УДК 004.056:061.68

Совин Я. Р., Хома В. В., Тишик І. Я., Чінка В. М., Решетар Я. В. (НУ «ЛП», Україна)

ЕФЕКТИВНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ДСТУ ГОСТ 28147-89 ДЛЯ 8/16/32-БІТНИХ ВБУДОВАНИХ СИСТЕМ

Вступ

Необхідність захисту інформації у вбудованих системах привела до інтенсивних досліджень шляхів ефективної реалізації криптографічних алгоритмів, за умови обмежень, які накладаються цими системами. Ресурси вбудованих систем обмежені продуктивністю процесорного ядра, споживаною потужністю, розміром доступної постійної та оперативної пам'яті. Прикладом можуть бути безпровідні сенсорні мережі (БСМ), RFID-мітки, інтелектуальні карти, OTP-токени, системи охоронно-пожежної сигналізації і контролю доступу, системи промислово-побутової автоматизації і моніторингу тощо. Це привело до появи так званої lightweight-криптографії, яка займається розробленням та дослідженням криптоалгоритмів для вбудованих систем з обмеженими ресурсами, а самі алгоритми отримали назву lightweight-алгоритми [1, 2, 3].

У багатьох вбудованих системах, де ціна та витрати енергії виходять на перший план, обчислювальна потужність сконцентрована у недорогих центральних процесорах, які входять до складу мікроконтролерів загального призначення. Тому існує потреба у криптоалгоритмах, які б ефективно функціонували при реалізації на цих платформах.

Аналіз останніх досліджень і публікацій

Критеріями оцінки криптографічних алгоритмів з огляду на придатність використання у вбудованих системах є необхідний розмір внутрішньої пам'яті (коду) для їх реалізації та час виконання (продуктивність). Перший критерій зумовлений тим, що розмір коду програми безпосередньо впливає на вартість мікропроцесора, який переважно є найдорожчим компонентом системи. Час виконання є критичним з огляду на енергоспоживання, оскільки, як правило, у вбудованих системах центральний процесор більшу частину часу знаходиться в режимі пониженого енергоспоживання, виходячи з нього лише на короткий час для збирання та передачі інформації. Відповідно, час виконання криптографічного алгоритму прямо пропорційний споживаній потужності пристрою [2, 3].

Як вказує аналіз публікацій [1, 2, 3, 4, 5, 6], найпоширенішим симетричним алгоритмом захисту інформації у вбудованих системах є алгоритм AES [7]. Цей алгоритм при програмній реалізації демонструє найкраще співвідношення продуктивність/пам'ять у порівнянні з іншими відомими криптоалгоритмами [1]. Для більшості вбудованих застосувань достатньо 128-бітного ключа [3], що зумовлює переважне використання алгоритму AES у варіанті AES-128.

Оцінці продуктивності AES для різних мікропроцесорних архітектур присвячені численні дослідження, при цьому особливу увагу приділяють сімействам 8/16/32-бітних мікроконтролерів AVR, MSP430 та ARM7TDMI відповідно, з огляду на їх особливості описані нижче.

Сімейство 8-бітних AVR-мікроконтролерів фірми Atmel це недорогі, високопродуктивні RISC-мікроконтролери з розвиненим набором периферійних модулів, що зумовило їх широке використання у вбудованих системах [8]. AVR-мікроконтролери поєднують високу продуктивність та низьке енергоспоживання, що робить їх лідерами у класі 8-бітних платформ. Також у цьому сімействі крім мікроконтролерів загального призначення присутні спеціалізовані захищені мікроконтролери (Secure Microcontrollers), орієнтовані на використання в інтелектуальних картах, USB-ключах та інших апаратних криптографічних засобах.

Мікроконтролери з ядром MSP430 є високопродуктивними 16-бітними процесорами з

наднизьким енергоспоживанням. Мікроконтролери сімейства MSP430 містять 16-бітне RISC ядро виконане за Прінстонською архітектурою та високоточні периферійні аналогові модул [9]. Однією з ключових переваг сімейства MSP430 є ультранизьке енергоспоживання (до 18(мкА/MIPS), що забезпечило їм широку популярність у вбудованих системах і особливо у БСМ.

Сімейство ARM-ядер спеціально розроблено для 32-бітних високопродуктивних процесорів з низьким енергоспоживанням. Мікроконтролери з ARM-ядром у 2009 році становили до 90% ринку 32-бітних RISC-мікроконтролерів і наразі за енергоспоживанням та ціною наблизилися до 8-бітних моделей, складаючи останнім серйозну конкуренцію у їх традиційних сегментах використання.

Коротко проаналізуємо відомі реалізації AES для вказаних сімейств мікроконтролерів, результати яких представлені у табл. 1.

У статті [10] розглянуто оптимізовану програмну реалізацію алгоритму AES-128 для різних процесорних платформ, зокрема і ARM7TDMI, орієнтовану на використання у вбудованих системах, смарт-картах і ПК. Оскільки смарт-карти накладають достатньо жорсткі вимоги щодо об'єму пам'яті, автори роблять акцент на збільшенні швидкодії за рахунок ретельного програмування внутрішніх перетворень алгоритму AES, без використання LUT-таблиць (за винятком S-Box операцій). Програмування алгоритму здійснювалося на мові С, для двох варіантів: з розгортанням ключа в процесі шифрування/дешифрування (on-the-fly) і попереднім обчисленням та збереженням у пам'яті раундових підключів (unrolling), при цьому попереднє обчислення ключа додатково потребує ще 634 такти.

Робота [11] присвячена дослідженню шляхів ефективної реалізації алгоритму AES-128 для ARM-процесорів на мові С. Пріоритетом було досягнення максимальної продуктивності за рахунок використання як LUT-таблиць для здійснення внутрішніх операцій, так і особливостей кеш-пам'яті ARM-архітектури. Як і слід було очікувати, це привело до зростання продуктивності більш ніж у 2.5 рази, порівняно з реалізацією [10]. Проте достатньо високі вимоги до пам'яті – 5966 байт (з них 2570 байт для LUT-таблиць) значно обмежують коло потенційних застосувань у вбудованих системах.

У [5] наведено результати реалізації алгоритму AES-128 на мові С для 16-бітних мікроконтролерів з архітектурою MSP430 для використання у БСМ та вбудованих системах. Досліджено вплив різних шляхів оптимізації, як на рівні програми, так і компілятора, на продуктивність та обсяг необхідної пам'яті. На підставі аналізу результатів були відібрані найдоцільніші прийоми оптимізації, що у поєднанні з unrolling-підходом до формування підключів раунду (потребує додатково 260 байт RAM) дають змогу досягнути показників представлених у табл. 1. Також розглянуто переваги і недоліки апаратної реалізації алгоритму AES у мікроконтролерах та радіомодулях орієнтованих на застосування у БСМ.

У публікації [12] також здійснено оцінку параметрів С-реалізації алгоритму AES-128 для мікроконтролерів сімейства MSP430. За рахунок розгортання ключа «на льоту» і відсутності спеціальних заходів з оптимізації, показники продуктивності поступаються результатам роботи [5].

У [4] наведено результати асемблерної реалізації алгоритму AES-128 для архітектури MSP430, що входить до складу криптобібліотеки, призначеної для вбудованих систем з обмеженими ресурсами. Основною вимогою до реалізацій криптоалгоритмів даної бібліотеки було забезпечення мінімальних вимог до ROM- і RAM-пам'яті та максимальної продуктивності при вказаних обмеженнях. Це зумовило використання більш ефективної мови асемблера замість С та здійснення розгортання ключа «на льоту», при цьому вдалося досягнути мінімального обсягу необхідної пам'яті зі всіх відомих реалізацій AES-128 на 8/16-бітних мікроконтролерах.

Також у [4] подано результати асемблерної реалізації алгоритму AES-128 для 8-бітних мікроконтролерів з архітектурою AVR. За рахунок архітектурних особливостей AVR-ядра,

розвиненої системи команд та орієнтації алгоритму AES на 8- та 32-бітні платформи, показники продуктивності даної реалізації виявилися навіть вищими ніж для 16-бітних мікроконтролерів сімейства MSP430.

Варто відзначити, що при необхідності, програмні реалізації AES-128 вищезгаданої бібліотеки можуть бути доповнені спеціальними заходами для протидії Side-Channel атакам (в першу чергу DPA-атакам), які проте приводять до зниження продуктивності.

Робота [1] містить результати виконання алгоритму AES-128 на AVR-мікроконтролерах, з точки зору вимог lightweight-криптографії. Відповідно, це також зумовило використання асемблеру та відмову від LUT-таблиць, хоча продуктивність у кінцевому підсумку виявилася на рівні з найкращим результатом досягнутим для 16-бітних MSP430-мікроконтролерів.

Таблиця 1

Результати реалізації алгоритму AES-128 для сімейств AVR, MSP430, ARM7TDMI

Платформа	Шифрування,тактів/блок	Дешифрування, тактів/блок	Продуктивність шифрування*, Кбіт/с	Продуктивність дешифрування*, Кбіт/с	Розмір ROM, байт
Асемблер					
AVR [4]	4009	6073	255.4	168.6	3100
AVR [1]	3766	4558	271.9	224.7	3410
MSP430 [4]	5432	8802	188.5	116.3	2536
C/C++					
MSP430 [12]	~6600	~8400	155.2	121.9	4262
MSP430 [5]	~3564	~4277	287.3	239.4	5160
ARM7TDMI (unrolling) [10]	1675	2074	611.3	493.7	Дані відсутні
ARM7TDMI (on-the-fly) [10]	2074	2378	493.7	430.6	Дані відсутні
ARM7TDMI [11]	639	638	1602.5	1605.0	5966

* Продуктивність обрахована для тактової частоти 8 МГц.

Аналізуючи наведені результати в цілому, слід зауважити, що попри широке використання алгоритму AES у вбудованих системах, вимоги, які він висуває до необхідного об'єму пам'яті, є досить високі. Збільшення продуктивності можливе за рахунок додаткового значного зростання об'єму коду та LUT-таблиць. Крім цього, достатньо складна операція розширення ключа потребує використання 176 байт для збереження всіх можливих підключів. Відчутна несиметричність продуктивності шифрування/дешифрування створює додаткові незручності у системах реального часу з інтенсивним обміном даними.

Це зумовлює інтерес до дослідження інших відомих криптоалгоритмів, їх адаптації до можливостей вбудованих систем з обмеженими ресурсами, а також розроблення нових алгоритмів, які б задовольняли існуючі вимоги до продуктивності, пам'яті, енергоспоживання та захищеності у вказаних системах.

Мета статті

Метою статті є дослідити шляхи ефективної програмної реалізації симетричного криптоалгоритму ДСТУ ГОСТ 28147-89 на найпоширеніших 8/16/32-бітних процесорних платформах, оцінити продуктивність та вимоги до пам'яті одержаних реалізацій, що дасть змогу провести порівняння з аналогічними показниками алгоритму AES-128 і вибрати оптимальне рішення при розробленні механізмів захисту у вбудованих системах.

Структурні особливості алгоритму ГОСТ 28147-89

Алгоритм ГОСТ 28147-89 затверджений в Україні як чинний стандарт і має офіційну назву ДСТУ ГОСТ 28147:2009 [13].

Алгоритм ГОСТ 28147-89 шифрує дані поблоково, з розміром блоку 64 біти, довжин ключа становить 256 біт. Алгоритм має структуру мережі Фейстеля з 32-х раундів т використовує операції додавання за модулем 2^{32}, додавання за модулем 2, нелінійної замін S та циклічного зсуву, які легко реалізуються у мікроконтролерах за рахунок підтримки н рівні системи команд.

Структурна схема раунду алгоритму ГОСТ 28147-89 наведена на рис. 1.

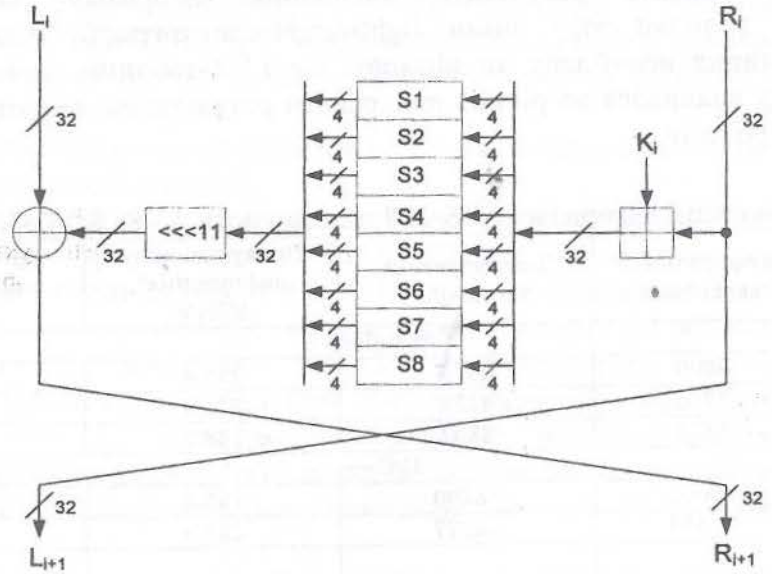


Рис. 1. Структурна схема раунду алгоритму ГОСТ 28147-89

Перед початком шифрування 64-бітний блок відкритого тексту заноситься в 32-бітні регістри $L_0||R_0$.

Алгоритм шифрування складається з 32 раундів. У кожному раунді вміст регістру R_i арифметично додається з 32-бітним підключем K_i. Результат сумування перетворюється в блоці підстановки S і циклічно зсувається на 11 розрядів вліво. Результат раундової функції $F(K_i, R_i)$ додається за модулем 2 з 32-бітним вмістом регістру L_i. Отриманий результат записується в R_{i+1}, при цьому старе заповнення регістру R_i переписується в L_{i+1}:

$$L_{i+1} = R_i, \quad R_{i+1} = L_i \oplus (S(K_i + R_i \bmod 2^{32}) <<< 11),$$

де $\oplus$ – позначає побітову суму за модулем 2, $<<<11$ – циклічний зсув вліво на 11 розрядів.

В останньому раунді обмін не здійснюється, тобто $R_{32} = R_{31}$ та $L_{32} = L_{31} \oplus (S(K_{31} + R_{31} \bmod 2^{32}) <<< 11)$.

В алгоритмі ГОСТ 28147-89 не використовується процедура розширення ключа. Ключ K довжиною 256 біт розглядається як вісім 32-бітних підключів: $K = K_0 \| K_1 \| K_2 \| K_3 \| K_4 \| K_5 \| K_6 \| K_7$. У раундах шифрування з номерами $0 \le r \le 23$ раундовий підключ K_i визначається як $K_i = K_{(r \bmod 8)}$, а для останніх восьми раундів $24 \le r \le 31$ як $K_i = K_{7-(r \bmod 8)}$. Під час розшифрування порядок підключів зворотний.

Блок підстановки S складається з восьми вузлів заміни *S1-S8* з пам'яттю на 64 біти кожен. Вузол заміни представляє собою таблицю з 16 елементів, по 4 біти кожен. На вхід блоку підстановки поступає 32-бітний вектор, який розбивається на вісім послідовних 4-бітних векторів, кожен з яких перетворюється в 4-бітний вектор відповідним вузлом заміни. Вхідний вектор визначає адресу елементу всередині вузла заміни, вміст елементу є вихідним вектором.

У стандарті ГОСТ 28147-89 не визначено блок підстановки, проте рекомендовані Державною службою спеціального зв'язку та захисту інформації України таблиці блоків підстановки для застосування у засобах криптографічного захисту інформації можна знайти в [14].

На сьогоднішній день алгоритм ГОСТ 28147-89 залишається стійким до відомих атак [15].

Попередній аналіз дає змогу стверджувати, що простота раундової функції та процедури генерування підключів, помірна потреба у обчислювальних ресурсах і висока криптостійкість робить перспективним використання алгоритму ГОСТ 28147-89 у вбудованих системах.

Архітектурні особливості платформ для реалізації алгоритму ГОСТ 28147-89

AVR-мікроконтролери. Серед особливостей AVR-ядра, важливих в контексті криптографії для вбудованих систем варто відзначити, що пам'ять організована за Гарвардською архітектурою з 8-бітною пам'яттю даних типу SRAM та 16-бітною пам'яттю програм типу Flash. Регістровий файл містить 32 регістри загального призначення безпосередньо підключених до АЛП. Система команд достатньо розвинена і складається з понад 130 інструкцій, більшість з яких завдяки Гарвардській архітектурі та дворівневому конвеєру виконуються за один такт.

AVR-мікроконтролери підтримують безпосередню, пряму та непряму адресації (з використанням 16-бітних індексних регістрів X, Y, Z). Наявність режимів предекременту та постінкременту при непрямій адресації дає змогу ефективно обробляти масиви даних в процесі виконання криптоалгоритму, генеруючи компактний програмний код. Для доступу до даних у Flash-пам'яті (S-Box, Look-Up таблиці, ключі) використовується непряма адресація на базі Z-регістра.

Тактова частота AVR-мікроконтролерів може досягати 32 МГц, що при продуктивності 32 MIPS забезпечує достатньо високу обчислювальну потужність для 8-бітних платформ.

MSP430-мікроконтролери. Компактне та недороге 16-бітне RISC-ядро MSP430 містить 16 регістрів, з яких дванадцять (R4-R15) є регістрами загального призначення. Система команд складається з 27 інструкцій ядра, і підтримує сім режимів адресації та формати даних байт або слово. Flash-пам'ять може використовуватися як для зберігання коду програми так і для даних, що виключає необхідність копіювати дані в ОЗП перед подальшим використанням. Завдяки однотактовим регістровим операціям та ортогональній архітектурі забезпечується компактність коду та висока продуктивність. Тактова частота MSP430-мікроконтролерів становить до 25 МГц.

ARM7TDMI-мікроконтролери. Фірма ARM (Advanced RISC Machine) надає широкий вибір процесорних ядер зі спільною архітектурою. Високопродуктивні ARM-ядра з малим енергоспоживанням ліцензовані основними виробниками мікроконтролерів (NXP, Atmel, Texas Instruments, Analog Devices, ST Microelectronics, Samsung, Toshiba та ін.). Ядра відрізняються організацією основної та кеш-пам'яті, роботою конвеєра, розширеннями системи команд та ін. Одним з найпоширеніших ARM-ядер серед мікроконтролерів є ядро ARM7TDMI.

ARM7TDMI – універсальний 32-бітний процесор, який споживає відносно малу потужність і при цьому забезпечує продуктивність до 130 MIPS. Наявність 3-крокового конвеєру дозволяє виконувати послідовні команди за один такт. У будь-якому з 7 режимів роботи процесору доступні 16 регістрів загального призначення R0-R15. Оскільки ядро ARM7TDMI належить до RISC-процесорів набір команд відносно невеликий. Арифметично-логічний пристрій має 32-бітний блок зсуву, який дозволяє одночасно з виконанням операції здійснювати логічний або циклічний зсув одного з операндів [16].

Результати дослідження ефективності реалізації криптоалгоритму ГОСТ 28147-89

Оскільки в пристроях різного призначення основними вимогами до криптоалгоритму

можуть бути як економне використання пам'яті, так і максимальна продуктивність, т доцільно дослідити алгоритм ГОСТ 28147-89 для цих двох варіантів, з метою зіставлення ефективності у 8/16/32-бітних платформах. Також для кожного з випадків розглянут реалізація на мові асемблера (ASM) та С (С).

Розробку програмного забезпечення на мовах асемблера та С, оцінку кількості тактів розміру коду здійснено в середовищах IAR Embedded Workbench for Atmel AVR 5.20, IAR Embedded Workbench Kickstart for MSP430 5.10.6 та IAR Embedded Workbench for ARM 5.30.

Варіант максимальної продуктивності (SPEED).

Для досягнення максимальної продуктивності програми слід оптимізувати ті частини алгоритму, які найчастіше виконуються і є найскладнішими в обчислювальному плані використавши для них найефективніші команди та способи адресації, що потребують мінімальної кількості тактів мікроконтролера.

З цієї точки зору очевидно, що для досягнення максимальної швидкодії виконання алгоритму ГОСТ 28147-89 основна увага повинна бути спрямована на функцію раунду (рис. 1), при цьому слід максимально врахувати особливості архітектури ядра. Оскільки операції додавання та сумування за модулем два є по-суті атомарними і реалізуються з допомогою відповідних команд процесора, то швидкодія алгоритму в цілому буде визначатися швидкістю виконання операцій підстановки та циклічного зсуву.

Основний метод підвищення швидкодії алгоритму ГОСТ 28147-89 пов'язаний з ретельним програмуванням циклу підстановок і, насамперед, у зменшенні числа ітерацій цього циклу.

Стандарт ГОСТ 28147-89 передбачає виконання восьми ітерацій циклу підстановок, в кожній з яких здійснюється одна чотирибітна підстановка. Для зберігання восьми таблиць чотирибітних підстановок *S1*...*S8* достатньо 64 байтів. Щоб зменшити кількість ітерацій для мікроконтролерів, які за одне звернення до пам'яті зчитують не менше одного байту, доцільним є одночасне виконання двох 4-бітних підстановок за рахунок збільшення розміру таблиць підстановок. У цьому випадку одна розширена таблиця (ES, Extended S-Box) буде містити 256 байт, а загальний розмір чотирьох таблиць восьмибітних підстановок ES12, ES34, ES56, ES78 становитиме 1 Кбайт. Такі розширені таблиці попередньо формуються із довгострокового ключового елементу і зберігаються в ROM (Flash-пам'яті програм), а їх розмір є прийнятним навіть для 8-розрядних мікроконтролерів.

Додатковим ресурсом збільшення продуктивності є оптимізація виконання циклічного зсуву. Операція циклічного зсуву 32-бітного значення на 11 розрядів вліво може розглядатися як послідовний зсув на 8 та 3 розряди. Очевидно, що при цьому циклічний зсув на 8 розрядів еквівалентний обміну вмісту між 8-бітними регістрами і може бути суміщений з операцією підстановки. Таким чином, потрібно буде здійснити програмний зсув лише на 3 розряд вліво. Цей підхід використано для мікроконтролерів AVR та MSP430. На рис. 2, а наведена структурна схема раунду алгоритму ГОСТ 28147-89 орієнтована на максимальну швидкодію.

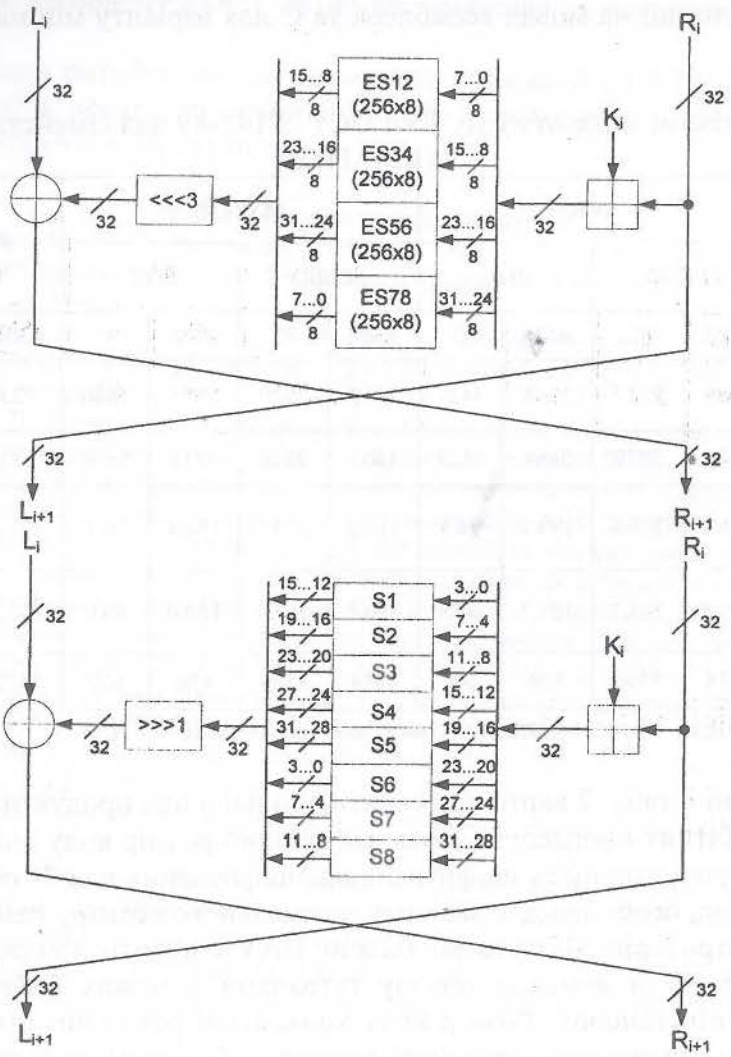
Для ARM7TDMI-мікроконтролерів існує можливість виконання логічної операції з одночасним циклічним зсувом одного з операндів вправо. Тому для них операція циклічного зсуву операнду на 11 розрядів вліво може бути суміщена з операцією додавання за модулем два з регістром L_i, що проте не приводить до збільшення часу виконання команди. При цьому слід врахувати, що циклічний зсув 32-бітного значення на 11 розрядів вліво еквівалентний циклічному зсуву вправо на 21 розряд:

$$EOR \quad R2, R1, R3, ROR\ \#21 \quad ; R2 = R1 \oplus (R3 >>> 21)$$

Щоб зменшити кількість тактів потрібних для 32-х викликів функції раунду, її включено безпосередньо в тіло програми (inline-функція).

Завдяки наявності в мікроконтролерах AVR/MSP430/ARM7TDMI великої кількості регістрів загального призначення, є можливість розташувати усі проміжні змінні алгоритму у

цих регістрах, зменшивши кількість звернень до ROM і RAM пам'яті, а отже збільшивши швидкодію.



а) б)

Рис. 2. Структурна схема раунду алгоритму ГОСТ 28147-89 для варіантів максимальної продуктивності (а) та мінімального розміру коду (б)

Flash-пам'ять програм використовується для зберігання коду, таблиць заміни і ключа. Для мікроконтролерів сімейства AVR, кількість тактів необхідних для зчитування операнду відрізняється в залежності від типу пам'яті, в якій він зберігається: оперативна чи постійна, а саме, зчитування з ОЗП відбувається за 2 такти, а з ПЗП – за 3 такти. Оскільки пріоритетом є досягнення максимальної швидкодії перед виконанням шифрування/дешифрування здійснюється копіювання ключа в ОЗП.

Результати реалізації на мовах асемблера та С для варіанту максимальної швидкодії наведено у табл. 2.

Варіант мінімального розміру коду (SIZE).

Даний варіант має три відмінності від варіанту максимальної швидкодії. По-перше, не використовуються розширені таблиці заміни, по-друге здійснюється виклик функції раунду. Третя відмінність стосується виконання операції циклічного зсуву для мікроконтролерів AVR та MSP430. Циклічний зсув 32-бітного значення на 11 розрядів вліво розглядається як послідовний зсув на 12 розрядів вліво та 1 розряд вправо. Очевидно, що при цьому циклічний зсув на 8 розрядів еквівалентний обміну вмісту між 8-бітними регістрами, і разом з операцією зсуву ще на 4 розряди вліво, може бути суміщений з операцією підстановки,

коли дані з вузлів заміни заносяться у відповідні розряди (рис. 2, б). Таким чином, цей підхід потребує програмного виконання лише одного зсуву вправо.

Результати реалізації на мовах асемблера та С для варіанту мінімального розміру коду наведено у табл. 2.

Таблиця 2

Результати реалізації алгоритму ДСТУ ГОСТ 28147-89 для сімейств AVR, MSP430, ARM7TDMI

Параметр реалізації	AVR				MSP430				ARM7TDMI			
	SPEED		SIZE		SPEED		SIZE		SPEED		SIZE	
	ASM	C	ASM	C	ASM	C	ASM	C	ASM	C	ASM	C
Шифрування, тактів/блок	2089	2517	3664	5427	1619	2320	3381	5680	723	707	1556	1502
Дешифрування, тактів/блок	2089	2519	3664	5429	1651	2328	3413	5688	723	681	1556	1502
Продуктивність шифрування*, Кбіт/сек	245.1	203.4	139.7	94.3	316.2	220.7	151.4	90.1	708.2	724.2	329.0	340.9
Продуктивність дешифрування*, Кбіт/сек	245.1	203.3	139.7	94.3	310.1	219.9	150.0	90.0	708.2	751.8	329.0	340.9
Розмір Flash-пам'яті, байт	1614	1796	508	640	1488	1566	456	670	1472	2308	516	700

* Продуктивність обрахована для тактової частоти 8 МГц.

Аналізуючи дані в табл. 2 варто відзначити, що найвища продуктивність досягнута, як і очікувалося, для 32-бітних процесорів, а ось найменший розмір коду одержано для 16-бітних мікроконтролерів. Продуктивність шифрування/дешифрування для 8- та 16-бітних платформ відрізняється незначно, особливо для варіанту мінімального розміру пам'яті. Це пояснюється тим, що у мікроконтролерів MSP430 порівняно з AVR досить обмежена система команд, відсутні бітові операції та команди обміну тетрадами в межах байту, які необхідні для виконання 4-бітних підстановок. Розмір коду конкретної реалізації практично не залежить від розрядності мікроконтролера, при цьому варіант SIZE потребує в середньому приблизно втричі менше пам'яті ніж варіант максимальної швидкодії SPEED. Продуктивності шифрування та дешифрування відрізняються незначно.

Порівняння даних в табл.1 та табл. 2 дає змогу стверджувати, що алгоритм ГОСТ 28147-89 перш за все дає суттєвий виграш в розмірі коду програми при співмірній продуктивності. Так, наприклад, мінімальний розмір ROM-пам'яті для реалізації алгоритму AES є в 5.5 разів більший, ніж мінімальний розмір ROM для реалізації ГОСТ 28147-89. Для обох варіантів реалізації (SPEED, SIZE), незалежно від мови програмування та процесорної архітектури, алгоритм ГОСТ 28147-89 демонструє значний виграш в розмірі коду.

Продуктивність реалізацій ГОСТ 28147-89 та AES для 8- і 16-бітних платформ відрізняються незначно, з відхиленнями як в одну так і другу сторону. Для ARM-процесорів продуктивність ГОСТ 28147-89 у варіанті максимальної швидкодії випереджає відповідні реалізації AES, які не використовують LUT-таблиці. Найпродуктивніша реалізація алгоритму AES для процесорів ARM7TDMI в 2.3 раз перевершує відповідну реалізацію ГОСТ 28147-89, що проте досягається за рахунок суттєвого збільшення розміру коду.

Висновки

Результати досліджень підтверджують перспективність та доцільність застосування у вбудованих системах алгоритму ГОСТ 28147-89, особливо за жорстких вимог щодо об'єму пам'яті.

Розглянуті методи оптимізації алгоритму при реалізації на 8/16/32-бітних вбудованих платформах дозволяють досягнути компромісу між параметрами ціна/енергоспоживання в залежності від конкретного застосування.

Перелік використаної літератури

1. Rinne S., Eisenbarth T., Paar C. Performance Analysis of Contemporary Light-Weight Block Ciphers on 8-bit Microcontrollers // ECRYPT Workshop Software Performance Enhancement for Encryption and Decryption, June 11-12, 2007, Amsterdam, NL, pp. 33-43.

2. T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, L. Uhsadel. A Survey of Lightweight Cryptography Implementations // IEEE Design & Test of Computers - Special Issue on Secure ICs for Secure Embedded Computing Vol. 24, Nr. 6, pp. 522-533, November 2007.

3. Jinwala D., Patel D., Dasgupta K. Investigating and Analyzing the Light-weight ciphers for Wireless Sensor Networks // INFOCOMP Journal of Computer Science, Vol. 8, Issue 2, pp. 39-50, 2009.

4. Feldhofer M. Efficient Data Protection for Devices with Scarce Resources // Workshop Developing Secure Applications for Mobile Wireless Environments (WISCY'10), December 2, 2010, Gent Belgium.

5. Didla S., Ault A., Bagchi S. Optimizing AES for embedded devices and wireless sensor networks // Proceedings of the 4th International Conference on Testbeds and research infrastructures for the development of networks & communities (TridentCom'08), March 17-20, 2008, Innsbruck, Austria, pp. 1-10, ICST, Brussels, (2008).

6. Çakiroğlu M. Software implementation and performance comparison of popular block ciphers on 8-bit low-cost microcontroller // International Journal of the Physical Sciences Vol. 5(9), pp. 1338-1343, 18 August, 2010.

7. FIPS-197: Advanced Encryption Standard (AES). Federal Information Processing Standard, National Institute of Standards and Technology, U.S. Dept. of Commerce, November 26, 2001.

8. Евстифеев А. В. Микроконтроллеры AVR семейства Mega. Руководство пользователя. – М.: Издательский дом «Додэка-XXI», 2007. – 592 с.

9. Семейство микроконтроллеров MSP430x1xx. Руководство пользователя: Пер. с англ. – М.: ЗАО «Компэл», 2004. – 368 с.

10. Gura N., Patel A., Wander A., Eberle H., Shantz S. Efficient Software Implementation of AES on 32-Bit Platforms // Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03), September 8-10, 2003, Cologne, Germany, Vol. 2779, pp. 159-171, Springer, Heidelberg (2003).

11. Atasu K., Breveglieri L., Macchetti M. Efficient AES implementations for ARM based platforms // Proceedings of the 2004 ACM symposium on Applied computing, March 14 -17, 2004, Nicosia, Cyprus, pp. 841-845, ACM New York (2004).

12. Kretzschmar U. AES128 – A C Implementation for Encryption and Decryption // Application Report, SLAA397A, March 2009, 7 pp., Texas Instruments 2009.

13. Системы обработки информации. Защита криптографическая. Алгоритмы криптографического преобразования: ДСТУ ГОСТ 28147:2009. – [Чинний від 2009-02-01]. – К.: Держспоживстандарт України, 2008. – 28 с. – (Національний стандарт України).

14. Адміністрація Державної служби спеціального зв'язку та захисту інформації України, наказ №114 від 12.06.2007, "Інструкція про порядок постачання і використання ключів до засобів криптографічного захисту інформації".

15. Панасенко С. П. Алгоритмы шифрования. Специальный справочник. – СПб.: БХВ-Петербург, 2009. – 576 с.

16. Редькин П. П. Микроконтроллеры ARM7 семейства LPC2000. Руководство пользователя. – М. Издательский дом «Додэка-XXI», 2007. – 560 с.

В статті здійснено оцінку продуктивності та вимог до об'єму пам'яті алгоритму симетричного шифрування ДСТУ ГОСТ 28147-89 при реалізації на 8/16/32-бітних сімействах мікроконтролерів AVR, MSP430, ARM7TDMI. Здійснено порівняння отриманих показників з показниками алгоритму AES. Показано, що алгоритм ДСТУ ГОСТ 28147-89 у порівнянні з AES має значно менші вимоги до розміру коду при співмірній продуктивності.

Ключові слова: ГОСТ 28147-89, AES, AVR, MSP430, ARM7TDMI, мікроконтролери, вбудовані системи, lightweight-криптографія.

В статье осуществлено оценку производительности и требований к объему памяти алгоритма симметричного шифрования ДСТУ ГОСТ 28147-89 при реализации на 8/16/32-битных семействах микроконтроллеров AVR, MSP430, ARM7TDMI. Произведено сравнение полученных показателей с

показателями алгоритма AES. Продемонстрировано, что алгоритм ДСТУ ГОСТ 28147-89 в сравнении с AE имеет значительно меньшие требования к размеру кода при сопоставимой производительности.

Ключевые слова: ГОСТ 28147-89, AES, AVR, MSP430, ARM7TDMI, микроконтроллеры, встраиваемы системы, lightweight-криптография.

In the article the estimation of the performance and code size of the symmetric-key chipper algorithm of DST GOST 28147-89 during realization on 8/16/32-bit microcontrollers of AVR, MSP430, ARM7TDMI. Comparing of th obtained results with the results of the algorithm AES. It is shown that the algorithm DSTU GOST 28147-89 i comparison from AES has significantly less requirements to the code size at the comparable performance.

Keywords: GOST 28147-89, AES, AVR, MSP430, ARM7TDMI, microcontrollers, embedded systems lightweight-cryptography.

Рецензент: д.т.н., проф. Дудикевич В.Б
Надійшла 31.01.201

УДК 004.056.5

Бобок И.И.
(Одеський національний політехнічний університет)

СТЕГАНОАНАЛИТИЧЕСКИЙ МЕТОД ДЛЯ ЦИФРОВОГО СИГНАЛА-КОНТЕЙНЕРА, ХРАНЯЩЕГОСЯ В ФОРМАТЕ С ПОТЕРЯМИ

Активизация научной деятельности в области стеганографии, где скрывается сам факт существования тайного сообщения, связанная с запретом шифрования на законодательном уровне во многих странах мира, привела в настоящий момент к чрезвычайно негативному последствию - росту частоты использования получаемых новых разработок различными террористическими структурами [1-5]. В силу этого в настоящий момент трудно переоценить *актуальность* решения вопросов, связанных с повышением эффективности стеганоанализа (СА) [1-5]. Общей чертой стеганографических методов является то, что скрываемое сообщение, или дополнительная информация (ДИ), встраивается в некоторый непривлекающий внимание объект – основное сообщение (ОС), или контейнер (не ограничивая общности рассуждений [7], в качестве ОС рассматривается цифровое изображение (ЦИ) в градациях серого). Процесс погружения ДИ в контейнер будем называть стеганопреобразованием (СП), а результат этого погружения – стеганосообщением (СС). После встраивания информации СС открыто транспортируется адресату по каналу связи или хранится в таком виде.

Под СА будем понимать процесс выявления или констатации отсутствия определенных характерных признаков в анализируемом информационном контенте, позволяющих делать вывод о факте внедрения секретной информации или об отсутствии такового.

В настоящее время хранение и передача цифровых сигналов, в частности, ЦИ по каналам телекоммуникаций в связи со значительным увеличением объемов информации осуществляется в сжатом состоянии. Этот немаловажный факт не может не учитываться при разработке подхода к решению задачи СА. Одним из самых популярных форматов хранения ЦИ сегодня является формат JPEG (с потерями), который и будет рассматриваться ниже как формат хранения ОС (для определенности рассматривается JPEG, основанный на дискретном косинусном преобразовании (ДКП)).

При организации стеганографического канала связи на сегодняшний день очень широко используется метод модификации наименьшего значащего бита (LSB) [2,6], что явилось побуждающим фактором для автора работы, в первую очередь, рассмотреть характерные особенности и найти способы выявления последствий работы этого метода.

Целью настоящей работы является разработка стеганоаналитического алгоритма (САА) детектирования наличия секретного сообщения, погруженного в JPEG-контейнер с использованием LSB-метода.