Volodymyr KHOMA[1], Małgorzata ZYGARLICKA[1], Yaroslav SOVYN[2], Yaroslav RESHETAR[2]

Opole University of Technology, Department of Electrical Engineering, Automatic Control and Informatics (1),
"Lviv Polytechnic" National University, Institute of Computer Technologies, Automation and Metrology (2), Ukraina

doi:10.15199/48.2018.04.37

Implementacja algorytmu kryptograficznego „Kalyna" w systemach wbudowanych

Abstract. *This paper describes an implementation of novel cryptographic algorithm "Kalyna", which is the national encryption standard of Ukraine. The performance and the code size requirement were estimated for implementation of this algorithm on general purpose 8/16/32-bit microcontrollers. Also, a comparative analysis of these parameters with other modern cryptographic algorithms was made.* ***An implementation of novel cryptographic algorithm "Kalyna"***

Streszczenie. *W pracy opisano sposoby realizacji nowego algorytmu kryptograficznego „Kalyna", będącego narodowym standardem szyfrowania Ukrainy. Dokonano estymacji wydajności oraz objętości kodu w przypadku realizacji kryptoalgorytmu na 8/16/32-bitowych mikrokontrolerach ogólnego przeznaczenia. Również przeprowadzono analizę porównawczą wymienionych parametrów z innymi nowoczesnymi kryptoalgorytmami.*

Keywords: symmetric block ciphers, DSTU 7624:2014, Kalyna, embedded systems, performance.
Słowa kluczowe: symetryczne szyfry blokowe, DSTU 7624:2014, Kalyna, systemy wbudowane, wydajność.

Wprowadzenie

Szyfry symetryczne stanowią podstawowe zabezpieczenie kryptograficzne podczas transmisji danych w systemach telekomunikacyjnych. Ponad to bardzo ważnym zastosowaniem tego rodzaju szyfrów są różne prymitywy kryptograficzne, takie jak generatory liczb losowych, funkcje skrótu, kody uwierzytelniania wiadomości itp. [1]. Obecnie istnieje wiele algorytmów szyfrowania symetrycznego, które zbudowane są według różnych zasad (sieć Feistela, SPN, ARX), o różnych parametrach (długość klucza, rozmiar bloku, liczba rund) zapewniających różne poziomy mocy kryptografacznej i złożoność realizacji (algorytmy lekkie). Niektóre najbardziej znane algorytmy kryptograficzne zatwierdzone zostały jako normy międzynarodowe i krajowe.

Dla kryptoalgorytmów symetrycznych oprócz zapewnienia wysokiej odporności na złamanie ważnym jest też duża szybkość przetwarzania, niskie zapotrzebowanie mocy obliczeniowej i pamięci procesora oraz przydatność do implementacji na szerokiej gamie platform sprzętowych i programowych. Jest to szczególnie aktualnym zagadnieniem w odniesieniu do systemów wbudowanych (Embedded Systems), gdyż cena i zużycie energii jest priorytetem, ponieważ moc obliczeniowa jest skoncentrowana w niedrogich procesorach centralnych (CPU) będących częścią mikrokontrolerów ogólnego przeznaczenia. Takie systemy są coraz częściej wykorzystywane do budowy sieci czujników bezprzewodowych (Wireless Sensors Network - WSN), układów przemysłowych, konsumenckich, medycznych, motoryzacyjnych i cyber-fizychnych, urządzeń Internetu rzeczy (Internet of Things - IoT), kart inteligentnych, OTP-tokenów, systemów sygnalizacji pożaru, zabezpieczeń i kontroli dostępu, automatyzacji i monitoringu w zastosowaniach przemsłowych i domowych, urządzeń ubieralnych (fitness trackery, zegarki i okulary inteligentne) i inne [2,3].

Potrzeba ochrony informacji w systemach wbudowanych doprowadziła do intensywnych badań nad sposobem skutecznej realizacji algorytmów kryptograficznych z uwzględnieniem właściwych tym systemom ograniczeń. Zasoby systemów wbudowanych typowo są ograniczone wydajnością rdzenia procesorowego, zużywaną mocą, przestrzenią dostępnej pamięci ulotnej i nieulotnej [3,4]. Długość kodu programu ma bezpośredni wpływ na koszt mikroprocesora, będącym często najdroższym składnikiem systemu. Czas wykonania operacji jest krytyczny ze względu na zużywaną moc, ponieważ zwykle w systemach wbudowanych CPU przez większość czasu przebywa w trybie zmniejszonego zużycia energii, opuszczając go na krótko w celu akwizycji, przetwarzania i przekazywania informacji. W związku z tym czas wykonania algorytmu kryptograficznego jest proporcjonalny do zużycia energii przez urządzenie.

Analiza ostatnich badań i publikacji

W krajach należących do byłego ZSRR od 1990 roku jako standart państwowy został przyjęty kryptoalgorytm GOST (tytuł oficjalny ГОСТ 28147-89) o rozmiarze przetwarzanego bloku 64 bity i długości kluczy 256 bitów [5]. Po rozwiązaniu ZSRR w niepodległych państwach Rosji, Ukrainie, Białorusi ten szyfr zdobył status normy państwowej i w tym statusie był wykorzystywany do niedawna. Obecnie znane są metody teoretyczne kryptoanalizy o znacznie mniejszej złożoności obliczeniowej od ataku brutalnego [6,7]. Szybki rozwój technologii telekomunikacyjnych, a w konsekwencji wzrost objętości przetwarzanej i transmitowanej informacji, doprowadziły do tego, że 64-bitowa długość przetwarzanego bloku nie jest już wystarczająca. Z tych powodów szyfr GOST został wycofany jako standard w Białorusi, Ukrainie i Rosji.

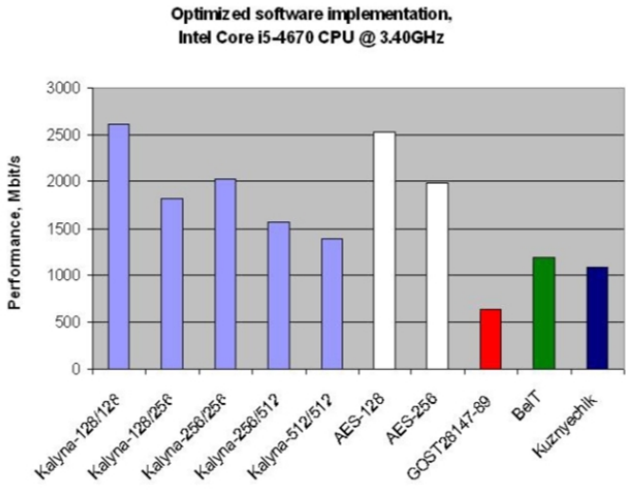
Ponadto pod względem wydajności szyfr GOST przegrywa z nowoczesnymi szyframi, przykładowo AES [8]. Jednak zastąpienie GOST na międzynarodowy standard i najpopularniejszy obecnie szyfr AES nie byłoby słusznym rozwiązaniem dla Ukrainy, gdyż globalne trendy wskazują na stopniowe wycofywanie tego szyfru podczas wyboru perspektywicznych rozwiązań w międzynarodowych konkursach kryptograficznych [9]. Na przykład, niektóre firmy, liderzy branży IT takie jak np. Google już wykorzystują nowe kryptoalgorytmy zamiast AES.

W dniu 1 lipca 2015 roku w Ukrainie wprowadzono nowy narodowy standard kryptograficzny „Kalyna" (nazwa oficjalna ДСТУ 7624:2014) [10]. Ten nowy kryptoalgorytm podtrzymuje rozmiar bloku i długość klucza 128, 256 i 512 bitów, zapewniając odpowiednio normalny, wysoki i bardzo wysoki poziom ochrony. Obecnie „Kalyna" to jedyny na świecie kryptoalgorytm symetryczny wspierający 512-bitowy klucz.

W artykułach [9,11] przedstawiono wyniki porównania wydajności szyfru „Kalyna" (wszystkie kombinacje rozmiaru bloku i długości klucza), AES-128, AES-256 (AES został standardem rządu federalnego USA od 26 maja 2002 roku), GOST 28147- 89 (poprzednia norma w Rosji, Ukrainie i Białorusi), „BelT" (СТБ 34.101.31-2011 – nowy standard

Narodowy Białorusi od 31 stycznia 2011 roku) oraz algorytm „Kuznieczik" (ГОСТ Р 34.12-2015 - nowy standard Rosyjskiej Federacji od 1 stycznia 2016 roku) [12].

Badania przeprowadzono na 64-bitowym komputerze z procesorem Intel Core i5-4670, który był taktowany zegarem 3.40 GHz i z systemem operacyjnym Linux (Ubuntu). W realizacji programowej AES nie był używany zestaw instrukcji AES-NI. Wyniki testów przedstawiono na rysunku 1.



Rys. 1. Szybkość realizaacji programowych kryptoalgorytmów w wersjach zoptymizowanych [11]

Badanie szybkości na platformie 64-bitowej kryptoalgorytmy „Kalyna" i AES wykazały podobne wyniki. W porównaniu z nowymi standardami Rosji i Białorusi szyfr „Kalyna" zapiewnia około 2 razy wyższą szybkość przetwarzania.

W pracy [13] przedstawiono wyniki obliczeń teoretycznych dotyczących ilości niezbędnych instrukcji procesora do przetwarzania jednego bajtu danych. Jako platformę obliczeniową wybrano 64-bitowy mikroprocesor ogólnego przeznaczenia o architekturze x86-64 firmy Intel, a oprócz „Kalyny" były badane wcześniej wymienione szyfry GOST, AES oraz narodowy standard Białorusi szyfr „BelT" (СТБ 34.101.31-2011). Podane w tabeli 1 wyniki wskazują na przewagę szyfru „Kalyna".

Tabela 1. Teoretyczne oszacowanie ilości instrukcji procesorowych potrzebnych do szyfrowania 1 bajtu danych [13]

	Rodzaj algorytmu kryptograficznego			
	Kalyna (128/128)	GOST	BelT	AES
operacji na 1 Bajt	40,375	72	40,5	45,375

Według publikacji [9,11] szyfr „Kalyna" jest ukierunkowany na osiągnięcie wysokiej wydajności na 64-bitowych nowoczesnych mikroprocesorach ogólnego przyznaczenia (Intel, AMD). Jak zadeklarowali sami autorzy szyfru „Kalyna" [9], wymogi skutecznej implementacji tego szyfru na układach o ograniczonej mocy obliczeniowej uznano jako drugorzędne, ze względu na brak w Ukrainie własnej produkcji mikroprocesorów i niezdolność do zapewnienia właściwej kontroli układów zagranicznych.

Cel artykułu

Artykuł ma na celu zbadanie sposobów skutecznej realizacji programowej kryptoalgorytmu „Kalyna" na najbardziej popularnych 8/16/32-bitowych platformach procesorowych, aby ocenić szybkość i wymagania dla pamięci oraz przeprowadzić analizę porównawczą z innymi szyframi symetryczymi, takimi jak AES, GOST, „Kuznieczik". Wyniki badań są ważne pod kątem wyboru najlepszego rozwiązania przy projektowaniu mechanizmów ochrony w systemach wbudowanych.

Osobliwości strukturalne algorytmu kryptograficznego „Kalyna"

Standard szyfrowania „Kalyna" [10] odnosi się do SPN-szyfrów bajtowo zorientowanych. Podstawowe parametry szyfru, takie jak rozmiar bloku danych l, długość klucza k, liczba rund t oraz liczba kolumn macierzy stanu c są związane zależnościami podanymi w tabeli 2. Rozmiar bloku i długość klucza występują w notacji szyfru jako wskaźnik, czyli „Kalyna-l/k".

Tabela 2. Parametry podstawowe szyfru „Kalyna"

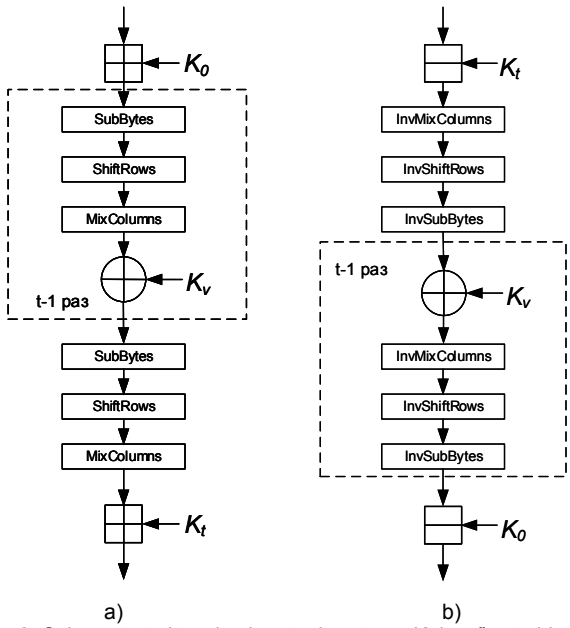
Długość klucza *k*, bitów	Rozmiar bloku, bitów	Liczba rund *t*	Liczba kolumn macierzy stanów *c*
128, 256	128	10	2
256, 512	256	14	4
512	512	18	8

Podczas wykonywania szyfrowania i deszyfrowania przetwarzania są przeprowadzane na dwuwymiarowej tabeli bajtów, zwanej aktualnym stanem szyfru (***State***). Bieżący stan szyfru można przedstawić jako macierz o wymiarze 8x*c* bajtów (osiem wierszy po *c* bajtów):

$$State = (s_{i,j}), \quad (1)$$

gdzie $i = 0...7, j = 0...c-1$.

Algorytm kryptograficzny wykorzystuje operacje arytmetycznego dodawania ($\boxplus$) i odejmowania ($\boxminus$) modulo 2^{64}, dodawanie modulo 2 ($\oplus$), zamiany tablicowe (***SubBytes***, ***InvSubBytes***), przesunięcia cyklicznego wierszy (***ShiftRows***, ***InvShiftRows***) oraz transformacje liniowe (***MixColumns***, ***InvMixColumns***). Struktura kryptoalgorytmu „Kalyna" została pokazana na rysunku 2 [10].



Rys. 2. Schemat strukturalny kryptoalgorytmu „Kalyna" w trybie szyfrowania (a) i deszyfrowania (b)

Rozważmy bardzej szczegółowo każdą z tych operacji.

Operacje dodawania i odejmowania dokonują dodawania lub odejmowania arytmetycznego kolumn macierzy stanu ***State*** i kolumn podklucza cyklicznego modulo 2^{64}. Liczby w kolumnach są reprezentowane w formacie *little-endian*, czyli najmniej znaczące bajty posiadają mniejszy wskaźnik. Operacja dodawania modulo

2 dokonuje dodawania modulo 2 macierzy stanu ***State*** i podklucza cyklicznego.

Operacje ***SubBytes*** i ***InvSubBytes*** - zamieniają każdy bajt macierzy stanu na odpowiadający mu bajt jednej z czterech tabel ***S0-S3*** albo ***inv_S0-inv_S3*** odpowiednio dla trybu szyfrowania oraz deszyfrowania. Każda tabela ma rozmiar 256 bajtów. Numer tabeli zamiany jest wyznaczany jako wskaźnik wiersza modulo 4 ciąg bajtów kodu wymiana modulo 4:

$$(2) \qquad s_{i,j} = S_{i \bmod 4}(s_{i,j}) \text{ lub } s_{i,j} = inv\_S_{i \bmod 4}(s_{i,j}).$$

Operacje ***ShiftRows*** i ***InvShiftRows*** dokonują przesunięcia cyklicznego bajtów wierszy w prawo lub w lewo. Liczba pozycji o którą wiersz jest przesuwany, zależy od numeru wiersza *i* oraz długości bloku *l* i wyliczana jest za pomocą wzoru:

$$(3) \qquad \delta_i = \left\lfloor \frac{i \cdot l}{512} \right\rfloor .$$

Istota operacji ***MixColumns*** i ***InvMixColumns*** polega na przekształcaniu kolumn macierzy stanu w sposób wykonania operacji mnożenia i dodawania w skończonym ciele Galois $GF(2^8)$ w oparciu o wielomian nierozkładalny $\psi = x^8 + x^4 + x^3 + x^2 + 1$. Każdy element macierzy wynikowej stanu $W = (w_{i,j})$ jest wyliczany w ciele $GF(2^8)$ jako iloczyn skalarny wiersza macierzy ***v (inv_v)*** oraz kolumny macierzy stanu ***State*** w następujący sposób:

$$(4)\ w_{i,j} = (v >>> i) \otimes S_j,\ w_{i,j} = (inv\_v <<< i) \otimes S_j,$$

gdzie *v* = (0x01, 0x01, 0x05, 0x01, 0x08, 0x06, 0x07, 0x04), *inv_v* = (0xAD, 0x95, 0x76, 0xA8, 0x2F, 0x49, 0xD7, 0xCA); *Sj* - *j*-tą kolumnę macierzy stanu ***State***; *v* >>> *i* oraz *v* <<< i - operacje przesunięcia cyklicznego wektora bajtowego *v* w prawo lub w lewo o *i* pozycji, odpowiednio.

Dla uzyskania podkluczy rund z początkowego klucza głównego służy procedura rozwinięcia kluczy, podczas której wykonane są opisane powyżej przetwarzania.

Cechy architektury mikrokontrolerów do implementacji kryptoalgorytmu „Kalyna"

W celu dalszych badań implementacji kryptoalgorytmu „Kalyna" do systemów wbudowanych, wybrano po jednej najbardziej popularnej na rynku architekturze wśród każdej z trzech klas 8/16/32-bitowych mikrokontrolerów (MCU).

Mikrokontrolery AVR (8-bitowe). Jako platformę 8-bitową, rozważmy rodzinę mikrokontrolerów AVR firmy Atmel. Wybór ten jest uzasadniony zestawem instrukcji procesora tej rodziny MCU, ukierunkowanych na skuteczne wykonanie programów napisanych w językach wysokiego poziomu [14,15].

Wśród cech mikrokontrolerów z rodziny AVR ważnych ze względu na implementację kryptoalgorytmów do systemów wbudowanych należy wymienić harvardską organizację 8-bitowej pamięci danych typu SRAM oraz 16-bitową pamięć programu typu Flash. Wszystkie mikrokontrolery rodziny AVR są oparte na typowej architekturze RISC, którego schemat został przedstawiony na rysunku 3a. Register File zawiera 32 ośmiobitowe rejestry ogólnego przeznaczenia, które są bezpośrednio związane z jednostką arytmetyczno-logiczną ALU. System rozkazów jest wystarczająco rozbudowany i zawiera ponad 130 instrukcji, które są wykonywane potokowo – podczas gdy jedna instrukcja jest wykonywana, następna jest już pobierana z pamięci programu. Dzięki takiemu rozwiązaniu większość instrukcji jest wykonywanych w jednym cyklu zegara [15].

Mikrokontrolery AVR wykorzystują bezpośrednie i pośrednie adresowanie. Dostępność trybów predekrementu i postinkrementu przy wykorzystaniu adresowania pośredniego pozwala skutecznie obsługiwać zbiory danych podczas realizacji algorytmu kryptograficznego, generując krótki kod programu. Dostęp do danych w pamięci Flash (***S-Box***, tabeli ***Look-Up***, klucze) jest zapewniany w trybie adresowania pośredniego.

Mikrokontroler MSP430 (16-bitowy). Aby przetestować działanie szyfru „Kalyna" na platformach 16-bitowych został wybrany mikrokontroler rodziny MSP430 (firma Texas Instruments), jako jeden z najbardziej popularnych w swoim segmencie. Jedną z głównych zalet rodziny MSP430 jest ultra niskie zużycie energii, co zapewnia im szeroką popularność w systemach wbudowanych, zwłaszcza w bezprzewodowych sieciach sensorowych WSN [16].

Kompaktowy wysokowydajny 16-bitowy rdzeń RISC kontrolera MSP430 zbudowany jest według architektury Princeton ze wspólną przestrzenią adresową dla instrukcji i danych. Rdzeń tego MCU zawiera 16 rejestrów, w tym dwanaście rejestrów (R4-R15) to rejestry ogólnego przeznaczenia (rys. 3b). Rejestry R0-R3 - wykonują funkcje specjalne (*Program Counter*, *Stack Pointer*, *Status Register*, *Constant Generator*). Zestaw instrukcji jest bardzo prosty i jest reprezentowany 27 oryginalnymi oraz 24 instrukcjami emulowanymi, które są zoptymalizowane pod kątem efektywnego wykorzystania języków programowania wysokiego poziomu. Wszystkie instrukcje są 16-bitowe i mogą obsługiwać zarówno 8-bitowe, jak i 16-bitowe operandy. Kontroler podtrzymuje siedem trybów adresowania [16,17].

Do przechowywania kodu programu i danych można zastosować pamięć Flash, dlatego nie ma potrzeby kopiowania danych do pamięci RAM przed ich wykorzystaniem. Dzięki jednocyklowym operacjom na rejestrach oraz architekturze ortogonalnej zapewniana jest wysoka wydajność i gęstość upakowania kodu programu. W kontekście kryptografii ważną cechą procesora MSP430 jest również bezpośrednia (z pominięciem rejestrów) wymiana danych między komórkami pamięci.

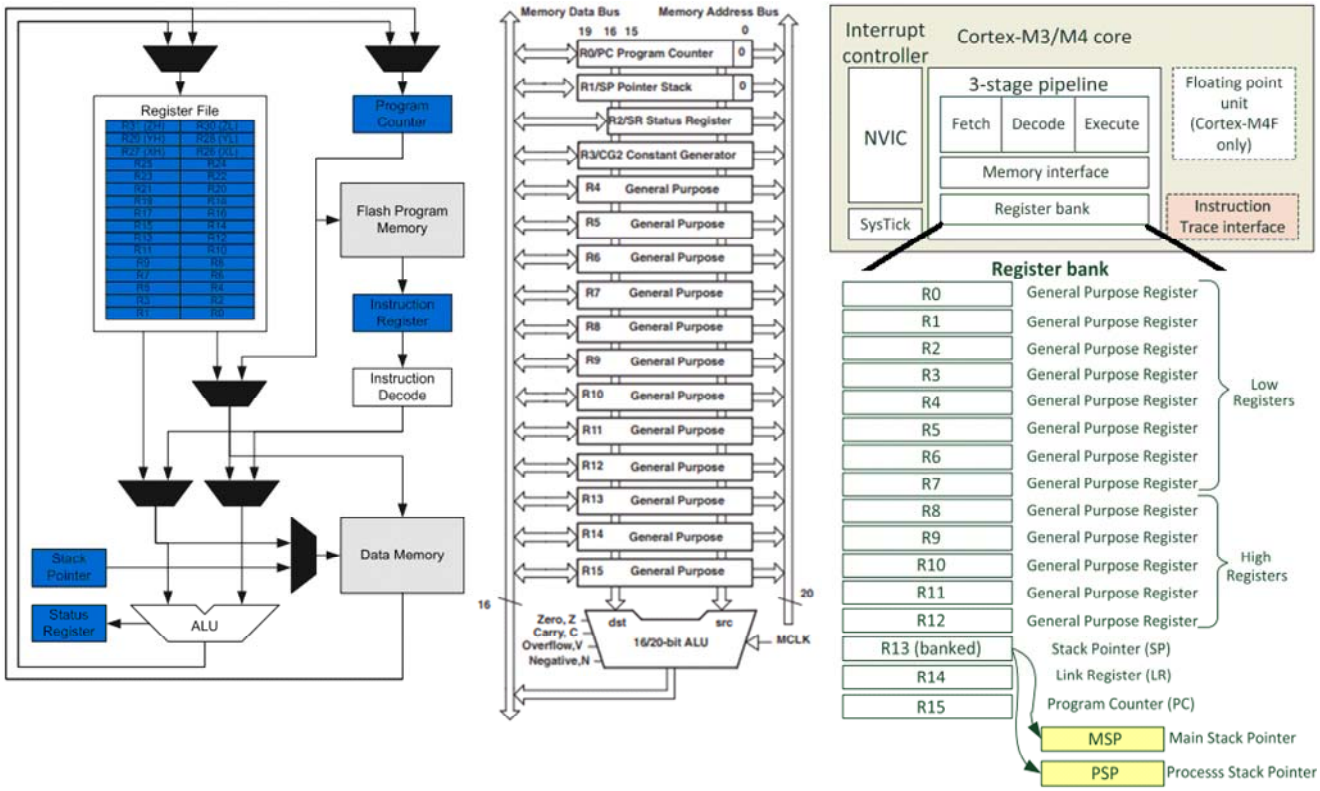
Mikrokontrolery ARM Cortex-M3 (32-bitowe). Wdrożenie platformy 32-bitowej zostało oparte na procesorze ARM Cortex (firma ARM), ponieważ mikrokontrolery z rdzeniem ARM zajmują do 90% rynku 32-bitowych mikrokontrolerów RISC. Ponadto pod względem ceny i zużywanej energii wymienione MCU zbliżają się do modeli 8-bitowych, stwarzając poważną konkurencję dla tradycyjnych zastosowań procesorów 8-bitowych.

Procesory ARM Cortex są dostępne w trzech aplikacyjnych wersjach [18]:

- Cortex A - procesory przeznaczone do stosowania w dużych systemach z zaimplementowanymi systemami operacyjnymi w celu uruchamiania różnych aplikacji,
- Cortex R - procesory przeznaczone do wykorzystania w systemach czasu rzeczywistego przeznaczone do implementowania w systemach (np. system ABS);
- Cortex-M - rdzenie zoptymalizowane cenowo i ukierunkowane na zastosowanie w systemach wbudowanych.

Ze względu na tematykę artykułu rozważana będzie wersja ARM Cortex-M3.

ARM Cortex-M3 jest 32-bitowym procesorem opartym na architekturze harvardzkiej z 3-stopniowym przetwarzaniem potokowym (pipeline), realizującym listę poleceń Thumb i Thumb-2 (rys. 3c).



a) b) c)

Rys. 3. Architektura procesora centralnego mikrokontrolerów AVR (a), MSP430 (b) i ARM Cortex-M3 (c)

Rdzeń Cortex-M3 zawiera 16 rejestrów: R0-R15, w tym rejestry R0-R12 są rejestrami ogólnego przeznaczenia. Rejestr R13 jest wskaźnikiem stosu podzielonym na dwa oddzielne rejestry bankowane (MSP i PSP), co oznacza, że w danym momencie jest widoczny tylko jeden z nich. Główny wskaźnik stosu (MSP – *Main Stack Pointer*) jest używany przez przerwania i jądro systemu operacyjnego, pracującego w trybie uprzywilejowanym, zaś procesowy wskaźnik stosu (PSP - *Process Stack Pointer*) jest używany przez program użytkownika, uruchomionym pod nadzorem systemu operacyjnego.

Rejestr R14 (*Link Register*) zawiera w sobie adres powrotu podczas wywoływania procedury. Rejestr R15 (*Program Counter*) zawiera adres aktualnie wykonywanej instrukcji.

Jednostka arytmetyczno-logiczna zawiera 32-bitowy układ przesunięcia, umożliwiający przesunięcie jednego operandu w czasie wykonania operacji [18]. Rdzeń obsługuje dwa poziomy dostępu do kodu źródłowego (uprzywilejowanego i użytkownika), które zapewniają bezpieczne korzystanie z krytycznych obszarów pamięci, jak również wdrożenie modelu mechanizmu ochrony. Podział przestrzeni adresowej jest ustalony na sztywno.

Metodologia przeprowadzania badań

Ponieważ szyfr „Kalyna” posiada wiele wariantów kombinacji długości bloku i klucza, więc w celu ułatwienia porównania z innymi kryptoalgorytmami badania zostały przeprowadzone dla rozmiaru bloku/klucza 128/128 oraz 128/256 bitów. Dlatego w analizie porównawczej uczestniczyły wybrane kryptoalgorytmy o parametrach podanych w tabeli 3.

Dla każdego rodzaju MCU: AVR (8 bitów), MSP430 (16-bit), ARM Cortex-M3 (32 bit), algorytmy te zostały zaimplementowane w języku C z użyciem najnowszych wersji IDE odpowiednio: IAR Embedded Workbench dla AVR (v6.70), IAR embedded Workbench dla MSP430 (v6.40) i IAR embedded Workbench dla ARM (v7.60). W środowiskach tych przeprowadzono oszacowanie liczby cykli i rozmiar kodu.

Tabela 3. Parametry badanych kryptoalgorytmów

Algorytm	Długość bloku/klucza, bity	Liczba rund
„Kalyna”	128/128	10
	128/256	14
AES	128/128	10
	128/256	14
GOST	64/256	32
„Kuznieczik”	128/256	10

Mierzonymi podczas badań parametrami były: szybkość **szyfrowania / deszyfrowania**, wyrażona w cyklach / bajt i uśredniona dla 100 bloków, **rozmiar pamięci stałej** (Flash), na który składa się rozmiar programu i tabeli umieszczonych w pamięci Flash i **rozmiar pamięci operacyjnej** (SRAM), reprezentowany tabelami w SRAM i potrzebnym rozmiarem stosu.

Ponieważ w urządzeniach różnego przeznaczenia podstawowymi wymaganiami dotyczącymi kryptoalgorytmów może być oszczędne wykorzystanie pamięci oraz duża szybkość działania, zatem sens ma zbadanie algorytmów właśnie pod tym kątem. W celu uzyskania maksymalnej wydajności programu należy zoptymalizować te części algorytmu, które są najbardzej złożone obliczeniowo, używając tych poleceń oraz sposobów adresowania, które wymagają minimalnej liczby cykli mikrokontrolera. Operacje dodawania arytmetycznego i modulo dwa są zasadniczo atomowe oraz są realizowane za pomocą odpowiedniego polecenia procesora, dlatego wydajność algorytmów w całości będzie określana przez prędkość realizacji przetwarzania liniowego.

Dla algorytmów „Kalyna”, AES i „Kuznieczik” zaproponowano kilka sposobów zarządzania skutecznością szyfrowania, które przede wszystkim różnią się podejściem do realizacji najbardziej złożonej obliczeniowo operacji mnożenia w ciele Galois podczas przekształcania liniowego ***MixColumns***. Na tej zasadzie stworzono następujące profile (tryby):

- ***Profile SOFT***. Realizacja programowa operacji mnożenia w ciałach Galois - wersja ta ma na celu osiągnięcie minimalnego rozmiaru kodu kosztem niskiej wydajności. W trybie SOFT mnożenie w ciele Galois jest wykonywane w sposób konwencjonalny za pomocą dość powolnej funkcji ***Kalyna_Software_Mult*** (wymagane jest 65 taktów na wywołanie samej funkcji ***Kalyna_Software_Mult*** oraz 3400 taktów dla realizacji przekształcania ***MixColumns***).

- ***Profile FAST***. Jest to również realizacja programowa mnożenia w ciałach Galois, ale wykorzystywane są metody heurystyczne w celu przyspieszenia realizacji mnożenia i skutecznego wykonania przekształcania ***MixColumns***. Metody te uwzględniają cechy każdej architektury procesora (jego pojemność bitową, liczbę rejestrów ogólnego przeznaczenia, sposoby adresowania, szybkość operacji logicznych), a także regularności (szablony) występujące w macierzy ***MixColumns***, co pozwala zmniejszyć liczbę operacji. Tryb FAST jest zrównoważony pomiędzy szybkością i rozmiarem kodu.

- ***Profile LOCK-UP***. Tutaj wykorzystywane jest mnożenie tablicowe - wersja zapewnia osiągnięcie wysokiej wydajności z umiarkowanym wzrostem objętości kodu (dzięki z góry wyliczonym tabelom).

- ***Profile MDS***. Są wykorzystywane wstępnie wyliczane tabeli dla realizacji mnożenia oraz zamiany nieliniowej. Ta wersja jest skierowana na osiągnięcie maksymalnej szybkości kosztem znacznego wzrostu rozmiaru kodu. Istota tej techniki polega na łączeniu operacji ***SubBytes***, ***ShiftRows*** i ***MixColumns*** w jedną za pomocą wstępnie obliczanej tabeli. W efekcie dla podanego bajta wejściowego od razu można odczytać gotowy rezultat wyjściowy.

Dla operacji deszyfrowania używany jest algorytm równoważny obliczeniowo szyfrowaniu. Aby zapewnić symetrię procedury szyfrowania i deszyfrowania niezbędna jest wstępna obróbka kluczy rund podczas deszyfrowania.

W celu zwiększenia wydajności dla wszystkich trybów i architektur mikrokontrolerów ewentualne wartości pośrednie i stanu szyfrowania zostały ulokowane do rejestrów ogólnego przeznaczenia. Również były stosowane takie techniki jak rozwinięcia cykli, używanie makra zamiast funkcji, uwzględnienie symetrii wstępnie wyliczonych tabeli i inne.

Podczas pomiaru wydajności szyfrowania stosowane były również wygenerowane wstępnie podklucze rundowe, które były przechowywane w pamięci RAM. Jest to dość realistyczne podejście do systemów wbudowanych, które wykorzystują jeden klucz tajny w ciągu całego cyklu życia lub klucze sesji z dość trwałymi seansami wymiany danych. Jednakże do oszacowania pojemności pamięci uwzględniono funkcje wytwarzania (rozwinięcia) kluczy rund.

Algorytm GOST nie zawiera mnożenia w ciałach Galois, dlatego przydzielono mu: tryb SOFT wykorzystujący 4-bitowe węzły zamiany, tryb FAST wykorzystujący 8-bitowe węzły zamiany, tryb MDS wykorzystujący 8-bitowe węzły zamiany oraz tabelaryczne przedstawienie operacji zamiany nieliniowej wymiany i przesunięcia cyklicznego przez 11 bitów.

Dla mikrokontrolerów AVR liczba cykli do odczytu danych z pamięci Flash i pamięci SRAM różni się (odpowiednio 3 vs 2). Tak więc w zależności od miejsca pamięci w której są przechowywane tabele, szybkość przetwarzań kryptograficznych będzie różna. Dlatego dla trybów ukierunkowanych na wysokie szybkości takie jak FAST, LOCK-UP i MDS, tabele należy rozmieścić w SRAM. Dla mikrokontrolerów MSP430 i ARM Cortex-M3 nie występuje podobna zależność i żadne z wyników nie są identyczne. Biorąc pod uwagę, że pamięć RAM jest zazwyczaj bardziej ograniczonym i cennieszym zasobem niż ROM, więc wszystkie tabele dla rozważanych architekturach są zapisywane w pamięci Flash.

Wyniki badań działań kryptoalgorytmów na różnych platformach mikrokontrolerów

W tabelach 4, 5 i 6 zostały podane wyniki badań szybkości szyfrowania/deszyfrowania oraz zapotrzebowania pamięci dla różnych kryptoalgorytmów na 8-, 16- i 32-bitowych mikrokontrolerach.

Tabela 4. Parametry realizacji programowej badanych kryptoalgoorytmów na 8-bitowym mikrokontrolerze AVR

Parametr		Tryb	Kryptoalgorytmy					
			GOST	AES		Kuznie-czik	Kalyna	
				128/ 128	128/ 256		128/ 128	128/ 256
Szybkość, cyk/Bajt	Szyfr. / Deszyfr.	Soft	585/ 585	569/ 1034	797/ 1469	8983/ 9103	2335/ 7983	3266/ 11165
		Fast	262/ 283	221/ 340	308/ 479	903/ 901	439/ 843	608/ 1173
		Lock-up	-	206/ 338	286/ 477	689/ 712	449/ 624	621/ 865
		MDS	359/ 359	240/ 238	334/ 331	1022/ 1018	390/ 411	539/ 564
Pamięć, KBajt	Flash / SRAM	Soft	0,5/ 0,05	1,5/ 0,21	1,6/ 0,28	1,2/ 0,26	3,5/ 0,28	3,5/ 0,36
		Fast	2,3/ 1,05	3,8/ 0,95	3,9/ 1,01	2,8/ 0,73	7,3/ 2,3	7,7/ 2,37
		Lock-up	-	4,7/ 2,19	4,8/ 2,25	8,7/ 2,48	17,0/ 5,04	17,1/ 5,11
		MDS	4,7/ 4,04	13,0/ 8,71	12,9/ 8,77	130,0/ 0,48	23,6/ 15,26	23,7/ 15,33

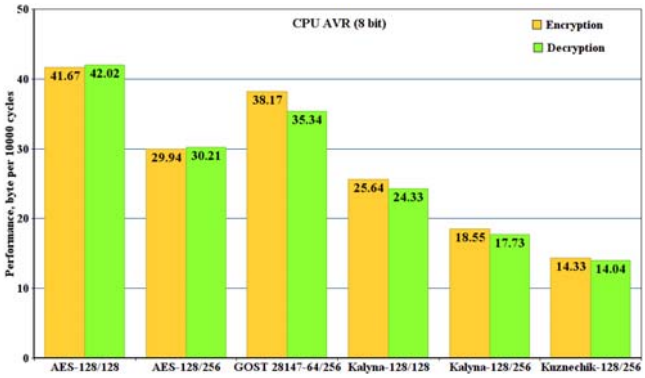
Tabela 5. Parametry realizacji programowej badanych kryptoalgoorytmów na 16-bitowym mikrokontrolerze MSP430

Parametr		Tryb	Kryptoalgorytmy					
			GOST	AES		Kuznie-czik	Kalyna	
				128/ 128	128/ 256		128/ 128	128/ 256
Szybkość, cyk/Bajt	Szyfr. / Deszyfr.	Soft	499/ 495	782/ 3724	1202/ 5450	9876/ 9926	2616/ 9796	3360/ 13697
		Fast	279/ 294	284/ 398	401/ 566	1243/ 1269	482/ 1010	673/ 1411
		Lock-up	-	241/ 229	339/ 323	807/ 813	388/ 469	540/ 654
		MDS	229/ 239	140/ 141	195/ 196	555/ 615	266/ 283	372/ 386
Pamięć, KBajt	Flash / SRAM	Soft	0,5/ 0,06	1,6/ 0,21	1,6/ 0,28	1,3/ 0,27	3,5/ 0,29	3,5/ 0,37
		Fast	2,0/ 0,08	3,4/ 0,23	3,4/ 0,29	2,3/ 0,27	5,5/ 0,29	5,6/ 0,37
		Lock-up	-	3,8/ 0,2	3,9/ 0,27	7,8/ 0,24	6,8/ 0,29	6,9/ 0,37
		MDS	5,0/ 0,048	11,5/ 0,22	11,7/ 0,28	129,4/ 0,25	51,5/ 0,28	51,8/ 0,33

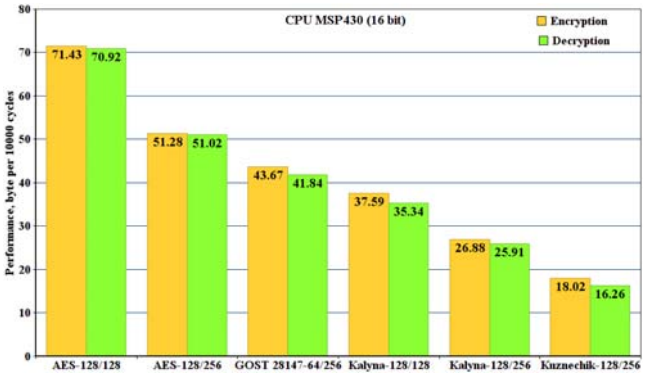
Dla lepszego oszacowania i porównania wyników zawartych w tabelach 4-6, na rysunkach 4, 5 i 6 przedstawiono tryby FAST różnych realizacji o maksymalnej prędkości szyfrowania i deszyfrowania, przy czym nie są one reprezentowane w bajtach na cykl, a jako bajty zaszyfrowane w ciągu 10000 cykli. Oznacza to, że większej prędkości przetwarzania odpowiadają większej wartości na wykresie.

Tabela 6. Parametry realizacji programowej badanych krypto-algoorytmów na 32-bitowym mikrokontrolerze ARM Cortex-M3

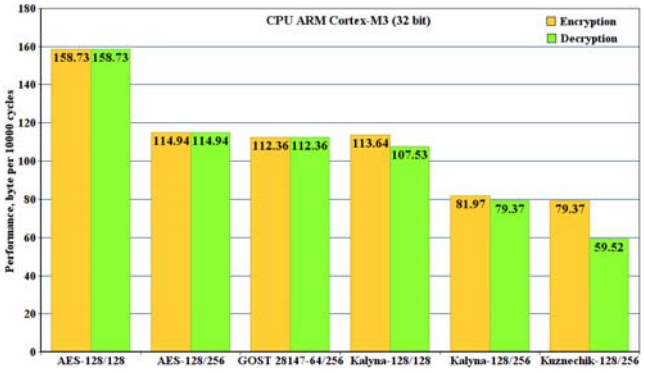
Parametr		Tryb	Kryptoalgorytmy					
			GOST	AES		Kuznie-czik	Kalyna	
				128/ 128	128/ 256		128/ 128	128/ 256
Szybkość, cyk/Bajt	Szyfr. / Deszyfr.	Soft	267/ 267	166/ 321	234/ 459	7440/ 7463	2221/ 7944	3109/ 11111
		Fast	89/ 89	101/ 248	141/ 354	690/ 673	183/ 349	254/ 488
		Lock-up	-	101/ 189	141/ 269	443/ 465	291/ 409	407/ 572
		MDS	92/ 89	63/ 63	87/ 87	126/ 168	88/ 93	122/ 126
Pamięć, KBajt	Flash / SRAM	Soft	0,3/ 0,11	1,3/ 0,23	1,4/ 0,30	1,3/ 0,28	3,1/ 0,30	3,2/ 0,37
		Fast	1,9/ 0,07	1,9/ 0,23	2,0/ 0,29	2,5/ 0,24	4,7/ 0,31	4,8/ 0,39
		Lock-up	-	3,2/ 0,22	3,3/ 0,28	7,2/ 0,24	7,4/ 0,31	7,5/ 0,39
		MDS	6,9/ 0,11	10,8/ 0,22	10,9/ 0,28	129,9/ 0,24	53,0/ 0,28	52,9/ 0,36



Rys. 4. Analiza porównawcza szybkości kryptoalgorytmów na 8-bitowym procesorze AVR



Rys. 5. Analiza porównawcza szybkości kryptoalgorytmów na 16-bitowym procesorze MSP430



Rys. 6. Analiza porównawcza szybkości kryptoalgorytmów na 32-bitowym procesorze ARM Cortex-M3

Z tabel 4-6 wynika, że dla wszystkich architektur i trybów realizacji, kryptoalgorytm „Kalyna" przegrywa z szyframi AES i GOST co do prędkości, jak i zapotrzebowania na pamięć, ale wyprzedza według tych parametrów szyfr "Kuznieczik".

W przypadku procesorów 8-bitowych dla maksymalnej wydajności szyfr „Kalyna" jest wolniejszy od AES o około 1,6 raza, a od GOST - 2 razy, dla procesorów 16-bitowych odpowiednio 2 razy i 1,6 raza, a dla 32-bitowych około 1,4 raza. Dla procesorów 32-bitowych parametry szyfru „Kalyna" są zbliżone do szyfrów AES-128/256 i GOST.

Dla 8-bitowego mikrokontrolera w trybie z maksymalną prędkością MDS szyfr „Kalyna" wymaga 24 i 15 kB odpowiednio pamięci ROM i RAM, co jest dość znaczące dla tej klasy systemów wbudowanych. Dla 16 i 32-bitowych mikrokontrolerów w trybie MDS potrzeby w pamięci ROM wynoszą około 52-54 kB, co znacznie ogranicza wybór modeli mikrokontrolerów. Jeśli maksymalna wydajność nie jest zbyt ważna, wymagania dotyczące pamięci Flash dla szyfru „Kalyna" są całkiem przyzwoite i współmierne do innych kryptoalgorytmów symetrycznych. Należy również zauważyć, że szyfr „Kuznieczik" o wymogach 130 kB pamięci ROM sprawia, że jego implementacja do systemów wbudowanych jest problematyczna, zwłaszcza na 8 i 16-bitowych platformach.

Podsumowanie

Badania wykazały, że dla systemów wbudowanych o normalnym i wysokim poziomie ochrony informacji, szyfry AES i GOST poza prędkością przetwarzania i zapotrzebowaniem pamięci są lepsze niż kryptoalgorytm „Kalyna". Szczególnie konieczne jest zastosowanie szyfru GOST zamiast „Kalyna" w sytuacji surowych wymagań dotyczących objętości kodu. Została potwierdzona teza deweloperów szyfru „Kalyna" o jego ukierunkowaniu na 64-bitowe wysokowydajne mikroprocesory ogólnego przeznaczenia, a nie na systemy z ograniczonymi zasobami.

Algorytmy AES i GOST z powodu stosowania prostego schematu rozwinięcia kluczy rund (zwłaszcza dotyczy to algorytmu GOST) są podatne na stosunkowo łatwe odtworzenie master klucza na podstawie fragmentów kluczy rund, uzyskanych poprzez atak *side-channel*. W związku z tym, dla wymienionych kryptoalgorytmów problem z ochroną przed atakami *side-channel* jest bardzo poważny i, właśnie, może być rozwiązany poprzez maskowanie lub ukrywanie wykowanego kodu.

W algorytmie „Kalyna" wprowadzono nowe podejście - jednokierunkowy system rozwinięcia kluczy rund, bazujący na przekształcaniu cyklicznym, co zapewnia odporność do ataków *side-channel* oraz znanych metod analizy, zorientowanych na schemat rozwinięcia kluczy rund. Ponadto w algorytmie „Kalyna", zastosowano dodatkowe zabezpieczenie przed wieloma sposobami kryptoanalizy, ukierunkowanych w szczególności na sprzętową lub programową realizację przetwarzania [7]. Innymi słowy, szyfr „Kalyna" już na poziomie samego algorytmu jest bardziej zabezpieczony do ataku typu *side-channel*.

Realizacja specjalnych środków zaradczych w celu wyeliminowania tego rodzaju podatności realizacji fizycznych kryptoalgorytmów AES i GOST doprowadzi do zmniejszenia prędkości, a co za tym idzie do wyrównania z szyfrem „Kalyna" tego wskaźnika.

Rozważane sposoby implementacji szyfru „Kalyna" do 8/16/32-bitowych platform systemów wbudowanych pozwalają na osiągnięcie kompromisu pomiędzy parametrami wydajność/cena/pobór mocy w zależności od konkretnego zastosowania.

***Autorzy**: Prof., dr hab. inż. Volodymyr Khoma,* Politechnika Opolska, Instytut Automatyki, *E-mail: v.khoma@po.opole.pl;*

LITERATURA

[1] Stallings W. Cryptography and Network Security: Principles and Practice *(6th Edition) Prentice Hall Press*, 2013, 752 p.

[2] Sohraby K., I Minoli D., Znati T. „Wireless Sensor Networks: Technology, Protocols, and Applications", *Wiley-Interscience*, 2007, 328 p.

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

[4] 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), 33-43

[5] Schneier B. (1996). Applied cryptography : protocols, algorithms, and source code in C (2. ed.). *John Wiley & Sons, Inc. New York*, 1995 , 758 p.

[6] Courtois N. T. Security evaluation of GOST 28147-89 in view of international standardisation. *Cryptologia 36.1* (2012): 2-13.

[7] Takanori Isobe. A Single-Key Attack on the Full GOST Block Cipher // *Fast Software Encryption 18th International Workshop* (FSE-2011), 290-305, Springer LNCS 6733, 2011

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

[9] Oliynykov R. A New Encryption Standard of Ukraine: The Block Cipher Kalyna // *15th Central European Conference on Cryptology, CECC 2015*, 8-10 July 2015, Alpen-Adria-Universität Klagenfurt - Mode of access: www. URL: https://conference.aau.at/event/14/session/2/contribution/14

[10] State Service of Special Communication and Information Protection of Ukraine. Statement on Public Competition of Cryptographic Algorithms. - Mode of access: www. URL: http://www.dstszi.gov.ua/dstszi/control/ua/publish/printable/article?art id=48387, 2006 (in Ukrainian)

[11] Oliynykov R., Gorbenko I., Kazymyrov O., Ruzhentsev V., Kuznetsov O., Gorbenko Y., Dyrda O., Dolgov V., Pushkaryov A., Mordvinov R., Kaidalov D. A New Encryption Standard of Ukraine: The Kalyna Block Cipher // *Norwegian Information Security Conference* (NISK-2015), 2015, 113 p.

[12] National Standard of the Russian Federation GOST R 34.12–2015 - Mode of access: www. URL: http://tc26.ru/en/standard/gost/GOST_R_34_12_2015_ENG.pdf

[13] Oliynykov R. Related-key cryptanalysis of perspective symmetric block cipher / R. Oliynykov, D. Kaidalov // *Prikladnaya radioelektronika*. - 2014. - T. 13, Nr 3, p. 192-200. - Mode of access: http://nbuv.gov.ua/UJRN/Prre_2014_13_3_5.

[14] Baranowski Rafał, Mikrokontrolery AVR Atmega, BTC, Warszawa 2005.

[15] 8-bit AVR Instruction Set - Mode of access: www. URL: www.atmel.com

[16] User's Guide. MSP430x5xx and MSP430x6xx Family // Texas Instruments, 2014, 1145 p. 15. Yiu J. The Definitive Guide to the ARM Cortex-M3 and ARM Cortex-M4 Processors. Third Edition. – Elsevier, 2014, 1055 p.

[17] MSP430 - ultra-low-power Microcontrollers. - Mode of access: www. URL: http://www.ti.com/lsds/ti/microcontrollers_16-bit_32-bit/msp/overview.page?DCMP=MCU_other&HQS=msp430

[18] Yiu J. The Definitive Guide to the ARM Cortex-M3 and ARM Cortex-M4 Processors. Third Edition. – Elsevier, 2014, 1055 p.

[19] Ambrose J., Ignjatovic A., Parameswaran S. Power Analysis Side Channel Attacks: The Processor Design-level Context. *Omniscriptum Gmbh & Company Kg.,* 2010, 300 p.