НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
НАВЧАЛЬНО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Лабораторна робота №3
З дисципліни
«Операційні системи»

Виконав:
студент групи КА-65
Іванов Д. С.
Перевірив:
Коваленко А.Є.

Київ 2017

Засоби захисту і політики безпеки операційних систем

Мета. Основна цілі роботи:

1. навчитись використовувати програмні засоби операційних систем для шифрування і дешифрування даних

2. ознайомитись з особливостями налагодження політики безпеки операційних систем

3. ознайомитись з класифікацією і порядком застосування антивірусних програм; набути досвіду налагодження і використання антивірусних програм.

Завдання:

1. Побудувати ряд файлів з визначенними розмірами (10, 20, 30, 40, 50, 60, 80)МБ
2. Зашифрувати створені файли, визначивши час шифрування (системний, в області користувача). Підрахувати сумарний час.
3. Отримані зашифровані дані записати у файли шифрованих данних.
4.Виконати дешифрування зашифрованих даних, визначивши час шифрування (системний, в області користувача).
5. Відобразити у звіті послідовність виконання дій і отримані результати у вигляді тіблиці.
6.Описати основні характеристики алгоритмів шифрування (довжина ключа, режим, час створення. Використання).
7.Побудувати гістограму часу шифрування і дешифрування для двох алгоритмів.
8.Зробити висновки відноно продуктивності алгоритмів шифрування за даними шифрування і дешифрування (системному, в області користвача, загальний час).
9.Оформити звіт до роботи.
10. Відповісти на контрольні питання та поясники параметри алгоритмів.

AES

Advanced Encryption Standard (AES), також відомий під назвою Rijndael — симетричний алгоритм блочного шифрування (розмір блока 128 біт, ключ 128/192/256 біт), фіналіст конкурсу AES і прийнятий як американський стандарт шифрування урядом США. Вибір припав на AES з розрахуванням на широке використання і активний аналіз алгоритму, як це було із його попередником, DES. Державний інститут стандартів і технологій (англ. National Institute of Standards and Technology, NIST) США опублікував попередню специфікацію AES 26 жовтня 2001 року, після п'ятилітньої підготовки. 26 травня 2002 року AES оголошено стандартом

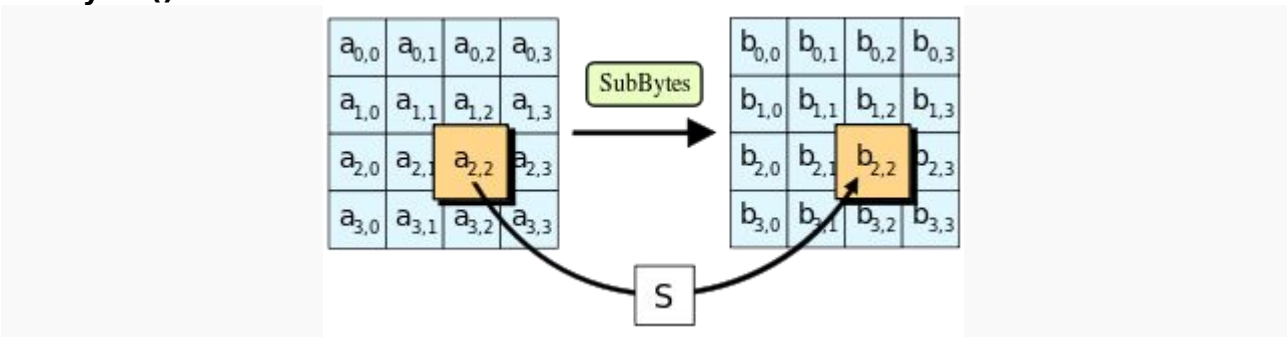
шифрування. Станом на 2009 рік AES є одним із найпоширеніших алгоритмів симетричного шифрування.[1]

В принципі, алгоритм, запропонований Рейменом і Дейцменом, і AES не одне і те ж. Алгоритм *Рейндол*[3] підтримує широкий діапазон розміру блоку та ключа. AES має фіксовану довжину у 128 біт, а розмір ключа може приймати значення 128, 192 або 256 біт. В той час як Рейндол підтримує розмірність блоку та ключа із кроком 32 біт у діапазоні від 128 до 256. Через фіксований розмір блоку AES оперує із масивом 4×4 байт, що називається *станом* (версії алгоритму із більшим розміром блоку мають додаткові колонки).

Для ключа 128 біт алгоритм має 10 раундів у яких послідовно виконуються операції

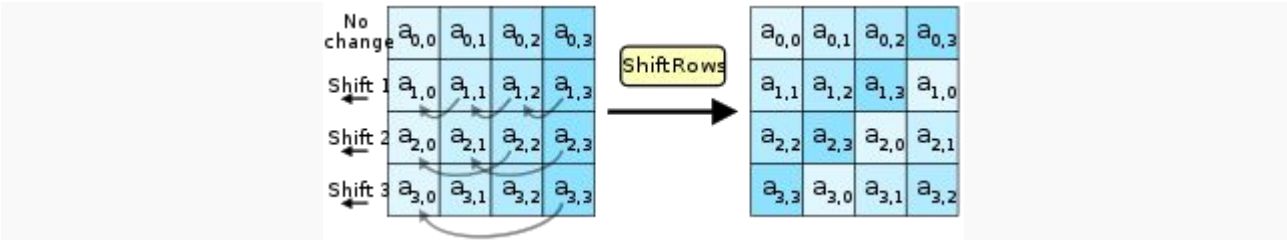
subBytes()
shiftRows()
mixcolumns() (у 10-му раунді пропускається)
xorRoundKey()

SubBytes()



Процедура SubBytes() обробляє кожен байт стану незалежно, проводячи нелінійну заміну байтів використовуючи таблицю замін (S-box). Така операція забезпечує нелінійність алгоритму шифрування.

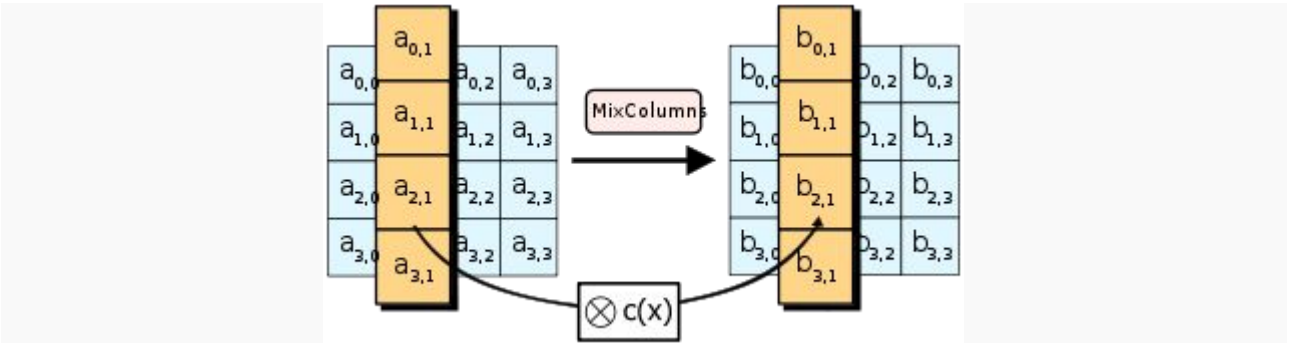
ShiftRows()



ShiftRows працює з рядками таблиці State. При цій трансформації рядка стану циклічно зсуваються на r байтів по горизонталі, залежно від номера рядка. Для нульового рядка r = 0, для першого рядка r = 1 і т. д. Таким чином кожна колонка вихідного стану після застосування процедури ShiftRows складається з байтів з кожної колонки початкового стану. Для алгоритму Rijndael патерн зсуву рядків для 128 - і 192-бітних рядків однаковий. Однак для блоку розміром 256 біт відрізняється від попередніх тим, що 2, 3, і 4-і рядки зміщуються на 1, 3, і 4 байти, відповідно.

Фактично це проста перестановка байтів таблиці 4x4 State.

MixColumns()



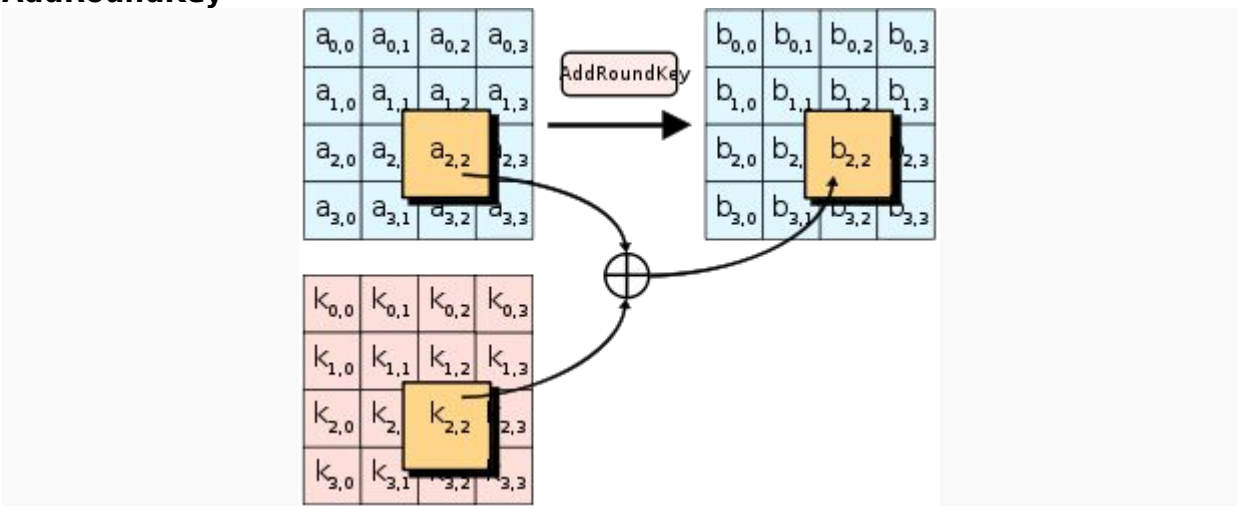
У процедурі MixColumns , чотири байти кожної колонки State змішуються, використовуючи для цього зворотну лінійну трансформацію. MixColumns опрацьовує стан по колонках, трактуючи кожну з них як поліном четвертого степеня. Над цими

поліномами виконується множення в по модулю на фіксований

многочлен . Разом з ShiftRows , MixColumns вносить дифузію в шифр.

Під час цієї операції, кожен стовпчик множиться на матрицю, яка для 128-бітного ключа має вигляд

AddRoundKey



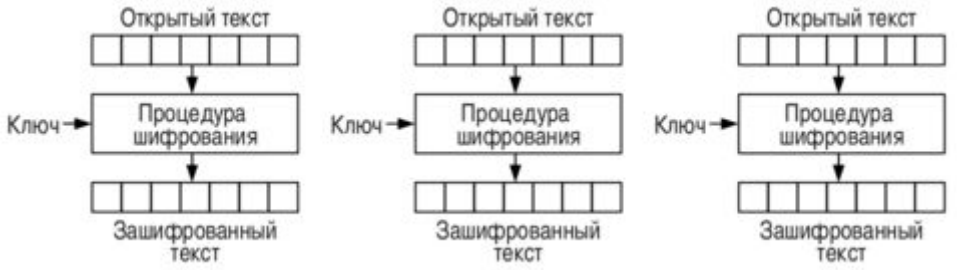
У процедурі AddRoundKey , RoundKey кожного раунду об'єднується зі State. Для кожного раунду Roundkey виходить з CipherKey використовуючи процедуру KeyExpansion ; кожен RoundKey такого ж розміру, що і State. Процедура виробляє побітовий XOR кожного байта State з кожним байтом RoundKey .

Фактично це звичайний побайтовий XOR байт ключа з байтами таблиці State.

Electronic code book (ECB)

В ГОСТ 28147—89 этот режим называется **режимом простой замены**.

Зашифрование:



Особенности:

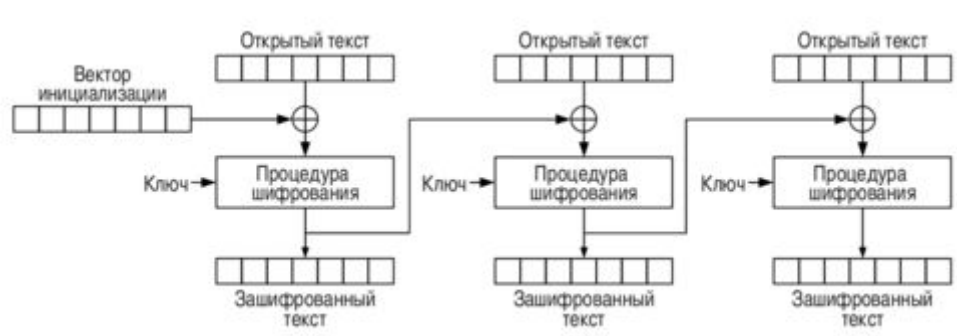
каждый блок шифруется/расшифровывается независимо от других блоков.

Достоинства ECB:

постоянная скорость обработки блоков (скорость определяется эффективностью реализации шифра);
возможно распараллеливание вычислений (так как блоки не связаны между собой).

Cipher block chaining (CBC)

Для шифрования некоторого сообщения выполняются следующие действия[4].



Недостатки CBC:

возможность определения при изменении шифротекста;
возможность изменения открытого текста при перемещении[блоков;
возможность изменения блока шифротекста {\displaystyle C_{i-1}} путём изменения блока сообщения {\displaystyle P_{i}} ;
невозможность распараллеливания шифрования (поскольку для шифрования каждого i-го блока требуется блок, зашифрованный на предыдущем шаге (блоки связаны между собой))[1].

Достоинства CBC:

постоянная скорость обработки блоков (скорость определяется эффективностью реализации шифра; время выполнения операции «хог» пренебрежимо мало);
отсутствие статистических особенностей, характерных для режима ECB (поскольку каждый блок открытого текста «смешивается» с блоком шифротекста, полученном на предыдущем шаге шифрования);
возможность распараллеливания расшифровки[1].

Blowfish

Blowfish-криптографічний алгоритм, який реалізує блочне симетричне шифрування.

Розроблений Брюсом Шнайєром в 1993 році. Являє собою шифр на основі мережі Фейстеля. Виконано на простих і швидких операціях: XOR, підстановка, додавання. Не запатентований і вільно поширюваний.

Blowfish зарекомендував себе, як надійний алгоритм, тому реалізований у багатьох програмах, де не потрібна часта зміна ключа і необхідна висока швидкість шифрування / розшифрування.

Хешування паролів
Захист електронної пошти і файлів
GnuPG (безпечне зберігання і передача)
В лініях зв'язку: зв'язка ElGamal (не запатентований) або RSA (дія патенту закінчилося в 2000 році) і Blowfish замість IDEA
В маршрутизаторі Intel Express 8100 з ключем довжиною 144 біта
Забезпечення безпеки в протоколах мережного і транспортного рівня
PuTTY (мережевий рівень)
SSH (транспортний рівень)
OpenVPN (створення зашифрованих каналів).

Алгоритм Blowfish[ред. • ред. код]
Розділений на 2 етапи:

Підготовчий — формування ключів шифрування по секретному ключу.
Ініціалізація масивів P і S за допомогою секретного ключа K
Ініціалізація P1-P18 фіксованим рядком, що складається з шістнадцяткових цифр мантиси числа пі.
Проводиться операція XOR над P1 з першими 32 бітами ключа K, над P2 з другими 32-бітами і так далі.
Якщо ключ K коротше, то він накладається циклічно.
Шифрування ключів і таблиць замін
Алгоритм шифрування 64-бітного блоку, використовуючи початкові ключі P1-P18 і таблицю замін S1-S4, шифрує 64 бітну нульовий (0x0000000000000000) рядок. Результат записується в P1, P2.

P1 і P2 шифруються зміненими значеннями ключів і таблиць замін. Результат записується в P3 і P4.
Шифрування триває до зміни всіх ключів P1-P18 і таблиць замін S1-S4.
Шифрування тексту отриманими ключами і F(x), з попереднім розбиттям на блоки по 64 біти. Якщо неможливо розбити початковий текст точно на блоки по 64 біти, використовуються різні режими шифрування для побудови повідомлення, що складається з цілого числа блоків. Сумарні необхідна пам'ять 4168 байт: P1-P18: 18 змінних по 32 біта; S1-S4: 4x256 змінних по 32 бита.
Розшифрування відбувається аналогічно, тільки P1-P18 застосовуються у зворотному порядку.

Завдання 1

```
daniel@daniel-Aspire-E3-112 ~ $ cd ivanov/lab3/
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ nano com31
yes "hehe" | head -c $[$1*10485760]>$2

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ chmod a+x com31
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ PATH=$PATH:~/ivanov/lab3

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 1 f1
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 2 f2
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 3 f3
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 4 f4
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 5 f5
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 6 f6
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 7 f7
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ com31 8 f8
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ ls -l

total 368644
-rwxrwxr-x 1 daniel daniel       40 Nov 28 11:41 com31
-rw-rw-r-- 1 daniel daniel 10485760 Nov 28 11:43 f1
-rw-rw-r-- 1 daniel daniel 20971520 Nov 28 11:43 f2
-rw-rw-r-- 1 daniel daniel 31457280 Nov 28 11:43 f3
-rw-rw-r-- 1 daniel daniel 41943040 Nov 28 11:43 f4
-rw-rw-r-- 1 daniel daniel 52428800 Nov 28 11:44 f5
-rw-rw-r-- 1 daniel daniel 62914560 Nov 28 11:44 f6
-rw-rw-r-- 1 daniel daniel 73400320 Nov 28 11:44 f7
-rw-rw-r-- 1 daniel daniel 83886080 Nov 28 11:44 f8
```

2. Зашифрувати створені файли, визначивши час шифрування (системний, в області користувача). Підрахувати сумарний час.

```
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f1 -out ff1

real    0m0.413s
```

user 0m0.268s
sys 0m0.032s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f2 -out ff2

real 0m0.589s
user 0m0.516s
sys 0m0.064s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f3 -out ff3

real 0m0.938s
user 0m0.804s
sys 0m0.072s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f4 -out ff4

real 0m1.153s
user 0m1.020s
sys 0m0.124s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f5 -out ff5

real 0m1.439s
user 0m1.216s
sys 0m0.208s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f6 -out ff6

real 0m1.715s
user 0m1.492s
sys 0m0.216s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -aes-192-ecb -k hehe -in f8 -out ff8

real 0m2.283s
user 0m2.020s
sys 0m0.256s
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $

Далі використаемо bf-cbc

```
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f1 -out f1f1

real 0m0.178s
user 0m0.140s
sys 0m0.028s
```

```
daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f2 -out f1f2

real       0m0.353s
user       0m0.288s
sys        0m0.048s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f3 -out f1f3

real       0m0.501s
user       0m0.380s
sys        0m0.116s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f4 -out f1f4

real       0m0.699s
user       0m0.508s
sys        0m0.144s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f5 -out f1f5

real       0m0.821s
user       0m0.688s
sys        0m0.120s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f6 -out f1f6

real       0m0.979s
user       0m0.744s
sys        0m0.220s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -bf-cbc -k hehe
-in f8 -out f1f8
```

```
real    0m1.282s
user    0m1.000s
sys     0m0.276s
```

4. Виконати дешифрування зашифрованих даних, визначивши час шифрування (системний, в області користувача)

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff1 -out d1

real 0m0.458s
user 0m0.312s
sys 0m0.056s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff2 -out d2

real 0m0.716s
user 0m0.620s
sys 0m0.080s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff3 -out d3

real 0m1.058s
user 0m0.892s
sys 0m0.152s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff4 -out d4

real 0m1.410s
user 0m1.228s
sys 0m0.168s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff5 -out d5

real 0m1.748s
user 0m1.556s
sys 0m0.184s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff6 -out d6

real 0m2.127s
user 0m1.856s
sys 0m0.236s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -aes-192-ecb -k hehe -in ff8 -out d8

real 0m2.808s
user 0m2.448s
sys 0m0.328s

Тепер для bf-cbc

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f1 -out dd1

real 0m0.179s
user 0m0.128s
sys 0m0.048s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f2 -out dd2

real 0m0.350s
user 0m0.260s
sys 0m0.080s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f3 -out dd3

real 0m0.529s
user 0m0.416s
sys 0m0.092s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f4 -out dd4

real 0m0.663s
user 0m0.488s
sys 0m0.172s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f5 -out dd5

real 0m0.829s
user 0m0.640s
sys 0m0.180s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f6 -out dd6

real 0m0.989s
user 0m0.756s
sys 0m0.228s

daniel@daniel-Aspire-E3-112 ~/ivanov/lab3 $ time openssl enc -d -bf-cbc -k hehe -in f1f8 -out dd8

real 0m1.366s
user 0m1.016s
sys 0m0.292s

Порівняння алгоритмів:

Алгоритм aes-192-ecb

Шифрування				Розшифрування			
Розмір файлу	sys	user	сума	Розмір файлу	sys	user	сума
10	0m0.032s	0m0.268s	0m0.413s	10	0m0.056s	0m0.312s	0m0.458s
20	0m0.064s	0m0.516s	0m0.589s	20	0m0.080s	0m0.620s	0m0.716s
30	0m0.072s	0m0.804s	0m0.938s	30	0m0.152s	0m0.892s	0m1.058s
40	0m0.124s	0m1.020s	0m1.153s	40	0m0.168s	0m1.228s	0m1.410s
50	0m0.208s	0m1.216s	0m1.439s	50	0m0.184s	0m1.556s	0m1.748s
60	0m0.216s	0m1.492s	0m1.715s	60	0m0.236s	0m1.856s	0m2.127s
70	0m0.216s	0m1.772s	0m1.997s	70	0m0.244s	0m2.188s	0m2.450s
80	0m0.256s	0m2.020s	0m2.283s	80	0m0.328s	0m2.448s	0m2.808s

Алгоритм bf-cbc

Шифрування				Розшифрування			
Розмір файлу	sys	user	сума	Розмір файлу	sys	user	сума
10	0m0.028s	0m0.140s	0m0.178s	10	0m0.048s	0m0.128s	0m0.179s
20	0m0.056s	0m0.272s	0m0.335s	20	0m0.080s	0m0.260s	0m0.350s
30	0m0.116s	0m0.380s	0m0.501s	30	0m0.092s	0m0.416s	0m0.529s
40	0m0.144s	0m0.508s	0m0.699s	40	0m0.172s	0m0.488s	0m0.663s
50	0m0.120s	0m0.688s	0m0.821s	50	0m0.180s	0m0.640s	0m0.829s
60	0m0.220s	0m0.744s	0m0.979s	60	0m0.228s	0m0.756s	0m0.989s
70	0m0.268s	0m0.852s	0m1.132s	70	0m0.248s	0m0.896s	0m1.171s
80	0m0.276s	0m1.000s	0m1.282s	80	0m0.292s	0m1.016s	0m1.366s

Як бачимо, bf-cbc показує набагато кращі результати, ніж aes-192-ecb при шифруванні та дешифруванні.

Контрольні питання по роботі:

Пояснити основні принципи шифрування і дешифрування даних на основі типових програмних засобів

При симетричному шифруванні використовується один ключ: секретне ключове слово, пароль або парольну фразу. Довжина ключа визначає захищеність даних від спроб зловмисників заволодіти секретними даними. В операційній системі Ubuntu для симетричного шифрування можна застосовувати пакет OpenSSL, який не потребує ліцензування.

Як задати ключ у даному алгоритмі шифрування.

Пароль можна вказати у самій команді, наприклад:
Openssl –e des3 –k password –in 1 –out e1
Де –k password – вказує на пароль
Або –k file<file>: перший рядок у вказаному файлі є паролем.

Пояснити призначення і значення опцій заданого алгоритму шифрування.

-e :Шифрування
-d :Дешифрування
-in<file>: Вхідний файл
-out<file>: Файл, який отримують
-k<text>: Наступний аргумент є паролем
–k file<file>: перший рядок у вказаному файлі є паролем.

В чому полягає реалізація політики безпеки з використанням операційних систем?

Безпека ОС базується на двох ідеях:

1. ОС надає прямий чи непрямий доступ до ресурсів на кшталт файлів на локальному диску, привілейованих системних викликів, особистої інформації про користувачів та служб, представлених запущеними програмами;

2. ОС може розділити запити ресурсів від авторизованих користувачів, дозволивши доступ, та неавторизованих, заборонивши його.

Запити, в свою чергу, також діляться на два типи:

1. Внутрішня безпека — вже запущені програми. На деяких системах програма, оскільки вона вже запущена, не має ніяких обмежень, але все ж типово вона має ідентифікатор, котрий використовується для перевірки запитів до ресурсів.

2. Зовнішня безпека — нові запити з-за меж комп'ютера, як наприклад реєстрація з консолі чи мережеве з'єднання. В цьому випадку відбувається процес авторизації за допомогою імені користувача та паролю, що його підтверджує, чи інших способів як наприклад магнітні картки чи біометричні дані.

Висновок:

В ході лабораторної роботи я навчився використовувати програмні засоби операційних систем для шифрування і дешифрування даних; ознайомився з особливостями налагодження політики безпеки операційних систем, ознайомився з класифікацією і порядком застосування антивірусних програм;