КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (14), 2021
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2021.14.118130
УДК 004.42

Глинчук Людмила Ярославівна
канд. фіз.-мат. наук, доцент кафедри комп'ютерних наук та кібербезпеки
Волинський національний університет імені Лесі Українки, Луцьк, Україна
ORCID ID 0000-0002-8943-9604
Hlynchuk.Ludmila@vnu.edu.ua

Гришанович Тетяна Олександрівна
канд. фіз.-мат. наук, доцент кафедри комп'ютерних наук та кібербезпеки
Волинський національний університет імені Лесі Українки, Луцьк, Україна
ORCID ID 0000-0002-3595-6964
Hryshanovych.Tatiana@vnu.edu.ua

Ступінь Андрій Петрович
студент 4 курсу факультету інформаційних технологій і математики
Волинський національний університет імені Лесі Українки, м. Луцьк, Україна
Stupin.Andrij2018@vnu.edu.ua

РЕАЛІЗАЦІЯ СТАНДАРТУ СИМЕТРИЧНОГО ШИФРУВАННЯ DES МОВОЮ ПРОГРАМУВАННЯ С ТА ПОРІВНЯННЯ ЧАСУ ЙОГО РОБОТИ З ВІДОМИМИ УТИЛІТАМИ

Анотація. Дане дослідження присвячено огляду, реалізації та аналізу алгоритму симетричного шифрування, а саме – DES (Data Encryption Standard), який виконує як шифрування тексту, так і його дешифрування. Для даного алгоритму наведено не лише словесний опис, а схеми його роботи та приклади програмної реалізації. Проміжні результати та результати шифрування/дешифрування інформації із використанням реалізованого алгоритму перевірені на прикладах, тому можна вважати, що алгоритм реалізовано вірно. Порівняння часу виконання запропонованої реалізації алгоритму DES виконувалось для двох утиліт. Одна із них – OpenSSL написана на мові Assembler та використовує можливості мови програмування С, інша ж реалізована із використанням мови програмування Java. Порівняння проводилось за трьома критеріями: повний час від початку роботи утиліти до її завершення; час, витрачений процесором на виконання утиліти (при цьому не враховується час простою і час, коли процесор виконував інші завдання); час, який затратила операційна система для роботи утиліти, наприклад, читання файлу або його запис на диск. Аналіз показав, що повний час виконання алгоритму не рівний загальній кількості часу, витраченого і процесором, і операційною системо на його виконання. Це зумовлено наступним: загальний час виконання – це реальний час, який витрачено на виконання утиліти, його можна виміряти секундоміром. Тоді як час, який був витрачений процесором на виконання утиліти, вимірюється дещо інакше, а саме: якщо два ядра будуть виконувати одну і ту ж утиліту впродовж 1 секунди, то загальний час її виконання буде дорівнювати 2 секундам, хоча насправді пройшла одна секунда часу. З проведеного порівняння слідує висновок: час, який затрачений на шифрування, менший від часу, затраченого на розшифрування. Час виконання різних утиліт – різний: час утиліти OpenSSL виявився найкращим, адже така реалізація найбільш адаптована до апаратного забезпечення. Утиліта на Java виявилася за часом виконання найгіршою. Запропонована нами реалізація алгоритму DES близька за часом виконання до найшвидшої із розглянутих. Оскільки для стандарту симетричного шифрування DES було знайдено ряд можливостей злому, зокрема через невелику кількість можливих ключів, існує загроза їх повного перебору. Тому для збільшення криптостійкості було розроблено інші версії цього алгоритму: double DES (2DES), triple DES (3DES), DESX, G-DES. У перспективі планується розробити на основі запропонованої нами реалізації алгоритму DES утиліти і для демонстрації роботи його модифікацій.

ISSN 2663 - 4023

Ключові слова: алгоритм симетричного шифрування; алгоритм DES; утиліта; шифрування; дешифрування; час виконання алгоритму.

ВСТУП

При вивченні курсу “Криптографічний та стеганографічний захист інформації”, що входить до переліку нормативних навчальних дисциплін спеціальності 125 Кібербезпека першого (бакалаврського) освітнього ступеня, та при вивченні курсу “Технології захисту інформації”, що входить до переліку нормативних навчальних дисциплін спеціальності 122 Комп’ютерні науки першого (бакалаврського) освітнього ступеня, обов’язковою є тема «Стандарти симетричного шифрування».

Постановка проблеми. Вивчення стандартів симетричного шифрування здобувачі розпочинають із відомого та достатньо добре дослідженого блокового алгоритму DES (Data Encryption Standard) [1]. Такий вибір зумовлено тим, що цей алгоритм є цікавим та зрозумілим за своїм змістом, але програмна реалізація не є простою. Особливо це стосується здобувача, який тільки почав опановувати навчальний матеріал та має на меті реалізувати самостійно алгоритми такого виду. Зважаючи на те, що алгоритм DES є найстарішим методом симетричного шифрування, він пройшов ряд випробувань на безпеку і для нього наперед відомим є число випробувань для злому. Також на даному алгоритмі ґрунтуються такі його багаторазові версії як 2DES з двома ключами та 3DES з трьома ключами, що дозволяють показувати свою істотну стійкість до атак грубої сили [2]. Тому розпочинати вивчення стандартів шифрування доцільно розпочати саме із алгоритму DES.

Аналіз останніх досліджень і публікацій. Криптографія, як процес перетворення інформації, включає в себе дві частини: шифрування і дешифрування. Шифрування – це процес перетворення звичайного тексту (читабельного) у зашифрований текст (нечитабельного) і дешифрування – це процес перетворення зашифрованого тексту (нечитабельного) у звичайний текст (читабельний) [3]. Є декілька підходів до класифікації криптографічних алгоритмів. Одним із них є класифікація, що базується на кількості ключів, які використовуються для шифрування та дешифрування. На рисунку 1 зображено цю класифікацію. Симетричний алгоритм (із приватним ключем) використовує один і той же ключ для шифрування та дешифрування. До таких алгоритмів, наприклад, відносяться AES, DES, TDES, RC2 і RC6 [4]. Асиметричні алгоритми шифрування (із відкритим ключем) використовують різні ключі для шифрування та дешифрування. Це алгоритм Діффі-Хеллмана, RSA та DSA [5, 6].

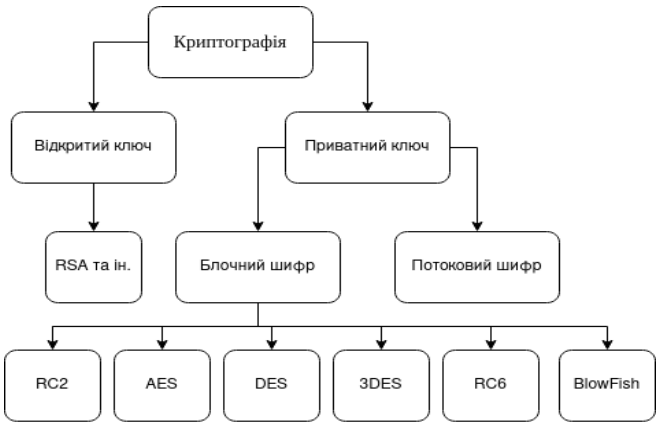

Рис. 1. Класифікація криптографічних алгоритмів

ISSN 2663 - 4023

Алгоритм симетричного шифрування DES представлено в 1976 році. Він виник як насідок розвитку проєкту «Люцифер» (Lucifer) [7], дослідного проєкту компанії IBM, метою якого було створення криптостійкого блочного шифру. Алгоритм DES розроблено для захисту конфіденційних урядових даних і офіційно прийнято в 1977 році для використання федеральними агентствами в США. DES – перший комерційний шифр, який став стандартом, що використовувався на урядовому рівні. Алгоритм шифрування DES був одним з тих, який використовували в версії 1.0 і 1.1 TLS (transport layer security) [8]. DES перетворює 64-бітні блоки даних відкритого тексту в зашифрований текст шляхом поділу на два окремих 32-бітних блока, застосовуючи процес шифрування до кожного окремо. Включає в себе 16 циклів різних процесів, через які будуть проходити дані в зашифрованому вигляді. В кінцевому підсумку 64-бітові блоки зашифрованого тексту створюються в якості вихідних даних.

DES є прикладом мережі Фейстеля [9], демонструє декілька режимів шифрування (ECB, CBC, CFB, OFB, CTR) [10], використовує такі основні перетворення як перестановка, підстановка, зсуви біт, інволюція, перетворення з однієї системи числення в іншу, використання раундового шифрування [11]. Програмування та аналіз такого виду алгоритмів дозволить здобувачеві покращити свої навички та вміння програмувати, порівнювати, оптимізовувати та досліджувати алгоритми.

Мета статті є особливості реалізації алгоритму симетричного блокового шифрування мовою програмування С та порівняння результатів роботи з іншими програмами-утилітами, які використовують алгоритми симетричного шифрування.

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Опишемо схему роботи алгоритму, схему генерування ключів та роботу головної функції шифрування алгоритмом DES та наведемо програмну реалізацію кожної із цих схем.

Схема роботи алгоритму DES зображена на рисунку 2.

Крок 1. Розбиваємо відкритий текст на блоки по 64 біт.

Крок 2. Виконуємо початкову перестановку IP біт 64-бітного блоку.

Крок 3. Ділимо 64-бітний блок на два блоки по 32 біт: лівий L_0 та правий R_0.

Крок 4. Виконуємо заміну та перестановку біт блоку R_0 за допомогою нелінійної функції f (буде описана пізніше), яка використовує ключ шифрування K_1.

Крок 5. Додавши по модулю 2 блок L_0 і значення функції $f(R_0, K_1)$, отримуємо правий 32-х бітний блок R_1. Лівий 32-х бітний блок L_1 приймаємо рівним R_0.

Крок 6. Виконуємо 4-5 у циклі ще 15 разів (раунди 2-16).

Крок 7. Об’єднуємо лівий та правий 32-х бітні блоки L_{16} та R_{16} в один 64-бітний блок.

Крок 8. Виконуємо кінцеву перестановкуIP^{-1} біт64-бітного блоку обернену до початкової перестановки IP.

ISSN 2663 - 4023

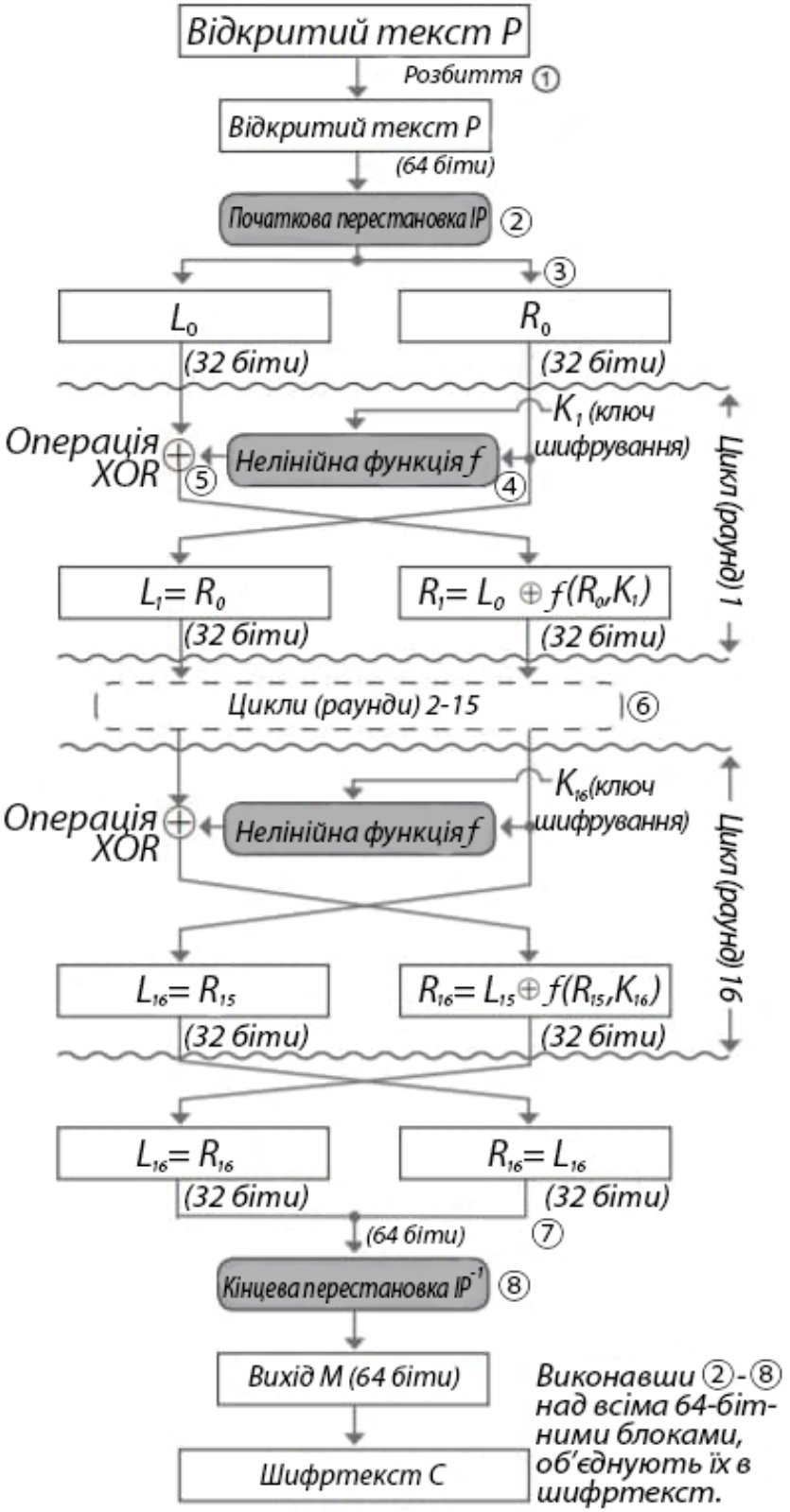

Рис. 2. Порядок шифрування DES (генерування зашифрованого тексту)

Розглянемо як генеруються ключі шифрування для алгоритму DES. Схема генерування зображена на рисунку 3. Ключі K_1, K_2, K_3, …, K_{16} генеруються так, щоб ключі, які використовуються в кожному раунді, відрізнялися один від одного.

ISSN 2663 - 4023

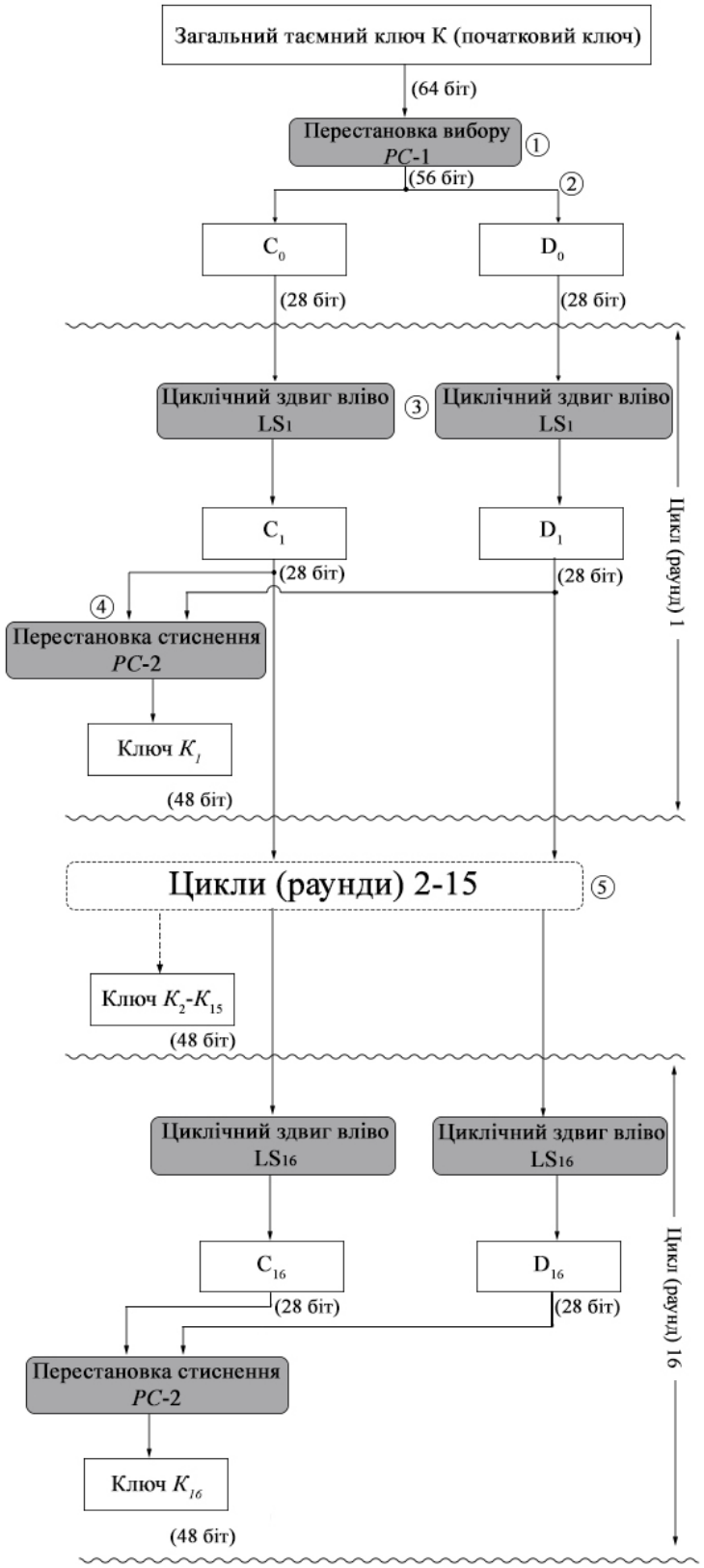

Рис. 3. Порядок генерування ключів шифрування і розшифрування DES

Опишемо процес покроково.

Крок 1. Забираємо 8 біт перевірки парності із 64-бітного початкового ключа, а біти що лишилися, переставляємо.

Крок 2. Ділимо 56 біт на 2 блоки по 28 біт: лівий C_0 і правий D_0.

Крок 3. Циклічно зсуваємо на визначену кількість біт (LS_1) блоки C_0 і D_0, отримуємо C_1 і D_1.

Крок 4. Об'єднуємо C_1 і D_1, переставляємо біти зі стисненням (PC-2) і отримуємо48-бітний ключ K_1.

Крок 5. Виконуючи обчислення 3-4 у циклі, отримуємо ключі K_n (n = 2, 3, …, 16) для кожного із раундів шифрування.

При генеруванні ключів розшифрування замість циклічних зсувів вліво виконуються циклічні зсуви вправо. Крім того, ключі розшифрування генеруються у зворотному порядку: K_{16}, K_{15}, K_{14}, … , K_1.

І, нарешті, розглянемо, як працює головна функція алгоритму. Схема її роботи описана на на рис. 4.

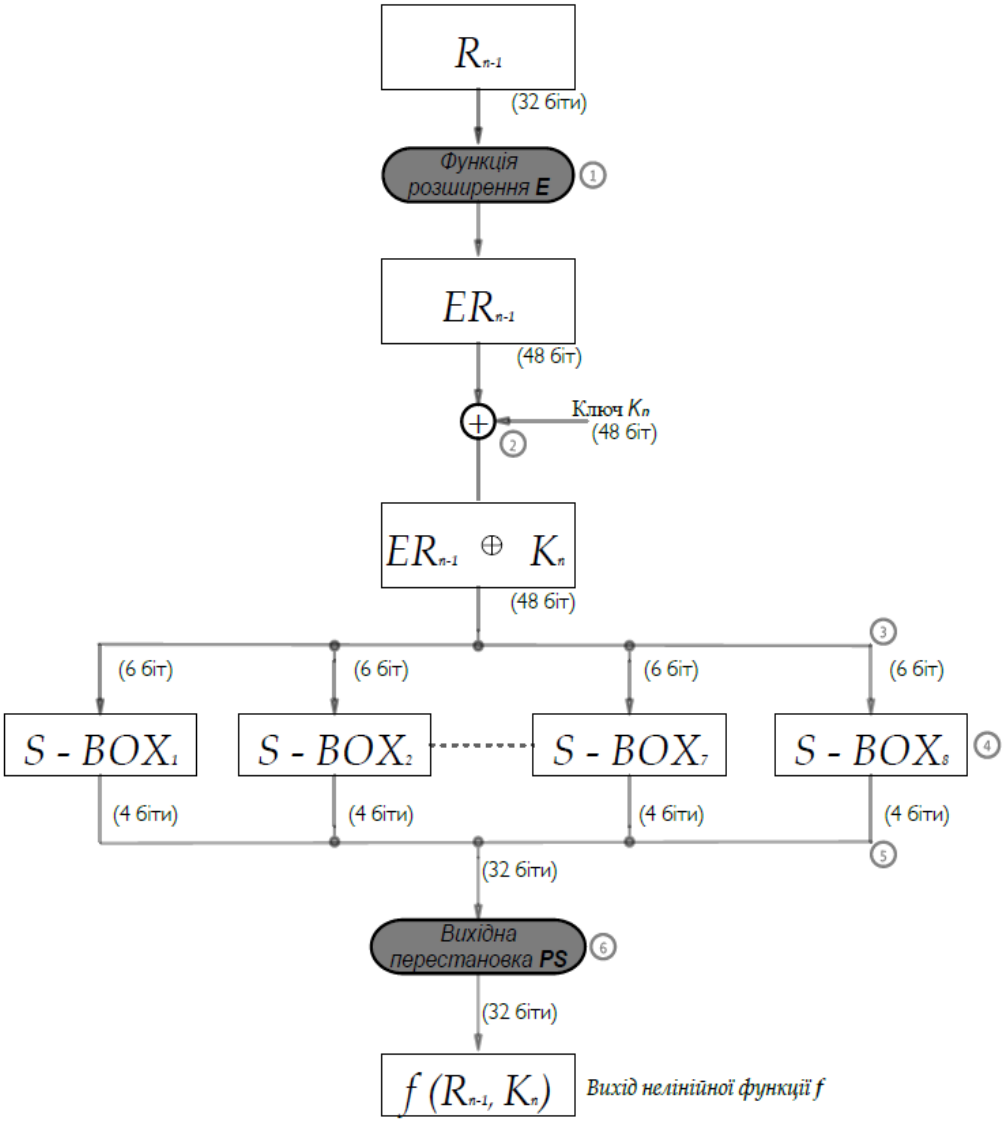

Рис. 4. Нелінійна функція f шифру DES

Покроковий алгоритм виконання функції *f* виглядає наступним чином:

Крок 1. Виконуємо розширення 32-х бітного блоку до 48 біт (довжина ключа).

Крок 2. Отриманий 48-бітний блок додаємо по модулю 2 з ключем.

Крок 3. Розбиваємо результат на вісім 6-ти бітних блоків.

Крок 4. Перетворюємо всі 6-ти бітні блоки у 4-бітні допомогою восьми S-блоків (таблиць заміни).

Крок 5. Об'єднуємо 4-бітні блоки у 32-бітний блок у порядку їх слідування.

Крок 6. Виконавши вихідну перестановку PS, отримаємо вихідне значення функції *f.i*

Розшифрування відбувається за аналогічною схемою як на рисунку 2.

Програмна реалізація алгоритму виконана на мові програмування С. Для першого пункту вхідний масив даних розбивається на блоки розміром 8 байт кожен, далі викликається функція шифрування одного блоку з переданими ключами, після чого цей блок записується у масив вихідних даних.

Програмна реалізація наведена нижче на рисунках 5 та 6.

```
size_t DES_encrypt_ecb( const uint8_t *input_data,
                        size_t bytes_count,
                        uint8_t *out_buffer,
                        const uint64_t keys48b[16],
                        uint64_t(*padding)(const uint8_t*, size_t))
{
    uint64_t block, encrypted_block;
    size_t i = 0;
    while(i+7 < bytes_count) {

        MAKE_64B_FROM_8BYTES(block, input_data+i)
1       encrypted_block = DES_encrypt_block(block, keys48b);
        WRITE_64B_TO_8BYTES(encrypted_block, out_buffer+i)

        i+=8;
    }

    size_t rest = bytes_count - i;
    if(padding != NULL) {
        block = padding(input_data+i,rest);
        encrypted_block = DES_encrypt_block(block, keys48b);
        WRITE_64B_TO_8BYTES(encrypted_block, out_buffer+i)
        return i+8;
```

Рис. 5. Програмна реалізація 1-го кроку алгоритму DES (розбиття тексту на блоки)

Генерування ключів відбувається за допомогою функції DES_keys_generate (рисунок 7), яка на вхід приймає 64-бітний ключ та масив 48-бітних ключів у кількості 16 штук. В результаті її виконання відносно вхідного ключа у буде згенеровано 16 ключів, які помістяться в масив із 48-бітних ключів. Програмна реалізація схеми з рис. 3 має наступний вигляд, як на рисунку 7.

Розглянемо програмну реалізацію нелінійної функції *f* шифру DES (програмний код представлено на рисунку 8), що зображена на рис. 5. Функція розширення носить

ISSN 2663 - 4023

назву E, операція xor реалізована оператором ^, функція для S-box носить назву S, кінцева перестановка - P.

```
220  uint64_t DES_encrypt_block(uint64_t  block, const uint64_t keys48b[16])
221  {
222      block = IP(block);                                  <- 2
223      uint32_t Li, Ri;
224      DIVIDE_TWO(block, Li, Ri, 32)                       <- 3
225
226      for(unsigned i = 0; i < 16; ++i) {                  <- 6
227          uint32_t Li_plus = Ri;
228          uint32_t Ri_plus = Li ^ F(Ri, keys48b[i]);      <- 4-5
229          Li = Li_plus;
230          Ri = Ri_plus;
231      }
232
233      COMBINE_TWO(Ri, Li, block, 32)                      <- 7
234      block = IP_reverse(block);                          <- 8
235
236      return block;
237  }
```

Рис. 6. Програмна реалізація кроків 2-8 алгоритму DES

```
189  #ifdef __cplusplus
190  extern "C"
191  #endif
192  void DES_keys_generate(uint64_t key, uint64_t keys48b[16])
193  {
194      uint64_t key56b = PC1(key);                         <- 1
195
196      uint32_t C,D;
197      DIVIDE_TWO_CLEAR(key56b,C,D,28);                    <- 2
198
199      for(size_t i = 0; i < 16; ++i) {
200          LEFT_CYCLE_SHIFT(C,28,SHIFT_table[i])           <- 3
201          LEFT_CYCLE_SHIFT(D,28,SHIFT_table[i])
202
203          uint64_t CiDi = 0x0;                            <- 4
204          COMBINE_TWO(C,D,CiDi,28)
205
206          keys48b[i] = PC2(CiDi);                         <- 5
207      }
208
209  }
```

Рис. 7. Програмна реалізація генерування ключів (схема з рис. 3)

Варто зауважити оптимізацію пунктів 3-5. У коді не створюється окремо масив з восьми 6-бітних блоків, кожен з яких проходить S-box та, врезультаті, об'єднуються у один 32-бітний блок, а це все здійснюється безпосередньо з операціями для кінцевого 32-бітного блоку у циклі на 8 ітерацій. Код наведено нижче (рисунок 8).

ISSN 2663 - 4023

```
static inline uint32_t F(uint32_t Ri, uint64_t key48b)
{
    uint64_t exp48b = E(Ri);                    <- 1
    uint64_t B8     = exp48b ^ key48b;          <- 2
    uint32_t result = S(B8);                    <- 3-5
    return            P(result);                <- 6
}
```

Рис. 8. Програмна реалізація функції f (схема з рис. 5)

Детальніше кроки 3-5 із функції *f* описано нижче на рисунку 9 (рядки 408 – 417).

```
static inline uint32_t S(uint64_t B8)
{
    uint32_t res = 0x0;
    for(int i = 0; i < 8; ++i) {
        uint8_t Bi = B8 >> (42 - i*6) & 0b00111111;

        uint8_t row = (Bi>>4&2) | (Bi&1) ;
        uint8_t col = (Bi>>1)&0xf;
        uint32_t S = S_table[i][row][col];


        res |= S << (28 - i *4);
    }
    return res;
}
```

Рис. 9. Детальна реалізація пунктів 3-5 функції f (схема з рис. 4)

Порівняння ефективності даної реалізації шифру DES виконувалось для двох утиліт:

- утиліта від OpenSSL, написана на мові assembler з використанням мови C, для можливості API викликів [12];
- утиліта, написана на мові Java [13].

Порівняння проводилось за трьома параметрами:

real - повний час від початку роботи утиліти до її завершення;

user — час, який затратив процесор на виконання утиліти, не враховується час простою і час, коли процесор виконував інші завдання;

sys — час, який затратила операційна система для роботи утиліти, наприклад, читання файлу або його запис на диск.

Результати порівняння представлені на рисунку 10.

Із проведеного аналізу бачимо, що user + sys != real. Чому так? Насправді час real показує час, який витрачено на виконання утиліти, тобто його можемо виміряти секундоміром. Тоді як user – це час, який був витрачений процесором на виконання утиліти, тобто якщо два ядра будуть виконувати одну і ту ж утиліту впродовж 1 секунди, то загальний час user буде дорівнювати 2 секундам, хоча насправді пройшла одна секунда часу. Проміжні результати та результати шифрування/дешифрування перевірені на прикладах на правильність, тому з впевненістю можна вважати, що алгоритм реалізовано вірно.

З проведеного порівняння можемо зробити висновок, що час, який затрачений на шифрування, менший від часу, затраченого на розшифрування. Час виконання різних утиліт – різний, а саме: час утиліти на ASSEMBLER виявився найкращим. Це і зрозуміло,

ISSN 2663 - 4023

бо така реалізація найближча для обчислювальної платформи і буде найшвидша. Розроблена нами утиліта близька за часом до попередньої, не дивлячись на те, що для реалізації була вибрана мова програмування С. Утиліта на Java виявилася за часом виконання гірша від двох попередніх. тобто можна зробити висновки, що рівень програмування та оптимізації нашої утиліти виконано на високому рівні та враховано всі особливості методу.

```
andrew@Naris-Ubuntu:/mnt/fast-storage/Sandbox/DES-bench-mark$ bash benchmark.sh

Шифрування...
ЧАС     JAVA реалізація від "deadlytea" :
real    0m18,256s
user    0m18,414s
sys     0m0,491s

ЧАС     ASSEMBLER реалізація від OpenSSL:
real    0m0,004s
user    0m0,007s
sys     0m0,000s

ЧАС     C реалізація, створена утиліта:
real    0m0,012s
user    0m0,012s
sys     0m0,000s

Розшифрування...
ЧАС     JAVA реалізація від "deadlytea" :
real    0m3,352s
user    0m3,678s
sys     0m0,432s

ЧАС     ASSEMBLER реалізація від OpenSSL:
real    0m0,005s
user    0m0,003s
sys     0m0,001s

ЧАС     C реалізація, створена утиліта:
real    0m0,007s
user    0m0,007s
sys     0m0,000s
```

Рис. 10. Час виконання запрограмованого алгоритму DES різними утилітами

ВИСНОВКИ ТА ПЕРСПЕКТИВИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

У роботі детально розглянуто симетричний алгоритм шифрування DES та його основні кроки. За допомогою мови програмування С здійснено реалізацію алгоритму в обидві сторони (шифрування/дешифрування). Для кращого розуміння основних елементів реалізації схематично показано (рис. 2-4) як відбуваються процеси в алгоритмі. Програмування такого виду алгоритмів покращує навики, дозволяє виконати оптимізацію коду і прагнути досягти гарних результатів по часу.

Оскільки для стандарту симетричного шифрування DES було знайдено ряд можливостей злому, зокрема через невелику кількість можливих ключів, існує загроза їх повного перебору. Тому для збільшення криптостійкості було розроблено інші версії цього алгоритму: double DES (2DES), triple DES (3DES), DESX, G-DES [14]. У перспективі планується розробити на основі запропонованої нами реалізації алгоритму DES утиліти і для демонстрації роботи його модифікацій..

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Глинчук, Л., Яцюк, С., Кузьмич, О., Багнюк, Н., & Чернящук, Н. (2020). Аналіз вимог та методологія підбору тем для вивчення основ криптографічного захисту інформації. *COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION*, (41), 16–22. https://doi.org/10.36910/6775-2524-0560-2020-41-03

2 Hoobi, M. M. (2017). Strong triple data encryption standard algorithm using nth degree truncated polynomial ring unit. *Iraqi Journal of Science*, *58*(3C). https://doi.org/10.24996/ijs.2017.58.3c.19.

3 Кузнецов, О. О. та ін. (2014). Обгрунтування вимог, побудування та аналіз перспективних симетричних криптоперетворень на основі блочних шифрів . *Вісник Національного університету "Львівська політехніка"*, 806, 124–142.

4 Charbathia, S., Sharma, S. (2014). A comparative study of rivest cipher algorithms. *International journal of information & computation technology*, 4(17), 1831–1838.

5 Imamoto, K., & Sakurai, K. (2005). Design and Analysis of Diffie-Hellman-Based Key Exchange Using One-time ID by SVO Logic. *Electronic Notes in Theoretical Computer Science*, *135*(1), 79–94. https://doi.org/10.1016/j.entcs.2005.06.003

6 Масааки, М., Синъити, С. (2019). *Занимательная информатика. Криптография. Манга*. Москва : ДМК Пресс.

7 Бабинюк, О. І., Нагірна, А. М., Нагорна, О. В. (2019). Алгоритм шифрування Lucifer та його криптоаналіз. *Моделювання та інформаційні системи в економіці : зб. наук. Пр*, *98*, 13–24.

8 Cryptology - the data encryption standard and the advanced encryption standard. *Encyclopedia Britannica*. https://www.britannica.com/topic/cryptology/The-Data-Encryption-Standard-and-the-Advanced-Encryption-Standard#ref794749

9 Zhang, X., Zhou, Z., & Niu, Y. (2018). An Image Encryption Method Based on the Feistel Network and Dynamic DNA Encoding. *IEEE Photonics Journal*, *10*(4), 1–14. https://doi.org/10.1109/jphot.2018.2859257

10 Kiernan & Mueller. (2021). Standardizing Security: Surveillance, Human Rights, and the Battle Over Tls 1.3. *Journal of Information Policy*, *11*, 1-25. https://doi.org/10.5325/jinfopoli.11.2021.0001

11 Babar, P. K., Bhope, V. P. (2016). Design and implement dynamic key generation to enhance DES algorithm. *Nternational journal for research in applied science & engineering technology*, 4, 465–468.

12 OpenSSL Quick Reference Guide | DigiCert.com. *SSL Digital Certificate Authority - Encryption & Authentication*. https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm.

13 GitHub - deadlytea/DES: Java implementation of DES for university cryptography course. *GitHub*. https://github.com/deadlytea/DES.

14 Smekal, D., Hajny, J., & Martinasek, Z. (2018). Comparative Analysis of Different Implementations of Encryption Algorithms on FPGA Network Cards. *IFAC-PapersOnLine*, *51*(6), 312–317. https://doi.org/10.1016/j.ifacol.2018.07.172

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 2 (14), 2021
ISSN 2663 - 4023

Liudmyla Y. Hlynchuk
Cand. Phys. and Math. Sc. (Ph.D), Associate Professor at the
Department of Computer Sciences and Cybersecurity
Lesya Ukrainka Volyn National University, Lutsk, Ukraine
ORCID ID: 0000-0002-8943-9604
Hlynchuk.Ludmila@vnu.edu.ua

Tetiana O. Hryshanovych
Cand. Phys. and Math. Sc. (Ph.D), Associate Professor at the
Department of Computer Sciences and Cybersecurity
Lesya Ukrainka Volyn National University, Lutsk, Ukraine
ORCID ID: 0000-0002-3595-6964
Hryshanovych.Tatiana@vnu.edu.ua

Andrii P. Stupin
student of the faculty of information technologies and mathematics,
Lesya Ukrainka Volyn National University, Lutsk, Ukraine
stupin.andrij2018@vnu.edu.ua

IMPLEMENTATION OF THE SYMMETRICAL ENCRYPTION STANDARD DES USING C PROGRAMMING LANGUAGE AND COMPARISON ITS EXECUTION TIME WITH OTHER UTILITIES

Abstract. This research dedicated to the review, implementation and analysis of the symmetric encryption algorithm, namely - DES (Data Encryption Standard) that encrypts and decrypts text information. For this algorithm represented not only a verbal description, but also schemes of its execution and examples of implementation. Intermediate results and the results of information encryption / decryption in the implemented algorithm were verified using examples, so we can assume that the algorithm implemented correctly. Comparison of the execution time for the DES algorithm proposed implementation made for two utilities. One of them is OpenSSL, developed using assembler and the capabilities of the C programming language. The other utility developed using programming language Java. The comparison was made according to three criteria: full time from the utility execution start to its completion; the time spent by the process to execute the utility (downtime and time when the processor perform other tasks not accounted); the time taken by the operating system to run a utility, such as reading or writing the file. The analysis showed that the total execution time is not equal to the total amount of time spent by both the processor and the operating system to execute the utilities. This is due to the following: the total execution time is the real time spent on the execution of the utility; it can measure with a stopwatch. Whereas the time spent by the processor to execute the utility is measured somewhat differently: if two cores execute the same utility for 1 second, the total execution time will be 2 seconds, although in fact one second of time has passed. From the comparison follows the next conclusion: the time spent on encryption is less than the time spent on decryption. The execution time for different utilities is different: the time for OpenSSL utility turned out to be the best, because such an implementation is most adapted to the hardware. The utility in Java turned out to be the worst in terms of execution time. We propose the implementation of the DES algorithm of the nearest execution time to the fastest of the considered. Because a number of hacking possibilities have been found for the symmetric encryption standard DES, in particular due to the small number of possible keys, there is a risk of overriding them. Therefore, to increase crypto currency, other versions of this algorithm have been developed: double DES (2DES), triple DES (3DES), DESX, G-DES. In the future, it is planned to develop a utility based on our proposed implementation of the DES algorithm and to demonstrate the operation of its modifications.

Keywords: algorithm DES; utility; encryption; decryption; algorithm execution time.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1 Hlynchuk, L., Yatsiuk, C., Kuzmych, O., Bahniuk, N., & Cherniashchuk, N. (2020). Analiz vymoh ta metodolohiia pidboru tem dlia vyvchennia osnov kryptohrafichnoho zakhystu informatsii. COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION, (41), 16–22. https://doi.org/10.36910/6775-2524-0560-2020-41-03

2 Hoobi, M. M. (2017). Strong triple data encryption standard algorithm using nth degree truncated polynomial ring unit. Iraqi Journal of Science, 58(3C). https://doi.org/10.24996/ijs.2017.58.3c.19.

3 Kuznetsov, O. O. ta in. (2014). Obhruntuvannia vymoh, pobuduvannia ta analiz perspektyvnykh symetrychnykh kryptoperetvoren na osnovi blochnykh shyfriv . Visnyk Natsionalnoho universytetu "Lvivska politekhnika", 806, 124–142.

4 Charbathia, S., Sharma, S. (2014). A comparative study of rivest cipher algorithms. International journal of information & computation technology, 4(17), 1831–1838.

5 Imamoto, K., & Sakurai, K. (2005). Design and Analysis of Diffie-Hellman-Based Key Exchange Using One-time ID by SVO Logic. Electronic Notes in Theoretical Computer Science, 135(1), 79–94. https://doi.org/10.1016/j.entcs.2005.06.003

6 Masaaky, M., Synъyty, S. (2019). Zanymatelnaia ynformatyka. Kryptohrafyia. Manha. Moskva : DMK Press.

7 Babyniuk, O. I., Nahirna, A. M., Nahorna, O. V. (2019). Alhorytm shyfruvannia Lucifer ta yoho kryptoanaliz. Modeliuvannia ta informatsiini systemy v ekonomitsi : zb. nauk. Pr, 98, 13–24.

8 Cryptology - the data encryption standard and the advanced encryption standard. Encyclopedia Britannica. https://www.britannica.com/topic/cryptology/The-Data-Encryption-Standard-and-the-Advanced-Encryption-Standard#ref794749

9 Zhang, X., Zhou, Z., & Niu, Y. (2018). An Image Encryption Method Based on the Feistel Network and Dynamic DNA Encoding. IEEE Photonics Journal, 10(4), 1–14. https://doi.org/10.1109/jphot.2018.2859257

10 Kiernan & Mueller. (2021). Standardizing Security: Surveillance, Human Rights, and the Battle Over Tls 1.3. Journal of Information Policy, 11, 1-25. https://doi.org/10.5325/jinfopoli.11.2021.0001

11 Babar, P. K., Bhope, V. P. (2016). Design and implement dynamic key generation to enhance DES algorithm. Nternational journal for research in applied science & engineering technology, 4, 465–468.

12 OpenSSL Quick Reference Guide | DigiCert.com. SSL Digital Certificate Authority - Encryption & Authentication. https://www.digicert.com/kb/ssl-support/openssl-quick-reference-guide.htm.

13 GitHub - deadlytea/DES: Java implementation of DES for university cryptography course. GitHub. https://github.com/deadlytea/DES.

14 Smekal, D., Hajny, J., & Martinasek, Z. (2018). Comparative Analysis of Different Implementations of Encryption Algorithms on FPGA Network Cards. IFAC-PapersOnLine, 51(6), 312–317. https://doi.org/10.1016/j.ifacol.2018.07.172