УДК 004.056:061.68

Я.Р. Совин, В.В. Хома, Я.В. Решетар
Національний університет "Львівська політехніка",
кафедра захисту інформації

РЕАЛІЗАЦІЯ КРИПТОГРАФІЧНОГО АЛГОРИТМУ ЗГІДНО З ДСТУ ГОСТ 28147:2009 ДЛЯ ВБУДОВАНИХ СИСТЕМ НА БАЗІ ARM-ПРОЦЕСОРІВ

Досліджено шляхи програмної реалізації криптографічного алгоритму згідно з ДСТУ ГОСТ 28147:2009 для найпоширеніших у вбудованих системах 32-бітних ARM-ядер: ARM7TDMI-S і ARM Cortex-M3. Оцінено продуктивність та розмір коду мовами С та асемблера. Порівняно одержані показники з відомими реалізаціями іншого криптографічного алгоритму – AES.

Ключові слова: симетричні криптоалгоритми, програмна реалізація, ARM-процесори, шифрування для вбудованих систем.

In this paper we investigate different ways of GOST 28147 cryptographic algorithm software implementation for the most popular 32-bit ARM-core embedded system processors (i.e. ARM7TDMI-S and ARM Cortex-M3). Performance and C/assembler code size evaluation was done. As well as the comparison to known Advanced Encryption Standard (AES) ARM-based implementations.

Key words: symmetric-key algorithms, software implementation, ARM-core processors, encryption for embedded systems.

Вступ

Функціональна насиченість вбудованих систем, яка постійно зростає, спричиняє підвищення вимог до продуктивності їх основних компонентів – мікроконтролерів. Тривалий час у вбудованих системах кількісно значно переважали 8-бітні мікроконтролери, які за показниками ціна – функціональність – енергоспоживання були поза конкуренцією. В останні роки почався стрімкий перехід з 8- та 16-бітних вбудованих систем на 32-бітні, зумовлений двома основними причинами. По-перше, це масовий перехід напівпровідникової індустрії на технологію 180 нм, що зробило співмірними вартість та розміри 8- та 32-бітних кристалів, і, по-друге, насичення 32-бітних процесорів периферією та системними модулями, характерними для 8-бітних мікроконтролерів.

Домінуюче становище на ринку 32-бітних процесорних ядер займає архітектура ARM (у 2009 р. 90 %), спеціально розроблена для високопродуктивних RISC-процесорів з низьким енергоспоживанням. Фірма ARM (Advanced RISC Machine) надає широкий вибір процесорних ядер зі спільною архітектурою, які ліцензовані такими основними виробниками мікроконтролерів, як NXP, Atmel, Texas Instruments, ST Microelectronics, Infineon, NEC, Renesas, Analog Devices, Samsung, Toshiba та ін. Ядра відрізняються організацією пам'яті, роботою конвеєра, розширеннями системи команд тощо. Найпоширенішими ARM-ядрами у вбудованих системах є ARM7TDMI-S та ARM Cortex-M3, на базі яких випускаються сотні типів мікроконтролерів для найрізноманітніших сфер застосування. Мікроконтролери з 32-бітним ARM-ядром завдяки низькій вартості та малому енергоспоживанню можуть успішно конкурувати з набагато слабкішими за обчислювальними можливостями 8- і 16-бітними моделями.

У багатьох вбудованих системах необхідно реалізувати функції захисту інформації, що стимулює інтенсивні дослідження шляхів ефективної реалізації криптографічних алгоритмів, за умови обмежень, які накладають ці системи. Головними обмеженнями є продуктивність процесорного ядра, споживана потужність, розмір постійної (ROM) та оперативної (RAM) пам'яті.

Залежно від конкретного застосування різними є і вимоги до реалізації – наприклад, мінімальні ресурси пам'яті у разі RFID-міток чи смарт-карт, максимальна продуктивність для USB-ключів або мінімальне енергоспоживання у безпровідних сенсорних мережах.

Найпоширенішим симетричним криптоалгоритмом, який застосовується у вбудованих системах є Advanced Encryption Standard (AES) [1], спеціально розроблений для ефективної програмної реалізації як на 8-, так і на 32-бітних процесорах. AES демонструє найкраще співвідношення продуктивність/пам'ять порівняно з іншими відомими криптоалгоритмами [2–6]. Вітчизняний стандарт шифрування ДСТУ ГОСТ 28147:2009 [7], який має важливе значення для національної інформаційної безпеки, також орієнтований на 32-бітні платформи і має невисокі вимоги до складності реалізації у вбудованих системах. З огляду на це, становить інтерес порівняння параметрів вказаних криптоалгоритмів під час їх реалізації на ARM-процесорах.

Аналіз останніх досліджень і публікацій

Критеріями оцінки криптографічних алгоритмів з огляду на придатність використання у вбудованих системах є необхідний розмір внутрішньої пам'яті 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. Продуктивність є критичною з огляду на енергоспоживання, оскільки, як правило, у вбудованих системах центральний процесор більшу частину часу перебуває в режимі пониженого енергоспоживання, короткочасно виходячи з нього для збирання, оброблення та передавання інформації. Отже, час виконання криптографічного алгоритму є прямо пропорційний споживаній потужності [3, 4].

Особливостям реалізації криптоалгоритму AES на різних мікропроцесорних архітектурах присвячені численні дослідження [2–6, 8, 9], чого не можна сказати про ГОСТ 28147, для якого практично відсутні відповідні публікації в науковій та технічній літературі. Для більшості вбудованих систем достатньо криптостійкості на рівні 128-бітного ключа [4], що зумовлює переважне використання алгоритму AES у варіанті AES-128, якому надалі буде приділена основна увага. Аналіз публікацій показує, що найкращі реалізації алгоритму AES-128 для 8-, 16-бітних мікроконтролерів мають продуктивність на рівні 220–350 тактів/байт та потребують 2,5–5 Кбайт ROM [2–6].

Недостатня продуктивність 8- та 16-бітних процесорів загального призначення під час реалізації криптоалгоритмів стимулювала виробників до включення у мікроконтролери спеціальних криптомодулів – апаратних прискорювачів (акселераторів). Так, наприклад, у 8-бітних мікроконтролерах сімейства AVR XMEGA (фірма Atmel) криптомодуль AES-128 здатний виконувати шифрування/дешифрування одного блока даних за 375 тактів при максимальній тактовій частоті 32 МГц. Перед операціями шифрування/дешифрування дані та ключ потрібно побайтно записати у спеціальні регістри, а після завершення операції – прочитати з них, що потребує додаткових тактів. Криптоакселератори AES-128 присутні також у сімействах 8-бітних мікроконтролерів STM8L16x (фірма STMicroelectronics) з продуктивністю 892 такти на блок при максимальній тактовій частоті 16 МГц, 16-бітних мікроконтролерів CC430 (фірма Texas Instruments) зі швидкодією шифрування/дешифрування одного блока 167 та 214 тактів відповідно та ін. Достатньо велика кількість тактів, необхідність побайтового читання/записування даних і ключів через спеціальні регістри та порівняно невисока тактова частота центральних процесорів 8-, 16-бітних мікроконтролерів обмежують продуктивність виконання алгоритму AES на цих платформах.

Подальше підвищення продуктивності можливе під час використання мікросхем, в яких на одному кристалі розміщено ядро мікроконтролера та ПЛІС. Прикладом можуть слугувати мікросхеми

класу Field Programmable System Level Integrated Circuit (FPSLIC) сімейства AT40K (фірма Atmel), які поєднують стандартне 8-бітне мікроконтролерне AVR-ядро та ПЛІС. У такому разі на базі ПЛІС реалізується апаратний криптопроцесор, який обмінюється даними з AVR-ядром через спільний двопортовий ОЗП (ДП-ОЗП) – рис. 1. При цьому з використанням ітеративної конвеєрної архітектури можна досягнути показників 10/32 тактів на блок для AES-128 та ГОСТ 28147 відповідно. Незважаючи на вдвічі більшу довжину ключа, порівняно з AES-128, ГОСТ 28147 при апаратній реалізації потребує значно менше ресурсів: 800 еквівалентних вентилів проти 3100 [10].

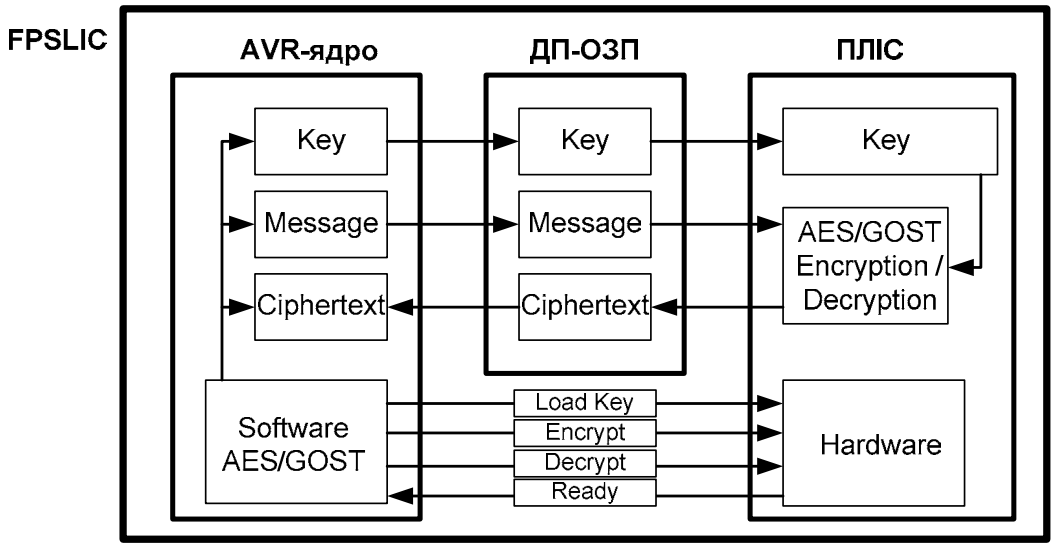

Рис. 1. Реалізація апаратного криптопроцесора на базі FPSLIC

Коротко проаналізуємо відомі програмні реалізації AES для ARM-мікроконтролерів, результати яких наведені у табл. 1.

У статті [8] розглянуто оптимізовану програмну реалізацію алгоритму AES-128 для різних процесорних платформ, зокрема і ARM7TDMI, орієнтовану на використання у вбудованих системах, смарт-картках і ПК. Оскільки смарт-картки накладають достатньо жорсткі вимоги щодо обсягу 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 здійснювали мовою С, для двох варіантів: з розгортанням ключа під час шифрування/дешифрування (on-the-fly) і попереднім обчисленням та збереженням у пам'яті раундових підключів (unrolling), до того ж попереднє обчислення ключа додатково потребує ще 634 такти.

Робота [9] присвячена дослідженню шляхів ефективної реалізації алгоритму AES для ARM-процесорів мовою С. Пріоритетом було досягнення максимальної продуктивності за рахунок використання як LUT-таблиць для здійснення внутрішніх операцій, так і особливостей системи пам'яті ARM-архітектури. Це привело до зростання продуктивності AES-128 більше ніж у 2,5 раза, порівняно з реалізацією [8], 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 [11] розглянуто вплив різноманітних прийомів оптимізації операцій шифрування/ дешифрування та розгортання ключа на продуктивність і обсяг пам'яті кінцевої реалізації AES-128 для ARM7TDMI-процесора. Автори дійшли висновку, що відмова від циклу під час операції розгортання ключа приводить до надмірного збільшення розміру коду (900 проти 176 байт), і всі їх результати отримані для циклічного розгортання ключа. Також досліджено вплив використання таблиць для виконання операцій SubBytes, InvSubBytes та MixColumns, заміни функцій макросами, розгортання циклу шифрування в лінійний код, розгортання ключа в RAM під час обчислень тощо. Найбільш значимі, на наш погляд, результати наведені в табл. 1.

Оцінку параметрів реалізації AES-128 для мікроконтролерів сімейства Stellaris фірми Texas Instruments з ядром ARM Cortex-M3 здійснено в публікації [12]. Порівняно невисокі результати реалізації пояснюються використанням лише по одній LUT-таблиці для операцій SubBytes та InvSubBytes під час шифрування/дешифрування. Розгортання ключа здійснюється в RAM, перед процесом обчислень, і потребує додатково 546/2387 тактів та 356 байт пам'яті програм.

Окремі моделі ARM-мікроконтролерів також оснащені апаратними криптомодулями AES.

Так, наприклад, мікроконтролери AT91SAM7X/XC з ARM7TDMI-ядром (фірма Atmel) підтримують апаратне шифрування/дешифрування алгоритмом AES-128 за 12 тактів. На апаратному рівні також реалізовані заходи з протидії диференційним атакам через аналіз енергоспоживання (DPA-атакам). Мікроконтролери сімейства EFM32G фірми Energy Micro з ядром ARM Cortex-M3 мають внутрішній апаратний акселератор алгоритму AES з довжиною ключа 128 і 256 біт. Для 128-бітного ключа один блок даних шифрується за 54 такти, а для 256-бітного – за 75 тактів. Перед початком шифрування/розшифрування дані і ключ повинні бути завантажені в спеціальні регістри, що потребує додаткових циклів та зменшує кінцеву продуктивність. Також потрібно відзначити невисоку максимальну тактову частоту цих мікроконтролерів – 32 МГц. У мікроконтролерах сімейства STM32F205/207/215/217xx з ядром ARM Cortex-M3 присутній криптографічний процесор з підтримкою AES-128/192/256, якому для оброблення одного блока потрібно 14/16/18 тактів відповідно при максимальній тактовій частоті 120 МГц. Крім того вони містять генератор випадкових чисел на базі аналогового джерела шуму та геш-процесор з підтримкою алгоритмів SHA-1 (66 тактів), MD-5 (50 тактів) і HMAC.

Варто відзначити, що криптоакселераторами оснащено, як правило, лише декілька моделей зі всього сімейства, з найвищими характеристиками і відповідно найвищою ціною та енергоспоживанням, що часто не відповідає вимогам, які висунуто з конкретного застосування.

Таблиця 1

Результати програмної реалізації алгоритму AES для ядер ARM7TDMI-S та ARM Cortex-M3

Платформа	Шифрування,тактів/блок	Дешифрування, тактів/блок	Шифрування, тактів/байт	Дешифрування, тактів/байт	ROM, байт
AES-128					
ARM7TDMI (unrolling) [8]	1675	2074	105	130	Дані відсутні
ARM7TDMI (on-the-fly) [8]	2074	2378	130	149	Дані відсутні
ARM7TDMI [9]	639	638	40	40	5966
ARM7TDMI [11]	1994	1994	125	125	7944
ARM7TDMI [11]	5542	5542	346	346	2292
ARM Cortex-M3 [12]	1329	1347	83	84	5272
AES-192					
ARM7TDMI [9]	747	746	47	47	5966
AES-256					
ARM7TDMI [9]	855	854	53	53	5966

Аналізуючи наведені результати, загалом, зауважимо, що попри широке використання алгоритму AES у вбудованих системах, його вимоги до необхідного обсягу пам'яті є доволі значимими. Зменшення розміру коду приводить до непропорційного зменшення продуктивності, отже, доволі важко здійснювати обмін «продуктивність/розмір коду» залежно від конкретного застосування. Крім того, достатньо складна операція розширення ключа потребує використання значних додаткових ресурсів пам'яті та часу, а також ускладнює реалізацію багатоключових аплікацій.

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 наявні вимоги до продуктивності, пам'яті, енергоспоживання та захищеності у вбудованих системах.

Метою статті є дослідити шляхи програмної реалізації симетричного криптоалгоритму згідно з ДСТУ ГОСТ 28147:2009 (надалі ГОСТ 28147) на 32-бітних процесорних платформах з ядрами ARM7TDMI-S та ARM Cortex-M3; оцінити продуктивність та вимоги до пам'яті одержаних реалізацій, що дасть змогу порівняти з аналогічними показниками алгоритм AES і вибрати оптимальне рішення під час розроблення механізмів захисту у вбудованих системах.

Структурні особливості алгоритму ГОСТ 28147

Стандарт [7] визначає поблокове шифрування даних, з розміром блока 64 біти та довжиною ключа 256 біт. Алгоритм криптографічного перетворення має структуру мережі Фейстеля з 32 раундів та використовує операції додавання за модулем 2^{32}, додавання за модулем 2, нелінійної заміни S та циклічного зсуву, які легко реалізуються у ARM-мікроконтролерах за рахунок підтримки на рівні системи команд.

Структурна схема раунду алгоритму ГОСТ 28147 наведена на рис. 2.

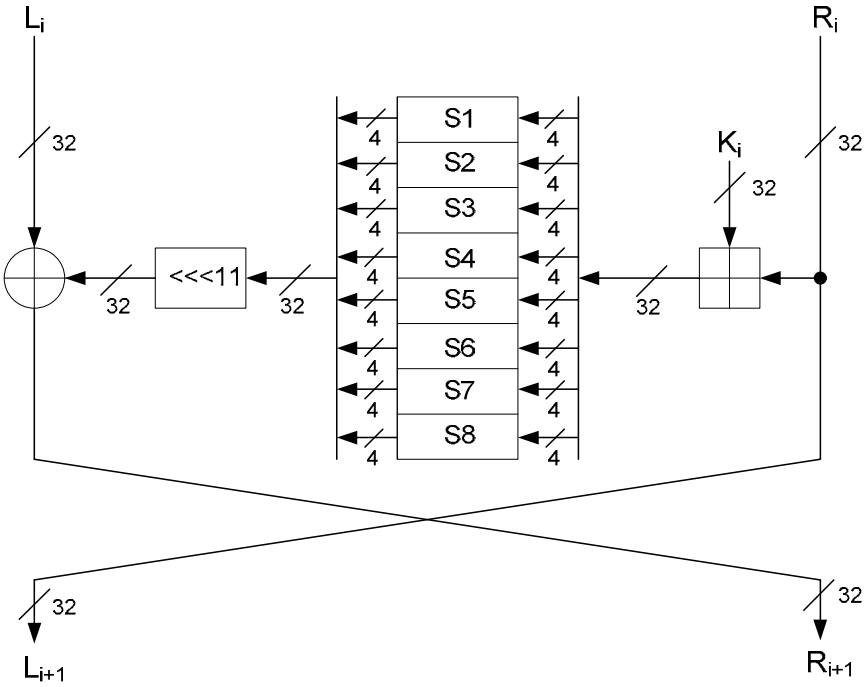

Рис. 2. Структурна схема раунду алгоритму ГОСТ 28147

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 підсумовування 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}, а попереднє 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 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 являє собою таблицю з 16 елементів, по 4 біти кожен. На вхід блока підстановки надходить 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 елемента всередині вузла заміни, вміст елемента є вихідним вектором.

У стандарті 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 [13].

Сьогодні алгоритм ГОСТ 28147 залишається стійким до відомих атак [14]. Наведені в публікації [15] дані про знайдені вразливості, не становлять перешкоди для використання алгоритму у вбудованих системах, оскільки здебільшого для них достатньо довжини ключа 128 біт, а наведена атака лише зменшує еквіваленту довжину ключа ГОСТ 28147 з 256 біт до 225.

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

Архітектурні особливості платформ для реалізації алгоритму ГОСТ 28147

ARM7TDMI-ядро. ARM7TDMI-S – універсальний 32-бітний процесор з Прінстонською архітектурою, який споживає малу потужність і може працювати на частоті до 133 МГц. Наявність трирівневого конвеєра дозволяє виконувати прості команди за один такт. У будь-якому з 7 режимів роботи процесору доступні 16 регістрів загального призначення R0-R15. Система команд складається з достатньо повного, як для RISC-процесора, набору ортогональних інструкцій з підтримкою префіксів умовного виконання та потужних індексних режимів адресації. Арифметико-логічний пристрій має блок зсуву, який дає змогу одночасно з виконанням операції здійснювати логічний або циклічний зсув одного з операндів на довільну кількість розрядів [16].

Важливою особливістю ARM7TDMI-ядра, яка впливає на продуктивність та розмір програми, є підтримка двох наборів команд – 32-бітного набору команд ARM і 16-бітного набору THUMB, що є стисненою підмножиною набору ARM. У режимі THUMB команди не мають повного доступу до всіх регістрів регістрового файла, відсутні деякі інструкції, скорочені поля даних і адресних зміщень у форматі команд тощо. Використання режиму THUMB дає орієнтовно 30% економію пам'яті програм за рахунок зменшення продуктивності на 40%.

Cortex M3-ядро. Cortex-M3 – це стандартизоване, орієнтоване на використання в мікро-контролерах, ядро, яке, на відміну від ARM7TDMI, крім центрального процесора містить такі системні елементи, як контролер вкладених переривань, системний таймер та налагоджувальні модулі. Завдяки Гарвардській архітектурі ядро має окремі шини даних і команд, що разом з трирівневим конвеєром збільшує швидкодію. Регістровий файл містить 16 регістрів R0-R15, з них R0-R12 – користувацькі, а R13-R15 – виконують спеціальні функції. Пристрій циклічного зсуву дозволяє, виконуючи команду, зсувати значення другого операнду на величину до 32 біт [17].

В ядрі Cortex-M3 використовується лише набір команд THUMB-2, орієнтований на компілятори мов C/C++. THUMB-2 складається з 16- та 32-бітних команд, що дає змогу поєднати

продуктивність режиму ARM та економічність THUMB. Набір THUMB-2 доповнено новими інструкціями: ділення, маніпуляцій з бітами, умовних команд тощо. Порівняно з режимом ARM досягається приблизно 26% економії пам'яті за однакової продуктивності.

Максимальна тактова частота ядра Cortex-M3 може досягати 264 МГц, сьогодні мікроконтролери з ядром Cortex-M3 працюють на частотах до 150 МГц.

Результати дослідження ефективності реалізації криптоалгоритму ГОСТ 28147

Оскільки в пристроях різного призначення основними вимогами до криптоалгоритму можуть бути як економне використання пам'яті, так і максимальна продуктивність, то доцільно дослідити алгоритм ГОСТ 28147 для цих двох варіантів. Також для кожного з випадків розглянута реалізація мовою асемблера (ASM) та С (С).

У асемблерних реалізаціях функцій шифрування/розшифрування ГОСТ 28147 використовується С-інтерфейс, що дає можливість викликати їх з програм мовами С/С++. Для виклику асемблерних функцій з С-програми вони повинні дотримуватися правил передачі і повернення параметрів, згідно з розробленим для ARM-процесорів стандартом Procedure Call Standard for the ARM Architecture (AAPCS) [18]. AAPCS визначає, що перші чотири параметри передаються функції через регістри R0-R3. Функція повинна зберігати вміст регістрів R4-R11, R13 та R14 між викликами, а вміст регістрів R0-R3 та R12 може змінюватися.

Всі результати для ARM7TDMI-ядра отримані в режимі ARM, оскільки режим THUMB, як показали дослідження, не дає істотного виграшу в розмірі коду при майже двократному зменшенні продуктивності.

Розробку програмного забезпечення мовами асемблера та С, оцінку кількості тактів і розміру коду здійснено в середовищі IAR Embedded Workbench for ARM 5.30.

Варіант з максимальною продуктивністю (SPEED). Для досягнення максимальної продуктивності програми потрібно оптимізувати ті частини алгоритму, які найчастіше виконуються і є найскладнішими в обчислювальному сенсі, використавши для них найефективніші команди та способи адресації, що потребують мінімальної кількості тактів мікроконтролера.

З цього погляду очевидно, що для досягнення максимальної швидкодії виконання алгоритму ГОСТ 28147 основна увага повинна бути спрямована на функцію раунду (рис. 2), потрібно максимально врахувати особливості архітектури ядра. Оскільки операції додавання та підсумовування за модулем два є по-суті атомарними і реалізуються за допомогою відповідних команд процесора, то швидкодія алгоритму, загалом, буде визначатися швидкістю виконання операцій підстановки.

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

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

На рис. 3 наведена структурна схема раунду алгоритму ГОСТ 28147, орієнтована на максимальну швидкодію.

Для ARM-процесорів 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. До того ж необхідно врахувати, що циклічний зсув 32-бітного значення на 11 розрядів вліво еквівалентний циклічному зсуву вправо на 21 розряд:

EOR R2, R1, R3, ROR #21 ; R2 = R1 ⊕ (R3 >>> 21).

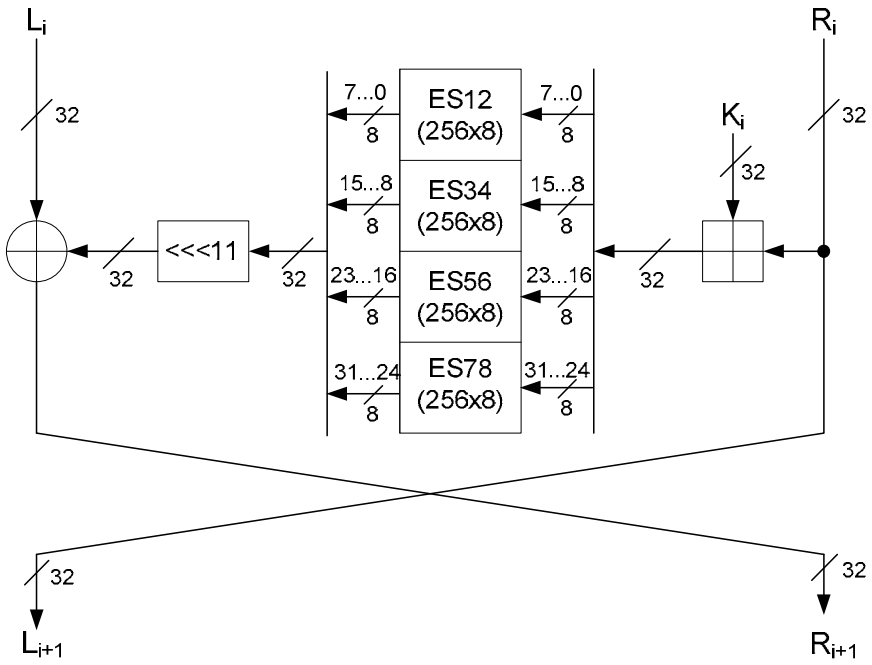

Рис. 3. Структурна схема раунду алгоритму ГОСТ 28147 для варіанта максимальної продуктивності

Структура програми складається з чотирьох ітерацій по 8 раундів. Щоб зменшити кількість тактів, потрібних для 32 викликів функції раунду, її включено безпосередньо в тіло програми (inline-функція).

Завдяки наявності в ARM-ядрі великої кількості регістрів загального призначення, всі проміжні змінні алгоритму розташовуються у регістрах, що зменшує кількість звернень до пам'яті, а отже, збільшує швидкодію. Для ядра ARM Cortex-M3 проміжні змінні потрібно намагатися розташовувати у регістрах R0-R7, оскільки в цьому разі генеруються 16-бітні команди, замість 32-бітних.

Результати реалізації мовами асемблера та С для варіанта максимальної швидкодії наведено у табл. 2.

Варіант з мінімальним розміром коду (SIZE). Цей варіант має дві відмінності від варіанта максимальної швидкодії. По-перше, не використовуються розширені таблиці заміни, по-друге, здійснюється виклик функції раунду.

Результати реалізації мовами асемблера та С для варіанта мінімального розміру коду наведено у табл. 2.

Таблиця 2

Результати реалізації алгоритму ГОСТ 28147 для ядер ARM Cortex-M3 та ARM7TDMI-S

Параметр реалізації	ARM Cortex-M3				ARM7TDMI-S			
	SPEED		SIZE		SPEED		SIZE	
	ASM	C	ASM	C	ASM	C	ASM	C
Шифрування, тактів/блок	477	611	1210	1269	608	702	1566	1499
Дешифрування, тактів/блок	479	595	1212	1265	610	678	1568	1501
Шифрування, тактів/байт	60	76	151	159	76	88	196	187
Дешифрування, тактів/байт	60	74	152	158	76	85	196	188
Розмір коду ROM, байт	1810	2116	434	474	1972	2152	420	620

Аналізуючи дані в табл. 2 варто відзначити, що ядро ARM Cortex-M3, загалом, має кращі показники, як за швидкодією, так і розміром коду. Продуктивність шифрування/дешифрування відрізняється незначно. Варіант SIZE потребує приблизно в 3,5–4,5 раза менше пам'яті ніж варіант максимальної швидкодії SPEED. Швидкодія для варіанта SPEED, при використанні мови асемблера, у 2,5 раза перевершує швидкодію реалізації SIZE, для мови С – у 2 рази.

Для типової максимальної тактової частоти ARM7TDMI-мікроконтролерів 60 МГц можна досягнути продуктивності шифрування/дешифрування на рівні 6 Мбіт/с. Мікроконтролери з ядром Cortex-M3 при тактовій частоті 120 МГц забезпечують продуктивність порядку 16 Мбіт/с. Враховуючи, що найбільш швидкісний стандартний інтерфейс USB, присутній в багатьох моделях ARM-мікроконтролерів, забезпечує швидкість передачі даних до 12 Мбіт/с (у режимі Full Speed), стає зрозуміло, що продуктивність реалізації алгоритму ГОСТ 28147 буде обмежуватися передусім не продуктивністю центрального процесора, а продуктивністю інтерфейсів вводу-виводу.

Порівняння даних у табл. 1 та 2 дає змогу стверджувати, що для обох варіантів реалізації (SPEED, SIZE), незалежно від мови програмування та процесорної архітектури, алгоритм ГОСТ 28147 демонструє перевагу у економії ресурсів пам'яті при співмірній продуктивності. Так, мінімальний розмір ROM-пам'яті для реалізації алгоритму AES є в 5,4 раза більший, ніж мінімальний розмір ROM для реалізації ГОСТ 28147. Відмова від LUT-таблиць під час реалізації алгоритму AES дозволяє отримати приблизно однакові з варіантом SPEED розміри коду у разі істотно меншої продуктивності (76 тактів/байт для ГОСТ 28147 проти 346 для AES-128).

Найшвидша реалізація AES-128 у 1,5 раза перевершує за швидкодією відповідну реалізацію ГОСТ 28147 для ядра ARM Cortex-M3 та у 1,9 раза для ядра ARM7TDMI-S. При однакових довжинах ключа, алгоритм AES-256 практично втрачає перевагу в швидкодії, випереджаючи реалізації ГОСТ 28147 у 1,1 та 1,4 раза для ядер ARM Cortex-M3 і ARM7TDMI-S відповідно.

Висновки

Було досліджено методи програмної реалізації вітчизняного криптоалгоритму ГОСТ 28147 для забезпечення шифрування у вбудованих системах на 32-бітних процесорах з архітектурою ARM.

Розглянуті варіанти реалізації алгоритму з мінімальним розміром коду та максимальною продуктивністю дозволяють досягнути компромісу між параметрами ціна – розмір коду – енергоспоживання залежно від конкретного застосування.

Результати порівняння з відомими реалізаціями іншого криптоалгоритму – AES, підтвердили перспективність та доцільність застосування у вбудованих системах на базі ARM-ядер алгоритму ГОСТ 28147, особливо за жорстких вимог щодо обсягу пам'яті.

1. FIPS-197: Advanced Encryption Standard (AES). Federal Information Processing Standard, National Institute of Standards and Technology, U.S. Dept. of Commerce, November 26, 2001. 2. 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, 2007, Amsterdam, NL, pp. 33–43. 3. 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. 4. 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. 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), 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, 2010. 7. 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). 8. Bertoni G., Breveglieri L., Fragneto P., Macchetti M., Marchesin S. Efficient Software Implementation of AES on 32-Bit Platforms // Proceedings of the 5th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'03), 2003, Cologne, Germany, Vol. 2779, pp. 159–171, Springer, Heidelberg (2003). 9. Atasu K., Breveglieri L., Macchetti M. Efficient AES implementations for ARM based platforms // Proceedings of the 2004 ACM symposium on Applied computing, 2004, Nicosia, Cyprus, pp. 841–845, ACM New York (2004). 10. Poschmann A., Ling S., Wang H. 256 Bit Standardized Crypto for 650 GE – GOST Revisited // Proceedings of the 12th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'10), 2010, Santa Barbara, USA, Vol. 6225, pp. 219-233, Springer, Heidelberg (2010). 11. Matthew Darnall, Doug Kuhlman. AES Software Implementations on ARM7TDMI // Progress in Cryptology INDOCRYPT 2006, vol. 4329, pp. 424–435. Springer 2006. 12. Using AES Encryption and Decryption with Stellaris Microcontrollers // Application Note, AN01251-03, 17 pp, Texas Instruments 2010. 13. 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". 14. Панасенко С. П. Алгоритмы шифрования. Специальный справочник. – СПб.: БХВ-Петербург, 2009. – 576 с. 15. Isobe T. A Single-Key Attack on the Full GOST Block Cipher // Proceedings of the 18th International Workshop on Fast Software Encryption (FSE'11), 2011, Lyngby, Denmark, Vol. 6733, pp. 290–305, Springer, Heidelberg (2011). 16. Редькин П. П. Микроконтроллеры ARM7 семейства LPC2000. Руководство пользователя. – М. Издательский дом «Додэка-XXI», 2007. – 560 с. 17. Yiu J. The Definitive Guide to the ARM Cortex-M3. Second Edition – Elsevier, 2010, 457 pp. 18. Procedure Call Standard for the ARM Architecture. – ARM IHI 0042D 34 pp., 2009.