УДК 004.056.55; 004.051

В. Яковина, Т. Климаш
Національний університет “Львівська політехніка”,
кафедра програмного забезпечення

ПРОГРАМНІ ЗАСОБИ ДЛЯ ДОСЛІДЖЕННЯ ШВИДКОДІЇ АЛГОРИТМУ СИМЕТРИЧНОГО ШИФРУВАННЯ RC5

Розроблено програмне середовище для дослідження швидкодії алгоритму симетричного шифрування RC5, яке за рахунок використання низькорівневого програмування та відсутності впливу операційної системи, дає змогу підвищити точність та відтворюваність результатів. Розроблені програмні засоби можуть бути використані для експериментального визначення швидкодії будь-яких алгоритмів, які не вимагають захищеного режиму роботи процесора та обсягу оперативної пам'яті понад 32 Мбайтів.

Ключові слова: шифрування, середовище, швидкодія, симетричний.

The software tool for RC5 algorithm software performance investigation has been developed. The tool due to low level programming use as well as absence of operating system influence allows to increase investigation accuracy and reproducibility. The developed tool can be used for experimental evaluation of software performance of any algorithms which do not need CPU protected mode and RAM amount more than 32 Mbytes.

Keywords: software.

Вступ

Алгоритми симетричного шифрування є основним засобом забезпечення конфіденційності інформації завдяки насамперед високій швидкодії порівняно з алгоритмами шифрування з відкритим ключем [1, 2]. Перевагами криптографічних засобів захисту інформації порівняно з іншими є [3, 4]:

- еквівалентність або зведеність загрози до математичної задачі, що дає змогу довести безпеку інформаційної системи за умови, що зазначена задача є складною;
- можливість прогнозування безпеки інформаційної системи;
- можливість порівняння однотипних засобів захисту інформації, що забезпечують захист від одних і тих самих загроз, і вибору найкращого варіанта.

Серед алгоритмів симетричного можна виділити родину алгоритмів RC5 завдяки ряду переваг, зокрема [1, 5, 6]:

- Придатність для апаратної та програмної реалізації. В RC5 використовуються тільки елементарні обчислювальні операції, які зазвичай застосовуються в мікропроцесорах.
- Швидкість виконання. RC5 є простим алгоритмом, що працює з даними розміром в машинне слово. Усі основні операції передбачають також роботу з даними завдовжки в слово.
- Адаптованість до процесорів з різною довжиною слова. Довжина слова в бітах є параметром RC5 – із зміною довжини слова змінюється сам алгоритм.
- Змінна кількість раундів. Кількість раундів є другим параметром RC5. Цей параметр дає змогу вибрати оптимальне співвідношення необхідної швидкості роботи і вимог до ступеня захисту.
- Змінна довжина ключа. Довжина ключа є третім параметром RC5. Як і в попередньому випадку, цей параметр дає змогу знайти прийнятний компроміс між швидкістю роботи та необхідним рівнем безпеки.
- Простота. Структура RC5 дуже проста не тільки для реалізації, але й для оцінки її криптоаналітичної стійкості.

- Низькі вимоги до пам'яті. Низькі вимоги до пам'яті роблять RC5 придатним для використання в смарт-картах та інших подібних пристроях з обмеженим обсягом пам'яті.
- Високий ступінь захисту. RC5 покликаний забезпечити високий ступінь захисту за умови вибору відповідних значень параметрів.
- Залежність циклічних зсувів від даних. В RC5 використовуються циклічні зсуви, величина яких залежить від даних, що повинно підвищувати криптоаналітичну стійкість алгоритму.

У попередніх роботах автори досліджували характеристики криптографічних алгоритмів для використання в розподілених інформаційних системах, зокрема в системі теплового проектування електронних пристроїв [7, 8]. Зокрема було показано, що швидкість шифрування алгоритму RC5-32/12/16 на процесорі AMD Athlon X2 5000+ становить 212 Мбайт/с [7], що майже на порядок перевищує швидкість програмної реалізації алгоритму DES [8]. Однак, незважаючи на максимальну можливу коректність проведених програмних експериментів, при роботі в операційній системі розділення часу, якими, зокрема, є усі версії MS Windows, Linux, FreeBSD тощо, неможливо гарантувати відтворюваність результатів та точність досліджень внаслідок дії механізму витісняючої багатозадачності [9]. У літературі описано декілька WinAPI функцій, які зводять ці недоліки до мінімуму [9], однак не ползбавляють від них. Крім того, сучасне використання засобів захисту інформації передбачає використання алгоритмів шифрування до завантаження операційної системи. Такою, наприклад, є технологія Bitlocker, реалізована в настільних і серверних операційних системах MS Windows, починаючи з Windows Vista [10].

Отже, проблема експериментального визначення швидкості програмної реалізації алгоритмів шифрування на сучасних апаратних платформах без впливу інших процесів системи є актуальною прикладною задачею, розв’язання якої і є метою цієї роботи.

Опис алгоритму та методики дослідження

RC5 – алгоритм симетричного шифрування – розробив Рон Райвест у середині 90-х років [5]. Цей алгоритм вбудований в багатьох основних продуктах компанії RSA Data Security Inc., зокрема BSAFE, JSAFE та S/MAIL. RC5 фактично являє собою родину алгоритмів шифрування, що визначається трьома такими параметрами: розмір слова, кількість раундів шифрування, довжина таємного ключа.

Алгоритм RC5 шифрує блоки відкритого тексту завдовжки 32, 64 чи 128 бітів у блоки шифрованого тексту тієї самої довжини. Довжина ключа може змінюватись від 0 до 2040 бітів. Конкретна версія RC5 позначається RC5-*w*/*r*/*b*. Наприклад, RC5-32/12/16 використовує 32-бітові слова (64-бітові блоки відкритого і шифрованого тексту), 12 раундів шифрування і ключ завдовжки 16 байтів (128 бітів). Райвест пропонує використовувати RC5-32/12/16 як "стандартну" версію RC5.

В алгоритмі RC5 виконуються три елементарні операції (а також обернені до них):

- Додавання. Додавання слів виконується за модулем 2^w. Оберненою операцією є віднімання за модулем 2^w.
- Побітове виключне АБО.
- Циклічний зсув ліворуч. В алгоритмі використовується циклічний зсув слова *x* ліворуч на *y* бітів, оберненою операцією є циклічний зсув слова *x* праворуч на *y* бітів.

Двома найважливішими особливостями RC5 є простота алгоритму та використання керованих даними циклічних зсувів. Циклічні зсуви – єдина нелінійна складова цього алгоритму. Райвест стверджує [5], що у зв'язку з тим, що величина зсуву визначається даними, що обробляються алгоритмом, лінійний та диференційний криптоаналіз алгоритму буде значно ускладнений.

З метою забезпечення можливості ефективного використання RC5 в неоднорідному середовищі, специфікація RFC 2040 [6] визначає чотири різні режими роботи цього алгоритму.

- Блоковий шифр RC5. Алгоритм прямого шифрування, при якому береться блок даних заданого розміру (2^w бітів) і з нього за допомогою залежного від ключа перетворення генерується блок шифрованого тексту такого самого розміру. Цей режим часто називають режимом ECB (режим електронної шифрувальної книги).
- RC5-CBC. Режим зв'язаних шифрованих блоків для RC5. У режимі CBC обробляються повідомлення, довжина яких кратна розміру блоку RC5 (тобто кратна 2*w* бітам). Режим CBC

забезпечує вищий ступінь захисту, ніж ECB, оскільки генерує різні блоки шифрованого тексту для однакових повторних блоків відкритого тексту.

- RC5-CBC-Pad. Модифікація режиму CBC, призначена для роботи з відкритим текстом будь-якої довжини. Довжина шифрованого тексту в цьому режимі перевищує довжину відкритого тексту не більш ніж на довжину одного блоку RC5.
- RC5-CTS. Режим запозичення шифрованого тексту (ciphertext stealing) теж є модифікацією CBC. У цьому режимі допускається обробка відкритого тексту будь-якої довжини і генерується шифрований текст тієї самої довжини.

Тобто алгоритм шифрування RC5 має переваги, які дають змогу його використовувати на процесорах з різною архітектурою та різною довжиною машинного слова, ефективно використовувати апаратні та програмні реалізації алгоритму, роблять його програмну реалізацію ефективною з погляду використання обчислювальних потужностей процесора і пам'яті.

Для вимірювання швидкодії програмної реалізації було створено програмне середовище, яке дає змогу викликати програму, в якій реалізовано досліджуваний алгоритм та функцію вимірювання швидкості його роботи. З метою збільшення швидкості програмної реалізації алгоритму весь програмний код (і середовища, і основної програми) було написано мовою assembler [11]. Для уникнення впливу швидкодії жорсткого диска на швидкість шифрування алгоритму дослідження проводились шляхом циклічного виклику функції шифрування, вихідні дані якої є одночасно вхідними даними для подальшого проходження циклу шифрування. Необхідний обсяг даних для шифрування визначався кількістю таких циклів. Початковими вхідними значеннями для функції шифрування є два випадкові 32-бітові слова.

Оскільки задачею цієї роботи є вимірювання швидкодії алгоритму без неконтрольованого впливу операційної системи, програмне середовище для дослідження швидкості програмної реалізації алгоритму повинно завантажуватись замість операційної системи, а отже, необхідно модифікувати завантажувальний сектор (boot sector) носія (флеш-диска, компакт-диска тощо). Завантажувальний сектор – сектор жорстких дисків, дискет або аналогічних пристроїв зберігання даних, що містить код для завантаження програми (зазвичай, але не обов'язково, операційними системами), що зберігаються в інших частинах диска.

На IBM PC сумісних машинах BIOS вибирає завантажувальний пристрій, а потім копіює перший сектор з пристрою (який може бути MBR, VBR або будь-який виконуваний код), в адресу пам'яті 0x7c00. Залежно від типу операційної системи завантажувальний сектор (bootsector) має різну структуру. На рис. 1 наведено структуру завантажувального сектора DOS для файлової системи FATxx, а саме FAT12 – файлової системи, що використовується на дискетах, оскільки при завантаженні з CD-ROM практично відбувається таке саме завантаження з дискети, образ якої записаний на компакт-диску.

Структура завантажувального сектору DOS		
Зміщення	**Довжина (байт)**	**Вміст**
0000h	3	JMP xx xx перехід на код завантаження
0003h	8	OEM-ім’я компанії і версія системи
000Bh	2	SectSiz байт на сектор
000Dh	1	ClustSiz секторів на одиницю розподілу (кластер)
000Eh	2	ResSecs резервних секторів (секторів перед першою FAT)
0010h	1	FatCnt кількість таблиць FAT
0011h	2	RootSiz максимальна кількість 32-байтових елементів кореневого каталогу
0013h	2	TotSecs загальна кількість секторів на носію (розділ DOS)
0015h	1	Media дескриптор носія (те саме, що 1-й байт FAT)
0016h	2	FatSize кількість секторів в одній FAT
0018h	2	TrkSecs секторів на доріжку (циліндр)
001Ah	2	HeadCnt кількість головок читання/запису (поверхонь)
001Bh	2	HidnSec прихованих секторів
001Eh	480	Розмір форматованої порції кореневого сектора, початок коду і даних завантаження
01FEh	2	Сигнатура завантажувального сектора (55AAh)

Рис. 1. Структура завантажувального сектора файлової системи FAT12

У лістингу 1 наведено вихідний код середовища для запуску основної програми, яке є модифікованим завантажувальним сектором.

Лістинг 1. Середовище для запуску основної програми

```
; bootsect.asm
org 7C00h
use16                                   ; 16 – бітний код
jmp Beginning                   ; стрибок на початок коду
nop             ; оскільки попередня команда займає 2 байти, команда nop заповнює третій
db '@TarasKl'                           ; 8 байт завантажувального сектора
SectSize dw 00200h
ClustSize db 001h
ResSecs dw 0000Ch
FatCnt db 002h
RootSiz dw 000E0h
TotSecs dw 00B40h
Media db 0F0h
FatSize dw 00009h
TrkSecs dw 00012h
HeadCnt dw 00002h
HidnSec dw 00000h
Beginning:                      ; початок програми
cli                                     ; підготовлення регістрів
mov ax, cs
mov ds, ax
mov es, ax
mov ss, ax
mov sp, 7c00h
sti
; Команди cli і sti (заборонити і дозволити переривання) ; необов'язкові, але за їх відсутності якщо станеться переривання ; таймера, то оскільки не заповнена таблиця переривань (або заповнена не ; повністю), перехід, скоріш за все, здійсниться туди, де знаходяться ; дані або взагалі нічого немає.
xor ax,ax
mov es,ax
mov bx,7e00h
mov ah,02h
mov al,8
xor dx,dx
mov ch,00000000b
mov cl,2
int 13h
; Сигнатура завантажувального сектора 55AAh
times (512-2-($-7C00h)) db 0
db 055H,0AAH
; Підключення файла з основною програмою.
include 'main.asm'
```

Для визначення швидкості шифрування використано системний таймер. Частота кварцового генератора ділиться на величину лічильника таймера – 655536 – отримуємо мінімальний час, який можна зафіксувати – 18,2 тіки за секунду або – 55 мілісекунд. Отже, для отримання коректних і відтворюваних результатів визначення швидкості шифрування алгоритму потрібно задавати як вхідні дані таку кількість інформації, час шифрування якої принаймні вдвічі перевищує 55 мс.

У лістингу 2 наведено вихідний код функції, яка задає обсяг даних для шифрування, викликає функцію шифрування заданого алгоритму (в цьому випадку RC5 32/12/32 EBC) та визначає час шифрування.

Лістинг 2. Функція визначення часу шифрування

```
CryptoProcess:
; в edx – кількість шифрованої інформації
pushad
push edx
```

```
mov eax, 0
mov ecx, 0
mov edx, 0
int 1Ah
mov ax, cx
shl eax, 16
or eax, edx
mov [time_start], eax
;*** Початок шифрування ****
pop edx
mov ax, 20h
mov es, ax
mov eax, 0df1171e2h
mov ebx, 0b154d0a3h
@crypt10:
 call RC5_ENCRYPT
add edx, -1
jnz @crypt10
;*** Кінець шифрування ****
mov eax, 0
mov ecx, 0
mov edx, 0
int 1Ah
mov ax, cx
shl eax, 16
or eax, edx
mov [time_end], eax
mov eax, [time_start]
push [pos]
pusha
call PrintDec
popa
pop [pos]
add [pos], 12h
mov [color], 0Ah
mov eax, [time_end]
push [pos]
pusha
call PrintDec
popa
pop [pos]
add [pos], 12h
mov [color], 0Ah
mov eax, [time_end]
sub eax, [time_start]
mov ecx, 55
mul ecx
push [pos]
pusha
call PrintDec
popa
pop [pos]
popad
ret
```

Уся програмна реалізація написана з використанням тільки функцій BIOS, оскільки жодної операційної системи не буде завантажено. Оскільки в цій реалізації використано реальний режим роботи процесора, а завантажувальний сектор використовує 16-бітний код, то максимальний доступний програмі об'єм оперативної пам’яті становить 32 Мбайти.

Для оптимізації швидкодії програмної реалізації алгоритму RC5, який передбачає циклічні зсуви 32-бітових чисел на величину розміром 32 біти, було використано таку властивість циклічних зсувів: $\mathbf{V}^{(i)} = \mathbf{V}^{(i \bmod n)}$, де $\mathbf{V}$ – бітовий вектор, що піддається зсуву, i – величина зсуву, n – розрядність вектора $\mathbf{V}$.

Експерименти проводили, використовуючи вхідні дані розміром від 10 МБ до 10 ГБ, які шифрувались з використанням однакового ключа шифрування, при цьому програмно вимірювався час шифрування кожного файла. Експерименти проводили 5 разів для кожного розміру вхідних даних, а результати усереднювались. Швидкодія програмної реалізації алгоритму визначалась як коефіцієнт нахилу прямої, що описує залежність розміру відкритого тексту від часу шифрування. Експерименти проводились на комп'ютері з процесором Intel Celeron CPU 2533 MHz.

Крім того, для визначення впливу операційних систем, було здійснено реалізацію програмного засобу для визначення швидкості шифрування цього алгоритму мовою C++, який був відкомпільований різними компіляторами, а саме TC++ (операційна система MS DOS), Borland C++ (Windows XP), GCC/G++ (Linux Ubuntu 8.04). Та проведено аналогічні дослідження в середовищі цих операційних систем. У середовищі Windows XP було здійснено реалізацію проекту як GUI Application (яка запускалась на виконання з різними базовими пріоритетами) та CLI Application.

Аналіз отриманих результатів

Значення часу шифрування алгоритму RC5-32/12/32 залежно від розміру вхідних даних для кожного з 5 експериментів в реалізації середовища на основі завантажувального сектора наведено на рис. 2. Розкид значень між експериментами не перевищує 4,8%, що свідчить про хорошу відтворюваність результатів.

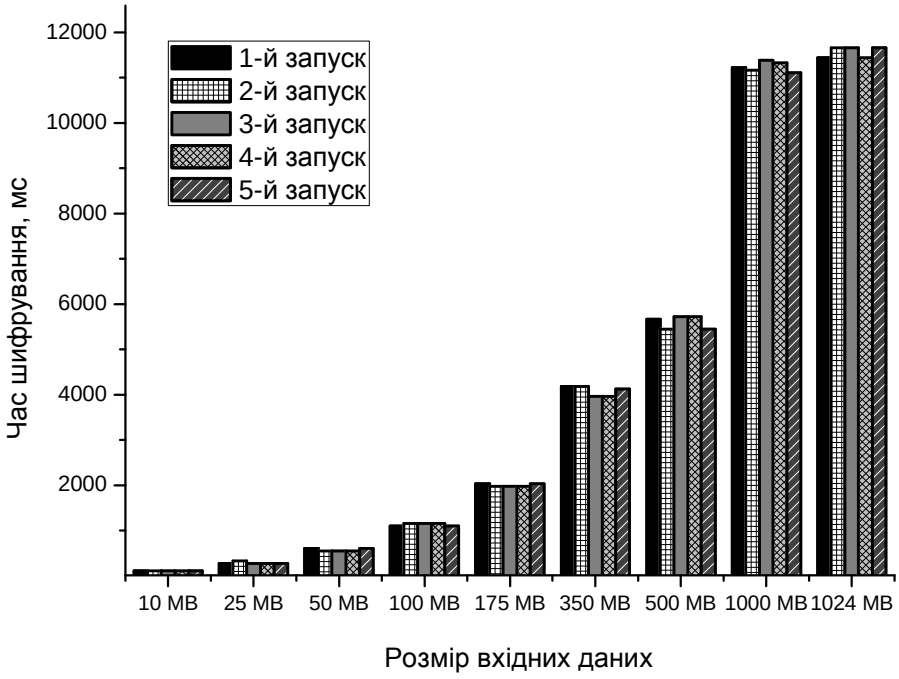

Рис. 2. Значення часу шифрування алгоритму RC5-32/12/16 для 5 запусків програми на основі завантажувального сектора

Натомість розкид значень часу шифрування для трьох експериментів, проведених з використанням додатка з графічним інтерфейсом користувача в операційній системі Windows XP, виконаного з нормальним базовим пріоритетом (рис. 3), показує, що відтворюваність результатів при повторних запусках програми є вкрай низькою. Так, різниця між часом шифрування одного і того самого об'єму вхідних даних для різних запусків програми може сягати 470 % (звертаємо увагу на логарифмічний масштаб осі ординат на рис. 3). Це ще раз підтверджує факт, що операційні системи розділення часу мало придатні для відтворюваних експериментів з визначення швидкодії програмних засобів без ретельного врахування їх особливостей при програмуванні.

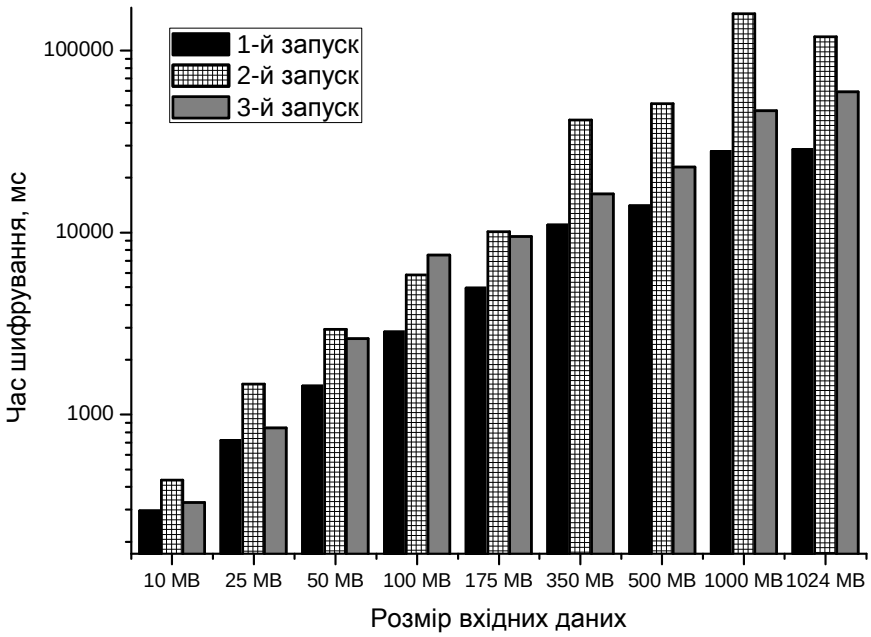

Рис. 3. Значення часу шифрування алгоритму RC5-32/12/16 для 3 запусків програми під управлінням Windows XP

Усереднене для 5 експериментів значення часу шифрування алгоритму RC5-32/12/32 залежно від розміру вхідних даних для виконання програми у різних операційних системах наведено на рис. 4.

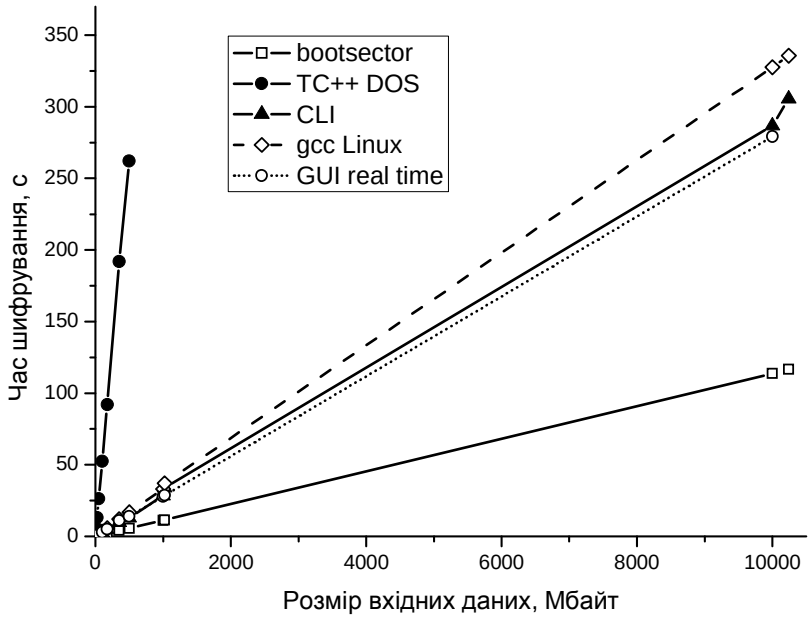

Рис. 4. Усереднене значення часу шифрування програмної реалізації алгоритму RC5-32/12/16 в різних операційних системах

Як видно з рис. 3, практично усі залежності є лінійними, при цьому найбільша швидкодія програмної реалізації спостерігається у випадку виконання програми з середовища на основі завантажувального сектора. Найменша – в середовищі MS DOS, що може пояснюватись 16-розрядною операційною системою, тоді як алгоритм RC5-32/12/32 призначений для 32-розрядної архітектури. Швидкодія алгоритму в ОС Linux є дещо повільнішою, ніж у Windows XP, однак усі вони значно поступаються реалізації без операційної системи (на основі завантажувального сектора), що підтверджує доцільність використання розробленого програмного середовища для визначення швидкості програмної реалізації

криптоалгоритмів та високу точність і відтворюваність його результатів. Крім того, як видно з рис. 4, при великих значеннях об'єму вхідних даних для реалізації у вигляді програми командного рядка (CLI Application) залежність часу шифрування від об'єму вхідних даних перестає бути лінійною, що свідчить про вплив сторонніх факторів у роботу алгоритму шифрування.

Методом найменших квадратів було апроксимовано залежність часу шифрування від розміру файла лінійною функцією і показано, що експериментальні дані з високою точністю апроксимуються лінійною залежністю (квадрат коефіцієнта кореляції дорівнює 1 для реалізації на основі завантажувального сектора і 0,99998 для реалізації у вигляді додатка з графічним інтерфейсом – GUI Application – виконаного з пріоритетом реального часу).

Швидкодію програмної реалізації алгоритму визначали як коефіцієнт нахилу прямої, що описує залежність розміру відкритого тексту від часу шифрування. Розрахована так швидкість шифрування реалізації алгоритму RC5-32/12/32 на основі завантажувального сектора становить 87,92±0,06 Мбайт/с, а для GUI Application з пріоритетом реального часу – 35,81±0,05 Мбайт/с, що підтверджує високу швидкість роботи алгоритму, описану в літературі [5]. Крім того, існуючі дані [5] показують, що алгоритм RC5 32/12/32 на процесорі 50 MHz 486 SLC виконується зі швидкістю 1,2 Мбайт/с. Якщо привести частоту процесора до 2533 МГц, то швидкість шифрування повинна становити 60,8 Мбайт/с, що майже на 50% гірше за отримані нами результати.

Останнім етапом досліджень було вивчення впливу базового пріоритету процесу в операційній системі Windows XP на час шифрування алгоритму. Для цього одну і ту саму реалізацію алгоритму у вигляді додатка з графічним інтерфейсом користувача виконувалась з базовими пріоритетами real time, normal та idle. Усереднене значення часу шифрування як функції об'єму вхідних даних для процесів з різними пріоритетами наведено на рис. 5. Як видно з рис. 5, базовий пріоритет процесу в операційній системі Windows XP значно впливає на час виконання програми: так, для процесу з пріоритетом реального часу швидкість шифрування становить 35,81±0,05 Мбайт/с, тоді як для процесу з низьким пріоритетом – 31,52±0,07 Мбайт/с. Слід зазначити, що в цьому випадку залежність часу шифрування від об'єму вхідних даних залишається лінійною з квадратом коефіцієнта кореляції 0,99995. Тобто зміна пріоритету процесу в операційній системі з витісняючою багатозадачністю (зокрема у Windows XP) призводить до зменшення швидкості роботи програмної реалізації алгоритму шифрування на 13,6 %.

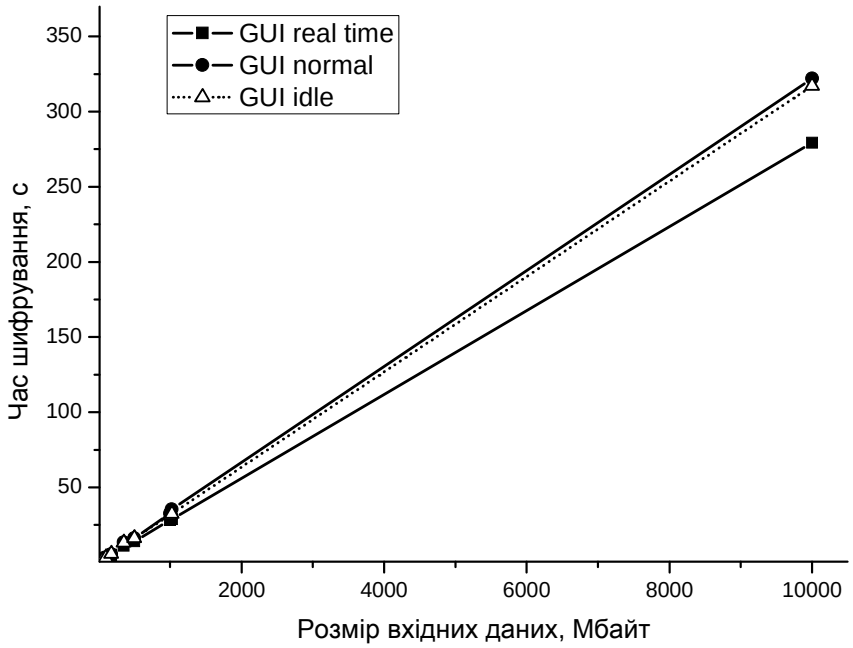

Рис. 5. Усереднене значення часу шифрування програмної реалізації алгоритму RC5-32/12/16 в операційній системі Windows XP для процесів з різними базовими пріоритетами

Висновки

Розроблено програмне середовище для дослідження швидкодії алгоритму симетричного шифрування RC5, яке за рахунок використання низькорівневого програмування та відсутності впливу операційної системи дає змогу підвищити точність та відтворюваність результатів.

Експериментально досліджено розроблене середовище на прикладі алгоритму симетричного шифрування RC5-32/12/32. Встановлено, що швидкість шифрування алгоритму RC5-32/12/32 на процесорі Intel Celeron CPU 2533 MHz без впливу операційної системи становить 87,92±0,06 Мбайт/с, тоді як найкраща швидкість для виконання програми під управління операційної системи становила 35,81±0,05 Мбайт/с. Швидкодія програмної реалізації алгоритму є лінійною в усьому дослідженому діапазоні розмірів вхідних даних.

Показано, що на швидкість програмної реалізації впливає як тип операційної системи (MS DOS, Linux, Windows), так і базовий пріоритет процесу в багатозадачних операційних системах. Зокрема в операційній системі Windows XP зміна пріоритету процесу з низького на реального часу підвищує час виконання алгоритму на 13,6 %.

Розроблені програмні засоби можуть бути використані для експериментального визначення швидкодії будь-яких алгоритмів, які не вимагають захищеного режиму роботи процесора та обсягу оперативної пам'яті понад 32 Мбайтів.

1. Столлингс В. Криптография и защита сетей: принципы и практика. – М.: Вильямс, 2001. – 672 с. 2. Яковина В., Федасюк Д., Сенів М., Білас О. Порівняння швидкодії програмної реалізації алгоритмів симетричного (DES) та асиметричного (RSA) шифрування // Вісник Нац. ун-ту "Львівська політехніка" Комп'ютерні науки та інформаційні технології. – № 598 (2007). – С. 181–185. 3. Ростовцев А.Г., Маховенко Е.Б. Теоретическая криптография. – СПб.: НПО "Профессионал", 2004. – 478 с. 4. Б. Шнайер Прикладная криптография: Протоколы, алгоритмы, исходные тексты на языке Си. – М.: ТРИУМФ, 2003. – 816 с. 5. Rivest R.L. The RC5 Encryption Algorithm // Proceedings of the Second International Workshop on Fast Software Encryption, Leuven Belgium, 1994, pp. 86-96. 6. R. Baldwin, R. Rivest The RC5, RC5-CBC, RC5-CBC-Pad, and RC5-CTS Algorithms // RFC 2040, October 1996. 7. Яковина В.С., Одуха О.В., Сенів М.М., Білас О.Є. Дослідження основних характеристик алгоритму симетричного шифрування RC5 для побудови модуля захисту розподіленої системи теплового проектування // Вісник Нац. ун-ту "Львівська політехніка" Комп'ютерні науки та інформаційні технології. – № 616 (2008). – С. 143–150. 8. Яковина В.С., Федасюк Д.В., Салій С.І., Сенів М.М. Дослідження характеристик криптостійкості алгоритму симетричного шифрування DES // Вісник Нац. ун-ту "Львівська політехніка" Комп'ютерні системи проектування. Теорія і практика, 2008. – № 626. – С. 55–62. 9. Рихтер Дж. Windows для профессионалов: создание эффективных Win32 приложений с учетом специфики 64-разрядной версии Windows / Пер. с англ. – 4-е изд. – СПб.: Питер; М.: Издательско-торговый дом "Русская Редакция", 2001. – 752 с. 10. Хайнз Б. Защита данных с помощью шифрования диска BitLocker [Електронний ресурс] // TechNet Magazine, 2007, № 6. – Режим доступу до журн. http://technet.microsoft.com/ru-ru/magazine/2007.06.bitlocker.aspx. 11. Зубков С.В. Assembler для DOS, Windows и UNIX. – М.: ДМК Пресс, 2000. – 608 с.