**РЕФЕРАТ**

З дисципліни «Комп'ютерна криптографія»

На тему: «Дослідження алгоритму симетричного криптографічного перетворення ДСТУ 7624:2014 «Калина»

|  |  |
| --- | --- |
| **Виконали**  Студенти групи КВ-61м Щербакова Г.В.  Бондарчук М.Ю.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | **Прийняв**  Доцент кафедри СПСКС  Тесленко О.К.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

ВСТУП

До 2015 року ГОСТ 28147-89 був основним блочним, що використовувався в Україні. Навіть зараз даний шифр забезпечує достатній рівень безпеки. Проте його програмна реалізація є значно повільнішою та менш ефективною в сучасних критеріях у порівнянні з рішеннями, такими як AES.

Використовуючи досвід західних держав у проведенні тендерів для криптографічних рішень (наприклад, AES чи NESSIE) Державна служба спеціального зв'язку та захисту інформації України провела тендер для блочних шифрів, що могли стати державним стандартом. Основними вимогами були високий рівень криптографічної безпеки, гнучкий розмір блоку та ключа (128, 256 та 512 біт) та прийнятна швидкодія при програмній реалізації. Серед усіх алгоритмів було вибрано «Калину», що була встановлена як ДСТУ 7326:2014. Державний стандарт описує блочний шифр та 10 типів операцій до нього.

«Калина» – симетричний блочний алгоритм шифрування, розроблений ЗАТ «Інститут інформаційних технологій» м. Харків [1-2]. Алгоритм має розмір блоку 128, 256 і 512 бітів і підтримує такі ж довжини ключів. При побудові шифру основна увага розробників була приділена забезпеченню високого рівня криптографічної стійкості, а також досягненню високих показників продуктивності в апаратній та програмній реалізаціях. Був врахований інтелектуальний внесок в практику проектування та криптоаналізу блочних алгоритмів шифрування провідних криптографів, що отримав значний розвиток в ході проведення міжнародних проектів NESSIE і AES. Широко застосовувалися результати наукових робіт по методам криптоаналізу і проектування симетричних шифрів останніх років, що були опубліковані українськими авторами, а також власні дослідження і розробки ЗАТ «ІІТ» в цьому напрямку.

ЗАГАЛЬНА СТРУКТУРА АЛГОРИТМУ

Алгоритм шифрування «Калина» є ітеративною процедурою, що складається з початкової і кінцевої рандомізації, а також двох різних ітеративних послідовних шифрувальних перетворень. Структура алгоритму аналогічна структурі AES, та забезпечує гарне розсіювання і перемішування. На вхід кожного шифрувального перетворення подається поточний стан шифру та необхідна кількість ключів (основний та раундовий). Відкритий текст копіюється в поточну матрицю стану перед початком шифрування, а по завершенню шифрування в поточному стані знаходиться зашифрований текст. Кількість циклів шифрування залежить від довжини ключа (мастер-ключа), при цьому довжина ключа не може бути менше розміру блоку, що шифрується.

ПОЗНАЧЕННЯ В АЛГОРИТМІ

|  |  |
| --- | --- |
| 0x | – префікс шістнадцяткових літералів |
|  | поле Галуа для поліному |
| ⊕ | сума по модулю 2 (XOR) |
|  | ціла частина числа |
|  | довжина бінарної послідовн |
|  | функція, що повертає менш значимих бітів бінарної послідовності довжиною |
|  | функція, що повертає більш значимих бітів бінарної послідовності довжиною |
| >> | логічний зсув праворуч |
| << | логічний зсув ліворуч |
| >>> | циклічний зсув праворуч |
| <<< | циклічний зсув ліворуч |
| + | додавання по модулю |
| Ⓧ | скалярний добуток векторів |
|  | розмір блоку (128, 256, 512 біт) |
|  | розмір ключа (128, 256, 512 біт) ( або ) |
|  | кількість рядків в матриці стану |
|  | вектор розміру |
|  | базове перетворення при шифруванні |
|  | базове перетворення при розшифруванні |
|  | Конкатенація бінарних послідовностей |
|  | послідовне виконання операцій (справа наліво) |
| t | число перетворень при виконанні чи |
|  | послідовне виконання операцій , , … , , починаючи з |
|  | представлення невід’ємного цілого числа у вигляді бінарної послідовності з бітів |
|  | виконання чи для розміру блоку та розміру ключа |

ПАРАМЕТРИ АЛГОРИТМУ

Кількість раундів для різних комбінацій довжин блоків і ключів.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Довжина блоку | Довжина ключа | Кількість раундів | Кількість рядків внутрішньої матриці стану |
| 1 | 128 | 128 | 10 | 2 |
| 2 | 256 | 14 |
| 3 | 256 | 256 | 14 | 4 |
| 4 | 512 | 18 |
| 5 | 512 | 512 | 18 | 8 |

ВХІДНІ ТА ВИХІДНІ ДАНІ

Базовий процес перетворення обробляє вхідний блок розміром біт (відкритого чи зашифрованого тексту). Вводиться поняття матриці внутрішнього стану , яка представляється як , де , та . Матриця внутрішнього стану заповнюється вхідними байтами , , …, по рядкам. Іншими словами, вхідний блок розбивається на колонки по 8 байт. Наприклад, якщо вхідний блок розміром 128 біт представлено як D5810DDF7F8157482705CADC256DF03B, то матриця внутрішнього стану матиме вигляд:

|  |  |  |
| --- | --- | --- |
|  | j=0 | j=1 |
| i=0 | D5 | 27 |
| i=1 | 81 | 05 |
| i=2 | 0D | CA |
| i=3 | DF | DC |
| i=4 | 7F | 25 |
| i=5 | 81 | 6D |
| i=6 | 57 | F0 |
| i=7 | 48 | 3B |

ПРОЦЕС ШИФРУВАННЯ

Процес шифрування можна описати формулою

де – ключ шифрування довжиною біт,

– функція додавання по модулю матриці внутрішнього стану з раундовим ключем ,

- функція нелінійного взаємно-однозначного відображення (S-box), що обробляє вектори,

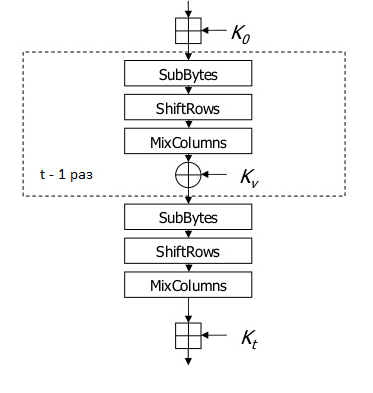
– перестановка елементів матриці внутрішнього стану (циклічний зсув байтів вправо),

– лінійне перетворення елементів матриці внутрішнього стану з використанням алгебри Галуа,

– функція додавання по модулю 2 (XOR) раундового ключа і матриці внутрішнього стану.

В функціях , та вхідним параметром є , а вихідне значення , подається у вигляді матриць розміром , залежно від .

Алгоритм можна подати в більш наглядному вигляді:



Функція додавання по модулю

Функція виконує попарне додавання колонок матриці внутрішнього стану та колонок матриці раундового ключа використовуючи додавання за модулем 264. Результатом є також матриця (матриця внутрішнього стану після додавання раундового ключа).

Розглянемо приклад. Нехай до поточного зашифрованого тексту 6A8B9BD8A0A7984F71A52D25636F3A87, що представлений матрицею внутрішнього стану

|  |  |
| --- | --- |
| 6A | C1 |
| 8B | A5 |
| 9B | 2D |
| D8 | 25 |
| A0 | 63 |
| A7 | 6F |
| 98 | 3A |
| 4F | 87 |

необхідно додати раундовий ключ 504F4E4D4C4B4A494847464544434241, що представлений матрицею

|  |  |
| --- | --- |
| 50 | 48 |
| 4F | 47 |
| 4E | 46 |
| 4D | 45 |
| 4C | 44 |
| 4B | 43 |
| 4A | 42 |
| 49 | 41 |

Тоді результатом буде

(6A8B9BD8A0A7984F + 504F4E4D4C4B4A49) mod 264 = BADAEA25ECF2E298

(4847464544434241 + C1A52D25636F3A87) mod 264 = 19EC736AA7B27CC8

|  |  |
| --- | --- |
| BA | 19 |
| DA | EC |
| EA | 73 |
| 25 | 6A |
| EC | A7 |
| F2 | B2 |
| E2 | 7C |
| 98 | C8 |

Або просто BADAEA25ECF2E29819EC736AA7B27CC8.

Функція нелінійного взаємно-однозначного відображення

Функція виконує операцію, що відома в блочних алгоритмах як S-box. Кожен байт матриці внутрішнього стану замінюєтья на байт , де , називається відображенням, а всі таблиці для відображень наведені в додатку А.

Розглянемо приклад. Нехай необхідно перетворити поточне значення AC309488D842CCE8806FBE500F257691:

|  |  |
| --- | --- |
| AC | 80 |
| 30 | 6F |
| 94 | BE |
| 88 | 50 |
| D8 | 0F |
| 42 | 25 |
| CC | 76 |
| E8 | 91 |

Тоді результатом буде 2D7C5005C4694B0B9B20DD3109F46C9F:

|  |  |
| --- | --- |
| (AC)=2D | (80)=9B |
| (30)=7C | (6F)=20 |
| (94)=50 | (BE)=DD |
| (88)=05 | (50)=31 |
| (D0)=C4 | (0F)=09 |
| (42)=69 | (25)=F4 |
| (CC)=4B | (76)=6C |
| (E8)=0B | (91)=9F |

Функція зсуву матриці внутрішнього стану

Функція виконує циклічний зсув кожного рядка матриці внутрішнього стану . Кількість елементів для зсуву залежить від номеру рядка , довжини блоку та обчислюється за формулою .

Розглянемо приклад. Нехай необхідно виконати зсув матриці внутрішнього стану для поточного значення 9638AD0DCB389479F6D17436193AA3337F4D72B36707F895E80566FDCEB99443 ():

|  |  |  |  |
| --- | --- | --- | --- |
| 96 | F6 | 7F | E8 |
| 38 | D1 | 4D | 05 |
| AD | 74 | 72 | 66 |
| 0D | 36 | B3 | FD |
| CB | 19 | 67 | CE |
| 38 | 3A | 07 | B9 |
| 94 | A3 | F8 | 94 |
| 79 | 33 | 95 | 43 |

Тоді результат отримуємо як:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| 0 |  | 96 | F6 | 7F | E8 |
| 1 |  | 38 | D1 | 4D | 05 |
| 2 |  | 66 | AD | 74 | 72 |
| 3 |  | FD | 0D | 36 | B3 |
| 4 |  | 67 | CE | CB | 19 |
| 5 |  | 07 | B9 | 38 | 3A |
| 6 |  | A3 | F8 | 94 | 94 |
| 7 |  | 33 | 95 | 43 | 79 |

Або 963866FD6707A333F6D1AD0DCEB9F8957F4D7436CB389443E8057 2B3193A9479.

Функція лінійного перетворення

Для виконання функції кожен байт матриці внутрішнього стану представляється як елемент скінченного поля , що формується за допомогою незвідного поліному чи .

Кожен байт результуючої матриці обчислюється за допомогою алгебри Галуа через формулу:

де – вектор, що в процесі множення циклічно зсувається вправо на 1 байт, – -й стовпчик матриці , – скалярний добуток векторів, де замість операції додавання використовується додавання за модулем 2 (xor), а замість звичайного множення – множення у полі Галуа.

Операцію множення многочленів можна мовою C# описати як

|  |
| --- |
| byte Gmul(int a, int b)  {  byte p = 0;  while (b != 0)  {  if ((b & 1) == 1)  p ^= (byte)a;  if ((a & 0x80) == 0x80)  a = (a << 1) ^ 0x11d;  else  a <<= 1;  b >>= 1;  }  return p;  } |

Розглянемо приклад. Нехай необхідно виконати лінійне перетворення C1D335E2C253A5BA71E5864DA37694AC:

|  |  |
| --- | --- |
| C1 | 71 |
| D3 | E5 |
| 35 | 86 |
| E2 | 4D |
| C2 | A3 |
| 53 | 76 |
| A5 | 94 |
| BA | AC |

Оскільки операція множення в полі Галуа описана у відповідному розділі математики, а в алгоритмі обчислюється за допомогою наведеного вище коду, то її описувати не будемо. Тоді результат матиме вигляд

|  |  |
| --- | --- |
| Gmul(C1,1)Gmul(D3,1)  Gmul(35,5)Gmul(E2,1)  Gmul(C2,8)Gmul(53,7)  Gmul(A5,6)Gmul(BA,4)=36 | Gmul(71,1)Gmul(E5,1)  Gmul(86,5)Gmul(4D,1)  Gmul(A3,8)Gmul(76,7)  Gmul(94,6)Gmul(AC,4)=64 |
| Gmul(C1,4)Gmul(D3,1)  Gmul(35,1)Gmul(E2,5)  Gmul(C2,1)Gmul(53,8)  Gmul(A5,7)Gmul(BA,6)=10 | Gmul(71,4)Gmul(E5,1)  Gmul(86,1)Gmul(4D,5)  Gmul(A3,1)Gmul(76,8)  Gmul(94,7)Gmul(AC,6)=D6 |
| Gmul(C1,6)Gmul(D3,4)  Gmul(35,1)Gmul(E2,1)  Gmul(C2,5)Gmul(53,1)  Gmul(A5,8)Gmul(BA,7)=85 | Gmul(71,6)Gmul(E5,4)  Gmul(86,1)Gmul(4D,1)  Gmul(A3,5)Gmul(76,1)  Gmul(94,8)Gmul(AC,7)=4A |
| Gmul(C1,7)Gmul(D3,6)  Gmul(35,4)Gmul(E2,1)  Gmul(C2,1)Gmul(53,5)  Gmul(A5,1)Gmul(BA,8)=55 | Gmul(71,7)Gmul(E5,6)  Gmul(86,4)Gmul(4D,1)  Gmul(A3,1)Gmul(76,5)  Gmul(94,1)Gmul(AC,8)=58 |
| Gmul(C1,1)Gmul(D3,1)  Gmul(35,5)Gmul(E2,1)  Gmul(C2,8)Gmul(53,7)  Gmul(A5,6)Gmul(BA,4)=92 | Gmul(71,8)Gmul(E5,7)  Gmul(86,6)Gmul(4D,4)  Gmul(A3,1)Gmul(76,1)  Gmul(94,5)Gmul(AC,1)=D0 |
| Gmul(C1,4)Gmul(D3,1)  Gmul(35,1)Gmul(E2,5)  Gmul(C2,1)Gmul(53,8)  Gmul(A5,7)Gmul(BA,6)=8C | Gmul(71,1)Gmul(E5,8)  Gmul(86,7)Gmul(4D,6)  Gmul(A3,4)Gmul(76,1)  Gmul(94,1)Gmul(AC,5)=B5 |
| Gmul(C1,6)Gmul(D3,4)  Gmul(35,1)Gmul(E2,1)  Gmul(C2,5)Gmul(53,1)  Gmul(A5,8)Gmul(BA,7)=C8 | Gmul(71,5)Gmul(E5,1)  Gmul(86,8)Gmul(4D,7)  Gmul(A3,6)Gmul(76,4)  Gmul(94,1)Gmul(AC,1)=09 |
| Gmul(C1,7)Gmul(D3,6)  Gmul(35,4)Gmul(E2,1)  Gmul(C2,1)Gmul(53,5)  Gmul(A5,1)Gmul(BA,8)=09 | Gmul(71,1)Gmul(E5,5)  Gmul(86,1)Gmul(4D,8)  Gmul(A3,7)Gmul(76,6)  Gmul(94,4)Gmul(AC,1)=87 |

Або 36108555928CC80964D64A58D0B50987.

Функція додавання за модулем 2

Функція використовує звичайне арифметичне додавання за модулем 2 (XOR) і не має жодних особливостей.

Функція , що залежить від параметра (раундовий ключ -ї ітерації), і в його якості отримує внутрішній стан шифротексту . І раундовий ключ , і внутрішній стан шифротексту представляються матрицями розміром , де . В функції дані матриці додаються використовуючи операцію XOR. Результатом є матриця розміром .

Розглянемо приклад. Нехай до поточного значення шифротексту 566E199501A3D4FE1C8FFE90C2B0ADA6 необхідно додати раундовий ключ FA5E8D1DD9E118169CE040C0CD95DB37. Матриці даних значень:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 56 | 1C |  | FA | 9C |
| 6E | 8F |  | 5E | E0 |
| 19 | FE |  | 8D | 40 |
| 95 | 90 |  | 1D | C0 |
| 01 | C2 |  | D9 | CD |
| A3 | B0 |  | E1 | 95 |
| D4 | AD |  | 18 | DB |
| FE | A5 |  | 16 | 37 |

Тоді результатом буде AC309488D842CCE8806FBE500F257691:

|  |  |
| --- | --- |
| 56⊕FA=AC | 1C⊕9C=80 |
| 6E⊕5E=30 | 8F⊕E0=6F |
| 19⊕8D=94 | FE⊕40=BE |
| 95⊕1D=88 | 90⊕C0=50 |
| 01⊕D9=D8 | C2⊕CD=0F |
| A3⊕E1=42 | B0⊕95=25 |
| D4⊕18=CC | AD⊕DB=76 |
| FE⊕16=E8 | A5⊕37=91 |

ПРОЦЕС РОЗШИФРУВАННЯ

Процес оберненим до процесу шифрування і описується формулою

де приставка означає функцію, обернену, до відповідної функції, що використовується при шифруванні.

Відмінності від процесу шифрування полягають в наступному:

* Замість функції додавання за модулем 264використовується функції віднімання за модулем 264.
* Функція нелінійного взаємно-однозначного відображення замість таблиць використовує таблиці .
* Функція зсуву матриці внутрішнього стану виконує зсув вліво замість зсуву вправо.
* Функція лінійного перетворення замість вектора використовує вектор і представлений він як .

ПРОЦЕС ГЕНЕРАЦІЇ РАУНДОВИХ КЛЮЧІВ

Довжина проміжного ключа дорівнює довжині блоку ( біт) та представляється матрицею розміром . Цей ключ генерується з ключа шифрування використовуючи наступне перетворення:

де , , , , – функції, що розглянуті в описі процесі шифрування.

Коли довжина блока дорівнює довжині ключа (), тоді

Якщо ж довжина блока не дорівнює довжині ключа (), то тоді і

Дану формулу можна спростити з точки зору програмної реалізації. Для цього необхідно розбити її на формулу для раундових ключів з парними і непарними номерами.

Раундові ключі з парними номерами

Обчислюються за формулою

де – функція, що визначається як , де – є довжиною внутрішнього стану шифротексту.

Раундові ключі з непарними номерами

Обчислюються за формулою

де – розмір внутрішнього стану шифротексту (в бітах), а

Тобто дана формула виконує циклічний зсув ключів, згенерованих на попередньому етапі.

ВИСНОВКИ

Застосування алгебраїчного аналізу дозволяє описати схему розгортування ключів алгоритму «Калина» [128/128) системо. рівнянь 3 ступня, що містить 25224 рівняння з 384 невідомими (або 256 невідомими для спрощеного варіанту, що відрізняється від наведеного в специфікації ІІТ)

Рішеннями такої системи є все можливі комбінації проміжних станів і ключів шифрування, а однозначне визначення ключа шифрування по відомому стану ключа за допомогою відомих методів неможливо. Складність визначення додаткових проміжних станів (по всій множині значень) для обчислення ключів шифрування є не меншою, ніж пошук з повним перебором.

Таким чином, підхід, використаний при розробці схеми розгортування ключів алгоритму «Калина» забезпечує стійкість до алгебраїчних методів криптоаналізу при спробі відновити значення ключа шифрування по відомим ключам, що, в свою чергу, забезпечує додатковий рівень захисту від атак на алгоритм.

В загальному, алгоритм «Калина» є дуже подібним до алгоритму шифрування AES і відрізняється лише в реалізації деяких функцій шифрування.

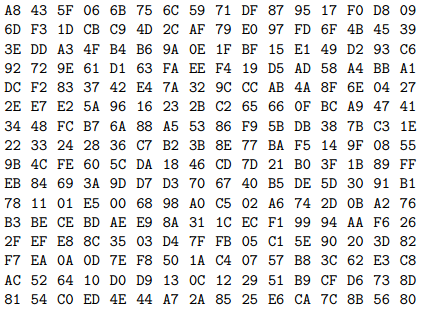
СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Government Committee of the USSR for Standards. GOST 28147-89. State Standard of the USSR. Information Processing Systems. Cryptographic protection. Algorithm of cryptographic transformation. Government Committee of the USSR for Standards, 1990 (in Russian).
2. National Institute of Standards and Technology (NIST). Advanced Encryption Standard (AES). Federal Information Processing Standards (FIPS) Publication 197, Nov. 2001.
3. Courtois, Nicolas T. Security evaluation of GOST 28147-89 in view of international standardisation. Cryptologia 36.1 (2012): 2-13.
4. National Institute of Standards and Technology (NIST). Announcing Development Of A Federal Information Processing Standard For Advanced Encryption Standard. [http://csrc.nist.gov/archive/aes/pre-round1/aes\_9701.txt. Jan. 1997](http://csrc.nist.gov/archive/aes/pre-round1/aes_9701.txt.%20Jan.%201997).
5. NESSIE New European Schemes for Signatures, Integrity, and Encryption. https://www.cosic.esat.kuleuven.be/nessie, 2004.
6. State Service of Special Communication and Information Protection of Ukraine. Statement on Public Competition of Cryptographic Algorithms. http://www.dstszi.gov.ua/dstszi/control/ua/publish/printable article? art id=48387, 2006 (in Ukrainian).
7. Oliynykov Roman, Gorbenko Ivan, Dolgov Victor, Ruzhentsev Victor. Results of Ukrainian National Public Cryptographic Competition. Tatra Mountains Mathematical Publications, 47(1), 99-113. 2009.
8. Roman Oliynykov, Ivan Gorbenko, Oleksandr Kazymyrov, Victor Ruzhentsev, Oleksandr Kuznetsov, Yurii Gorbenko, Oleksandr Dyrda, Viktor Dolgov, Andrii Pushkaryov, Ruslan Mordvinov, Dmytro Kaidalov. DSTU 7624:2014. National Standard of Ukraine. Information technologies. Cryptographic Data Security. Symmetric block transformation algorithm. Ministry of Economical Development and Trade of Ukraine, 2015 (in Ukrainian).
9. Горбенко, І. Д. Перспективний блоковий симетричний шифр «Калина» : основні положення та специфікації /
10. Долгов, В. И. Подстановочные конструкции современ- ных симметричных блочных шифров / В. И. Долгов, Р. В. Олейников, И. В. Лисицкая [та ін.] // Радіоелектронні і комп’ютерні системі. – 2009. – № 6 (40). – С. 89–93.
11. Кузнецов, О. О. Захист інформації в інформаційних систе- мах : монографія / О. О. Кузнецов, С. П. Євсеєв, О. Г. Король. – Х. : Вид-во ХНЕУ, 2010. – 511 с.
12. Казимиров, А. В. Алгебраические свойства схемы развора- чивания ключей блочного симметричного шифра «Кали- на» / А. В. Казимиров, Р. В. Олейников // Радіоелектронні і комп’ютерні системі. – 2010. – № 5 (46). – С. 61–66.

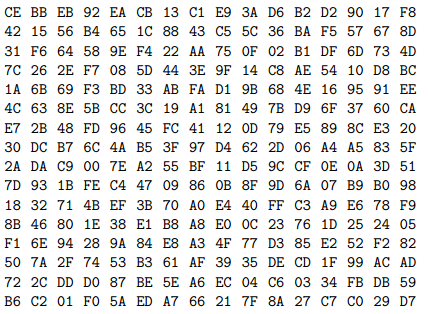
ДОДАТОК А

Матриці S-box для функції заміщення даних.

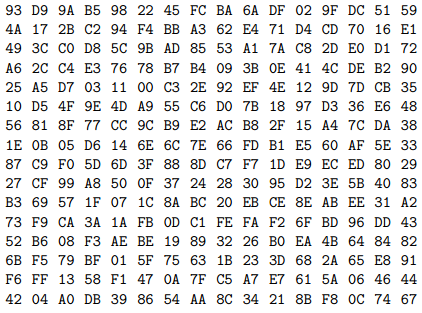
Таблиця



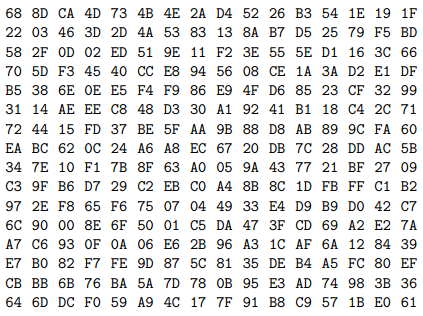
Таблиця



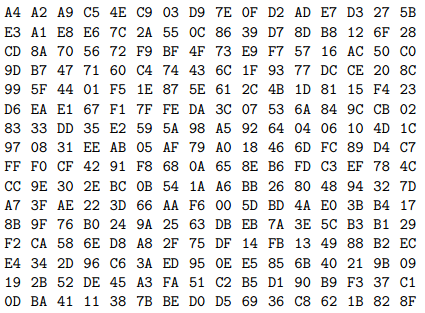
Таблиця



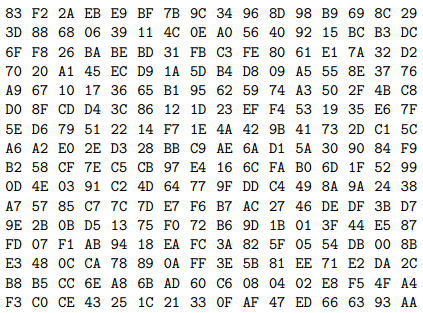
Таблиця



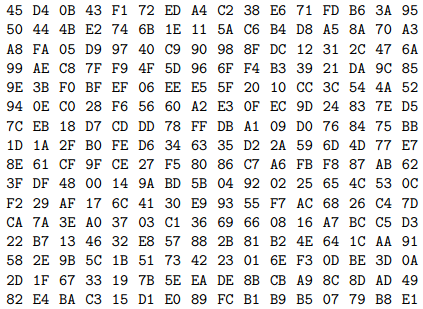
Таблиця



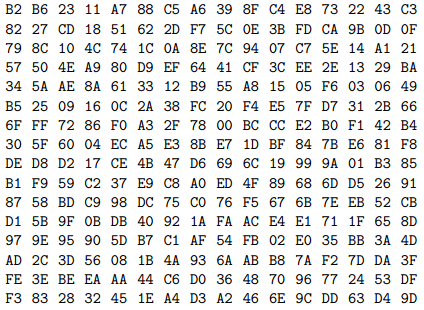
Таблиця



Таблиця



Таблиця



ДОДАТОК Б

Покрокові операції генерації раундових ключів, шифрування та розшифрування.

Етап генерації раундових ключів:

Round keys generation:

Key 504F4E4D4C4B4A494847464544434241

KT generation:

state[0].add\_rkey: 554F4E4D4C4B4A494847464544434241

state[0].s\_box: 16EECBCF8F4E4E4F9CFAC3F442F3D738

state[0].s\_row: 16EECBCF42F3D7389CFAC3F48F4E4E4F

state[0].m\_col: 32403B509B0BF9AB19FC3984D331D601

state[0].xor\_rkey: 620F751DD740B3E251BB7FC197729440

state[0].s\_box: FCF86E791A1A3A6BE77633C670B750B5

state[0].s\_row: FCF86E7970B750B5E77633C61A1A3A6B

state[0].m\_col: C7B1278C0805C1B5335D7D105D207E80

state[0].add\_rkey: 180075D954510BFE7BA4C355A163C0C1

state[0].s\_box: 79CE6E35966302E0F5EFF34811FD52C6

state[0].s\_row: 79CE6E3511FD52C6F5EFF348966302E0

state[0].m\_col: 698B9AD89FA7974F70A52C25626F3987

KT 698B9AD89FA7974F70A52C25626F3987

state[0].ShiftLeft (tmv): 01000100010001000100010001000100

state[0].Rotate (id): 504F4E4D4C4B4A494847464544434241

state[0].add\_rkey (tmv): 6A8B9BD8A0A7984F71A52D25636F3A87

state[0].add\_rkey (kt\_round): BADAEA25ECF2E298B9EC736AA7B27CC8

state[0].s\_box: F1DEE751CF0113A4EC34D6D8A0806096

state[0].s\_row: F1DEE751A0806096EC34D6D8CF0113A4

state[0].m\_col: 339EEA68B823D76E4A1CD2E9E1AABF27

state[0].xor\_rkey (kt\_round): 591571B018844F213BB9FFCC82C585A0

state[0].s\_box: 651C0B6C797E352FAD0C676AFE843F97

state[0].s\_row: 651C0B6CFE843F97AD0C676A797E352F

state[0].m\_col: 75B524F4F5339FABECE7F0B47DA8DC16

state[0].add\_rkey (tmv): 6A8B9BD8A0A7984F71A52D25636F3A87

state[0].add\_rkey (kt\_round): E040C0CD95DB37FA5E8D1DD9E118169D

state[2].ShiftLeft (tmv): 02000200020002000200020002000200

state[2].Rotate (id): 4847464544434241504F4E4D4C4B4A49

state[2].add\_rkey (tmv): 6B8B9CD8A1A7994F72A52E25646F3B87

state[2].add\_rkey (kt\_round): B3D2E31DE5EADB90C2F47C72B0BA85D0

state[2].s\_box: BD2F5879D9C668C3E85A6062B3233FE7

state[2].s\_row: BD2F5879B3233FE7E85A6062D9C668C3

state[2].m\_col: F3826228BBD1C7E62BF35E307369E26A

state[2].xor\_rkey (kt\_round): 9809FEF01A765EA9595670151706D9ED

state[2].s\_box: 673A7464973FE63365191E4AAF132398

state[2].s\_row: 673A7464AF13239865191E4A973FE633

state[2].m\_col: 48349827D4DA836E5A741109F4424F08

state[2].add\_rkey (tmv): 6B8B9CD8A1A7994F72A52E25646F3B87

state[2].add\_rkey (kt\_round): B3C0350076821CBDCD193F2F58B18A8F

state[4].ShiftLeft (tmv): 04000400040004000400040004000400

state[4].Rotate (id): 504F4E4D4C4B4A494847464544434241

state[4].add\_rkey (tmv): 6D8B9ED8A3A79B4F74A53025666F3D87

state[4].add\_rkey (kt\_round): BDDAED25EFF2E598BCEC766AAAB27FC8

state[4].s\_box: AADE06518D0147A494346CD8A6803396

state[4].s\_row: AADE0651A680339694346CD88D0147A4

state[4].m\_col: BEFF2B59C34FE791C1FCD0205C280439

state[4].xor\_rkey (kt\_round): D374B58160E87CDEB559E0053A4739BE

state[4].s\_box: 0D4AFB7E34EC6080E949F64BD5FA3BE2

state[4].s\_row: 0D4AFB7ED5FA3BE2E949F64B34EC6080

state[4].m\_col: C7467CE8EDDFE6480ABB2941DC9F9C16

state[4].add\_rkey (tmv): 6D8B9ED8A3A79B4F74A53025666F3D87

state[4].add\_rkey (kt\_round): 34D21BC1918781977F605967430ED99D

state[6].ShiftLeft (tmv): 08000800080008000800080008000800

state[6].Rotate (id): 4847464544434241504F4E4D4C4B4A49

state[6].add\_rkey (tmv): 718BA2D8A7A79F4F78A534256A6F4187

state[6].add\_rkey (kt\_round): B9D2E91DEBEAE190C8F48272B6BA8BD0

state[6].s\_box: EC2FA779B9C6FFC3FB5AF0628A23E9E7

state[6].s\_row: EC2FA7798A23E9E7FB5AF062B9C6FFC3

state[6].m\_col: 4BD314E75E8A1DA1371A86D7AD37E3D2

state[6].xor\_rkey (kt\_round): 3A58B63FF92D82EE4FBFB2F2C758A255

state[6].s\_box: D5810DDF256DF03B2705CADC7F815748

state[6].s\_row: D5810DDF7F8157482705CADC256DF03B

state[6].m\_col: C972CD621FE2F6677B0962CE4479CB4E

state[6].add\_rkey (tmv): 718BA2D8A7A79F4F78A534256A6F4187

state[6].add\_rkey (kt\_round): 3AFE703AC78A95B6F3AE96F3AEE90CD5

state[8].ShiftLeft (tmv): 10001000100010001000100010001000

state[8].Rotate (id): 504F4E4D4C4B4A494847464544434241

state[8].add\_rkey (tmv): 798BAAD8AFA7A74F80A53C25726F4987

state[8].add\_rkey (kt\_round): C9DAF925FBF2F198C8EC826AB6B28BC8

state[8].s\_box: 05DE3451CA0104A4FB34F0D88A80E996

state[8].s\_row: 05DE34518A80E996FB34F0D8CA0104A4

state[8].m\_col: AA8171A2F5265DABAE113811E56AD554

state[8].xor\_rkey (kt\_round): D30ADB7A5A81FAE42EB4043497059CD3

state[8].s\_box: 0DD668DB66DA21BA9338984070CB3EF7

state[8].s\_row: 0DD668DB70CB3EF79338984066DA21BA

state[8].m\_col: B3076C356281E723EDB471942DF20FA2

state[8].add\_rkey (tmv): 798BAAD8AFA7A74F80A53C25726F4987

state[8].add\_rkey (kt\_round): 2C93170E12298E726E59ADB9A0615929

state[10].ShiftLeft (tmv): 20002000200020002000200020002000

state[10].Rotate (id): 4847464544434241504F4E4D4C4B4A49

state[10].add\_rkey (tmv): 898BBAD8BFA7B74F90A54C25826F5987

state[10].add\_rkey (kt\_round): D1D3011E03EAF990E0F49A72CEBAA3D0

state[10].s\_box: EA74D9F506C634C3AC5A95623D231FE7

state[10].s\_row: EA74D9F53D231FE7AC5A956206C634C3

state[10].m\_col: 28D137477AE4548D560134C5762E2B4E

state[10].xor\_rkey (kt\_round): A15A8D9FC543E3C2C6A478E0F44172C9

state[10].s\_box: 117BEDB203F35893D4EF66CB4E6B05A3

state[10].s\_row: 117BEDB24E6B05A3D4EF66CB03F35893

state[10].m\_col: 26F31CDB53DD7E8DF8E56B94D4BFF2D7

state[10].add\_rkey (tmv): 898BBAD8BFA7B74F90A54C25826F5987

state[10].add\_rkey (kt\_round): B07ED7B4138535DC898AB7BA572F4C5E

round[1].r\_rkey: FA5E8D1DD9E118169CE040C0CD95DB37

round[2].r\_rkey: B3C0350076821CBDCD193F2F58B18A8F

round[3].r\_rkey: BDCD193F2F58B18A8EB3C0350076821C

round[4].r\_rkey: 34D21BC1918781977F605967430ED99D

round[5].r\_rkey: 977F605967430ED99D34D21BC1918781

round[6].r\_rkey: 3AFE703AC78A95B6F3AE96F3AEE90CD5

round[7].r\_rkey: B6F3AE96F3AEE90CD53AFE703AC78A95

round[8].r\_rkey: 2C93170E12298E726E59ADB9A0615929

round[9].r\_rkey: 726E59ADB9A06159292C93170E12298E

round[10].r\_rkey: B07ED7B4138535DC898AB7BA572F4C5E

Етап шифрування:

round[1].s\_box: 9638AD0DCB389479F6D17436193AA333

round[1].s\_row: 9638AD0D193AA333F6D17436CB389479

round[1].m\_col: 566E199501A3D4FE1C8FFE90C2B0ADA6

round[1].xor\_rkey: AC309488D842CCE8806FBE500F257691

round[2].s\_box: 2D7C5005C4694B0B9B20DD3109F46C9F

round[2].s\_row: 2D7C500509F46C9F9B20DD31C4694B0B

round[2].m\_col: 4CFC80357A697A36C16348F85DD772E9

round[2].xor\_rkey: FF3CB5350CEB668B0C7A77D70566F866

round[3].s\_box: 8054FBCC1703B977172D7E5C75FC8C5F

round[3].s\_row: 8054FBCC75FC8C5F172D7E5C1703B977

round[3].m\_col: 770756810DE3A5F486D8353658A2C3F8

round[3].xor\_rkey: CACA4FBE22BB147E086BF50358D441E4

round[4].s\_box: C1D335E2A37694AC71E5864DC253A5BA

round[4].s\_row: C1D335E2C253A5BA71E5864DA37694AC

round[4].m\_col: 36108555928CC80964D64A58D0B50987

round[4].xor\_rkey: 02C29E94030B499E1BB6133F93BBD01A

round[5].s\_box: 5F94402906B2EFC1FDB8C2DF3A766BB7

round[5].s\_row: 5F9440293A766BB7FDB8C2DF06B2EFC1

round[5].m\_col: B1D0CD13C56DD97158F735242262FE9C

round[5].xor\_rkey: 26AFAD4AA22ED7A8C5C3E73FE3F3791D

round[6].s\_box: 9AF9EED60173634903287FDF10F0FD79

round[6].s\_row: 9AF9EED610F0FD7903287FDF01736349

round[6].m\_col: 6E690AF2F08ACF9058D4EB5E1A02FCD8

round[6].xor\_rkey: 54977AC837005A26AB7A7DADB4EBF00D

round[7].s\_box: 9686B196EECE189E742DAFD0AE03421E

round[7].s\_row: 9686B196AE03421E742DAFD0EECE189E

round[7].m\_col: 71DC819D4133FD8617858613A25B767C

round[7].xor\_rkey: C72F2F0BB29D148AC2BF7863989CFCE9

round[8].s\_box: 7F4D72B3CEB99443E80566FD6707F895

round[8].s\_row: 7F4D72B36707F895E80566FDCEB99443

round[8].m\_col: 9722A3DF13A7B090A62E06363B20109B

round[8].xor\_rkey: BBB1B4D1018E3EE2C877AB8F9B4149B2

round[9].s\_box: 99461AB0433DB26BFB978E09DE6BEF00

round[9].s\_row: 99461AB0DE6BEF00FB978E09433DB26B

round[9].m\_col: 73B1868D4823A840CDCECBAE224F6C71

round[9].xor\_rkey: 01DFDF20F183C919E4E258B92C5D45FF

round[10].s\_box: 43AD91585400268AD0DDD04749370061

round[10].s\_row: 43AD915849370061D0DDD0475400268A

round[10].m\_col: 00DE8F5859609144873DE1855197367B

round[10].add\_rkey: B15D670C6CE5C72010C8993FA8C682D9

Етап розшифрування:

round[10].sub\_rkey: 00DE8F5859609144873DE1855197367B

round[10].m\_col: 43AD915849370061D0DDD0475400268A

round[10].s\_row: 43AD91585400268AD0DDD04749370061

round[10].s\_box: 01DFDF20F183C919E4E258B92C5D45FF

round[9].xor\_rkey: 73B1868D4823A840CDCECBAE224F6C71

round[9].m\_col: 99461AB0DE6BEF00FB978E09433DB26B

round[9].s\_row: 99461AB0433DB26BFB978E09DE6BEF00

round[9].s\_box: BBB1B4D1018E3EE2C877AB8F9B4149B2

round[8].xor\_rkey: 9722A3DF13A7B090A62E06363B20109B

round[8].m\_col: 7F4D72B36707F895E80566FDCEB99443

round[8].s\_row: 7F4D72B3CEB99443E80566FD6707F895

round[8].s\_box: C72F2F0BB29D148AC2BF7863989CFCE9

round[7].xor\_rkey: 71DC819D4133FD8617858613A25B767C

round[7].m\_col: 9686B196AE03421E742DAFD0EECE189E

round[7].s\_row: 9686B196EECE189E742DAFD0AE03421E

round[7].s\_box: 54977AC837005A26AB7A7DADB4EBF00D

round[6].xor\_rkey: 6E690AF2F08ACF9058D4EB5E1A02FCD8

round[6].m\_col: 9AF9EED610F0FD7903287FDF01736349

round[6].s\_row: 9AF9EED60173634903287FDF10F0FD79

round[6].s\_box: 26AFAD4AA22ED7A8C5C3E73FE3F3791D

round[5].xor\_rkey: B1D0CD13C56DD97158F735242262FE9C

round[5].m\_col: 5F9440293A766BB7FDB8C2DF06B2EFC1

round[5].s\_row: 5F94402906B2EFC1FDB8C2DF3A766BB7

round[5].s\_box: 02C29E94030B499E1BB6133F93BBD01A

round[4].xor\_rkey: 36108555928CC80964D64A58D0B50987

round[4].m\_col: C1D335E2C253A5BA71E5864DA37694AC

round[4].s\_row: C1D335E2A37694AC71E5864DC253A5BA

round[4].s\_box: CACA4FBE22BB147E086BF50358D441E4

round[3].xor\_rkey: 770756810DE3A5F486D8353658A2C3F8

round[3].m\_col: 8054FBCC75FC8C5F172D7E5C1703B977

round[3].s\_row: 8054FBCC1703B977172D7E5C75FC8C5F

round[3].s\_box: FF3CB5350CEB668B0C7A77D70566F866

round[2].xor\_rkey: 4CFC80357A697A36C16348F85DD772E9

round[2].m\_col: 2D7C500509F46C9F9B20DD31C4694B0B

round[2].s\_row: 2D7C5005C4694B0B9B20DD3109F46C9F

round[2].s\_box: AC309488D842CCE8806FBE500F257691

round[1].xor\_rkey: 566E199501A3D4FE1C8FFE90C2B0ADA6

round[1].m\_col: 9638AD0D193AA333F6D17436CB389479

round[1].s\_row: 9638AD0DCB389479F6D17436193AA333

round[1].s\_box: 54B4262213B4141DBE48FEEF390917A9

round[0].sub\_rkey: 747365547DD8DC235FBBE11557F1010C