УДК 004.056:061.68

Я. І. Грабовський, Я. Р. Совин, І. Я. Тишик
Національний університет “Львівська політехніка”,
кафедра захисту інформації

ПОРІВНЯННЯ РЕАЛІЗАЦІЙ НОВИХ АЛГОРИТМІВ ГЕШУВАННЯ SHA-3 ТА ГОСТ Р 34.11-2012 ДЛЯ 8/32-БІТОВИХ МІКРОКОНТРОЛЕРНИХ АРХІТЕКТУР

Виконано порівняльну оцінку продуктивності та розміру коду для криптографічних функцій гешування SHA-3 та ГОСТ Р 34.11-2012 у разі їх реалізації на 8-бітних мікроконтролерах родини AVR та 32-бітних мікроконтролерах з ядрами ARM Cortex M0/M0+/M3/M4.

Ключові слова: геш, криптографічна геш-функція, продуктивність, мікроконтролер, AVR, ARM, Cortex M.

COMPARISON IMPLEMENTATIONS OF NEW HASH ALGORITHMS SHA-3 AND GOST R 34.11-2012 FOR 8/32-BIT MICROCONTROLLER ARCHITECTURES

In this paper, the comparative evaluation of the performance and code size of cryptographic hash functions SHA-3 and GOST R 34.11-2012 is conducted for the implementation on 8-bit microcontroller family AVR and 32-bit microcontroller cores ARM Cortex M0/M0+/M3/M4.

Key words: hash, cryptographic hash function, performance, microcontroller, AVR, ARM, Cortex M.

Вступ

Сьогодні із поширенням електронних засобів криптографія відіграє вирішальну роль у забезпеченні конфіденційності та перевірки цілісності інформації. Зокрема, криптографічні функції гешування (КФГ) є одними з найважливіших примітивів, які використовують для створення криптографічних засобів захисту інформації. Вони призначені для підтвердження цілісності даних в електронному цифровому підписі та цифрових сертифікатах, електронних валютах, протоколах SSL, SSH, IPsec та TLS, різних протоколах автентифікації користувачів та повідомлень (наприклад, HMAC, HOTP/TOTP), комп’ютерних системах контролю цілісності та виявлення втручань тощо.

Широкий спектр застосувань КФГ і у вбудованих системах, це, зокрема, смарт-карти і RFID-мітки, сенсорні мережі, USB-ключі, інтелектуальні карти, OTP-токени, системи охоронно-пожежної сигналізації та контролю доступу, системи промислово-побутової автоматизації та моніторингу.

Необхідність захисту інформації у вбудованих системах привела до інтенсивних досліджень способів ефективної реалізації криптографічних алгоритмів, за умови обмежень, які накладають ці системи. Ресурси вбудованих систем обмежені продуктивністю процесорного ядра, споживаною потужністю, розміром доступної пам’яті.

У багатьох вбудованих системах, у яких ціна та витрати енергії є критичними, обчислювальна потужність сконцентрована у недорогих центральних процесорах, які входять до складу мікроконтро-

лерів. Тому існує потреба у КФГ, які б ефективно функціонували на цих платформах, а це стимулює дослідження їхньої продуктивності та пошуки методів оптимізації використовуваних ресурсів.

У 2007 р. Американський інститут стандартів і технологій оголосив трьохетапний конкурс на нову криптографічну функцію гешування SHA-3 (Secure Hash Algorithm Version 3), покликану замінити або доповнити діючу геш-функцію SHA-2. Основні вимоги до алгоритмів-конкурсантів передбачали створення класу геш-функцій, потенційно стійких до відомих атак, націлених на SHA-2, зі збереженням або збільшенням ефективності гешування порівняно з SHA-2. У жовтні 2012 р. переможцем серед п'яти фіналістів вибрано алгоритм Keccak.

Іншою важливою подією є прийняття Російською Федерацією нової КФГ як стандарту ГОСТ Р 34.11-2012, що набрав чинності з 1 січня 2013 року. Стандарт розроблений як заміна ГОСТ Р 34.11–94, у зв'язку з останніми результатами криптоаналізу, що вказують на його теоретичні недоліки, і виниклою потребою у створенні геш-функції, яка б відповідала сучасним вимогам до криптографічної стійкості та вимогам нового стандарту на електронний цифровий підпис ГОСТ Р 34.10–2012.

У цій роботі ми розглянемо ці алгоритми та опишемо основні особливості їх реалізації у вбудованих системах на базі популярних 8- і 32-бітних мікроконтролерних архітектур AVR і Cortex-M0/M0+/M3/M4 з представленням результатів вимірювання продуктивності.

Огляд відомих рішень

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

Оскільки обидва алгоритми гешування порівняно нові, то в мікропроцесорах і мікроконтролерах загального призначення поки що відсутні відповідні апаратні криптоакселератори, що зумовлює інтерес до програмних реалізацій.

Робота [2] містить детальну інформацію щодо оцінки продуктивності SHA-3 як в програмній, так і в апаратній реалізації. Також у цій роботі визначено теоретичні межі для оцінки кількості операцій, необхідних для виконання алгоритму. Основна увага приділена високопродуктивним платформам з архітектурами x86-32/64 та підтримкою SIMD-команд, для яких продуктивність коливається у межах 6–40 тактів/байт. Для тестування SHA-3 на мікроконтролерах з ядрами AVR та Cortex-M0/M3 алгоритм було написано на асемблері й для великих повідомлень досягнуто продуктивності 95 тактів/байт (для M3), 144 тактів/байт (для M0) та 1110 тактів/байт (для AVR).

Робота [4] досліджує усіх фіналістів третього етапу конкурсу SHA-3 на процесорі з ядром ARM11. Програмний код написано на асемблері для довжини результату 256 бітів. Досягнутий результат у випадку Keccak становить 72 такти/байт, за рахунок зменшення кількості операцій читання/запису та поєднання зсувів з арифметичними операціями.

У [5] можна знайти результати часу виконання та використання пам'яті програм алгоритмом SHA-3 для 16-бітних мікроконтролерів Freescale сім'ї S12X. Особливістю цієї реалізації є використання вбудованого XGATE-співпроцесора, що дає змогу розпаралелити виконання алгоритму між двома ядрами. Досягнута продуктивність становила 1005 тактів/байт (зі співпроцесором) та 2398 тактів/байт (без співпроцесора).

У роботі [6] подано результати реалізації SHA-3 та решти алгоритмів учасників другого етапу конкурсу на високопродуктивних процесорах, таких як Intel Core 2 Quad Q6600 (x86-32/64 архітектура), PowerPC 750 (PowerPC-архітектура), Broadcom BCM3302 (MIPS-архітектура), ARM920T (ARMv4-архітектура). Тестувалися реалізації мовами C та Java. Для C-коду автори прагнули досягнути максимальної сумісності та портативності, тому не використовували платформозалежні

можливості процесорів. Для 256-бітного гешу отримано такі результати: 20 тактів/байт (Intel), 128 тактів/байт (PowerPC), 253 такти/байт (BCM3302) та 300 тактів/байт (ARM920T). Реалізації мовою Java показали нижчу продуктивність.

Для 8-бітних AVR-мікроконтролерів у роботі [7] наведено результати використання оперативної пам'яті, розміру коду та кількості тактів для повідомлень з різною довжиною. Довжина результату гешування SHA-3 становила 256 бітів. Алгоритм, запрограмований мовою асемблера, дає змогу отримати 2796 тактів/байт.

Результати реалізації алгоритму SHA-3 для платформ x86-32 та x86-64 (Intel Core i5) мовою Java наведено у [8]. Найвищу продуктивність – близько 120 байтів/такт досягнуто на 64-бітній архітектурі під час гешування файлів великого розміру (256 Мбайт).

Що стосується публікацій про реалізації функції гешування ГОСТ Р 34.11-2012 на різних платформах, то їх поки що доволі небагато (порівняно з SHA-3), що пояснюється меншим часом публічного обговорення (відкритий конкурс науково-дослідних робіт з аналізу геш-функції ГОСТ Р 34.11-2012 заплановано провести з 21 жовтня 2013 року до 15 грудня 2014 року) та меншим колом потенційних користувачів.

У роботі [9] наведено шляхи оптимізації й відповідні реалізації алгоритмів теперішнього і попереднього стандартів ГОСТ Р 34.11-2012 і ГОСТ Р 34.11-94 для процесорів архітектури x86-64 (Intel Core i7-920) та графічних процесорів NVIDIA з підтримкою технології CUDA. Для ГОСТ Р 34.11-2012 досягнуто продуктивності 27 тактів/байт (архітектура x86-64), порівняно з 40 байтами/такт для алгоритму ГОСТ Р 34.11-94. Також у роботі зазначено, що нова КФГ може бути вдвічі швидша, ніж стара на сучасних мікропроцесорах, проте повільнішою на обмежених у ресурсах пристроях.

Дослідження продуктивності геш-функції ГОСТ Р 34.11-2012 провели також її розробники на 64-бітному процесорі Intel Xeon E5335 2 ГГц. Використовувалося одне ядро. Продуктивність становила 51 такт/байт [3].

Отже, аналіз публікацій показує, що під час оцінювання нових геш-функцій основна увага приділена високопродуктивним платформам, а використання у вбудованих системах SHA-3 і особливо ГОСТ Р 34.11-2012 досліджено недостатньо.

Мета статті

Мета статті – дослідити способи ефективної програмної реалізації геш-функцій SHA-3 та ГОСТ Р 34.11-2012 на найпоширеніших 8/32-бітних мікроконтролерних платформах, оцінити продуктивність та вимоги до пам'яті одержаних реалізацій, що дасть змогу вибирати оптимальне рішення для впровадження криптографічного захисту у вбудовані системи.

Аналіз структури досліджених функцій гешування

Алгоритм SHA-3

Під час проведення конкурсу до алгоритмів-претендентів ставилися аналогічні вимоги, що і до попередніх геш-функцій, які стосувалися розміру вихідного і вхідного блоків, стійкості до колізій, однорідності алгоритмів обчислення для різного розміру блоків тощо. Разом з тим, існували вимоги, спрямовані на вдосконалення наявних алгоритмів геш-функцій: можливість рандомізованого гешування, покращене розпаралелення, ефективне функціонування на широкому спектрі сучасних обчислювальних платформ (від 64-бітних процесорів до 8-бітних мікроконтролерів та смарт-карт).

Переможець конкурсу SHA-3 алгоритм Keccak (надалі просто алгоритм SHA-3) підтримує розміри вихідного гешу 224, 256, 384 та 512 біт. Дані обробляються блоками розміром 1152/1088/832/576 біт, при цьому внутрішній стан представляється у вигляді масиву 5×5, елементами якого є 64-бітні слова (тобто розмір стану становить 1600 бітів). Алгоритм складається з 24 раундів.

Алгоритм SHA-3 – це криптографічна функція гешування, в основу якої покладена конструкція "губка". У такій конструкції виділяють дві фази (рис. 1).

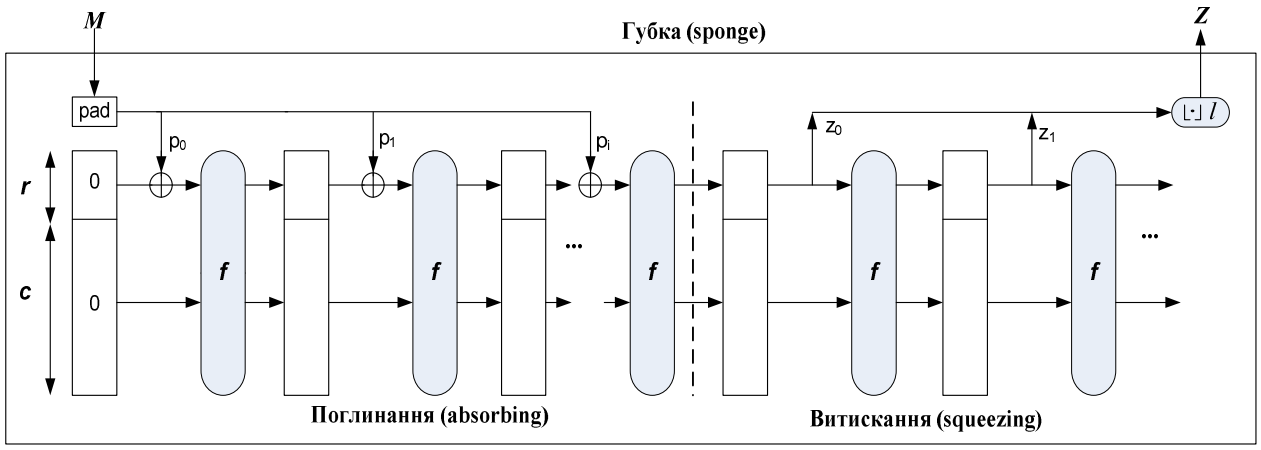

Рис. 1. Конструкція "губка", використана для геш-функції

Перша фаза – поглинання. Вхідне повідомлення розділяють на блоки розміром *r* бітів. Після додавання блока за модулем 2 до перших *r* бітів стану проводиться перестановка. Така послідовність дій виконується з усіма блоками повідомлення.

Друга фаза – витискання. На вихід подається *r* перших бітів стану. Якщо довжина результату повинна бути більша, виконується перестановка, і до попереднього результату додається ще *r* бітів, поки не буде досягнуто бажаної довжини. У разі перевищення потрібної довжини результату зайві біти відкидають. Розмір стану $b = 1600$ біт, де $b = r + c$. Параметр $c = 2n$, де n – довжина результату гешування. У разі подання повідомлення, розмір якого не кратний довжині блока, також застосовується доповнення.

Стан a організовано у вигляді тривимірного масиву 5×5×64 бітів або 64-бітних смуг, елементи якого належать скінченному полю *GF(2)*. Адреса кожного біта позначається як $a[x][y][z]$, де $x, y \in Z_5$, $z \in Z_{64}$. Якщо розглядати стан як одновимірний масив, співвідношення між адресами бітів таке: $s[64 \cdot (5y + x) + z] = a[x][y][z]$.

Перестановка складається з 24 раундів. Кожен раунд містить п'ять кроків: θ, ρ, π, χ, ι. На кроці θ до кожного біта $a[x][y][z]$ додаються за модулем 2 усі біти стовпців $a[x-1][\cdot][z]$ та $a[x+1][\cdot][z-1]$. Крок ρ виконує циклічні зсуви смуг на задані значення. На кроці π виконується перестановка смуг. Крок χ кожен біт додає за модулем 2 із результатом операції кон'юнкції двох наступних бітів ряду, перший з яких інвертований. Крок ι складається з додавання константи раунду до першої смуги стану. Константа залежить від номера раунду. Раунд повинен починатися із θ. Послідовність інших кроків не важлива [1].

Смуги стану подаються у вигляді процесорних слів. Якщо довжина слова процесора менша, можна застосувати побітове чергування. Ця методика полягає у представленні 64-бітної смуги у вигляді масиву із $s = \frac{64}{m}$ слів процесора, кожне з яких дорівнює m бітам, зі словом ζ, яке містить біти $z \equiv \zeta \pmod{s}$ смуги [2].

Алгоритм ГОСТ Р 34.11-2012

ГОСТ Р 34.11-2012 – це криптографічна функція, яка дає змогу отримати результати гешування довжиною 256 та 512 біт. Механізм роботи функції однаковий для обох випадків. Відмінність полягає у різних початкових значеннях і в тому, що для отримання 256 бітів потрібно відкинути таку ж кількість молодших бітів від результату. Стан функції та блоки повідомлення дорівнюють 512 бітам.

Функцію гешування можна розділити на три етапи (рис. 2).

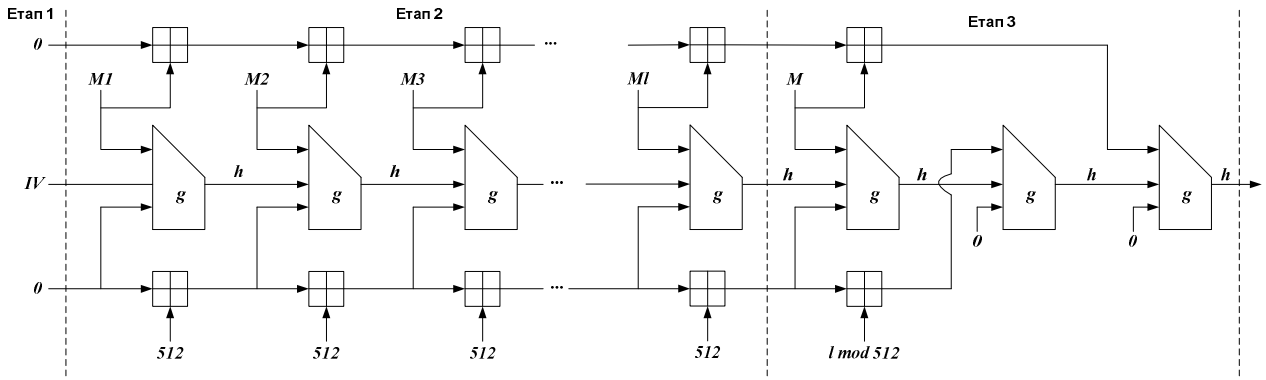

Рис. 2. Структура функції гешування ГОСТ Р 34.11-2012

Перший етап – ініціалізація значень. На другому етапі обробляються усі блоки повідомлень, крім останнього. Кожен блок подається на функцію стиснення g. Разом з цим, обчислюється кількість опрацьованих бітів N та сума повідомлення $\sum$, що розглядаються як числа за модулем 2^{512}. Функція стиснення g основана на 512-бітному блоковому шифрі, який являє собою варіант AES з 64-байтним станом та 12 раундами. Раундове перетворення складається з рівня нелінійних перетворень, байтової перестановки, лінійного перетворення та накладання раундового ключа.

Третій етап розпочинається доповненням останнього блока. Далі виконується така сама послідовність дій, як і на другому етапі. У кінці на функцію стиснення подається довжина повідомлення у бітах та сума.

Варто звернути увагу на перетворення *L, P, S* у функції стиснення. *S*-перетворення полягає у заміні кожного байта на відповідний байт із таблиці π. Перетворення *P* – перестановка байтів за заданою послідовністю. У *L*-перетворенні відбувається лінійне перетворення. Стан ділять на 8 64-бітні вектори, кожен з яких множиться на задану матрицю *A* над полем *GF(2)* [3].

Для збільшення швидкодії можна розширити матрицю у лінійному перетворенні *L* [10]. Розмір нової матриці становить 16 Кбайт.

Огляд досліджуваних процесорних архітектур

Архітектура 8-бітних AVR-мікроконтролерів

Мікроконтролери родини AVR компанії Atmel – це поширені 8-бітні високопродуктивні RISC-мікроконтролери загального призначення, які характеризуються низьким енергоспоживанням, високою продуктивністю за помірної ціни. У мікроконтролері міститься процесорне ядро та доволі багатий набір периферійних модулів. Ядро побудоване за статичною архітектурою, що дає змогу використовувати енергозбережні режими роботи. Також воно має арифметико-логічний пристрій, який з'єднаний із 32 регістрами загального призначення, об'єднаними у регістровий файл. Пам'ять організована за гарвардською архітектурою з розділеними пам'яттю програм та даних. Дворівневий конвеєр забезпечує виконання більшості команд за 1 такт. Майже усі команди займають одну 16-бітну комірку пам'яті. Таке подання забезпечує багату систему команд.

Пам'ять даних складається із трьох областей: регістрової пам'яті, статичної оперативної пам'яті та EEPROM-пам'яті. У регістровому файлі кожен регістр може бути джерелом або приймачем даних. Шість з 32 регістрів можуть використовуватися як три 16-бітові покажчики адреси у процесі непрямого адресування даних. AVR-мікроконтролери також підтримують різні види адресації. Така організація пам'яті та доступу до неї забезпечує написання ефективного програмного коду [11].

Архітектура 32-бітних мікроконтролерів ARM Cortex-M

Процесори Cortex-M компанії ARM розроблені для ринку 32-бітних мікроконтролерів. Самі мікроконтролери забезпечують високу продуктивність за низького енергоспоживання та ціни, мають вбудовану велику кількість додаткових модулів. Процесори Cortex-M є 32-бітними

RISC-процесорами. Вони мають 16 32-бітних регістрів, 13 з яких призначено для загального користування. Кожен з трьох інших регістрів має спеціальне призначення: лічильник команд, що містить адресу виконуваної команди; регістр, що зберігає адресу повернення у разі виклику підпрограми; покажчик стека, що складається з двох регістрів, проте в кожен момент часу доступний лише один з них. Є також три- або дворівневий конвеєр, а більшість команд виконується за один такт. У мікроконтролерах організовано фіксований розподіл адресного простору, але конструкція процесора дає змогу задіяти зазначені області по-різному. Передбачені два режими роботи процесора і два рівні доступу до коду програми. Із привілейованим доступом можна використовувати режим обробника та потоку, а за непривілейованого – лише останній. У режимі обробника може виконуватися код обробника переривання або системного винятку, в режимі потоку – код програми. Два рівні доступу забезпечують базову модель захисту [12, 13].

Тепер випускаються мікроконтролери з ядрами M0 (найменша продуктивність), M0+ (оптимізована версія M0), M3 та M4 (найвища продуктивність), для яких і будуть проводитися дослідження.

Результати

Для написання і компіляції коду та оцінки продуктивності використано інтегровані середовища розроблення компанії IAR Systems: IAR Embedded Workbench for ARM v6.60 та IAR Embedded Workbench for AVR v.5.51.

Тестування реалізованих версій криптографічних функцій гешування полягало у вимірюванні швидкості оброблення одного байта довгого повідомлення (100 КБайт), розміру коду програми та обсягу потрібної оперативної пам’яті. Також компілювання коду програми виконувалося з різним ступенем оптимізації, спрямованої на збільшення швидкості виконання коду. Отримані результати вимірювання наведено у табл. 1–6. Для обох видів мікроконтролерів функції гешування написано мовою С. Довжина результатів гешування – 512 біт.

Алгоритм SHA-3 для Cortex-M та AVR реалізований у вигляді простої послідовності кроків. Застосовано побітове чергування. Розмір слова – 32 біти.

Для написання алгоритму ГОСТ Р 34.11-2012 для Cortex-M використано 32-бітні вектори даних.

Таблиця 1

Результати вимірювання параметрів реалізації алгоритму SHA-3 на мікроконтролерах з архітектурою ARM Cortex-M

Оптимізація	Такт/байт		Розмір коду, байт		Обсяг оперативної пам’яті, байт	
	M0/M0+	*M3/M4*	*M0/M0+*	*M3/M4*	*M0/M0+*	*M3/M4*
відсутня	1176	1359	2160	1936	300	308
низька	1062	1263	1990	1750	288	300
середня	694	828	1562	1468	320	328
висока	554	719	1376	1346	304	324

Під час реалізації алгоритму ГОСТ Р 34.11-2012 розглядали два варіанти: без розширення матриці в *L*-перетворенні та з використанням розширеної матриці, що потребує 16384 байт пам’яті.

Для AVR довжина вектора даних становить 8 бітів. Застосовано просту послідовність кроків.

Таблиця 2

Результати вимірювання параметрів реалізації алгоритму ГОСТ Р 34.11-2012 на мікроконтролерах з архітектурою Cortex-M (матриця *A* не розширена)

Оптимізація	Такт/байт		Розмір коду, байт		Обсяг оперативної пам’яті, байт	
	M0/M0+	*M3/M4*	*M0/M0+*	*M3/M4*	*M0/M0+*	*M3/M4*
відсутня	6734	7429	2946	2828	540	556
низька	6249	7257	2836	2724	524	532
середня	3501	5523	2504	2554	540	548
висока	1805	3791	3984	4840	556	580

Таблиця 3

Результати вимірювання параметрів реалізації алгоритму ГОСТ Р 34.11-2012 на мікроконтролерах з архітектурою Cortex-M (матриця *A* розширена)

Оптимізація	Такт/байт		Розмір коду, байт		Обсяг оперативної пам'яті, байт	
	M0/M0+	*M3/M4*	*M0/M0+*	*M3/M4*	*M0/M0+*	*M3/M4*
відсутня	767	1068	21294	21632	516	532
низька	757	1062	21254	21566	500	516
середня	493	632	20026	20080	516	524
висока	324	434	20192	21330	532	544

Таблиця 4

Результати вимірювання параметрів реалізації алгоритму SHA-3 на мікроконтролерах з архітектурою AVR

Оптимізація	Такт/байт	Розмір коду, байт	Обсяг оперативної пам'яті, байт
Відсутня	8596	5202	277
Низька	8320	4670	269
Середня	7695	3698	303
Висока	7058	3464	274

Таблиця 5

Результати вимірювання параметрів реалізації алгоритму ГОСТ Р 34.11-2012 на мікроконтролерах з архітектурою AVR (матриця *A* не розширена)

Оптимізація	Такт/байт	Розмір коду, байт	Обсяг оперативної пам'яті, байт
Відсутня	54130	3288	430
Низька	53627	3234	414
Середня	24862	2858	414
Висока	10389	2804	420

Таблиця 6

Результати вимірювання параметрів реалізації алгоритму ГОСТ Р 34.11-2012 на мікроконтролерах з архітектурою AVR (матриця *A* розширена)

Оптимізація	Такт/байт	Розмір коду, байт	Обсяг оперативної пам'яті, байт
Відсутня	16630	19134	420
Низька	16355	19082	410
Середня	9343	19121	410
Висока	4780	18666	415

Висновки

Подано результати реалізацій функцій гешування SHA-3 і ГОСТ Р 34.11-2012 для мікроконтролерів AVR та ARM Cortex-M0/M0+/M3/M4.

Вища продуктивність ядер M0/M0+ порівняно з ядрами M3/M4 пояснюється тим, що у них команда доступу до пам'яті виконується за один такт, а у M3/M4 – за два такти. Разом з тим, незважаючи на обмежений набір команд, який підтримують ядра M0/M0+, порівняно з M3/M4 об'єм коду відрізняється незначно.

За даними табл. 1–3 видно, що алгоритм SHA-3 за високої оптимізації є у 3,3 разу швидшим для ядер M0/M0+ та у 5,3 разу для ядер M3/M4, ніж ГОСТ Р 34.11-2012 без розширеної матриці. Виграш у об'ємі коду алгоритму SHA-3 становить 2,9 разу для ядер M0/M0+ та 3,6 разу для ядер M3/M4, а необхідний об'єм ОЗП є у 1,8 разу менший. Зі зниженням рівня оптимізації якісна картина зберігається. Отже, можна стверджувати, що алгоритм SHA-3 загалом потребує менше ресурсів та є помітно швидшим за алгоритм ГОСТ Р 34.11-2012, якщо в останньому не використовуються спеціальні заходи.

Використання розширеної *A*-матриці в алгоритмі ГОСТ Р 34.11-2012 дає змогу підвищити продуктивність для ядер M0/M0+ від 5,6 до 8,3 разу (залежно від рівня оптимізації), а для ядер M3/M4 від 6,8 до 8,7 разу. При цьому об'єм коду зростає у 5,1–8,0 разів для ядер M0/M0+ та у 4,4–7,9 разу для ядер M3/M4.

Якщо ж порівнювати ГОСТ Р 34.11-2012 з розширеною *A*-матрицею з алгоритмом SHA-3, то за вищої у 1,3–1,7 разу продуктивності його розмір коду зростає у 9,8–14,7 разу для ядер M0/M0+ та 11,2–15,8 разу для ядер M3/M4. Отже, за співмірної з SHA-3 швидкодії алгоритм ГОСТ Р 34.11-2012 потребує значно більших ресурсів пам'яті програм, що суттєво обмежує коло потенційних застосувань. Об'єм пам'яті в 20 Кбайт, необхідний для реалізації алгоритму ГОСТ Р 34.11-2012 з розширеною матрицею, є досить великим навіть для мікроконтролерів з ядрами M3/M4, а для мікроконтролерів з ядрами M0/M0+ взагалі неприйнятним, оскільки вони створювалися як недорогі контролери з обмеженими ресурсами для критичних до вартості застосувань з типовим розміром Flash-пам'яті 16/32/64 Кбайт.

На мікроконтролерах AVR алгоритм SHA-3 теж виявився швидшим за алгоритм ГОСТ Р 34.11-2012 без використання розширеної матриці. За високої оптимізації алгоритм SHA-3 швидший приблизно у 1,5 разу, якщо відсутня оптимізація – у 6,3 разу. Розмір коду для алгоритму SHA-3 є дещо більшим за ГОСТ Р 34.11-2012: від 1,2 (оптимізація відсутня) до 1,6 (висока оптимізація) разу. Абсолютні розміри коду програм для обох алгоритмів прийнятні з практичного погляду для 8-бітних мікроконтролерів. За використанням оперативної пам'яті економнішим є алгоритм SHA-3 (приблизно у 1,5 разу).

Використання розширеної матриці для реалізації алгоритму ГОСТ Р 34.11-2012 на AVR-мікроконтролерах спричиняє збільшення об'єму коду до близько 19 Кбайт, що у багатьох випадках є неприйнятним для вбудованих 8-бітних систем. За рахунок шестикратного збільшення розміру програми виграш у швидкодії порівняно з SHA-3 досягається тільки за максимальної оптимізації та становить лише 1,5 разу, а програш у об'ємі коду алгоритму SHA-3 становить 5,4 разу.

Отже, на нашу думку, для використання у вбудованих системах придатнішим є геш-алгоритм SHA-3, який забезпечує кращий баланс швидкодія–пам'ять. Головним недоліком алгоритму ГОСТ Р 34.11-2012 у разі реалізації на мікроконтролерах є непропорційність обміну швидкодія–пам'ять, особливо для 8-бітних архітектур.

1. Bertoni G. The Keccak Reference // Bertoni G., Daemen J., Peeters M., Van Assche G. – 2011. – 69 p. 2. Bertoni G. Keccak implementation overview // Bertoni G., Daemen J., Peeters M., Van Assche G., Van Keer R. – 2012. – 59 p. 3. ГОСТ Р 34.11-2012: Национальный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Функция хеширования: изд. официальное. – М. Стандартинформ, 2012. 4. Schwabe P., Yang B., Yang S. SHA-3 on ARM11 processors // Proceedings of the 5th International Conference on Cryptology in Africa (AFRICACRYPT'12). – P. 324–341, Springer-Verlag Berlin, Heidelberg (2012). 5. Murvay P., Groza B. Performance improvements for SHA-3 finalists by exploiting microcontroller on-chip parallelism // 6th International Conference on Risk and Security of Internet and Systems (CRiSIS'11), 2011, Timisoara, RO. – P. 1–8. 6. Pornin T. Comparative Performance Review of the SHA-3 Second-Round Candidates // Proc. of the Second SHA-3 Candidate Conference, 2010. 7. Balasch J., Ege B., Eisenbarth T., Gérard B., Gong Z., Güneysu T., Heyse S., Kerckhof S., Koeune F., Plos T., Pöppelmann T., Regazzoni F., Standaert F-X., Van Assche G., Van Keer R., Van Oldeneel tot Oldenzeel L., Maurich I. Compact Implementation and Performance Evaluation of Hash Functions in ATtiny Devices // Proceedings of the 11th International Conference Smart Card Research and Advanced Application (CARDIS'12), 2012, Graz, Austria, LNCS, Vol. 7771. – P. 158–172, Springer-Verlag Berlin Heidelberg (2013). 8. Dahal R., Bhatta J., Dhamala T. Performance analysis of SHA-2 and SHA-3 finalists // International Journal on Cryptography and Information Security. – 2013. – Vol. 3, № 3. 9. Лебедев П. Сравнение старого и нового стандартов РФ на криптографическую хэш-функцию на ЦП и графических процессорах NVIDIA // Математические вопросы криптографии. – 2013. – Т. 4, Вып. 2. – С. 73–80. 10. Kazymyrov Oleksandr, Shevchuk Oleksii. Implementation of hash function Stribog. https://github.com/okazymyrov/stribog. 11. Ефстифеев А. В. Микроконтроллеры AVR семейств Tiny и Mega фирмы "ATMEL". – М.: Додэка-XXI, 2004. – 506 с. 12. The Definitive Guide to ARM Cortex-M3 and Cortex-M4 Processors. Third Edition // Joseph Yiu. – Elsevier Inc., 2014. – 1055 p. – ISBN-13: 978-0-12-408082-9. 13. The Definitive Guide to ARM Cortex-M0 // Joseph Yiu. – Elsevier Inc., 2011. – 529 p. – ISBN: 978-0-12-385477-3.