

It has been substantiated that the personal means of cryptographic information security SHIPKA prone to timing attacks. Algorithms used in RSA cryptosystem, such as Simple Modular Exponentiation, Montgomery reduction, Chinese Remainder Theorem, have been considered. It has been reported how timing attacks can be performed. Blinding has been suggested to be used in SHIPKA as countermeasures to prevent timing attacks on RSA.

Keywords: the personal means of cryptographic information security SHIPKA, hardware encryption, timing attacks, Simple Modular Exponentiation, Montgomery reduction, Chinese Remainder Theorem, Blinding.

Вступ

На сьогодні дедалі більшого використання набувають апаратні засоби захисту інформації, незважаючи на те, що таке обладнання зазвичай є набагато дорожчим за аналогічні програмні засоби. Насамперед це пов'язано з тим, що апаратна реалізація криптоалгоритмів забезпечує більший ступінь надійності. Перевагами апаратних засобів шифрування є:

– гарантія цілісності алгоритму шифрування, адже криптографічне програмне забезпечення не захищене від дії програм, що несуть загрозу цілісності програмного засобу, тобто здатні змінити структуру криптоалгоритму, модифікувавши код програми;

– наявність апаратного датчика псевдовипадкових чисел, який використовується для генерації ключів криптоалгоритмів;

– зберігання ключів шифрування не в оперативній пам'яті комп'ютера (як у випадку з програмною реалізацією), а в пам'яті мікропроцесора шифратора;

– ідентифікація та автентифікація користувача до завантаження операційної системи;

– можливість реалізувати системи розмежування доступу до комп'ютера та захисту інформації від несанкціонованого доступу;

– застосування спеціалізованого мікропроцесора для виконання всіх перетворень, що розвантажує центральний процесор комп'ютера.

Одним з прикладів апаратних шифраторів є персональний засіб криптографічного захисту інформації (ПЗКЗІ), шифрування, ідентифікація, підпис, коди автентифікації) ШИПКА. Криптографічна функціональність ШИПКА містить шифрування, електронно-цифровий підпис (ЕЦП), хешування, генерацію ключів, довготривале зберігання ключів і сертифікатів. Реалізація криптографічних операцій у всіх випадках апаратна.

Постановка проблеми

Особливістю ПЗКЗІ ШИПКА є використання мікроконтролера ATMEGA128, що має швидкий апаратний помножувач. Саме це робить можливим реалізацію за прийнятний час операції модульної експоненти, яка, наприклад, застосовується в алгоритмі RSA.

Тактова частота ATMEGA128 — 16 МГц. Хоча мікроконтролер виконує більшість операцій за один такт, помножувачу необхідно мінімум два такти для множення двох 8-бітних чисел. Оскільки під час своєї роботи пристрою ШИПКА доводиться обробляти і числа порядку 2056 біт, а також використовувати спеціальні алгоритми для більшої швидкості обчислень, криптосистема може бути вразливою до часових атак.

Часові атаки (*Timing attacks*) відносять до атак сторонніми каналами, їх часто застосовують саме до апаратних засобів захисту. Під час такої атаки аналізується, як довго виконується певне криптографічне перетворення.

© Петренко А. Б., Шматок О. С., Шматок С. А., Агеєнко Є. О., 2014

Криптоаналітик фіксує тривалість операцій, порівнює їх та проводить статистичний аналіз, що дозволяє йому визначити залежність між вхідними, вихідними даними, а також отримати секретний ключ. Вразливості даного типу мають такі алгоритми, як: RSA, DSS, Diffie-Hellman, DES, AES, IDEA, RC5 [1].

Аналіз досліджень і публікацій

Уперше часова атака була реалізована в 1996 р. на алгоритм RSA. В 1998 р. американський криптолог Пол Кошер у своїй роботі «Timing Attacks on Implementations of Diffe-Hellman, RSA, DSS, and Other Systems» докладно описав часову атаку на RSA, а також виявив подібні вразливості й в інших криптографічних системах [1].

Досить тривалмй час вважалося, що вразливі тільки апаратні засоби захисту, проте стенфордські вчені Девід Брамлі та Ден Боне довели, що часові атаки можуть бути реалізовані і на програмне забезпечення. Їхня робота «Remote Timing Attacks are Practical» показала, яким чином можливо отримати секретні ключі криптографічної бібліотеки OpenSSL, що використовується при роботі з протоколом https [2].

Дослідженням часових атак займалися також китайські вчені Йонг Бін Джой та Денг Гуо Фенг. Вони проаналізували, як саме змінювалися методи реалізації часових атак, а також які засоби та алгоритми захисту з'явилися протягом 10 років з часу виникнення атаки. У своїй статті «Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing» вони дають класифікацію атакам сторонніми каналами, а також описують американський стандарт FIPS 140-2, що містить алгоритми протидії [3]. Використання подібних алгоритмів не є стовідсотковою гарантією захисту, проте значно ускладнює криптоаналіз.

Цілі

Мета статті — аналіз часових атак на апаратні засоби шифрування на прикладі ПЗКЗІ ШИПКА. Для дослідження був обраний алгоритм ЕЦП RSA. У зв'язку з тим, що дані щодо реалізації RSA на ПЗКЗІ ШИПКА мають обмежений доступ, необхідно розглянути всі можливі варіанти алгоритмів обчислення модульної експоненти, серед яких: алгоритм швидкого піднесення до степеня, Китайська теорема про залишки, алгоритм Монгомері. Після аналізу мають бути виявлені можливі шляхи вдосконалення криптосистеми для зменшення ймовірності успішної реалізації часової атаки на RSA.

Криптосистема має бути модифікована таким чином, щоб час виконання операції не залежав від розрядності даних або значення певних біт, а отже, питання розробки подібних алгоритмів для апаратних пристроїв шифрування є досі актуальним.

Алгоритм швидкого піднесення до степеня

ПЗКЗІ ШИПКА працює з RSA-512 та RSA-2056. Для генерації цифрового підпису виконується операція модульної експоненти:

$$S = S_A(m) = m^d \bmod N, \qquad (1)$$

де S — цифровий підпис; m — відкритий текст; d — закритий ключ; N — модуль.

Головна мета криптоаналітика — знайти d. Для підвищення швидкості обчислень використовується бінарний алгоритм, який також називають *алгоритмом швидкого піднесення до степеня*. Він представлений у вигляді функції:

```
1: int squareAndMultiply(m, d, n) {
2:  int R[w-1];
3:  int M[w-1];
4:  M[0] = 1;
5:  for (int i = 0, i < w, i++) {
6:      if (d[i] = = 1)
7:          R[i] = (M[i] * m)%n;
8:      else
9:          R[i] = M[i];
10:     M[i+1] = (sqr(R[k]))%n;
11:     }
12:     return R[w–1];
13:     }
```

де w позначає розрядність d.

Можна помітити, що залежно від біту закритого ключа d буде виконуватися або операція множення за модулем, або операція присвоєння. Перша буде займати набагато більше часу в мікропроцесора ніж друга. Криптоаналітик біт за бітом намагається визначити ключ. Атака передбачає, що для кожного $d[i]$ попередні біти $d[0]\ldots d[i-1]$ вже відомі [1].

Для знаходження d аналіз починається з першого біта, коли відомих бітів ще немає, і продовжується до останнього. Для кожної ітерації проводяться вимірювання, метою яких є знаходження трьох характеристик:

t — загальний час виконання функції;

c — час, витрачений на обробку вже відомих бітів;

D — час, витрачений на виконання рядка 7.

Значення c залежить від кількості відомих бітів, що дорівнюють 1, а D визначають як різницю часу, необхідного на обробку біту 1 і біту 0. Тоді час, витрачений на обробку всіх невідомих бітів, наступних за $d[i]$, дорівнюватиме різниці перших двох характеристик при $d[i] = 0$, або ж $(t - c - D)$ при $d[i] = 1$ [4].

Отримані значення є нормально розподіленими випадковими величинами, для яких криптоаналітик спочатку знаходить математичне сподівання $\mu(X)$ та стандартне відхилення $\sigma(X)$, а потім розраховує ймовірності:

$$P(m,t,c,D|d_i=1)=\varphi\left(\frac{(t-c-D)-\mu(t-c-D)}{\sigma(t-c-D)}\right); \quad (2)$$

$$P(m,t,c,D|d_i=0)=\varphi\left(\frac{(t-c)-\mu(t-c)}{\sigma(t-c)}\right), \quad (3)$$

де $\varphi(x)$ — функція стандартного нормального розподілу:

$$\varphi(x)=\frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}}. \quad (4)$$

Використовуючи теорему Байєса, вирази (2) і (3) можна об'єднати так, щоб криптоаналітик отримував імовірність того, що $d_i=1$ при відомих значеннях m,t,c,D:

$$P(d_i=1|m,t,c,D)=\frac{P(m,t,c,D|d_i=1)}{\sum_{v[0,1]}P(m,t,c,D|d_i=v)}; \quad (5)$$

$$P(d_i=1)=\frac{\prod_{i=0}^{w-1}P(d_i=1|m_i,t_i,c_i,D_i)}{\sum_{v[0,1]}\prod_{i=0}^{w-1}P(d_i=v|m_i,t_i,c_i,D_i)}. \quad (6)$$

Таким чином, після всіх вимірювань та розрахунків біт за бітом отримують секретний ключ RSA.

Китайська теорема про залишки

Крім бінарного алгоритму, можливе використання Китайської теореми про залишки, що також застосовується для спрощення обчислення модульної експоненти. Цей алгоритм є дуже простим у реалізації, проте досить ефективним. Усі обчислення виконуються не за модулем N, а окремо за модулем p та q. Спочатку розраховують значення:

$$d_p=d \bmod (p-1); \quad (7)$$

$$d_q=d \bmod (q-1). \quad (8)$$

Для обчислення цифрового підпису $S_A(m)$ використовують $S_p(m)$ та $S_q(m)$:

$$S_p(m)=m^{d_p} \bmod p; \quad (9)$$

$$S_q(m)=m^{d_q} \bmod q. \quad (10)$$

Оскільки p та q значно менші N, розрахунки займають менше часу та ресурсів процесора. Криптоаналітик може отримати ці два параметри за допомогою часової атаки [5]. Вираз (9) буде обчислюватися тільки тоді, коли $m^{d_p}>p$, інакше відповідь буде дорівнювати m^{d_p}. У першому випадку операція займатиме набагато більше часу ніж у другому. Надсилаючи різні дані на вхід, поступово можна отримати значення p та q.

Алгоритм Монтгомері

В алгоритмі Монтгомері для спрощення розрахунків переходять до величин m' та S', з останньої в кінці обчислень отримають S. Для того, щоб працювати з числами за модулем N, підбирають певне значення R, що задовольняє дві умови: $R>N$; $НСД(R,N)=1$. Для одержання коректного результату R має обов'язково відповідати цим критеріям.

Наступні параметри мають бути обчислені під час ініціалізації, кожен буде використаний декілька разів:

– $R^{-1} \bmod N$ використовують під час модульного множення та отримання вихідних значень;

– $N'=N \bmod R$ збільшує ефективність розрахунків;

– $R \bmod N$ обчислюється для конвертування вхідних величин.

Деяке число a, таке що $0 \le a < N$, буде зведене до виду Монгомері під час операції:

$$a'=aR \bmod N. \quad (11)$$

Оберненою дією є:

$$a=a'R^{-1} \bmod N. \quad (12)$$

Далі виконуються такі обчислення:

$$c=am \bmod N; \quad (13)$$

$$c'=cR=aRmRR^{-1}=a'm'R^{-1} (\bmod N) \quad (14)$$

Цей алгоритм можна представити у вигляді функції:

```
1: int montgomery(c′, N′, R) {
2: m=((c′%R) N′)% R;
3: t=( c′+m*N)/R;
4: if (t>=N)
5: return t-N;
6: else
7: return t;
8: }
```

Час виконання цієї функції залежить від вхідного значення c', оскільки він впливає на умову $t >= N$, за якої виконується додаткове віднімання. Застосовується формула (6), тільки вже для обчислення ймовірності того, чи буде відбуватися ще одне додаткове віднімання для модульного множення випадкової змінної $u \in Z_n$.

Цей вираз поширюється і на множину Z_p і має вигляд:

$$ps_i(u) = \frac{u \bmod p}{2R}. \quad (15)$$

Далі застосовується той самий принцип, що й під час криптоаналізу алгоритму Китайської теореми про залишки.

При обчисленні операції за модулем p та q значення результату будуть збільшуватися в разі збільшення $u(u < p)$, проте далі відбудеться різкий спад, коли u «перестрибне» через p.

Атака складається з трьох фаз [4].

Спочатку знаходять інтервал $[u_1, u_2]$, якому можуть належати p та q. Під час другої фази інтервал зменшують і обирають тільки прості числа.

Остання фаза — це розрахунок для кожного з них та N найбільшого спільного дільника, поки це значення не буде дорівнювати u. Тоді вважають, що $p = u$, знаходять другий простий множник модуля і обчислюють секретний ключ.

Алгоритм протидії часовим атакам

Найбільш очевидний шлях запобігання атакам за часом полягає в тому, щоб усі дії та операції займали один і той самий час. На жаль, це часто буває важко реалізувати.

Створення спеціального програмного забезпечення таким чином, щоб воно виконувалося у фіксований час, досить важко, тому що оптимізація компілятора, попадання в програмний кеш, час виконання інструкцій і інші фактори можуть внести несподівані коливання під час виконання.

Якщо для затримки результатів до певного часу використовується таймер, то такі чинники, як «відгук системи» (*responsiveness*) або енергоспоживання, можуть служити ознаками фактичного закінчення операції. Крім того, реалізації з фіксованим часом будуть повільнішими, тому що тривалість усіх операцій має дорівнювати часу виконання найбільш повільної.

Тому було запропоновано розглянути методи Blinding, що використовуються для приховування цифрових підписів [1]. Вони можуть бути пристосовані до криптосистеми RSA.

Головна мета – модифікація вхідних даних функції модульного піднесення до степеня. Після цього криптоаналітик не може аналізувати часові характеристики, варіюючи відкритий текст, адже вони більше не пов'язані. Перед обчисленням модульної експоненти потрібно вибрати випадкову пару:

$$\nu_f^{-1} = \nu^x \bmod N. \quad (16)$$

Для RSA найкраще вибрати випадкове ν_f, взаємно просте з N, потім розрахувати

$$\nu_i = \left(\nu_f^{-1}\right)^e \bmod N, \quad (17)$$

де e — відкрита експонента.

Перед модульним піднесенням до степеня вхідні дані повинні бути помножені на $\nu_i (\bmod N)$, потім результат корегується множенням на $\nu_f (\bmod N)$.

При цьому система повинна відхиляти повідомлення, що дорівнюють $0 (\bmod N)$.

Обчислення інверсії mod N є повільним, тому не практично обирати нову випадкову пару (ν_i, ν_f) для кожної нової експоненти. Більше того, сам розрахунок виразу (16) може піддаватися часовому аналізу.

Проте пари (ν_i, ν_f) не мають використовуватися повторно, так як вони можуть бути розкриті часовим аналізом.

Розв'язком цієї проблеми є модернізація (ν_i, ν_f) шляхом піднесення до квадрата: $\nu'_i = \nu_i^2$ та $\nu'_f = \nu_f^2$. Ці значення можуть бути обчислені завчасно.

Якщо (ν_i, ν_f) тримати в секреті, то криптоаналітик не володіє ніякою корисною інформацією щодо вхідних даних функції модульної експоненти. Він може визначити тільки загальний розподіл часу для операції піднесення до степеня, а він близький до нормального, тож 2^w експонент не можна розпізнати.

Висновки

Апаратний шифратор ПЗКЗІ ШИПКА, як і більшість апаратних засобів захисту інформації, має вразливості до часових атак. Як показали дослідження, часові атаки є досить простими у реалізації.

Криптоаналітики можуть застосовувати дані принципи для будь-якої криптосистеми, де залежно від вхідних даних відбуваються умовні переходи з різними за складністю та часом виконання операціями.

Найбільш очевидний шлях запобігання атакам за часом полягає в тому, щоб усі дії та операції займали один і той самий час. Проте реалізації з фіксованим часом будуть повільнішими, оскільки тривалість усіх операцій має дорівнювати часу виконання найбільш повільної.

Отже, були розглянуті методи приховування цифрових підписів *Blinding* для криптосистеми RSA. Вдосконалений алгоритм дає змогу змінювати вхідні дані функції модульного піднесення до степеня таким чином, що криптоаналітик

більше не може визначити зв'язок між відкритим текстом і часовими характеристиками.

Тож, для ПЗКЗІ ШИПКА доцільно модифікувати всі реалізовані криптоалгоритми таким чином, щоб час виконання операції не залежав від розрядності даних або значення певних біт. Це дозволить значно ускладнити для зловмисника завдання криптоаналізу.

ЛІТЕРАТУРА

1. Kocher Paul C. Cryptoanalisis of Diffie Hellman, RSA, DSS, and other cryptosystem using timing attacks / Paul C. Kocher // Advances in cryptology. — 1995. — № 15. — P. 171–183.

2. Brumley David. Remote timing attacks are practical / David Brumley and Dan Boneh // USENIX Security. — 2003. — № 12. — P. 3–14.

3. Zhou Yong Bin. Side-Channel Attacks: Ten Years After Its Publication and the Impacts on Cryptographic Module Security Testing / Yong Bin Zhou, Deng Guo Feng // Physical Security Testing Workshop. — 2005. — № 3. — P. 26–60.

4. Mark van Cuijk Timing Attacks on RSA / Mark van Cuijk // Phedny. — 2009. — № 20. — P. 1–5.

5. Schindler Werner. A timing attack against rash with the chinese remainder theorem / Werner Schindler // Cryptographic Hardware and Embedded System. — 2000. — № 2. — P. 109–124.

Стаття надійшла до редакції 25.04.2014.