Ходаковський Олександр Сергійович,
спеціаліст системотехнік,
магістрант інформаційних технологій

Криптографічний захист інформації

ІН 11М

Комп`ютерний набір в редакторі Microsoft® Office® Word 2003 О.С Ходаковський.

Редагування, верстка , макетування та дизайн Р.М. Літнарович.

Науковий керівник Р.М. Літнарович, доцент, кандидат технічних наук

Міжнародний Економіко-Гуманітарний Університет ім.. акад.. Степана Дем`янчука

Кафедра математичного моделювання
33027, м.Рівне, Україна
Вул.акад. С.Дем`янчука, 4, корпус 1
Телефон(+00380) 362 23-73-09
Факс(+00380) 362 23-01-86
E-mail:mail@regi.rovno.ua
E-mail:Al_Dominator@mail.ru

Міністерство освіти і науки, молоді та спорту України

Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука

О.С. Ходаковський

Криптографічний захист інформації

Науковий керівник: Р. М. Літнарович,
канд.. техн. наук доцент

Рівне – 2012 р.

УДК 614.2

Ходаковський О.С. Криптографічний захист інформації. Науковий керівник Р.М. Літнарович. МЕГУ. Рівне, 2012. – 108 с. Khodakovskiy O.S. Cryptographic priv. Scientific leader R.M. Litnarovich. IEGU. Rivne, 2012.-108 p.

Рецензенти: В.О. Боровий, доктор техн. наук професcор
В.Г. Бурачек, доктор техн. наук професcор
Є.С. Парняков, доктор техн. наук професcор
Відповідальний за випуск:
Й.В. Джунь, доктор фіз.-мат. наук, професcор

Послідовно викладаються основні поняття шифрування у сучасних технологіях інформаційної безпеки з криптлогії.

Монографія містить актуальний матеріал довідково-аналітичного характеру, призначеного для студентів спеціальності "Інформаційні системи та технології" для вивчення дисципліни "Криптологія".

Ключові слова: інформація, захист, криптологія, безпека.

Последовательно рассматриваются основные понятия шифрования в современных технологиях компьютерной безопасности, в криптологии.

Монография содержит актуальный материал справочно-аналитического характера, предназначенного для студентов специальности «Информационные системы и технологии» для изучения дисциплины «Криптология».

Ключевые слова: информация, защита, криптология, безопасность.

The basic concepts of encipherement in modern technologies of computer safety are consistently examined, in kriptologii.

A monograph contains aktual'nyy material of certificate-analytical character, intended for the students of speciality the «Informative systems and technologies» for the study of discipline «Kriptologiya».

Keywords: information, defence, kriptology, safety.

13. Введение в криптографию/Под общей ред. В.В. Ященко. — С-Пб.: Питер, 2001. — 288 с.
14. Всемирная история шпионажа/Авт.-сост. М.И. Ушаков. — М.: Олимп; ООО “Фирма «Издательство АСТ»”, 2000. — 496 с.
15. Домарев В.В. Безопасность информационных технологий. Методология создания систем защиты. — К.: ООО “ДС”, 2001. — 688 с.
16. Защита программного обеспечения. Пер. с англ./Д. Гроувер, Р. Сатер, Дж. Фипс и др.; Под ред. Д. Гроувера. — М.: Мир, 1992. — 285 с.
17. Зегжда Д.П., Ивашко А.М. Основы безопасности информационных систем. — М.: Горячая линия — Телеком, 2000. — 452 с.
18. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях/Под ред. В.Ф. Шаньгина; 2-е изд., перераб. и доп. — М.: Радио и связь, 2001. — 376 с.

Список використаної літератури

1. Жельников В. Криптография от папируса до компьютера. — М.: ABF, 1996. — 336 с.
2. Вербицький О.В. Вступ до криптології. — Львів: Видавництво науково-технічної літератури, 1998. — 248 с.
3. Вєртузаев М.С., Юрченко О.М. Захист інформації в комп'ютерних системах від несанкціонованого доступу: Навч. посібник/За ред. С.Г. Лаптєва. — К.: Видавництво Європейського університету, 2001. — 201 с.
4. Герасименко В.А., Малюк А.А. Основы защиты информации. — М.: МГИФИ, 1997. — 538 с.
5. Дориченко С.А., Ященко В.В. 25 этюдов о шифрах. — М.: ТЕИС, 1994. — 69 с.
6. Жельников В. Криптография от папируса до компьютера. — М.: ABF, 1996. — 336 с.
7. Организация и современные методы защиты информации/Под общ. ред. С.А. Диева, А.Г. Шаваева. — М.: Коцерн “Банковский Деловой Центр”, 1998. — 472 с.
8. Полмар Н., Аллен Т.Б. Энциклопедия шпионажа/Пер. сангл. В. Смирнова. — М.: КРОН-ПРЕСС, 1999. — 816 с.
9. Росоловський В.М., Анкудович Г.Г., Катерноза К.О., Шевченко М.Ю. Основи інформаційної безпеки автоматизованої інформаційної системи державної податкової служби України: Навч. посібник/За ред. М.Я. Азарова. — Ірпінь: Академія ДПС України, 2003. — 466 с.
10. Хорев А.А. Способы и средства защиты информации. — М.: МО РФ, 2000. — 316 с.
11. Барабаш А.В., Шанкин г.п. История криптографии. Ч.l. — М.: Гелиос АРВ, 2002. — 240 с.
12. Болдырев А.И., Василевский И.В., Сталенков С.Е. Методические рекомендации по поиску и нейтрализации средств негласного съема информации. Практическое пособие. — М.: НЕЛК, 2001. — 138 с.

Ходаковський Олександр Сергійович, спеціаліст системотехнік, магістрант інформаційних технологій

Зміст

Закінчення лістингу 4

```
END SELECT

 FOR j% = 1 to 4
        mbits(((i% - 1) * 4) + j%)= ASC(MID$(NIBBLE$,
j%, 1)) - 48
    NEXT j%
  NEXT i%
END SUB
```

Інші режими використання алгоритму шифрування DES

Крім режиму ECB, алгоритм DES може використовуватися в ***режимі зчеплення блоків шифртекста*** (CBC — Cipher B1ock Chaining). Суть цього режиму полягає в тому, що повідомлення розбивається на блоки по 64 бітів, і їх послідовність зашифровується. Перед шифруванням (у режимі ECB), блок відкритого тексту порозрядний складається з попереднім блоком шифртекста. Для шифрування першого блоку шифртекста потрібний так званий *вектор ініціалізації* (IV — initialization vector). Останній не є секретним. Даний режим не дозволяє накопичуватися помилкам при передачі, оскільки помилка при передачі приведе до втрати тільки двох блоків початкового тексту. Окрім ECB і CBC, існують також режими ***шифрування із зворотним зв'язком*** (Cfв — Cipher Feedback) і ***шифрування із зовнішнім зворотним зв'язком*** (Ofb — Output Feedback).

Продовження лістингу 4

```
SELECT CASE X$

CASE "0"
     NIBBLE$ = "0000"
    CASE "1"
     NIBBLE$ = "0001"
    CASE "2"
     NIBBLE$ = "0010"
    CASE "3"
     NIBBLE$ = "0011"
    CASE "4"
     NIBBLE$ = "0100"
    CASE "5"
     NIBBLE$ = "0101"
    CASE "6"
     NIBBLE$ = "0110"
    CASE "7"
     NIBBLE$ = "0111"
    CASE "8"
     NIBBLE$ = "1000"
    CASE "9"
     NIBBLE$ = "1001"
    CASE "A"
     NIBBLE$ = "1010"
    CASE "B"
     NIBBLE$ = "1011"
    CASE "C"
     NIBBLE$ = "1100"
    CASE "D"
     NIBBLE$ = "1101"
    CASE "E"
     NIBBLE$ = "1110"
    CASE "F"
     NIBBLE$ = "1111"
    CASE ELSE
     Print "Не є 16-ричним значенням!"
     SYSTEM
```


Вступ

До недавнього часу криптографія представляла інтерес головним чином для військових і дипломатів. Приватні особи і комерційні організації рідко вважали за необхідне удаватися до шифрування для захисту своєї кореспонденції, а якщо і вдавалися, то, як правило, без достатньої ретельності. Проте через цілий ряд обставин інтерес до застосування криптографії різко підвищився.

В даний час кількість областей, в яких засоби електронного зв'язку замінюють паперове листування, швидко збільшується. В результаті збільшується і доступний для перехоплення об'єм інформації, а саме перехоплення стає більш легким. Проте ті ж самі чинники, які сприяють розповсюдженню електронних засобів зв'язку, помітно знижують витрати на криптографію.

У разі передачі даних електронною поштою перехоплення виявляється навіть ще більш легким, ніж у разі телефонного зв'язку, оскільки при телефонному зв'язку перехоплювач не має можливості розрізняти зміст повідомлення, якщо тільки не використовується людина-спостерігач. При передачі даних матеріал знаходиться у формі, придатній для сприйняття обчислювальною машиною, і подібних обмежень не виникає.

Вартість перехоплення з часом все більш зменшується. В результаті цього зростає інтерес до криптографії як у приватних осіб, так і у комерційних організацій. Особливу гостроту проблема криптографічного

захисту придбала з бурхливим розвитком електронної пошти і систем електронних платежів.

Основні поняття

Криптографія — наука про методи перетворення (шифрування) інформації з метою її захисту від зловмисників.

Інформація — основне поняття наукових напрямів, що вивчають процеси передачі, переробки і зберігання різних даних. Суть поняття інформації зазвичай пояснюється на прикладах. Формальне визначення дати дуже складно, оскільки поняття інформації відноситься до таких же фундаментальних понять, як матерія.

Інформація, яка потребує захисту, виникає в самих різних життєвих ситуаціях. Зазвичай в таких випадках говорять, що інформація містить таємницю або є такою, що захищається. Для найбільш типових ситуацій, що часто зустрічаються, введені спеціальні поняття: *державна таємниця*, *військова таємниця*, *комерційна таємниця*, *юридична таємниця*, *лікарська таємниця* і так далі

Причому, коли говорять про інформацію, що захищається, мають на увазі наступні ознаки такої інформації:

- є певний круг законних користувачів, які мають право володіти цією інформацією;
- є незаконні користувачі, які прагнуть оволодіти цією інформацією.

Шифр — спосіб (метод), перетворення інформації з метою її захисту від незаконних користувачів.

Стеганографія — набір засобів і методів захования факту передачі повідомлення.

Стеганографія приховує сам факт передачі повідомлення, а криптографія вважає, що повідомлення (у шифрованому вигляді) доступне незаконному користувачеві, але він не може витягувати з цього повідомлення інформацію, що захищається.

Перші сліди стеганографічних методів втрачаються в глибокій старовині. Відомий такий спосіб захования письмового повідомлення: рабу голили голову, на шкірі писали повідомлення і після відростання волосся раба відправляли до адресата. Відомі різні способи прихованого

Продовження лістингу 4

```
XR(1)= keyr(28): XR(29)= keyr(56)

letbe keyr(), XR(), 56
END SUB

SUB sboxinit (b() AS INTEGER)
RESTORE sboxesl
FOR i% = 1 TO 8
 FOR j% = 1 TO 64
  READ b(i%, j%)
 NEXT j%
NEXT i%
END SUB

SUB stob (a$, mbits() AS INTEGER)
FOR i% = 1 TO 8
 b$ = MYBIN$(ASC(MID$(a$, i%, 1)))
 FOR j% = 1 TO 8
  mbits(((i% - 1) * 8) + j%)= ASC(MID$(b$, j%, 1)) -
48
 NEXT j%
NEXT i%
END SUB

SUB transpose (datax() AS INTEGER, T() AS INTEGER,
nt%)
letbe XT(), datax(), 64
FOR i% = 1 TO nt%
 datax(i%)= XT(T(i%))
NEXT i%
END SUB

SUB xtob (a$, mbits() AS INTEGER)
LOCAL X$,NIBBLE$
FOR i% = 1 to 16
    X$ = MID$(a$,i%,1)
```

Продовження лістингу 4

```
r%)\ 4) MOD 2
 x(k4% - 1)= (s(k%, r%)\ 2) MOD 2: x(k4%)= s(k%, r%)
MOD 2
NEXT k%
transpose x(), ptr(), 32
END SUB
SUB init (x() AS INTEGER, n%)
 FOR i% = 1 TO n%
  READ x(i%)
 NEXT i%
END SUB

SUB letbe (target() AS INTEGER, source() AS INTEGER,
last%)
FOR il% = 1 TO last%
 target(il%)= source(il%)
NEXT il%
END SUB
FUNCTION MYBIN$ (n%)
STS$ = ""
p% = n%
 FOR i% = 1 TO 8
  IF (p% MOD 2) THEN
   ST$ = "1" + ST$
  ELSE
   ST$ = "0" + ST$
  END IF
  p% = p% \ 2
 NEXT i%
 MYBIN$ = ST$
END FUNCTION

SUB mrotate (keyr() AS INTEGER)
letbe XR(), keyr(), 56
FOR ir% = 56 TO 2 STEP -1
 XR(ir%)= XR(ir% - 1)
NEXT ir%
```


листа серед рядків звичайного, незахищеного листа: від молока до складних хімічних реактивів з подальшою обробкою.

Широко застосовується сучасний метод “мікрокрапки”: повідомлення записується за допомогою сучасної техніки на дуже маленький носій — “мікрокрапку”, яка пересилається із звичайним листом, наприклад, над маркою або де-небудь в іншому заздалегідь обумовленому місці.

Один типовий стеганографічний прийом тайнопису — акровірш. Акровіршем називається така організація віршованого тексту, при якій, наприклад, початкові букви кожного рядка утворюють приховуване повідомлення.

Зараз у зв'язку з широким застосуванням ЕОМ застосовуються різні методи “приховування” інформації, що захищається, усередині великих її об'ємів.

На відміну від стеганографії, криптографія займається методами перетворення інформації, які повинні перешкодити супротивникові у витяганні її з перехоплюваних повідомлень. При цьому по каналу зв'язку передається вже не сама інформація, що захищається, а результат її перетворення за допомогою шифру або коди, і для супротивника виникає складне завдання розтину шифру або коди.

Розтин шифру — процес отримання інформації (відкритого тексту), що захищається, з шифрованого повідомлення (шифртекста) без знання застосованого шифру.

Шифрування — процес застосування шифру і інформації, що захищається, тобто перетворення інформації, що захищається, в шифроване повідомлення за допомогою певних правил, що містяться в шифрі.

Дешифрування — процес, зворотний шифруванню, що полягає в перетворенні шифрованого повідомлення в інформацію, що захищається, за допомогою певних правил, що містяться в шифрі.

Під ***ключем*** в криптографії розуміють змінний елемент шифру, який застосовують для шифрування конкретних повідомлень.

Одне з центральних місць в понятійному апараті криптографії займає таке поняття, як стійкість шифру. Під ***стійкістю шифру*** розуміють здатність шифру протистояти всіляким методам розтину. Якісно зрозуміти його досить легко, але отримання строгих доказових оцінок стійкості для кожного конкретного шифру все ще залишається невирішеною проблемою. Це пояснюється тим, що до цих пір немає математичних результатів, необхідних для вирішення такої проблеми.

Тому стійкість конкретного шифру оцінюється тільки шляхом всіляких спроб його розтину і залежить від кваліфікації криптоаналітиків, що розкривають шифр. Подібну процедуру називають перевіркою криптостійкості.

Криптологія — наука, що складається з двох напрямів: криптографії і криптоаналізу. ***Криптоаналіз*** — це наука (і практика її застосування) про методи і способи розтину шифрів. Співвідношення криптографії і криптоаналізу очевидно: криптографія — це захист, тобто розробка шифрів, а криптоаналіз — розтин шифрів. Проте це дві науки зв'язано один з одним, і не буває хороших криптографів, що не володіють методами криптоаналізу. Річ у тому, що стійкість розробленого шифру можна довести за допомогою проведення різних спроб розтину шифру, стаючи в думках в положення супротивника.

Трохи історії

Довгий час заняття криптографією було долею одинаків. Серед них були обдаровані учені, дипломати і священнослужителі. Відомі випадки, коли криптографію вважали навіть чорною магією. Цей період розвитку криптографії, як мистецтва, тривав з незапам'ятних часів до початку XX століття, коли з'явилися перші шифрувальні машини. Розуміння математичного характеру вирішуваних криптографічних завдань прийшло тільки в середині XX століття, після робіт видатного американського ученого До. Шенона.

Перші відомості про використання шифрів у військовій справі пов'язані з ім'ям спартанського полководця Лісандра (шифр “Сциталь”, V століття д.н.э). Цезар використовував в листуванні шифр, який увійшов до історії як “шифру Цезаря”. У стародавній Греції був винайдений вид шифру, який надалі називався “Квадрат Політія”. Братерство франкмасонов з моменту свого виникнення (VIII століття) розробило і використовувало цілу систему особливих шифрів.

Одну з перших книг по криптографії написав абат І. Трітемій (1462-1516 рр.) що жив в Германії. У 1566 р. відомий механік і математик Д. Кардано опублікував роботу з описом винайденої ним системи шифрування (“грати Кардано”). Франція XVI століття залишила в історії

Продовження лістингу 4

```
FUNCTION desalg$ (a$)
 temp$ = "": stob a$, ades()
 transpose ades(), InitialTr(), 64
 transpose ades(), swappy(), 64
 FOR i% = 16 TO 1 STEP -1
  letbe bdes(), ades(), 64
  f i%, bdes(), xdes()
  FOR j% = 1 TO 32
   ades(j%)= (bdes(j% + 32)+ xdes(j%)) MOD 2
  NEXT j%
  FOR j% = 33 TO 64
   ades(j%)= bdes(j% - 32)
  NEXT j%
 NEXT i%
 transpose ades(), FinalTr(), 64
 btos ades(), temp$
 desalg$ = temp$
END FUNCTION

SUB f (i%, a() AS INTEGER, x() AS INTEGER)
h% = i%: letbe EF(), a(), 64
transpose EF(), etr(), 48
letbe ikeyf(), P2(), 64
transpose ikeyf(), KeyTr2(), 48
FOR j% = 1 TO rots(h%)
 mrotate P2()
NEXT j%

FOR j% = 1 TO 48
 yf(j%)= (EF(j%)+ ikeyf(j%)) MOD 2
NEXT j%
FOR k% = 1 TO 8
 k6% = 6 * k%: k4% = 4 * k%
 r% = (32 * yf(k6% - 5)) + (16 * yf(k6%)) + (8 *
yf(k6% - 4)) + (4 * yf(k6% - 3)) + (2 * yf(k6% - 2))
+ yf(k6% - 1)+ 1
 x(k4% - 3)= (s(k%, r%)\ 8) MOD 2: x(k4% - 2)= (s(k%,
```

Продовження лістингу 4

```
DATA 02,12,04,01,07,10,11,06,08,05,03,15,13,00,14,09
DATA 14,11,02,12,04,07,13,01,05,00,15,10,03,09,08,06

DATA 04,02,01,11,10,13,07,08,15,09,12,05,06,03,00,14
DATA 11,08,12,07,01,14,02,13,06,15,00,09,10,04,05,03

DATA 12,01,10,15,09,02,06,08,00,13,03,04,14,07,05,11
DATA 10,15,04,02,07,12,09,05,06,01,13,14,00,11,03,08
DATA 09,14,15,05,02,08,12,03,07,00,04,10,01,13,11,06
DATA 04,03,02,12,09,05,15,10,11,14,01,07,06,00,08,13

DATA 04,11,02,14,15,00,08,13,03,12,09,07,05,10,06,01
DATA 13,00,11,07,04,09,01,10,14,03,05,12,02,15,08,06
DATA 01,04,11,13,12,03,07,14,10,15,06,08,00,05,09,02
DATA 06,11,13,08,01,04,10,07,09,05,00,15,14,02,03,12

DATA 13,02,08,04,06,15,11,01,10,09,03,14,05,00,12,07
DATA 01,15,13,08,10,03,07,04,12,05,06,11,00,14,09,02
DATA 07,11,04,01,09,12,14,02,00,06,10,13,15,03,05,08
DATA 02,01,14,07,04,10,08,13,15,12,09,00,03,05,06,11

rotsl:
DATA 1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1

SUB btos (mbits() AS INTEGER, a$)
a$ = ""
FOR i% = 1 TO 8
 w% = 0
 FOR j% = 1 TO 8
  w% = w% + ((mbits(((i% - 1) * 8) + j%)) * (2 ^ (8 -
j%)))
 NEXT j%
 a$ = a$ + CHR$(w%)
NEXT i%
END SUB
```


криптографії шифри короля Генріха IX і Рішельє. У Росії найбільш відомим шифром є “цифрова азбука” 1700 року, автором якої був Петро I.

Деякі відомості про властивості шифрів і їх застосування можемо знайти в художній літературі і кіно. Хороше і докладне пояснення одного з простих шифрів — шифру заміни і методів його розтину міститься в двох відомих розповідях: “Золотий жук” Е. По і “Танцюючі чоловічки” А. Конан-дойля.

Розглянемо детальніше деякі приклади.

Шифр “Сциталь”. Цей шифр відомий з часів війни Спарти і Персії проти Афін. Спартанський полководець Лісандр підозрював персів в зраді, але не знав їх таємних планів. Його агент в стані персів прислав шифроване повідомлення, яке дозволило Лісандру випередити персів і розгромити їх. Шифроване повідомлення було написане на поясі офіційного гінця від персів таким чином: агент намотав пояс на сциталь (дерев'яний циліндр певного діаметру) і написав на поясі повідомлення уподовж сциталя; потім він розмотав пояс і вийшло, що упоперек поясу безладно написані букви. Гонець не здогадався, що узор на його красивому поясі насправді містить зашифровану інформацію. Лісандр узяв сциталь такого ж діаметру, акуратно намотав на нього пояс і уподовж сциталя прочитав повідомлення від свого агента.

Відзначимо, що в цьому шифрі перетворення відкритого тексту в шифрований полягає в певній перестановці букв відкритого тексту. Тому клас шифрів, до яких відноситься і шифр “Сциталь”, — це ***перестановочні шифри***.

Шифр Цезаря. Цей шифр реалізує наступні перетворення відкритого тексту: кожна буква відкритого тексту замінюється третьою після неї буквою в алфавіті, який вважається по кругу, тобто після букви “я” слідує буква “а”. Тому клас шифрів, до яких відноситься і шифр Цезаря, — це ***підстановочні шифри***.

Відзначимо, що Цезар замінював букву третьою після неї буквою, але можна замінювати і п'ятою, і будь-який інший. Головне, щоб той, кому посилається шифроване повідомлення, знав цю величину зрушення.

Шифр Віженера. Цей шифр відноситься до сімейства *поліалфавітних підстановочних шифрів*. Його найзручніше уявити, як шифр Цезаря із змінною величиною зрушення. Щоб знати, на скільки зрушувати чергову букву відкритого тексту, заздалегідь обмовляється спосіб запам'ятовування зрушень. Для цієї мети використовується ключове слово, кожна буква якого своїм номером в алфавіті указує величину зрушення. Ключове слово повторюється стільки раз, скільки потрібно для заміни всіх букв відкритого тексту.

Подальший розвиток ідеї ключового слова, а саме ідея запам'ятовувати спосіб перетворення відкритого тексту за допомогою якої-небудь книги, привів до виникнення різних видів так званих книжкових шифрів.

Результати криптографічних досліджень реалізуються зараз у вигляді шифруючих пристроїв, вбудованих в сучасні системи зв'язку. Тому криптографи обмежені у вибиранні засобів тим рівнем техніки і технології, який досягнутий на даний момент. Така залежність відбивається і на виборі використовуваного в криптографії математичного апарату.

Умовно можна виділити три принципово різних етапи в розвитку математичного апарату криптографії.

До 40-х років XX століття застосовувалися тільки електромеханічні шифромашини, тому і спектр математичних перетворень був обмежений, в основному, методами комбінаторного аналізу і теорії вірогідності.

Після появи електронної техніки, а тим більше комп'ютерів, сильно змінився і математичний апарат криптографії. Отримали розвиток прикладні ідеї і методи теорії інформації, алгебри, теорії кінцевих автоматів.

Роботи Діффі і Хеллмана (70-і роки) послужили поштовхом для бурхливого розвитку нових напрямів математики: теорії односторонніх функцій, доказів з нульовим розголошуванням. У наш час прогрес саме в цих напрямах визначає практичні можливості криптографії.

Проте для того, щоб криптографічні методи перетворення забезпечили ефективний захист інформації, вони повинні задовольняти ряду вимог. У стислому вигляді їх можна сформулювати таким чином:

- складність і стійкість криптографічного захисту повинні вибиратися залежно від об'єму і ступеня секретності даних;
- надійність захисту повинна бути такою, щоб секретність не порушувалася у тому випадку, коли зловмисникові стає відомий метод захисту;
- метод захисту, набір використовуваних ключів і механізм їх розподілу не можуть бути дуже складними;
- виконання процедур прямого і зворотного перетворень повинне бути формалізованим. Ці процедури не повинні залежати від довжини повідомлень;
- помилки, що виникають в процесі виконання перетворення, не повинні розповсюджуватися на текст повною мірою і по системі;
- надмірність, що вноситься процедурами захисту повинна бути мінімальною.

Продовження лістингу 4

```
KeyTr2l:
DATA 14,17,11,24,01,05,03,28,15,06,21,10,23,19,12,04
DATA 26,08,16,07,27,20,13,02,41,52,31,37,47,55,30,40
DATA 51,45,33,48,44,49,39,56,34,53,46,42,50,36,29,32

etrl:
DATA 32,01,02,03,04,05,04,05,06,07,08,09,08,09,10,11
DATA 12,13,12,13,14,15,16,17,16,17,18,19,20,21,20,21
DATA 22,23,24,25,24,25,26,27,28,29,28,29,30,31,32,01

ptrl:
DATA 16,07,20,21,29,12,28,17,01,15,23,26,05,18,31,10
DATA 02,08,24,14,32,27,03,09,19,13,30,06,22,11,04,25

sboxesl:
DATA 14,04,13,01,02,15,11,08,03,10,06,12,05,09,00,07
DATA 00,15,07,04,14,02,13,01,10,06,12,11,09,05,03,08
DATA 04,01,14,08,13,06,02,11,15,12,09,07,03,10,05,00
DATA 15,12,08,02,04,09,01,07,05,11,03,14,10,00,06,13

DATA 15,01,08,14,06,11,03,04,09,07,02,13,12,00,05,10
DATA 03,13,04,07,15,02,08,14,12,00,01,10,06,09,11,05
DATA 00,14,07,11,10,04,13,01,05,08,12,06,09,03,02,15
DATA 13,08,10,01,03,15,04,02,11,06,07,12,00,05,14,09

DATA 10,00,09,14,06,03,15,05,01,13,12,07,11,04,02,08
DATA 13,07,00,09,03,04,06,10,02,08,05,14,12,11,15,01
DATA 13,06,04,09,08,15,03,00,11,01,02,12,05,10,14,07
DATA 01,10,13,00,06,09,08,07,04,15,14,03,11,05,02,12

DATA 07,13,14,03,00,06,09,10,01,02,08,05,11,12,04,15
DATA 13,08,11,05,06,15,00,03,04,07,02,12,01,10,14,09
DATA 10,06,09,00,12,11,07,13,15,01,03,14,05,02,08,04
DATA 03,15,00,06,10,01,13,08,09,04,05,11,12,07,02,14
```

Продовження лістингу 4

```
IF rescue% THEN

PRINT "Спробуйте іншим способом розшифрувати цей
файл."
ELSE
 PRINT "Розшифровка по алгоритму DES завершена."
END IF
SYSTEM

' Дані і функції

InitialTrl:
DATA 58,50,42,34,26,18,10,02,60,52,44,36,28,20,12,04
DATA 62,54,46,38,30,22,14,06,64,56,48,40,32,24,16,08
DATA 57,49,41,33,25,17,09,01,59,51,43,35,27,19,11,03
DATA 61,53,45,37,29,21,13,05,63,55,47,39,31,23,15,07

FinalTrl:
DATA 40,08,48,16,56,24,64,32,39,07,47,15,55,23,63,31
DATA 38,06,46,14,54,22,62,30,37,05,45,13,53,21,61,29
DATA 36,04,44,12,52,20,60,28,35,03,43,11,51,19,59,27
DATA 34,02,42,10,50,18,58,26,33,01,41,09,49,17,57,25

swappyl:
DATA 33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48
DATA 49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64
DATA 01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16
DATA 17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32

KeyTr1l:
DATA 57,49,41,33,25,17,09,01,58,50,42,34,26,18,10,02
DATA 59,51,43,35,27,19,11,03,60,52,44,36
DATA 63,55,47,39,31,23,15,07,62,54,46,38,30,22,14,06
DATA 61,53,45,37,29,21,13,05,28,20,12,04
```

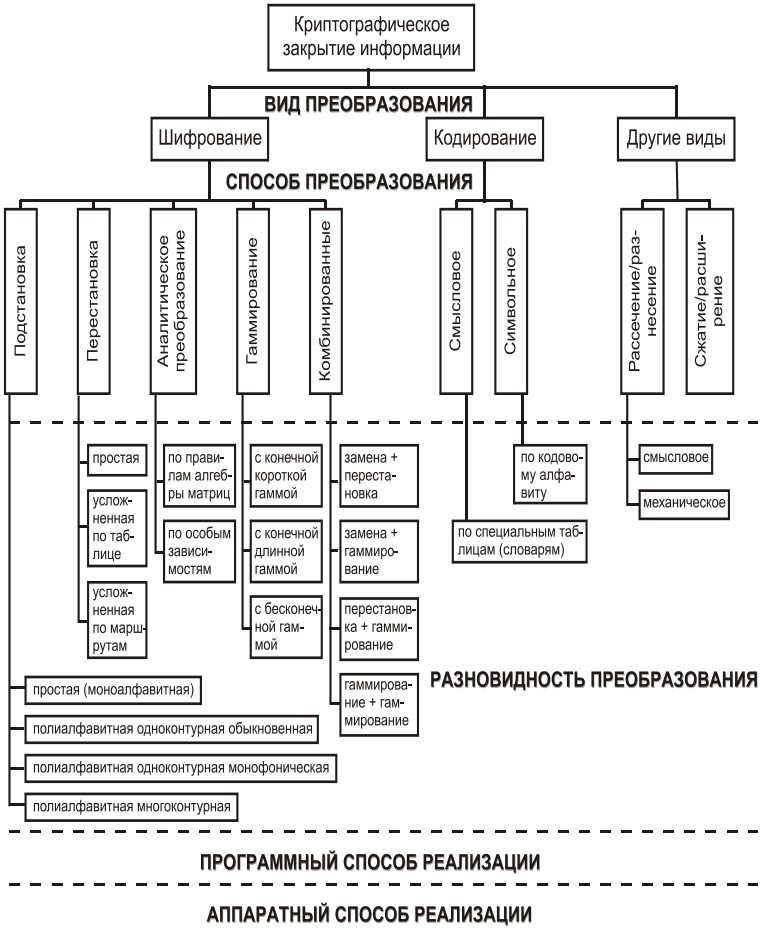

Класифікація криптографічних методів

В даний час не існує закінченої і загальноприйнятої класифікації криптографічних методів, оскільки багато які з них знаходяться у стадії розвитку і становлення. Найбільш доцільною представляється загальноприйнята російська класифікація, представлена на мал. 1.

Під *шифруванням* в даному випадку розуміється такий вид криптографічного захисту, при якому перетворенню піддається кожен символ повідомлення, що захищається. Всі відомі способи шифрування розбиті на п'ять груп: *підстановка* (заміна), *перестановка*, *аналітичне перетворення*, *гаммування* і *комбіноване шифрування*. Кожен з цих способів може мати декілька різновидів.

Під *кодуванням* розуміється такий вид криптографічного закриття, коли деякі елементи даних (не обов'язково окремі символи), що захищаються, замінюються заздалегідь вибраними кодами (цифровими, буквеними, буквено-цифровими поєднаннями і так далі). Цей метод має два різновиди: смислове і символьне кодування. При *смисловому кодуванні* кодовані елементи мають цілком певний сенс (слова, пропозиції, групи пропозицій). При *символьному кодуванні* кодується кожен символ тексту, що захищається. Символьне кодування по суті співпадає з підстановлювальним шифруванням.

До окремих видів криптографії відносяться методи *розтину* і *стиснення даних*. Розтин полягає в тому, що масив даних, що захищаються, ділиться (розтинається) на такі елементи, кожен з яких окремо не дозволяє розкрити зміст інформації, що захищається. Виділені таким чином елементи даних розносяться по різних зонах пам'яті або розташовуються на різних носіях. Стиснення даних є заміною однакових рядків даних або послідовностей однакових символів, що часто зустрічаються, деякими заздалегідь вибраними символами.



Мал. 1. Класифікація криптографічних методів

Продовження лістингу 4

```
LOCATE rescue% + 11, 21: PRINT ; "Розшифровка :   0
%";
bigblocks&=(blocks&-1)\ 32
large$=space$(256)
FOR m& = 1 TO bigblocks&
  outblock$=""
  get$ #1,256,large$
  for o%=1 to 256 step 8
   outblock$=outblock$+desalg$(mid$(large$,o%,8))
  next
  put$ #2,outblock$
  LOCATE rescue% + 11, 32: PRINT ; USING "###"; (m& /
(bigblocks&+1)) * 100;
next

FOR n& = (bigblocks&*32)+1 TO blocks& - 1
 GET$ #1,8,ciphertekst$
 plaintekst$ = desalg$(ciphertekst$)
 PUT$ #2, plaintekst$
 LOCATE rescue% + 11, 32: PRINT ; USING "###"; (n& /
blocks&) * 100;
NEXT n&
get$ #1,8,ciphertekst$
if len(ciphertekst$)> 0 then
   plaintekst$ = desalg$(ciphertekst$)
   IF rescue% THEN
     last$ = plaintekst$
   ELSE
     last$ = LEFT$(plaintekst$, lf& + 32 - LOF(1))
   END IF
   IF LEN(last$)> 0 THEN
      PUT$ #2, last$
   END IF
end if
CLOSE
LOCATE 11 + rescue%, 32: PRINT "100 %  готово": PRINT
```

Продовження лістингу 4

```
RIGHT$(plainf$, LEN(plainf$) - INSTR(plainf$, "*.") -
1)
 ELSE
  plainf$ = LEFT$(plainf$, INSTR(plainf$, "*.") - 1)+
oldplainf$ + RIGHT$(plainf$, LEN(plainf$) -
INSTR(plainf$, "*."))
 END IF
END IF
IF (RIGHT$(plainf$, 1)= "*") THEN
 IF plainf$ = "*" THEN
  plainf$ = oldplainf$
 ELSE
  IF (MID$(plainf$, LEN(plainf$) - 1, 1)= ".") THEN
   plainf$ = LEFT$(plainf$, INSTR(plainf$, ".") - 1)+
             RIGHT$(oldplainf$, LEN(oldplainf$) -
             INSTR(oldplainf$, ".")+ 1)
  ELSE
   plainf$ = LEFT$(plainf$, LEN(plainf$) - 1)+
oldplainf$
  END IF
 END IF
END IF
IF RIGHT$(plainf$, 1)= "\" THEN plainf$ = plainf$ +
oldplainf$

OPEN plainf$ FOR RANDOM AS 2
IF LOF(2)> 0 THEN
 CLOSE #2
 PRINT : PRINT "Вже є файл з таким ім'ям!"
 SYSTEM
ELSE
 CLOSE #2
 OPEN plainf$ FOR BINARY AS 2
END IF

plaintekst$ = ""
blocks& = (LOF(1)\ 8) - 3
```


Вимоги до криптографічних методів захисту інформації

Розкриття зашифрованих текстів (в першу чергу знаходження ключа) здійснюється за допомогою методів криптоаналізу. Основними методами криптоаналізу є:

- *статистичні*, при яких знаючи статистичні властивості відкритого тексту намагаються досліджувати статистичні закономірності шифротекста і на підставі виявлених закономірностей розкрити текст;
- *метод вірогідних слів*, в якому при зіставленні деякої невеликої частини шифротекста з відомим фрагментом відкритого тексту намагаються знайти ключ і з його допомогою розшифрувати весь текст. Необхідний фрагмент відкритого тексту можна знайти за допомогою статистичних методів або просто вгадати, виходячи з передбачуваного змісту або структури відкритого тексту.

Оскільки криптографічні методи застосовуються давно, то вже сформульовані основні вимоги до них.

1. Метод повинен бути надійним, тобто відновлення відкритого тексту при володінні тільки шифротекстом, але не ключем повинно бути практично нездійсненним завданням
2. Через труднощі запам'ятовування об'єм ключа не повинен бути великим.
3. Через труднощі, пов'язані з складними перетвореннями, процеси шифрування повинні бути простими.
4. Через можливості появи помилок передачі дешифровка шифротекста, що містить окремі помилки, не повинна привести до нескінченного збільшення помилок в отриманому передбачуваному відкритому тексті.
5. Через труднощі передачі об'єм шифротекста не повинен бути значно більше відкритого тексту.

Перераховані вимоги були розроблені для традиційної криптографії.

При сучасному розвитку техніки необхідність задоволення перерахованим вимогам зазнає істотні зміни.

У зв'язку з розвитком технології, що дозволяє з великою щільністю записати і тривалий час надійно зберігати великі об'єми інформації, умова невеликого об'єму ключа може бути ослаблена (по суті це умова, як і всі останні, набуває нового сенсу, відповідного досягнутому рівню техніки). У зв'язку з розвитком мікроелектроніки з'являється можливість розробки

дешевих пристроїв, що здійснюють швидко і точно (порівняно) складні перетворення інформації. З іншого боку, можливість збільшення швидкості передачі відстає від можливості збільшення швидкості обробки інформації. Це, поза сумнівом, дозволяє ослабити вимогу п. 3 без збитку для швидкості передачі, що практично досягається. В даний час устаткування є високонадійним, а методи виявлення і виправлення помилок — добре розвиненими. До того ж, зазвичай використовувані в комп'ютерних мережах протоколи сеансів зв'язку передбачають передачу будь-якого тексту навіть за наявності збоїв під час передачі. Тому вимога п. 4 значною мірою втратила свою актуальність. В окремих випадках, якщо канали зв'язку не переобтяжені, може бути ослаблена і вимога п. 5.

Таким чином, не зачепленою залишилася вимога п. 1, при розгляді якого слід врахувати дві обставини.

По-перше, в автоматизованих системах (АС) циркулюють великі об'єми інформації, а наявність великого об'єму шифротекста полегшує завдання криптоаналізу.

По-друге, для вирішення завдання криптоаналізу можна використовувати ЕОМ. Це дозволяє в нових умовах вимагати значного збільшення надійності. Іншим важливим негативним чинником застосування криптографії в АС є те, що часто використовуються мови з вельми обмеженим запасом слів і строгим синтаксисом (мови програмування).

У зв'язку з новими специфічними застосуваннями криптографічних методів можуть бути висунуті також інші вимоги. Так, наприклад, другою важливою областю застосування криптографічних методів є системи управління базами даних (СУБД). В цьому випадку до криптографічних методів пред'являються наступні додаткові вимоги.

1. Через неможливість читання і відновлення записів з середини файлу, шифрування і дешифровка кожного запису повинні проводитися незалежно від інших записів.
2. Для уникнення великих зручностей обробки і зайвого перевантаження системи допоміжними перетвореннями необхідно всі операції з файлами проводити з даними в зашифрованому вигляді.

Специфіка СУБД впливає на надійність захисту по наступних причинах:

- дані в СУБД тривалий час знаходяться в зашифрованому вигляді. Це ускладнює або навіть робить неможливою часту зміну ключів, і у зв'язку з цим ЗІ стає менш надійним;

Продовження лістингу 4

```
lf& = VAL(le$)

ev& = lf& + 24
IF (ev& MOD 8) THEN ev& = ev& + 8 - (ev& MOD 8)
rescue% = 0
IF (ev& <> lof1&) THEN
 PRINT "Невірна!! (можлива втрата даних)"
 PRINT "     Довжина початкового файлу :"; lf&
 PRINT "    Довжина вказаного файлу :"; lof1&
 PRINT "   Довжина файлу повинна бути :"; ev&
 INPUT ; "Спробувати відновити? (y/n) : ", q$:
         IF (INSTR(q$, "N") OR (INSTR(q$, "n"))) THEN
SYSTEM
 rescue% = 4: PRINT
ELSE
 PRINT lf&; ", Вірна!"
END IF
pl% = INSTR(12, header$, "#")
oldplainf$ = RIGHT$(header$, 24 - pl%)
PRINT "        Ім'я початкового файлу : ";
oldplainf$;
OPEN oldplainf$ FOR RANDOM AS 2
IF INSTR(oldplainf$, ".") THEN
 PRINT " ([*.*] ";
ELSE
 PRINT " ([*] ";
END IF
IF LOF(2)> 0 THEN PRINT "вже є в каталозі";
PRINT ")"
CLOSE #2
plainf$ = ""
INPUT "        Ім'я вихідного файлу : ", plainf$
IF plainf$ = "" THEN plainf$ = oldplainf$
plainf$ = RTRIM$(LTRIM$(plainf$))
IF INSTR(plainf$, "*.") THEN
 IF INSTR(oldplainf$, ".") THEN
  plainf$ = LEFT$(plainf$, INSTR(plainf$, "*.") - 1)+
LEFT$(oldplainf$, INSTR(oldplainf$, "."))+
```

Продовження лістингу 4

```
IF (LEN(PW$)< 8) THEN PW$ = PW$ + STRING$(8 -
LEN(PW$), 0)

IF len(pw$)= 16 then

   LOCATE 6, 1: PRINT "                    Пароль :
";
   STRING$(16, 15); STRING$(10 " ")
    PW$ = ucase$(PW$)
    xtob PW$, P2()
ELSE
   LOCATE 6, 1: PRINT "                     Пароль :
";
   STRING$(8, 15); STRING$(10 " ")
   PW$ = LEFT$(PW$, 8)
   stob PW$, P2()
END IF

PRINT "            Перевірка пароля : ";
transpose P2(), KeyTr1(), 56
get$ #1,24,cheader$
header$ = desalg$(LEFT$(cheader$, 8))
IF NOT (LEFT$(header$, 3)= "DES") THEN
 PRINT "Невірний!": PRINT : PRINT "Неправильний
пароль або ";
       cipherf$; " не є зашифрованим файлом!"
 SYSTEM
ELSE
 PRINT "Вірний!"
END IF

PRINT "       Перевірка довжини файлу :";
header$ = header$ + desalg$(MID$(cheader$, 9, 8))
header$ = header$ + desalg$(RIGHT$(cheader$, 8))
pl% = INSTR(header$, "#")
le$ = MID$(header$, pl% + 1 (11 - pl%))
```


- ключі можуть не передаватися по різних адресах, а зберігатися всі в одному місці. Це підвищує надійність системи через зменшення можливості оволодіння ключами сторонніми особами.

У файлових системах вірогідність появи помилки значно менше, чим в каналах зв'язку, тому вимога п. 4 для файлових систем не має великого практичного значення.

Поява швидкодіючих ЕОМ сприяє виникненню так званої обчислювальної криптографії, тісно пов'язаної з обчислювальною технікою.

Математика розділення секрету

Розглянемо наступну, у наш час цілком реальну ситуацію. Два співвласники коштовності хочуть покласти її на зберігання в сейф. Сейф сучасний, з цифровим замком на 16 цифр. Оскільки співвласники не довіряють один одному, то вони хочуть закрити сейф так, щоб вони могли відкрити його разом, але ніяк не порізно. Для цього вони запрошують третю особу, звану ділером, якому вони обидва довіряють. Ділер випадково вибирає 16 цифр як “ключ”, щоб закрити сейф, і потім повідомляє першого співвласника таємно від другого перші 8 цифр “ключа”, а другому співвласникові таємно від першого — останні 8 цифр “ключа”. Такий спосіб представляється з точки здорового глузду оптимальним, адже кожен із співвласників, отримавши “півключа”, не зможе їм скористатися без другої половини, а що може бути краще?! Недоліком даного прикладу є те, що будь-який із співвласників, залишившись наодинці з сейфом, може за пару хвилин знайти ті, що не дістають “півключа” за допомогою нескладного пристрою, призначеного для перебору ключів і працює на тактовій частоті 1 Мгц. Здається, що єдиний вихід — в збільшенні розміру “ключа”, скажімо, удвічі. Але є інший математичний вихід міркування здорового глузду. А саме, ділер незалежно вибирає дві випадкові послідовності по 16 цифр в кожній, повідомляє кожен із співвласників таємно від іншого “його” послідовність, а як “ключ”, щоб закрити сейф, використовує послідовність, отриману складанням по модулю 10 відповідних цифр двох вибраних послідовностей. Досить очевидно, що для кожного із співвласників всі 1016 можливих “ключів” однаково вірогідні і залишається тільки перебирати їх, що займе в середньому близько півтора роки для пристрою перебору ключів, обладнаного процесором з частотою 100 Мгц.

І з математичної, і з практичної точки зору нецікаво зупинятися на випадку двох учасників і слід розглянути загальну ситуацію. Неформально кажучи, ***схема, що розділяє секрет*** (СРС) дозволяє "розподілити" секрет **n** учасниками так, щоб заздалегідь задана дозволена кількість учасників могла однозначно відновити секрет (сукупність цих множин називається ***структурою доступу***), а недозволені — не отримували ніякої додаткової до наявної апріорної інформації про можливе значення секрету. СРС з останньою властивістю називаються *досконалими*.

Історія СРС починається з 1979 року, коли ця проблема була поставлена і багато в чому вирішена Блейклі і Шаміром для випадку порогових **(n, k)**-СРС (тобто дозволеними множинами є будь-які множини з **k** або більш елементами). Особливий інтерес викликали так звані *ідеальні СРС*, в яких об'єм інформації, що надається учасникові, не більше об'єму секрету. Не для будь-якої структури доступу можливе ідеальне розділення секрету. З іншого боку, було показано, що для будь-якого набору дозволених множин можна побудувати досконалу СРС, проте відомі побудови вельми неекономні. Розглянемо деякі алгебро-геометричні і комбінаторні завдання, що виникають при математичному аналізі СРС.

Говоритимемо, що сімейство підпросторів $\{L_0, \ldots, L_n\}$ скінченномірного векторного простору **L** над полем **K** задовольняє властивості "все або нічого", якщо для будь-якої множини $A \subset \{1, \ldots, n\}$ лінійна оболонка підпросторів $\{L_a: a \in A\}$ або містить підпростір L_0 цілком, або перетинається з ним тільки по вектору **0**. У підрозділі "Лінійне розділення секрету" ми побачимо, що таке сімейство задає "лінійну" СРС, у якої множина $A \subset \{1, \ldots, n\}$ є дозволеним, якщо і тільки якщо лінійна оболонка підпросторів $\{L_a: a \in A\}$ містить підпростір L_0 цілком. У зв'язку з цим поняттям виникає ряд питань. Наприклад, якщо поле **K** скінченне ($|K| = q$) і всі підпростори $\{L_0, \ldots, L_n\}$ одновимірні, то яке максимально можливе число учасників **n** для лінійних граничних (**n**, **k**)-СРС ($k > 1$)? Інакше кажучи, яке максимально можливе число векторів $\{h_0, \ldots, h_n\}$ таких, що будь-які **k** векторів, що містять вектор h_0, лінійно незалежні, а будь-які **k + 1** векторов, що містять вектор **h0**, лінійно залежні. Виявляється, що ця властивість еквівалентна наступній, на перший погляд сильнішій, властивості: будь-які **ho** векторів лінійно незалежні, а будь-які **k + 1** — лінійно залежні. Такі системи векторів вивчалися в геометрії як **N - множини (N = n +**

Продовження лістингу 4

```
DIM xdes(1 TO 64) AS shared INTEGER

DIM XT(1 TO 64)   AS shared INTEGER

DIM P2(1 TO 64) AS shared INTEGER

main:
CLS
parm$ = ltrim$(rtrim$(COMMAND$))+" "
IF LEN(parm$)> 1 THEN
 cipherf$ = LTRIM$(RTRIM$(LEFT$(parm$, INSTR(parm$ "
")))))
 PRINT "Ім'я  зашифрованого файлу : "; cipherf$
ELSE
 INPUT "Ім'я розшифрованого файлу : ", cipherf$
END IF
if len(cipherf$)=0 then
 print : print "ЗБІЙ: введіть ім'я файлу!"
 system
end if
OPEN cipherf$ FOR RANDOM AS 1
lof1& = LOF(1)
IF lof1& = 0 THEN
 CLOSE #1
 KILL cipherf$
 PRINT : PRINT "Файл не знайдений!";
 SYSTEM
ELSE
 CLOSE #1
 OPEN cipherf$ for binary access read as #1
END IF

PW$ = ""
LOCATE 6, 1
INPUT "                    Пароль : ", PW$
```

Продовження лістингу 4

```
RESTORE swappyl
DIM swappy(1 TO 64) AS shared INTEGER

init swappy(), 64

RESTORE KeyTr1l
DIM KeyTr1(1 TO 56) AS shared INTEGER
init KeyTr1(), 56

RESTORE KeyTr2l
DIM KeyTr2(1 TO 48) AS shared INTEGER
init KeyTr2(), 48

RESTORE etrl
DIM etr(1 TO 48) AS shared INTEGER
init etr(), 48

RESTORE ptrl
DIM ptr(1 TO 32) AS shared INTEGER
init ptr(), 32

sboxinit s()

RESTORE rotsl
DIM rots(1 TO 16) AS shared INTEGER
init rots(), 16

DIM XR(1 TO 56) AS shared INTEGER

DIM EF(1 TO 64) AS shared INTEGER
DIM ikeyf(1 TO 64) AS shared INTEGER
DIM yf(1 TO 64) AS shared INTEGER

DIM ades(1 TO 64) AS shared INTEGER
DIM bdes(1 TO 64) AS shared INTEGER
```


1) в кінцевій проектній геометрії **PG(k–1, q)**, в комбінаториці — як ортогональні таблиці індексу **л = 1**, в теорії кодування — як перевірочні матриці. Наприклад, конструкція таких множин з **N = q + 1**. Існує досить стара гіпотеза про те, що це і максимальне можливе **N**, за винятком двох випадків: випадку **q < до**, коли **N = до + 1**, і випадку **q = 2m**, **до = 3** або **до = q – 1**, коли **N = q + 2.**

Розділення секрету для довільних структур доступу

Почнемо з формальної математичної моделі. Є **n+1** безліч **S0**, **S1 .**, **Sn** і (сумісне) розподіл вірогідності **P** на їх декартовому просторі **S = S0 * .* Sn**. Відповідні випадкові величини позначаються через **Si**. Є також деяка безліч **Г** підмножин множини **{1 ., n}**, звана структурою доступу.

Визначення 1

Пара **(P,S)** називається досконалою імовірнісною СРС, що реалізовує структуру доступу **Г**, якщо

$$P(S_0 = c_0 \mid S_i = c_i, i \in A) \in \{0, 1\} \text{ для } A \in \Gamma, \quad (1)$$

$$P(S0 = c0 \mid Si = ci, i \in A) = P(S0 = c0) \text{ для } A \notin \Gamma \quad (2)$$

Це визначення можна пояснити наступним чином. Маємо множину S_0 всіх можливих секретів, із яких секрет s_0 вибирається з вірогідністю $p(s_0)$, і маємо СРС, яка "роздає" секрет s_0 між **n** учасниками, посилаючи "проекції" s_1, ., s_n секрету с вірогідністю $P_{S0}(s_1, ., s_n)$. Відмітимо, що **i**-ий учасник отримує свою "проекцію" $s_i \in S_i$ і не має інформації про значення других "проекцій", однак знає всі множини S_i, а також розподіли ймовірностей $p(s_0)$ и $P_{S0}(s_1, ., s_n)$. Ці розподіли можуть бути еквівалентно замінені одна на одну: $P(s_0, s_1, ., s_n) = p(s_0)P_{S0}(s_1, ., s_n)$. Ціль СРС, як вказувалося вище:

- учасники з дозволеної безлічі **A** (тобто $A \in \Gamma$) разом могли б однозначно відновити значення секрету — це властивість (1);
- учасники, утворюючі недозволену множину **A,** $\notin$ не могли б отримати додаткову інформацію про s0, тобто, щоб вірогідність того, що значення секрету S0 **= c0**, не залежала від значень "проекцій" Si при і $\in$ A — це властивість (2).

Визначення 2

Матриця **V** задає досконалу комбінаторну СРС, що реалізовує структуру доступу **Г**, якщо, по-перше, для будь-якої множини $\mathbf{A} \in \mathbf{\Gamma}$ нульова координата будь-якого рядка матриці **V** однозначно визначається значеннями її координат з множини **A,** і, по-друге, для будь-якої множини $\mathbf{A} \notin \mathbf{\Gamma}$ і будь-яких заданих значень координат з множини **A** число рядків матриці **V** з даним значенням **i** нульової координати не залежить від **i**.

Підставимо в досконалу імовірнісну СРС, пару **(P, S)**, матрицю **V**, що складається з рядків $\mathbf{s} \in \mathbf{S}$, таких що $\mathbf{P(s)} > \mathbf{0}$. Відмітимо, що якщо у визначенні 1 покласти всі ненульові значення **P** однаковими, а умови (1) і (2) переформулировать на комбінаторній мові, то вийде визначення 2. Це комбінаторне визначення декілька узагальнюється, якщо допустити в матриці **V** рядки, що повторюються, що еквівалентно імовірнісному визначенню 1, коли значення вірогідності **P(s)** — раціональні числа.

Лінійне розділення секрету

Почнемо з запропонованої Шаміром елегантної схеми розподілу секрету для граничних структур доступу. Нехай $\mathbf{K} = \mathbf{GF(q)}$ — кінцеве поле із **q** елементів (наприклад, $\mathbf{q = p}$ — просте число и $\mathbf{K} = \mathbf{Z_p}$) и $\mathbf{q > n}$. Співставимо з учасниками **n** різних ненульових елементів поля $\{\mathbf{a_1}, ., \mathbf{a_n}\}$ тоді $\mathbf{a_0 = 0}$. При розподілі секрету $\mathbf{s_0}$ ділер СРС генерує **k–1** незалежно рівномірно розподілених на **GF(q)** випадкових величин $\mathbf{f_j\ (j = 1, ., k–1)}$ и посилає **i**-му учаснику **(i = 1, ..., n)** "його" значення $\mathbf{s_i = f(a_i)}$ многочлена $\mathbf{f(x) = f_0 + f_1x + . + f_{k-1}x_{k-1}}$, де $\mathbf{f_0 = s_0}$. Так як любий многочлен степені **k-1** однозначно поновлюється по його значенням в будь-яких **k** точках то любі **k** учасники разом можуть поновити многочлен **f(x)** и, тоді, найти значення секрету як $\mathbf{s_0 = f(0)}$. По цій же причині для будь-яких **k–1** учасників, отримані ними значення проекцій $\mathbf{s_i}$ и значення секрету $\mathbf{s_0}$ існує рівно один многочлен,. такий, що $\mathbf{s_i = f(a_i)}$ и $\mathbf{s_0 = f(0)}$. Тоді, ця схема являється ідеальною в співвідношенні з

Лістинг 4. Приклад реалізації алгоритму DES на мові Basic для розшифровки файлів

```
$CPU 80386
$FLOAT NPX
$OPTIMIZE SPEED
$LIB ALL-
$OPTION CNTLBREAK ON

DECLARE FUNCTION MYBIN$ (n%)

DECLARE FUNCTION desalg$ (a$)

DECLARE SUB f (i%, a%(), x%())
DECLARE SUB transpose (datax%(), T%(), n%)
DECLARE SUB mrotate (keyx%())

DECLARE SUB stob (a$, mbits%())
DECLARE SUB btos (mbits%(), a$)
DECLARE SUB letbe (target%(), source%(), last%)
DECLARE SUB init (x() AS INTEGER, n%)
DECLARE SUB sboxinit (b() AS INTEGER)

DECLARE SUB xtob (a$, mbits%())

DIM s(1 TO 8, 1 TO 64) AS shared INTEGER

' Ініціалізація
RESTORE InitialTrl
DIM InitialTr(1 TO 64) AS shared INTEGER
init InitialTr(), 64

RESTORE FinalTrl
DIM FinalTr(1 TO 64) AS shared INTEGER
init FinalTr(), 64
```

Закінчення лістингу 3

```
CASE "5"

 NIBBLE$ = "0101"
    CASE "6"
     NIBBLE$ = "0110"
    CASE "7"
     NIBBLE$ = "0111"
    CASE "8"
     NIBBLE$ = "1000"
    CASE "9"
     NIBBLE$ = "1001"
    CASE "A"
     NIBBLE$ = "1010"
    CASE "B"
     NIBBLE$ = "1011"
    CASE "C"
     NIBBLE$ = "1100"
    CASE "D"
     NIBBLE$ = "1101"
    CASE "E"
     NIBBLE$ = "1110"
    CASE "F"
     NIBBLE$ = "1111"
    CASE ELSE
     Print "Не є 16-ричним значенням!"
     SYSTEM
    END SELECT
    FOR j% = 1 to 4
        mbits(((i% - 1) * 4) + j%)= ASC(MID$(NIBBLE$,
j%, 1)) - 48
    NEXT j%
  NEXT i%
END SUB
```


визначенням 2. “Лінійність” дана схема становиться ясна, якщо записати “розподіл секрету” в векторно-матричному вигляді:

s = fH (3)

де **s = (s0 ., sn), f = (f0 ., fk–1), до Ч (n+1)** — матриця **H = (hij) = (aij-1)** і **h00 = 1**. Відмітимо, що будь-які **до** стовпців цієї матриці лінійно незалежні, а максимально можливе число стовпців матриці **H** рівне **q**, щоб добитися обіцяного значення **q+1** треба додати стовпець, відповідний крапці “нескінченність”.

Візьмемо з (3) як **H** довільну **r Ч (n + 1)**- матрицю з елементами з поля **K**. Отримувану СРС, називатимемо *одновимірною лінійною СРС*. Вона є досконалою комбінаторною СРС із структурою доступу Г**,** що складається з множин А таких, що вектор h0 представимо у вигляді лінійної комбінації векторів {**hj**: **j ∈A},** де hj — це j-ий стовпець матриці H**.** Рядками матриці V**,** відповідною даною СРС, є, як видно з (3), лінійні комбінації рядків матриці H**.** Перепишемо (3) в наступному вигляді

sj = (f, hj) для **j = 0, 1 ., n**

де **(f, hj)** — скалярний твір векторів **f** і **hj**. Якщо **A ∈ Г**, тобто якщо **h0 = Улjhj**, то

$$\mathbf{s0 = (f, h0) = \left(f, У_{j}h_{j}\right) = Улj(f, hj)= Улjsj}$$

І, отже, значення секрету однозначно знаходиться по його “проекціях”. Розглянемо тепер випадок, коли вектор **h0** представимо у вигляді лінійної комбінації векторів **{hj: j ∈ A}**. Нам потрібно показати, що в цьому випадку для будь-яких заданих значень координат з множини **A** число рядків матриці **V** з даним значенням будь-якої координати не залежить від цього значення. У цьому не важко переконається, розглянувши (3) як систему лінійних рівнянь щодо невідомих **fi** і скориставшись тим, що система сумісна тоді і тільки тоді, коли ранг матриці коефіцієнтів рівний рангу розширеної матриці, а число рішень у сумісних систем однаково і рівно числу вирішень однорідної системи.

Розглянете дві системи: з “нульовим” рівнянням і без нього (тобто з вільним членом). Оскільки вектор **h0** не представимо у вигляді лінійної комбінації векторів **{hj: j ∈ A}**, то ранг матриці коефіцієнтів другої системи на 1 більше рангу матриці коефіцієнтів першої системи. Звідси

негайно витікає, що якщо перша система сумісна, то сумісна і друга при будь-якому **s0**.

Ця конструкція підводить нас до визначення загальної лінійною СРС. Хай секрет і його "проекції" представляються як скінченномірні вектори **si = (si¹ ., simi)** і генеруються по формулі **si = fHi**, де **Hi** — деякі **r** і **mi-матриці.** Зіставимо кожній матриці **Hi** лінійний простір **Li** її стовпців (тобто що складається зі всіх лінійних комбінацій вектор-стовпців матриці **Hi**). Нескладні міркування, аналогічні приведеним вище для одновимірного випадку (все **mi = 1**), показують, що дана конструкція дає досконалу СРС тоді і тільки тоді, коли сімейство лінійних підпросторів **{L0 ., Ln}** скінченномірного векторного простору $\mathbf{K}^{\mathbf{r}}$ задовольняє згаданій вище властивості "все або нічого". При цьому множина **A** є дозволеним **{La: a ∈ A}** що містить підпростір **L0** цілком. З іншого боку, множина **A** є недозволеним (**A ∉ Γ**), якщо і тільки якщо лінійна оболонка підпросторів **{La: a ∈ A}** перетинається з підпростором **L0** тільки по вектору **0**. Відзначимо, що якби для деякого **A** перетин **L0** і лінійної оболонки **{La: a ∈ A}** був нетривіальним, то учасники **A не** могли б відновити секрет однозначно, але отримували б деяку інформацію про нього, тобто схема не була б досконалою.

Приклад Розглянемо наступну структуру доступу для випадку чотирьох учасників, **Γmin, що** задається **= {1,2}, {2,3}, {3,4}**. Вона відома як перший побудований приклад структури доступу, для якої не існує ідеальної реалізації. Більш того, було доведено, що для будь-якої її досконалої реалізації **R(Γ) X 3/2**. З іншого боку, безпосередня перевірка показує, що вибір матриць **H0**, **H1 ..., H4**, приведених на мал. 2, дає досконалу лінійну СРС з **R = 3/2**, що реалізовує цю структуру, яка, отже, є і оптимальною (найбільш економною) СРС.

.

$$\mathbf{H}_0 = \begin{bmatrix} 10 \\ 01 \\ 00 \\ 00 \\ 00 \\ 00 \end{bmatrix}, \mathbf{H}_1 = \begin{bmatrix} 00 \\ 00 \\ 10 \\ 01 \\ 00 \\ 00 \end{bmatrix}, \mathbf{H}_2 = \begin{bmatrix} 100 \\ 010 \\ 100 \\ 010 \\ 001 \\ 000 \end{bmatrix}, \mathbf{H}_3 = \begin{bmatrix} 001 \\ 000 \\ 000 \\ 010 \\ 001 \\ 100 \end{bmatrix}, \mathbf{H}_4 = \begin{bmatrix} 00 \\ 01 \\ 00 \\ 00 \\ 10 \\ 01 \end{bmatrix},$$

Мал. 2. Матриці, реалізуючі ідеальну лінійну СРС

Продовження лістингу 3

```
NEXT i%

END SUB

SUB stob (a$, mbits() AS INTEGER)
FOR i% = 1 TO 8
 b$ = MYBIN$(ASC(MID$(a$, i%, 1)))
 FOR j% = 1 TO 8
  mbits(((i% - 1) * 8) + j%)= ASC(MID$(b$, j%, 1)) -
48
 NEXT j%
NEXT i%
END SUB
SUB transpose (datax() AS INTEGER, T() AS INTEGER,
n%)
letbe XT(), datax(), 64
FOR i% = 1 TO n%
 datax(i%)= XT(T(i%))
NEXT i%
END SUB

SUB xtob (a$, mbits() AS INTEGER)
LOCAL X$,NIBBLE$
FOR i% = 1 to 16
    X$ = MID$(a$,i%,1)
    SELECT CASE X$
    CASE "0"
     NIBBLE$ = "0000"
    CASE "1"
     NIBBLE$ = "0001"
    CASE "2"
     NIBBLE$ = "0010"
    CASE "3"
     NIBBLE$ = "0011"
    CASE "4"
     NIBBLE$ = "0100"
```

Продовження лістингу 3

```
SUB letbe (target() AS INTEGER, source() AS INTEGER,
LAST%)
FOR i% = 1 TO LAST%
 target(i%)= source(i%)
NEXT i%
END SUB

FUNCTION MYBIN$ (n%)
LOCAL ST$
p% = n%
ST$=""
FOR i% = 1 TO 8
 IF (p% MOD 2) THEN
  ST$ = "1" + ST$
 ELSE
  ST$ = "0" + ST$
 END IF
 p% = p% \ 2
NEXT i%
MYBIN$ = ST$
END FUNCTION

SUB mrotate (keyr() AS INTEGER)
letbe XR(), keyr(), 56
FOR i% = 1 TO 55
 XR(i%)= XR(i% + 1)
NEXT i%
XR(28)= keyr(1): XR(56)= keyr(29)
letbe keyr(), XR(), 56
END SUB
SUB sboxinit (b() AS INTEGER)
RESTORE sboxesl
FOR i% = 1 TO 8
 FOR j% = 1 TO 64
  READ b(i%, j%)
 NEXT j%
```


Ідеальне розділення секрету і матроїди

Почнемо з визначення ідеальних СРС. Для цього повернемося до комбінаторного визначення досконалої СРС. Наступне визначення досконалої СРС є навіть більш загальним, ніж імовірнісне визначення 1, оскільки умова (2) замінена в ньому на слабкіше.

Для довільної множини $\mathbf{B} \subseteq \{0, 1 ., n\}$ позначимо через **VB М Ч |B|**-матрицю, отриману з матриці **V** видаленням стовпців, номери яких не належать безлічі **B.** Нехай **||W||** позначає число різних рядків в матриці **W**.

Визначення 3

Матриця **V** задає ідеальну СРС, що реалізовує структуру доступу **Г**, якщо

$$||VA0|| = ||\cup V_A|| \text{ Ч } ||V_0||^{дГ(A)}, \quad (4)$$

де $дГ(A) = 0$, якщо $A \in Г$, і $дГ(A) = 1$ інакше.

Це визначення відрізняється від визначень 1 і 2 тим, що на недозволені множини **A** накладаються досить слабка умова, а саме, якщо безліч рядків **V** з даними значеннями координат з множини **A** непорожні, то всі можливі значення секрету зустрічаються в нульовій координаті цих рядків (без вимог "однаково часто" як в комбінаторному визначенні 2 або ж "з апріорною вірогідністю" як в імовірнісному визначенні 1). Легко бачити, що матриця будь-якою досконалою імовірнісною СРС задає ідеальну СРС, але зворотне невірно.

Для довільної комбінаторної СРС, матрицею, що задається, **V**, визначимо на множинах $A \subseteq \{0, 1 ., n\}$ функцію $h(A)= \log q\ ||V_A||$, де $q = |S_0|$. Легко перевірити, що $\max\{h(A), h(B)\} \le h(A \cup B) \le h(A)+ h(B)$ для будь-яких множин **A** і **B**, а умова (4) може бути переписане у вигляді

$$hq(VA0_{\cup})= hq(VA)+ дГ(A)\ hq(V0)$$

Лема. Для будь-якої ідеальної СРС якщо $A \notin Г$ і $\{A \cup i\} \in Г$, то $h(i) \ge h(0)$.

Доказ. За умовами леми

$h(A \cup 0)= h(A)+ h(0)$ і $h(A\ i \cup 0)= h(A \cup i)$. Отже

$$h(A)+ h(i) \ge h(A \cup i) = h(A \cup i \cup 0) \ge h(A \cup 0)= h(A)+ h(0)$$

Так ми припускаємо, що всі крапки $\mathbf{i \in \{1 ., n\}}$ істотні, тобто для будь-якого **i** знайдеться підмножина **A** таке, що $\mathbf{A \notin \Gamma}$ і $\mathbf{\{A \cup i\} \in \Gamma}$, то з леми витікає

Наслідок. Для будь-якої ідеальної СРС $\mathbf{|S_i| \geq |S_0|}$ для всіх $\mathbf{i = 1 ..., n}$.

Наслідок означає, як наголошувалося вище, що для досконалих СРС "розмір" проекції не може бути менше "розміру" секрету. Тому ідеальна СРС називається ідеальною, якщо $\mathbf{|S_i| = |S_0|}$ для всіх $\mathbf{i = 1 ..., n}$.

Зауваження. Нерівність $\mathbf{|S_i| \geq |S_0|}$ справедливо і для досконалих імовірнісних СРС, оскільки їх матриці задають ідеальну СРС.

Природне питання полягає в тому, для яких структур доступу Γ існують ті, що реалізовують їх ідеальні (імовірнісні або комбінаторні) СРС. Як вже наголошувалося, якнайкращу на сьогоднішній день відповідь використовує слово *матроїд*. Нагадаємо визначення матроїдів і деякі їх основні властивості.

Матроїдом називається кінцева множина **X** і сімейство **I** його підмножин, званих *незалежними* (решта множин називається залежними), якщо виконані наступні властивості:

$\mathbf{\varnothing \in I;}$ (5.1)$\in$

Якщо $\mathbf{A \in I}$ і $\mathbf{B \subset A}$, то $\mathbf{B \in I}$; (5.2)$\subset\in$

Якщо **A**, $\mathbf{B \in I}$ і $\mathbf{|A| = |B| + 1}$

то існує $\mathbf{a \in A \backslash B}$ таке, що $\mathbf{a \cup B \in I}$. (5.3)$\cup\in$

Приклад Множина **X** — це безліч векторів в деякому лінійному векторному просторі, а незалежні підмножини — це лінійно незалежні підмножини векторів.

Власне з цього прикладу і почалася теорія матроїдів, спочатку як спроба дати аксіоматичне визначення лінійної незалежності векторів через "внутрішні властивості", тобто не апелюючи до поняття вектора. На щастя, спроба не вдалася, оскільки знайшлися матроїди, не уявні як лінійні (тобто як системи векторів), а сама теорія матроїдів розрослася далеко за межі лінійної алгебри.

Приклад (матроїд Вамоса). Розглянемо наступну множину: $\mathbf{X = \{1, 2, 3, 4, 5, 6, 7, 8\}}$ і покладемо $\mathbf{a = \{1, 2\}}$, $\mathbf{b = \{3, 4\}}$, $\mathbf{з = \{5, 6\}}$ і $\mathbf{d = \{7, 8\}}$. Матроїд Вамоса визначається як матроїд, в якому множини $\mathbf{a \cup з}$, **a d, b**

Продовження лістингу 3

```
 transpose ades(), FinalTr(), 64

 btos ades(), temp$
 desalg$ = temp$
END FUNCTION

SUB f (i%, a() AS INTEGER, x() AS INTEGER)
h% = i%: letbe EF(), a(), 64
transpose EF(), etr(), 48
FOR j% = 1 TO rots(h%)
 mrotate P2()
NEXT j%
letbe ikeyf(), P2(), 64: transpose ikeyf(), KeyTr2(),
48
FOR j% = 1 TO 48
 yf(j%)= (EF(j%)+ ikeyf(j%)) MOD 2
NEXT j%
FOR k% = 1 TO 8
 k6% = 6 * k%: k4% = 4 * k%
 r% = (32 * yf(k6% - 5)) + (16 * yf(k6%)) + (8 *
yf(k6% - 4)) + (4 * yf(k6% - 3)) + (2 * yf(k6% - 2))
+ yf(k6% - 1)+ 1
 x(k4% - 3)= (s(k%, r%)\ 8) MOD 2: x(k4% - 2)= (s(k%,
r%)\ 4) MOD 2
 x(k4% - 1)= (s(k%, r%)\ 2) MOD 2: x(k4%)= s(k%, r%)
MOD 2
NEXT k%
transpose x(), ptr(), 32
END SUB

SUB init (x() AS INTEGER, n%)
 FOR i% = 1 TO n%
  READ x(i%)
 NEXT i%
END SUB
```

Продовження лістингу 3

```
DATA 13,02,08,04,06,15,11,01,10,09,03,14,05,00,12,07
DATA 01,15,13,08,10,03,07,04,12,05,06,11,00,14,09,02

DATA 07,11,04,01,09,12,14,02,00,06,10,13,15,03,05,08
DATA 02,01,14,07,04,10,08,13,15,12,09,00,03,05,06,11

rotsl:
DATA 1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1

SUB btos (mbits() AS INTEGER, a$)
a$ = ""
FOR i% = 1 TO 8
 w% = 0
 FOR j% = 1 TO 8
  w% = w% + ((mbits(((i% - 1) * 8) + j%)) * (2 ^ (8 -
j%)))
 NEXT j%
 a$ = a$ + CHR$(w%)
NEXT i%
END SUB

FUNCTION desalg$ (a$)
 temp$ = ""
 stob a$, ades()
 transpose ades(), InitialTr(), 64
 FOR i% = 1 TO 16
  letbe bdes(), ades(), 64
  FOR j% = 1 TO 32
   ades(j%)= bdes(j% + 32)
  NEXT j%
  f i%, ades(), xdes()
  FOR j% = 1 TO 32
   ades(j% + 32)= (bdes(j%)+ xdes(j%)) MOD 2
  NEXT j%
 NEXT i%
 transpose ades(), swappy(), 64
```


$\cup$ **з, b** $\cup$ **d, з** $\cup$ **d**, а також всі підмножини з п'яти або більш за елементи є залежними. Відомо, що цей матроїд не є лінійним.

Матроїд також можна визначити через так звану рангову функцію **r(A)** матроїда, визначувану як максимальна потужність незалежної підмножини **B** $\subseteq$ **A**. Очевидно, що незалежні множини (і лише вони) задаються умовою **r(A)= |A|**. Рангова функція матроїда володіє властивостями

r(A) $\in$**Z; r(**$\varnothing$**) = 0;** (6.1)$\in\varnothing$

r(A) $\leq$ **r(A** $\cup$ **b)** $\leq$ **r(A)+ 1;** (6.2)$\cup\leq$

Якщо **r(A** $\cup$ **b)= r (A** $\cup$ **з) = r(A)**, то **r (A** $\cup$ **b** $\cup$ **з) = r(A)**.
(6.3)$\cup\cup$

Нехай деяка функція **r(A)** володіє властивостями (6). Назвемо незалежними ті множини **A,** для яких **r(A)= |A|**. Тоді ці множини задають матроїд, а функція **r** є його ранговою функцією. Можливо також визначити матроїд через мінімальні залежні множини, звані циклами. Матроїд називається *зв'язним*, якщо для будь-яких двох його крапок існує цикл, що містить їх.

Тепер ми можемо сформулювати основний результат.

Теорема. Для будь-якої ідеальної СРС, що реалізовує структуру доступу **Г**, незалежні множини, описувані умовою **log|S0| ||V**$_A$**|| = |A|**, задають зв'язні матроїд на множині **{0, 1., n}**. Всі цикли цього матроїда, що містять крапку **0**, мають вигляд **0** $\cup$ **A,** де **A** $\in$ **Гmin**.

Доведення теореми приведене у відповідній літературі і виходить за рамки даної теми. Головною в доведенні теореми є "перевірка" цілочисельності функції **h(A)**. Насправді, **h(A)** очевидно володіє рештою властивостей (6) і, отже, за умови цілочисельності є ранговою функцією і задає матроїд.

Відзначимо, що з другої частини затвердження теореми виходить, що ідеальним СРС, які реалізовують дану структуру доступу **Г**, завжди відповідає один і той же матроїд, оскільки матроїд однозначно визначається всіма циклами, що проходять через фіксовану крапку. Тим самим, кожній ідеальній структурі доступу, що реалізовується, відповідає певний матроїд.

У зв'язку з теоремою виникає декілька природних питань. Перш за все, чи не породжують ідеальні СРС всі матроїди? Ні, наприклад, матроїд Вамоса не може бути отриманий як матроїд ідеальної СРС. З іншого боку

лінійні матроїди є ні що інше, як розглянуті ідеальні одновимірні лінійні СРС. У зв'язку з цим виникає питання про існування структури доступу **Г**, яку неможливо реалізувати у вигляді ідеальної одновимірної лінійною СРС, але можна у вигляді ідеальної багатовимірної лінійною СРС. Такі приклади є, і, значить, ми можемо говорити про багатовимірні лінійні матроїди як класи матроїдів більш загальні, ніж лінійні.

Отже, ідеальних СРС більше, ніж лінійних матроїдов, але менше, ніж всіх матроїдів. Уточнити, наскільки більше, представляється досить складним завданням. Зокрема, чи існує структура доступу **Г**, що ідеально реалізовується, яку неможливо реалізувати як ідеальну лінійну багатовимірну СРС?

Секретність і імітостійкість

Криптографічні перетворення забезпечують вирішення двох головних проблем ЗІ: ***проблеми секретності*** (позбавлення супротивника можливості витягувати інформацію з каналу зв'язку) і ***проблеми імітостійкості*** (позбавлення супротивника можливості ввести помилкову інформацію в канал зв'язку або змінити повідомлення так, щоб змінився його сенс).

У разі телефонного зв'язку головною є *проблема імітостійкості*, оскільки викликана сторона не може часто визначити, хто дзвонить. Підслуховування, що вимагає підключення до проводів, технічно складніше і юридично небезпечніше, ніж виклик кореспондента і видача себе за когось іншого. У разі радіозв'язку ситуація прямо протилежна. Перехоплення тут є пасивним і зв'язаний з незначною юридичною небезпекою, тоді як введення інформації пов'язане з ризиком виявлення незаконного передавача і юридичного переслідування.

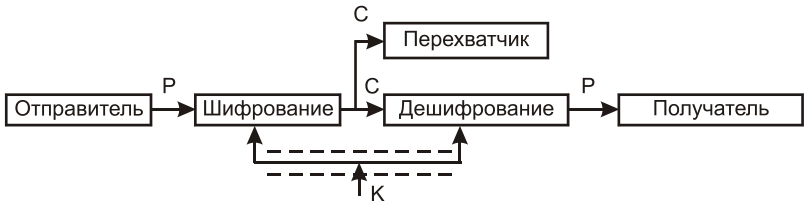
Проблема секретності

Проблеми секретності і імітостійкості між собою тісно зв'язані, тому методи вирішення однієї з них часто застосовні для вирішення іншої. З двох названих проблем проблема секретності зазвичай розглядається першою, як старіша і ширше відоміша. Розглянемо схему проходження потоку інформації в криптографічній системі, що забезпечує секретність (мал. 3).


```
sboxes1:
DATA 14,04,13,01,02,15,11,08,03,10,06,12,05,09,00,07
DATA 00,15,07,04,14,02,13,01,10,06,12,11,09,05,03,08
DATA 04,01,14,08,13,06,02,11,15,12,09,07,03,10,05,00
DATA 15,12,08,02,04,09,01,07,05,11,03,14,10,00,06,13

DATA 15,01,08,14,06,11,03,04,09,07,02,13,12,00,05,10
DATA 03,13,04,07,15,02,08,14,12,00,01,10,06,09,11,05
DATA 00,14,07,11,10,04,13,01,05,08,12,06,09,03,02,15
DATA 13,08,10,01,03,15,04,02,11,06,07,12,00,05,14,09

DATA 10,00,09,14,06,03,15,05,01,13,12,07,11,04,02,08
DATA 13,07,00,09,03,04,06,10,02,08,05,14,12,11,15,01
DATA 13,06,04,09,08,15,03,00,11,01,02,12,05,10,14,07
DATA 01,10,13,00,06,09,08,07,04,15,14,03,11,05,02,12

DATA 07,13,14,03,00,06,09,10,01,02,08,05,11,12,04,15
DATA 13,08,11,05,06,15,00,03,04,07,02,12,01,10,14,09
DATA 10,06,09,00,12,11,07,13,15,01,03,14,05,02,08,04
DATA 03,15,00,06,10,01,13,08,09,04,05,11,12,07,02,14

DATA 02,12,04,01,07,10,11,06,08,05,03,15,13,00,14,09
DATA 14,11,02,12,04,07,13,01,05,00,15,10,03,09,08,06
DATA 04,02,01,11,10,13,07,08,15,09,12,05,06,03,00,14
DATA 11,08,12,07,01,14,02,13,06,15,00,09,10,04,05,03



DATA 12,01,10,15,09,02,06,08,00,13,03,04,14,07,05,11
DATA 10,15,04,02,07,12,09,05,06,01,13,14,00,11,03,08
DATA 09,14,15,05,02,08,12,03,07,00,04,10,01,13,11,06
DATA 04,03,02,12,09,05,15,10,11,14,01,07,06,00,08,13

DATA 04,11,02,14,15,00,08,13,03,12,09,07,05,10,06,01
DATA 13,00,11,07,04,09,01,10,14,03,05,12,02,15,08,06
DATA 01,04,11,13,12,03,07,14,10,15,06,08,00,05,09,02
DATA 06,11,13,08,01,04,10,07,09,05,00,15,14,02,03,12
```

Продовження лістингу 3

```
DATA 62,54,46,38,30,22,14,06,64,56,48,40,32,24,16,08

DATA 57,49,41,33,25,17,09,01,59,51,43,35,27,19,11,03
DATA 61,53,45,37,29,21,13,05,63,55,47,39,31,23,15,07

FinalTrl:
DATA 40,08,48,16,56,24,64,32,39,07,47,15,55,23,63,31
DATA 38,06,46,14,54,22,62,30,37,05,45,13,53,21,61,29
DATA 36,04,44,12,52,20,60,28,35,03,43,11,51,19,59,27
DATA 34,02,42,10,50,18,58,26,33,01,41,09,49,17,57,25

swappyl:
DATA 33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48
DATA 49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64
DATA 01,02,03,04,05,06,07,08,09,10,11,12,13,14,15,16
DATA 17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32

KeyTr1l:
DATA 57,49,41,33,25,17,09,01,58,50,42,34,26,18,10,02
DATA 59,51,43,35,27,19,11,03,60,52,44,36
DATA 63,55,47,39,31,23,15,07,62,54,46,38,30,22,14,06
DATA 61,53,45,37,29,21,13,05,28,20,12,04

KeyTr2l:
DATA 14,17,11,24,01,05,03,28,15,06,21,10,23,19,12,04
DATA 26,08,16,07,27,20,13,02,41,52,31,37,47,55,30,40
DATA 51,45,33,48,44,49,39,56,34,53,46,42,50,36,29,32

etrl:
DATA 32,01,02,03,04,05,04,05,06,07,08,09,08,09,10,11
DATA 12,13,12,13,14,15,16,17,16,17,18,19,20,21,20,21
DATA 22,23,24,25,24,25,26,27,28,29,28,29,30,31,32,01

ptrl:
DATA 16,07,20,21,29,12,28,17,01,15,23,26,05,18,31,10
DATA 02,08,24,14,32,27,03,09,19,13,30,06,22,11,04,25
```

Мал. 3. Потік информації в криптографічній системі, забезпечуючій секретність

Відправник генерує відкритий текст, або шифроване повідомлення **P**, яке повинне бути передане одержувачеві по незахищеному каналу, що прослуховується. Для того, щоб перехоплювач не зміг дізнатися змісту повідомлення **P**, відправник шифрує або кодує його за допомогою оборотного перетворення **Sk** і отримує криптограму або шифрований текст

З = Sk(P). Одержувач, прийнявши повідомлення, дешифрує або декодує його за допомогою зворотного перетворення $\mathbf{S_k^{-1}}$ і отримує початкове повідомлення

$$\mathbf{S_k^{-1}(C) = S_k^{-1}[Sk(P)] = P}$$

Перетворення **Sk** вибирається з сімейства криптографічних перетворень, званих *криптографічною*, або *загальною*, системою.

Параметр, що вибирає окреме використовуване перетворення, називається ***ключем***.

Загальна система — це набір інструкцій, апаратних засобів і програмного забезпечення ЕОМ, за допомогою якого можна зашифрувати і розшифрувати текст різними способами, один з яких вибирається за допомогою конкретного ключа.

Кажучи формальніше, ***криптографічна система*** — це однопараметричне сімейство $\mathbf{(Sk)_{кК}}$ ϵоборотних перетворень **Sk:PC** $\rightarrow$з простору **P** повідомлень відкритого тексту в простір шифрованих повідомлень. Параметр, або ключ **До**, називається ***простором ключів***.

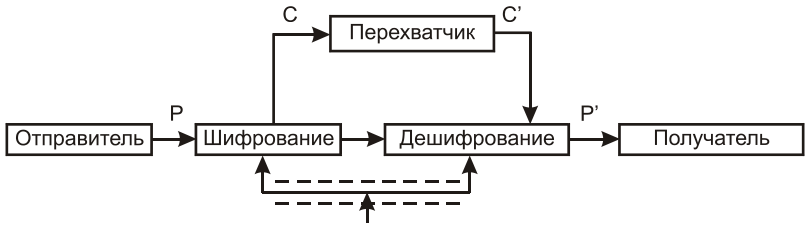
Зазвичай загальна система розглядається як загальнодоступна. З одного боку, відкрита для всіх частина загальної системи є предметом угоди, а з іншого боку, це відображає дуже важливе правило техніки захисту: захищеність системи не повинна залежати від секретності чого-небудь такого, що не можна швидко змінити у разі витоку секретної інформації. Зазвичай загальна система є деякою сукупністю апаратури і

програм, яку можна змінити тільки із значною витратою часу і засобів, тоді як ключем є легко змінний об'єкт.

Оскільки вся секретність зосереджена в секретності ключа, то його треба передавати відправникові і одержувачеві по захищеному каналу розповсюдження ключів, такому, як кур'єрська служба і так далі

Проблема імітостійкості

Тепер розглянемо схему проходження потоку інформації в криптографічній системі, забезпечуючою імітостійкість (мал. 4).



Мал. 4. Потік информації в криптографічній системі, забезпечуючій імітостійкість

При вирішенні проблеми імітостійкості супротивник може не тільки бачити всі криптограми, передавані по каналу, але може також змінювати їх по своєму бажанню. Законний одержувач захищає себе від обману, дешифруючи всі отримані повідомлення і приймаючи тільки ті повідомлення, які зашифровані правильним ключем.

Будь-яка спроба з боку перехоплювача розшифрувати криптограму для отримання відкритого тексту P або зашифрувати свій текст P' для отримання прийнятної криптограми З' без отримання ключа повинна бути повністю виключена.

Якщо криптоаналіз неможливий і криптоаналітик не може вивести **P** і **З** або **З'** з **P'** без попереднього отримання ключа, то така криптографічна система є ***криптостійкою***.

Безумовна і теоретична стійкість

Існує два принципово різних методу забезпечення стійкості криптографічних систем проти криптоаналітичного нападу.

У деяких системах об'єм доступною криптоаналітику інформації фактично недостатній для того, щоб знайти перетворення і дешифрування,

Продовження лістингу 3

```
for b%=1 to 256 step 8

outblock$ = outblock$+desalg$(mid$(inblock$,b%,8))
  next
  Put$ #2, outblock$
  LOCATE 10, 19: PRINT ; USING "###"; (n& / blocks&)
* 100;
NEXT n&

rest1 = lof1& MOD 256
rest2 = lof1& MOD 8
rest  = rest1-rest2

IF rest1 > 0 THEN
 outblock$=""
 get$ #1,rest1,inblock$
if rest2 > 0 then
  inblock$=inblock$+left$(anything$(8-rest2))
 end if
 for b%=1 to len(inblock$) step 8
  outblock$ = outblock$+desalg$(mid$(inblock$,b%,8))
 next
 Put$ #2, outblock$
END IF

CLOSE
LOCATE 10, 19: PRINT "100 %  завершене"
PRINT : PRINT "Шифрування по алгоритму DES
завершене."
PRINT
SYSTEM

' Дані і функції

InitialTrl:
DATA 58,50,42,34,26,18,10,02,60,52,44,36,28,20,12,04
```

Продовження лістингу 3

```
LOCATE 9, 8: PRINT "Пароль : "; STRING$(8, 15);
STRING$(10 " ")
   PW$ = LEFT$(PW$, 8)
   stob PW$, P2()

end IF
transpose P2(), KeyTr1(), 56
ciphertekst$ = ""
blocks& = lof1& \ 256

w = RND(-INT(TIMER))
anything$ = ""
FOR i% = 1 TO 12
 anything$ = anything$ + CHR$(128 + INT(127 *
RND(1)))
NEXT i%

header$ = "#" + LTRIM$(STR$(lof1&))
header$ = "DES" + LEFT$(anything$, 8 - LEN(header$))+
header$
header$ = header$ + RIGHT$(anything$, 12 -
LEN(bplainf$))+
          "#" + bplainf$

cheader$=desalg$(left$(header$,8))+desalg$(MID$(heade
r$,9,8))+desalg$(right$(header$,8))

put$ #2, cheader$

LOCATE 10, 8:  PRINT ; "Шифрування:   0 %";

inblock$=space$(256)
FOR n& = 1 TO blocks&
 get$ #1,256,inblock$
 outblock$=""
```


причому дана ситуація не залежить від того, які обчислювальні потужності має криптоаналітик.

Система такого роду називається ***безумовно стійкою***.

У тому випадку, коли перехоплений матеріал містить достатньо інформації для однозначного вирішення криптоаналітичного завдання, немає ніякої гарантії, що це рішення буде знайдено криптоаналітиком, що має певні обчислювальні ресурси. Отже, мета розробника криптографічної системи полягає в тому, щоб зменшити витрати на операції шифрування і дешифрування, але щоб в теж час будь-яка криптоаналітична операція була дуже складною і тому економічно невигідною. Іншими словами, необхідно, щоб завдання криптоаналізу, про яке відомо, що вона вирішувана при кінцевому об'ємі обчислень, було б таким громіздким, що для її вирішення не вистачило б фізичних обчислювальних ресурсів всього світу. Завдання такого об'єму називають ***обчислювально такою, що не реалізовується***, а пов'язану з нею криптографічну систему — ***обчислювально стійкою***.

У разі безумовних стійких систем їх стійкість може бути доведена, але що стосується теорії обчислювальної складності, то при нинішньому рівні її розвитку вона не в змозі продемонструвати таку, що не обчислювально реалізовує будь-яке криптографічне завдання. Тому в криптографії виникло і розвивається напрям, присвячений розробці формальних процесів перевірки стійкості. Такі процеси зводяться до криптоаналітичного нападу на пропоновані для перевірки криптографічні системи за умов, сприятливих для криптоаналітика.

Єдиною безумовно стійкою системою, що знаходиться в широкому користуванні, є ***стрічка одноразового використання***, в якій відкритий текст об'єднується з випадковим ключем такої ж довжини. Зазвичай відкритий текст є рядком з n бітів, яка об'єднуються з випадковим ключем такої ж довжини за допомогою складання по модулю 2. Як видно з самої назви, цей ключ ніколи більше не використовується.

Навіть якщо б криптоаналітик спробував здійснити дешифрування, використовуючи всі **2n** можливих ключів, він просто побачив би всі **2n** можливі відкриті тексти однакової довжини. Оскільки перехоплення криптограми не дозволяє криптоаналітику вивести яке-небудь повідомлення у вигляді відкритого тексту, то він не дізнається нічого, окрім довжини повідомлення. Клод Шеннон аналізував абсолютну стійкість детальніше. Якщо криптоаналітик має в своєму розпорядженні необмежений час для обчислень, то він не зв'язаний рамками обчислювальної ефективності і може провести повний криптоаналіз, випробовуючи всі можливі ключі і зберігаючи як результат всі осмислені

тексти. У разі стрічки одноразового використання необхідно зберегти всі осмислені тексти, що мають однакову з криптограмою довжину, але в інших безумовно стійких системах може бути менша кількість осмислених рішень. Наприклад, криптограма XMDA, отримана в результаті простої підстановки годиться для будь-якого чотирьохбуквеного слова з буквами, що не повторюються.

У міру того як кількість перехоплених текстів зростає, може бути досягнута крапка, в якій стає можливим отримання однозначного рішення. Шенон назвав це інтервалом однозначності **N0**. У разі стрічки одноразового використання цього ніколи не трапитися, і $\mathbf{N0} = \wp$, тоді як у разі простого підстановлювального шифру значення **N0** кінцеве. Шенон запропонував модель для прогнозу інтервалу однозначності шифру. Отримані за допомогою цієї моделі результати узгоджуються з практикою. Відповідно до цієї моделі "випадкового шифру"

$$\mathbf{N0} = \frac{\mathbf{H(K)}}{\mathbf{D}} \quad (7)$$

де **H(K)** — ентропія ключа (звичайно це просто довжина ключа, зміряна в бітах, або **log2** від кількості ключів), **D** — надмірність мови, зміряна в бітах на 1 знак. (Наприклад, в англійській мові за буквою Q завжди слідує буква U, яка є надмірною.) Якісно модель можна показати, переписавши (7) у вигляді вимоги для однозначності рішення

$$\mathbf{H(K)} \leq \mathbf{N0D} \quad (8) \leq$$

де **H(K)** характеризує кількість невідомих в двійковому представленні ключа, а **N0D** в широкому сенсі визначає кількість рівнянь, які необхідно вирішити для знаходження ключа. Коли кількість рівнянь *менше* кількості невідомих, однозначне рішення ***неможливе*** і система *є* безумовно стійкою. Коли кількість рівнянь *більше* кількості невідомих, тобто як в (8), однозначне рішення ***можливе*** і система *не є* безумовно стійкою, хоча вона все ще *може бути* обчислювано стійкою.

Не дивлячись на те, що в теорії кодування Шенона (тобто в припущенні, що криптоаналітик має в своєму розпорядженні необмежені ресурси) зазвичай розглядається напад за наявності тільки шифрованого тексту, але іноді використовуються і комбінації шифрованого і відкритого тексту, що підвищує надмірність.

Згідно Фрідмена, майже будь-яка криптограма з 25 букв і більш, отримана підстановкою, може бути розкрита. Оскільки криптоаналітик має в своєму розпорядженні обмежені обчислювальні можливості, він не

Продовження лістингу 3

```
LOOP WHILE sp% > 0
bplainf$ = RIGHT$(plainf$, LEN(plainf$) - (sp0%))
PRINT ім'я файлу, що "Зберігається: "; bplainf$
pp% = INSTR(sp0% + 1, plainf$, ".")
IF pp% = 0 THEN
 dcipherf$ = plainf$ + ".DES"
ELSE
 dcipherf$ = LEFT$(plainf$, pp% - 1)+ ".DES"
END IF
PRINT " За умовчанням : "; dcipherf$
INPUT "Вихідний файл : ", cipherf$
IF cipherf$ = "" THEN cipherf$ = dcipherf$
OPEN cipherf$ FOR RANDOM AS 2
IF LOF(2)> 0 THEN
 CLOSE #2
 PRINT : PRINT "Вихідний файл вже існує!"
 SYSTEM
ELSE
 CLOSE #2
 OPEN cipherf$ FOR binary AS 2
END IF

PW$ = ""
LOCATE 9, 1
INPUT ; "     Пароль : ", PW$

IF (LEN(PW$)< 8) THEN PW$ = PW$ + STRING$(8 -
LEN(PW$), 0)

IF len(pw$)= 16 then

   LOCATE 9, 8: PRINT "Пароль : "; STRING$(16, 15);
STRING$(10 " ")
   PW$ = ucase$(PW$)
   xtob PW$, P2()

ELSE
```

Продовження лістингу 3

```
DIM XT(1 TO 64)   AS shared INTEGER

DIM P2(1 TO 64)   AS shared INTEGER

' Головна програма
main:
CLS
parm$ = ltrim$(rtrim$(COMMAND$))+" "
IF LEN(parm$)> 1 THEN
 Plainf$ = LTRIM$(RTRIM$(LEFT$(parm$, INSTR(parm$ "
"))))
 PRINT "Початковий файл : "; Plainf$
ELSE
INPUT "Початковий файл : ", plainf$
END IF
if len(plainf$)=0 then
 print : print "ЗБІЙ: введіть ім'я файлу!"
 system
end if
OPEN plainf$ FOR RANDOM AS 1
lof1& = LOF(1)
IF lof1& = 0 THEN
 CLOSE #1
 KILL plainf$
 PRINT : PRINT "Файл не знайдений!";
 SYSTEM
ELSE
 IF lof1& > 9999999 THEN PRINT : PRINT "Початковий
файл дуже великий!": SYSTEM
 CLOSE #1
 OPEN plainf$ for binary access read as #1
END IF

cipherf$ = ""
sp0% = 0: sp% = 0
DO
 sp0% = sp%
 sp% = INSTR(sp% + 1, plainf$, "\")
```


може перепробувати все **26! ≈ 4.1026** ключів і повинен покладатися на субоптимальні методи, такі як частотний аналіз. Таким чином, можна сказати, що **N0** = 25 знаків.

У разі стрічки одноразового використання **H(K)=** $\wp$, звідки, згідно (7), **N0 =** $\wp$**.** $\notin \backslash\rangle \Sigma$ отримуємо **H(K)= log2(26!)= 88,4** бітів, тому для обчислення **N0** прийнято знаходити **D**. Кожен знак міг би переносити максимум **log2(26)= 4,7** біт інформації, якби всі комбінації були можливі. Але оскільки правила правопису і граматики забороняють використання більшості комбінацій, то в середньому кожен знак переносить всього лише **1,5** біт інформації. Залишки **3,2** біт виявляються надмірними, звідки **D = 3,2** біт/знак. Таким чином, рівняння (7) представляє величину **N0 = 28** знаків, що добре узгоджується з практикою.

Наприклад, якщо перехоплено повідомлення довжиною в **1000** знаків і відома деяка послідовність з **100** знаків відкритого тексту, то загальна надмірність складе не **3200** бітів, а (**900** знаків) × (**3,2** біт/знак) + (**100** знаків) × (**4,7** біт/знак) = **3350** бітів.

Стиснення даних усуває надмірність, збільшуючи тим самим інтервал однозначності. Надмірна інформація може бути додана після дешифрування. Досконале стиснення даних усунуло б всю надмірність і привело б до **N0 =** $\wp$ при будь-якій довжині ключа, але це досить дорогий захід.

Важливим підготовчим етапом для перевірки стійкості шифру є продумування різних передбачуваних можливостей, за допомогою яких супротивник може розкрити шифр. Поява таких можливостей у супротивника зазвичай не залежить від власне використовуваного криптографічного методу, а є наслідком деякої зовнішньої підказки, наявність якої істотно впливає на стійкість шифру. Тому оцінки стійкості шифру завжди містять ті припущення про супротивника, в умовах яких ці оцінки отримані.

Перш за все, зазвичай вважається, що супротивник знає сам шифр і має можливість його попереднього вивчення. Супротивник також знає деякі характеристики відкритих текстів (інформації, що захищається), наприклад, загальну тематику повідомлень, їх стиль, деякі стандарти, формати і так далі

Приведемо три приклади специфічних можливостей супротивника:

- супротивник може перехоплювати всі шифровані повідомлення, але не має відповідних їм відкритих текстів;
- супротивник може перехоплювати всі шифровані повідомлення і здобувати відповідні їм відкриті тексти;

- супротивник має доступ до шифру (але не до ключів!) і тому може зашифрувати будь-яку інформацію.

Впродовж багатьох століть серед фахівців не затихали спори про стійкість шифрів і про можливість побудови абсолютно стійкого шифру.

"Отець кібернетики" Норберт Вінер відзначав: "Будь-який шифр може бути розкритий, якщо тільки в цьому є необхідність і інформація, яку передбачається отримати, коштує витрачені засоби, зусилля і час."

Тому у користувача залишається єдиний шлях — отримання практичних оцінок стійкості. Цей шлях складається з наступних етапів.

1. Зрозуміти і чітко сформулювати, від якого супротивника необхідно захищати інформацію. Слід з'ясувати, що саме супротивник знає або може дізнатися про систему шифру, які сили і засоби він зможе застосувати для його розтину.
2. У думках стати в положення супротивника і спробувати з його позицій розкрити шифр, тобто розробити різні алгоритми розтину шифру, забезпечуючи при цьому в максимальній мірі моделювання сил, засобів і можливостей супротивника.
3. Якнайкращий з розроблених алгоритмів використовувати для практичної оцінки стійкості шифру.

Слід згадати про два прості методи розтини шифру: випадкового вгадування ключа (він спрацьовує з малою вірогідністю) і перебору всіх підряд ключів аж до знаходження істинного (він спрацьовує завжди, але має найвищу обчислювальну складність).

Аналіз основних криптографічних методів ЗІ

Іноді криптографічні методи ЗІ розділяють на три групи: ***методи підстановки***, ***методи перестановки*** і ***адитивні методи***. Методи перестановки і підстановки характеризуються хорошими ключами, а їх надійність пов'язана з складним алгоритмом перетворення. При адитивних методах користуються простими алгоритмами перетворення, забезпечуючи надійність за допомогою ключів великого об'єму.

Іноді говорять про ***блокові методи***, маючи на увазі перші дві групи, в яких алгоритм працює відразу над великим блоком інформації, і про ***потокові методи***, де шифрування відбувається знак за знаком. Проте при використанні адитивних методів перетворення може здійснюватися відразу над цілим машинним словом і метод набуває ознак блокового.

Продовження лістингу 3

```
RESTORE swappyl
DIM swappy(1 TO 64) AS shared INTEGER
init swappy(), 64

RESTORE KeyTr1l
DIM KeyTr1(1 TO 56) AS shared INTEGER
init KeyTr1(), 56

RESTORE KeyTr2l
DIM KeyTr2(1 TO 48) AS shared INTEGER
init KeyTr2(), 48

RESTORE etrl
DIM etr(1 TO 48) AS shared INTEGER
init etr(), 48

RESTORE ptrl
DIM ptr(1 TO 32) AS shared INTEGER
init ptr(), 32

sboxinit s()

RESTORE rotsl
DIM rots(1 TO 16) AS shared INTEGER
init rots(), 16

DIM XR(1 TO 56) AS shared INTEGER

DIM EF(1 TO 64) AS shared INTEGER
DIM ikeyf(1 TO 64) AS shared INTEGER
DIM yf(1 TO 64) AS shared INTEGER
DIM ades(1 TO 64) AS shared INTEGER
DIM bdes(1 TO 64) AS shared INTEGER
DIM xdes(1 TO 64) AS shared INTEGER
```

Приклад реалізації алгоритму DES представлений в лістингах 3 і 4 (компілятор — PowerBasic).

Лістинг 13. Приклад реалізації алгоритму DES на мові Basic для шифрування файлів

```
$CPU 80386
$FLOAT NPX
$OPTIMIZE SPEED
$LIB ALL-
$OPTION CNTLBREAK ON

DECLARE FUNCTION MYBIN$ (n%)

DECLARE FUNCTION desalg$ (a$)

DECLARE SUB f (i%, a%(), x%())
DECLARE SUB transpose (datax%(), T%(), n%)
DECLARE SUB mrotate (keyx%())

DECLARE SUB stob (a$, mbits%())
DECLARE SUB btos (mbits%(), a$)
DECLARE SUB letbe (target%(), source%(), LAST%)
DECLARE SUB init (x() AS INTEGER, n%)
DECLARE SUB sboxinit (b() AS INTEGER)

DECLARE SUB xtob (a$, mbits%())

DIM s(1 TO 8, 1 TO 64) AS shared INTEGER
' Ініціалізація

RESTORE InitialTrl
DIM InitialTr(1 TO 64) AS shared INTEGER
init InitialTr(), 64

RESTORE FinalTrl
DIM FinalTr(1 TO 64) AS shared INTEGER
init FinalTr(), 64
```


Шифрування методом підстановки (заміни)

У цьому, найбільш простому, методі символи шифрованого тексту замінюються іншими символами, узятими з одного (моноалфавітна підстановка) або декількох (поліалфавітна підстановка) алфавітів. Найпростішим різновидом є пряма заміна, коли букви шифрованого тексту замінюються іншими буквами того ж самого або деякого іншого алфавіту.

Якщо об'єм зашифрованого тексту великий, то частоти появи букв в зашифрованому тексті будуть ближчі до частот появи букв в алфавіті (тієї мови, на якій написаний текст) і розшифровка буде дуже простою.

Тому просту заміну в даний час використовують рідко і в тих випадках, коли шифрований текст короткий.

Для підвищення стійкості шифру використовують так звані поліалфавітні підстановки, в яких для заміни символів початкового тексту використовуються символи декількох алфавітів. Існує декілька різновидів поліалфавітної підстановки, найбільш відомими з яких є одно- (звичайна і монофонічна) і багатоконтурна.

При *поліалфавітній одноконтурній звичайній* підстановці для заміни символів початкового тексту використовується декілька алфавітів, причому зміна алфавітів здійснюється послідовно і циклічно, тобто перший символ замінюється відповідним символом першого алфавіту, другий — символом другого алфавіту і так далі до тих пір, поки не будуть використані всі вибрані алфавіти. Після цього використання алфавітів повторюється.

Як приклад розглянемо шифрування за допомогою таблиці Віженера. ***Таблиця Віженера*** є матрицею з **n2** елементами, де **n** — кількість символів використовуваного алфавіту. Кожен рядок матриці отриманий циклічним зрушенням алфавіту на символ. Для шифрування вибирається буквенний ключ, відповідно до якого формується робоча матриця шифрування. Здійснюється це таким чином. З повної таблиці вибирається перший рядок і ті рядки, перші букви яких відповідають буквам ключа. Першим розміщується перший рядок, а під нею — рядки, відповідні буквам ключа в порядку проходження цих букв в ключі.

Сам процес шифрування здійснюється таким чином:

- під кожною буквою шифрованого тексту записують букви ключа (ключ при цьому повторюється необхідну кількість разів);
- кожна буква шифрованого тексту замінюється по підматриці буквами, що знаходяться на перетині ліній, що сполучають букви шифрованого

тексту в першому рядку підматриці і букв ключа, що знаходяться під ними;

- отриманий текст може розбиватися на групи по декілька знаків.

Дослідження показали, що при використанні такого методу статистичні характеристики початкового тексту практично не виявляються в зашифрованому тексті. Неважко відмітити, що заміна по таблиці Віженера еквівалентна простій заміні з циклічною зміною алфавіту, тобто тут ми маємо поліалфавітну підстановку, причому число використовуваних алфавітів визначається числом букв в ключі. Тому стійкість такої заміни визначається прямою заміною на кількість використовуваних алфавітів, тобто на кількість букв в ключі.

Одним з недоліків шифрування по таблиці Віженера є те, що при невеликій довжині ключа надійність шифрування залишається невисокою, а формування довгих ключів зв'язане з певними труднощами.

Недоцільно вибирати ключі з буквами, що повторюються, оскільки при цьому стійкість шифру не зростає. В той же час ключ повинен легко запам'ятовуватися, щоб його можна було не записувати. Послідовність же букв, що не мають сенсу, запам'ятати важко.

З метою підвищення стійкості шифрування можна використовувати вдосконалені варіанти таблиці Віженера. Відзначимо деякі з них:

- у всіх (окрім першої) рядках таблиці букви розташовуються в довільному порядку;
- як ключ використовуються випадкові послідовності чисел.

З таблиці Віженера вибираються десять довільних рядків, які кодуються натуральними числами від 0 до 10. Ці рядки використовуються відповідно до чергування цифр у вибраному ключі.

Окремим випадком розглянутої поліалфавітної підстановки є так звана *монофонічна підстановка*. Особливість цього методу полягає в тому, що кількість і склад алфавітів вибираються так, щоб частота появи всіх символів в зашифрованому тексті була однаковою. При такому положенні важко провести криптоаналіз зашифрованого тексту за допомогою його статичної обробки. Вирівнювання частот появи символів досягається за рахунок того, що для символів початкового тексту, що часто зустрічаються, передбачається використання більшого числа замінюючих елементів, чим для тих, що рідко зустрічаються.

Поліалфавітна багатоконтурна підстановка полягає в тому, що для шифрування використовується декілька наборів (контурів) алфавітів, використовуваних циклічно, причому кожен контур в загальному випадку має свій індивідуальний період застосування. Цей період обчислюється, як

2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S7	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	4	11	2	14	15	0	8	13	3	32	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Механізм дії S-блоків

Перетворення, за допомогою якого 48-розрядний блок перетвориться в 32-розрядний, зводиться до вибірки восьми тетрад з 8 таблиць (S-блоків) розміром 4 × 16 (табл. 7). З кожного S-блока вибирається одна тетрада. Для цього 48-розрядний блок ділиться послідовно на 8 комбінацій по 6 бітів кожна. Перша комбінація (зліва) є входом в перший S-блок**,** друга — в другій і так далі При цьому перший і останній біти комбінації **задають** номер рядка, а останні 4 бита — номер стовпця S-блока, на перетині яких розташована відповідна тетрада. Конкретні значення Si (i **= 1 ., 8)** представлені в табл. 7.

Таблиця 7. Таблиці S -блоків для DES

S1	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1

правило, кількістю знаків, після зашифровування яких міняється контур алфавітів. Окремим випадком багатоконтурної поліалфавітної підстановки є заміна по таблиці Віжінера - для шифрування використовується декілька ключів, кожен з яких має свій період застосування.

Загальна модель шифрування підстановкою може бути представлена в наступному вигляді:

$$t_i^c = t_i^p + щ \bmod (К - 1)$$

де t_i^c — символ зашифрованого тексту; t_i^p — символ початкового тексту; **щ** — ціле число в діапазоні від **0** до (**К–1**); **До** — кількість символів використовуваного алфавіту.

Якщо **щ** фіксоване, то формула описує моноалфавітну підстановку, якщо **щ** вибирається з послідовності **щ1, щ2 ., щn**, то виходить полиалфавитная підстановка з періодом **n**.

Якщо в полиалфавитной підстановці **n > m** (де **m** — кількість знаків шифрованого тексту) і будь-яка послідовність **щ1, щ2 ., щn** використовується тільки один раз, то такий шифр є теоретично не розкриваним, якщо супротивник не має доступу до початкового тексту. Цей шифр називають шифром Вермана.

Шифрування методом перестановки

Цей метод полягає в тому, що символи шифрованого тексту переставляються по певних правилах усередині шифрованого блоку символів. Розглянемо деякі найбільш відомих різновидів цього методу, що часто зустрічаються: *простий*, *ускладнений по таблиці* і *ускладнений по маршрутах перестановки*.

Шифрування простою перестановкою

Шифрування простою перестановкою здійснюється таким чином:

- вибирається ключове слово з символами, що не повторюються;
- шифрований текст записується послідовними рядками під символами ключового слова;
- зашифрований текст виписується колонками в тій послідовності, в якій розташовуються в алфавіті букви ключа (або в порядку проходження цифр в натуральному ряду, якщо він цифровий).

Недоліком шифрування простою перестановкою обусловлюється тим, що при великій довжині шифрованого тексту в зашифрованому тексті

можуть виявитися закономірності символів ключа. Для усунення цього недоліку можна міняти ключ після зашифровуваної певної кількості знаків. При достатньо частій зміні ключа стійкість шифрування можна істотно підвищити. При цьому, проте, ускладнюється організація процесу шифрування і дешифровки.

Ускладнений метод перестановки по таблицях

Ускладнений метод перестановки по таблицях полягає в тому, що для запису символів шифрованого тексту використовується спеціальна таблиця, в яку введені деякі ускладнюючі елементи. Таблиця є матрицею, розміри якої можуть бути вибрані довільно (наприклад 10 × 10). У неї, як і у разі простої перестановки, записуються знаки шифрованого тексту. Ускладнення полягає в тому, що певне число кліток таблиці не використовується. Кількість і розташування невживаних елементів є додатковим ключем шифрування. Шифрований текст блоками по **m × n – S** елементів записується в таблицю (**m n** — розміри таблиці, **S** — кількість невживаних елементів). Далі процедура шифрування аналогічна простій перестановці.

Варіюючи розмірами таблиці, послідовністю символів ключа, кількістю і розташуванням невживаних елементів, можна отримати необхідну стійкість зашифрованого тексту.

Ускладнений метод перестановок по маршрутах

Вельми високу стійкість шифруванню можна забезпечити, використовуючи ускладнений метод перестановок по маршрутах типу гамильтоновских. При цьому для запису символів шифрованого тексту використовуються вершини деякого гіперкуба, а знаки зашифрованого тексту прочитуються по маршрутах Гамільтона, причому використовується декілька різних маршрутів (мал. 5).

Слід відмітити, що всі процедури шифрування і розшифровки по методу перестановки є достатньою мірою формалізованими і можуть бути реалізовані алгоритмічно.

10	2	59	51	43	35	27	14	6	61	53	45	37	29
19	11	3	60	52	44	36	21	13	5	28	20	12	4

Таблиця 3. Перетворення PC2

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Вибір бітів по таблицях 2–4 з відповідних блоків проводиться таким чином. Таблиця розглядається як послідовність її рядків, записаних один за одним, починаючи з першого рядка. Біти блоку даних відповідної довжини нумеруються зліва направо, починаючи з одиниці. Кожен елемент **s** таблиці розглядається, як номер бита **bs** в блоці даних. Перетворення полягає в заміні всіх елементів **s** на біти **bs**.

Таблиця 4. Відповідність зрушень номерам циклів DES

Номер циклу	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Зрушення вліво (шифрування)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1
Зрушення управо (розшифровка)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Циклова функція проводить наступні дії.

1. Розширення блоку **Ri-1** до 48 бітів за рахунок повторення бітів блоку за допомогою функції розширення **EP** (табл. 5).
2. Порозрядне складання результату з ключем **Xi**.
3. Перетворення отриманої суми за допомогою заміни (використовуючи так звані *S-блоки*), в результаті якого виходить блок завдовжки 32 біт.
4. Застосування перестановки **P** (табл. 6), що дає значення функції **Y = f(R,X)**.

Таблиця 5. Перетворення EP

32	1	2	3	4	5
4	5	6	7	8	9

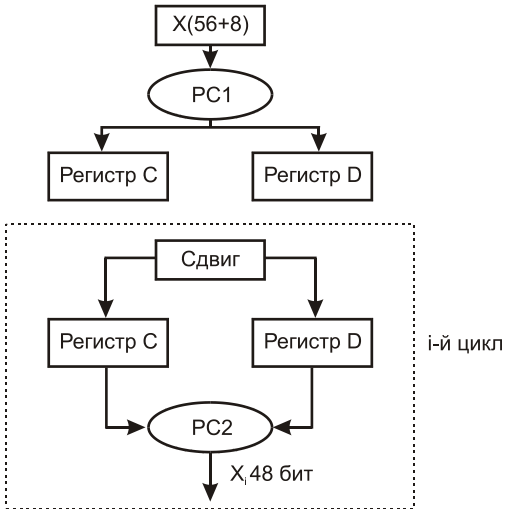
Таблиця 6. Перестановка P

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10

61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Процедура формування підключів

На кожному циклі (мал. 8) з ключа **X** довжиною 56 біт формується ключ **Xi** розміром 48 бітів. Сам ключ **X** розміщується у восьмибайтовому слові, причому восьмі розряди кожного байта є контрольними і в ключ не входять. Перед шифруванням, відповідно до процедури вибору **PC1** (табл. 2), з **X** вибираються 56 біт, якими заповнюються два регістри (**C** і **D**) завдовжки 28 бітів кожен. Надалі, при вході в черговий цикл з номером **i**, регістри зрушуються циклічно вліво. Величина зрушення залежить від номера циклу, але є фіксованою і заздалегідь відома. Після зрушення обидва підблоки об'єднуються в порядку (**C**, **D**). Потім, відповідно до функції вибору **PC2** (табл. 3), з них вибираються 48 біт підключа **Xi**. Шифрування і розшифровка відрізняються напрямом зрушень (табл. 4).



Мал. 8. Формування підключів

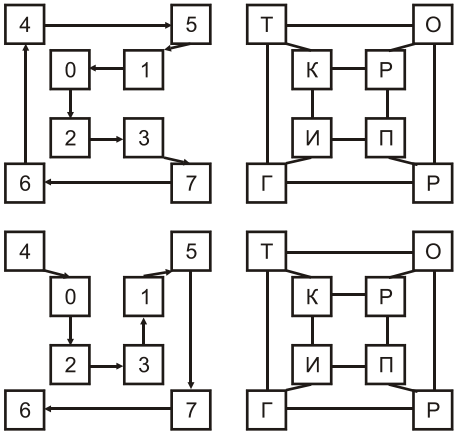
Таблиця 2. Перетворення PC1

Заповнення З							**Заповнення D**						
57	49	41	33	25	17	9	63	55	47	39	31	23	15
1	58	50	42	34	26	18	7	62	54	46	38	30	22

Шифрування за допомогою аналітичних перетворень

Достатньо надійне закриття інформації може забезпечуватися при використанні для шифрування аналітичних перетворень. Для цього можна застосовувати методи алгебри матриць, наприклад, множення матриці на вектор за правилом

$$||a_{ij}||\ b_j = C_j = \sum_j a_{ij} b_j$$



Мал. 5. Схема шифрування перестановкою по маршрутам Гамільтона.

Якщо матрицю $||a_{ij}||$ використовувати як ключ, а замість компоненти вектора b_j підставити символи початкового тексту, то компоненти вектора C_j будуть символами зашифрованого тексту.

Використовуємо як приклад цього методу квадратну матрицю третього порядку, яка гратиме роль ключа:

$$\begin{Vmatrix} 14 & 8 & 3 \\ 8 & 5 & 2 \\ 3 & 2 & 1 \end{Vmatrix}$$

Замінимо букви алфавіту цифрами, відповідними їх порядковому номеру в алфавіті. Тоді тексту **ВАТАЛІА** (текст довільний) відповідатиме

послідовність **3**, **0**, **19**, **0**, **12**, **0**. По прийнятому алгоритму шифрування виконаємо необхідні дії:

$$\begin{vmatrix} 14 & 8 & 3 \\ 8 & 5 & 2 \\ 3 & 2 & 1 \end{vmatrix} \times \begin{vmatrix} 3 \\ 0 \\ 19 \end{vmatrix} = \begin{vmatrix} 99 \\ 62 \\ 28 \end{vmatrix} \qquad \begin{vmatrix} 14 & 8 & 3 \\ 8 & 5 & 2 \\ 3 & 2 & 1 \end{vmatrix} \times \begin{vmatrix} 0 \\ 12 \\ 0 \end{vmatrix} = \begin{vmatrix} 96 \\ 60 \\ 24 \end{vmatrix}$$

Таким чином, зашифрований текст матиме наступний вигляд:

99, **62**, **28**, **96**, **60**, **24**

Розшифровка здійснюється з використанням того ж правила множення матриці на вектор, тільки як основа береться матриця, зворотна тій, за допомогою якої здійснюється захист, а як вектор-співмножник — відповідна кількість символів закритого тексту. Значеннями вектора-результату будуть цифрові еквіваленти знаків відкритого тексту.

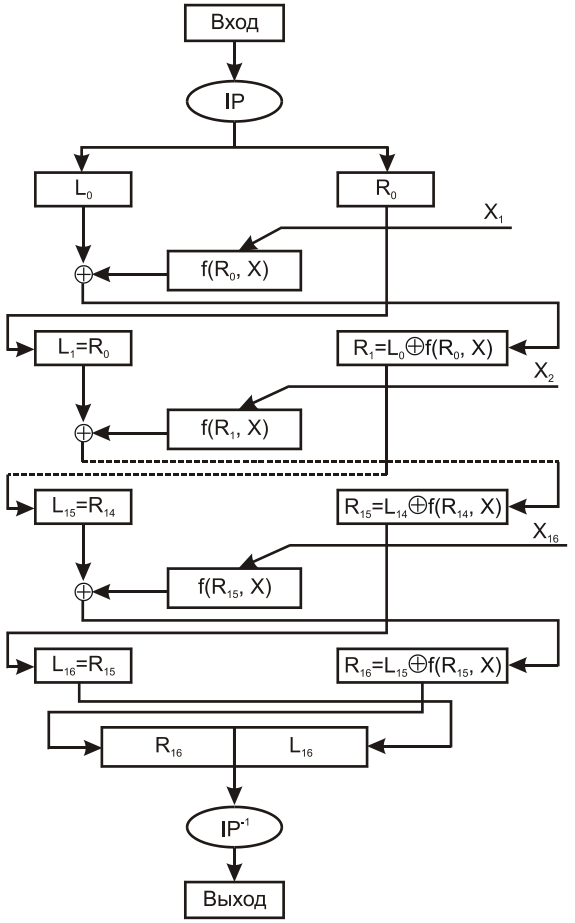
Шифрування методом гаммування

Суть цього методу полягає в тому, що символи шифрованого тексту послідовно складаються з символами деякої спеціальної послідовності, яка називається ***гаммою***. Іноді такий метод представляють як накладення гамми на початковий текст, тому він отримав назву “гаммування”.

Процедуру накладення гамми на початковий текст можна здійснити двома способами. У *першому способі* символи початкового тексту і гамми замінюються цифровими еквівалентами, які потім складаються по модулю **До**, де **До** — кількість символів в алфавіті, тобто **tc = (tp + tg) mod До**, де **tc**, **tp**, **tg** — символи відповідно зашифрованого тексту, початкового тексту і гамми.

При *другому способі* символи початкового тексту і гамми представляються у вигляді двійкового коду, а потім відповідні розряди складаються по модулю 2. Замість складання по модулю 2 при гаммуванні можна використовувати інші логічні операції, наприклад перетворення за правилом логічної еквівалентності або логічної нееквівалентності. Така заміна рівносильна введенню ще одного ключа, яким є вибором правила формування символів зашифрованого повідомлення з символів початкового тексту і гамми.

Стійкість шифрування методом гаммування визначається головним чином властивостями гамми — тривалістю періоду і рівномірністю статистичних характеристик. Останню властивість забезпечує відсутність закономірностей в появі різних символів в межах періоду.



Мал. 7. Блок-схема роботи алгоритму DES

Таблиця 1. Початкова перестановка IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3

Принцип роботи блокового шифру

Розглянемо принцип роботи блокового шифру. Входом в блоковий шифр і результатом його роботи є блок довжини **n** — послідовності, що складається з **n** бітів. Число **n** константа. При необхідності шифрування повідомлення довжиною, більшою **n**, воно розбивається на блоки, кожен з яких шифрується окремо. Різні режими роботи пов'язані з додатковими ускладненнями блокового шифру при переходах від блоку до блоку. У стандарті DES довжина блоку **n = 64.**

У режимі ECB шифрування блоку відкритого тексту **B** проводиться за 16 однотипних ітерацій, іменованих *циклами*. Схема перетворення приведена на мал. 7. Блок розглядається як конкатенація (зчеплення) двох підблоків рівної довжини: **B = (L, R)**. На кожному циклі застосовується свій ключ (**Xi**), що зазвичай виробляється з деякого основного ключа (X). Ключі, використовувані в циклах, називаються *підключами*.

Основним елементом шифру є несекретна *циклова функція* виду **Y = f(R,X)**. Входом в цикл є вихід з попереднього циклу. Якщо згаданий вхід має вигляд **(L, R)**, то вихід має вигляд **(R, L ⊕ f(R, X))**, де ⊕ — порозрядне складання по модулю 2**.** Наприклад**,** для виходу циклу з номером **i** це означає: **Ri = Li-1 ⊕ f(Ri-1, Xi), Li = Ri-1 (i = 1.,16)**.

У режимі ECB алгоритм DES зашифровує 64-бітовий блок за 16 циклів. Біти вхідного блоку перед першим циклом переставляються відповідно до табл. 1 в ході так званої *початкової перестановки* (**IP** — initial permutation). Після виходу з останнього циклу **L** і **R** переставляються місцями, після чого з'єднуються в блок. Біти отриманого блоку знову переставляються відповідно до перестановки **IP-1**, зворотної початкової. Результат приймається як блок шифртекста.

Розділяють два різновиди гаммування — з кінцевою і нескінченною гаммою. При хороших статистичних властивостях гамми стійкість шифрування визначається тільки довжиною її періоду. При цьому якщо довжина періоду гамми перевищує довжину шифрованого тексту, то такий шифр теоретично є абсолютно стійким. Це, проте, не означає, що дешифрування такого тексту взагалі не можливо: за наявності деякої додаткової інформації початковий текст може бути часткове або повністю відновлений навіть при використанні нескінченної гамми.

Як нескінченна гамма може бути використана будь-яка послідовність випадкових символів, наприклад, послідовність цифр числа π або e. При шифруванні за допомогою ЕОМ послідовність гамми формується за допомогою датчика псевдовипадкових чисел. В даний час розроблені алгоритми роботи таких датчиків, які забезпечують задовільні характеристики.

Комбіновані методи шифрування

Як вже наголошувалося, однією з найважливіших вимог, що пред'являються до системи шифрування, є її стійкість. Проте підвищення стійкості будь-якого методу шифрування приводить, як правило, до істотного ускладнення самого процесу шифрування і збільшення витрат ресурсів (часу, апаратних засобів, зменшенню пропускної спроможності і тому подібне).

Достатньо ефективним засобом підвищення стійкості шифрування є комбіноване використання декількох різних способів шифрування, тобто послідовне шифрування початкового тексту за допомогою двох або більш методів.

Стійкість комбінованого шифрування **Sk** не нижча за стійкість використовуваних способів **S: Sk ≥ П Si**.

Абсолютно очевидно, що якщо який-небудь спосіб шифрування при незалежному його застосуванні може забезпечити стійкість не нижче **Sk** (наприклад, гаммування з нескінченною гаммою), то комбінування цього способу з іншими буде доцільним лише при виконанні умови $\sum_i R_i < R^*$, де R_i — ресурсоємкість **i-го** способу, використовуваного при комбінованому шифруванні; **R*** — ресурсоємкість того способу, який забезпечує стійкість не нижче **Sk**.

Комбінувати можна будь-які методи шифрування і в будь-якій кількості, проте на практиці найбільшого поширення набули наступні комбінації:

- підстановка + гаммування;
- перестановка + гаммування;
- гаммування + гаммування;
- підстановка + перестановка.

Типовим прикладом комбінованого шифру є національний стандарт США криптографічного закриття даних (DES).

Кодування

Одним із засобів криптографічного закриття інформації, що також має тривалу історію практичного використання, є ***кодування***, під яким розуміється заміна елементів даних, що захищаються, деякими цифровими, буквенними або комбінованими поєднаннями — кодами. Неважко відмітити, що між кодуванням інформації і її шифруванням підстановкою існує значна аналогія. Проте між цими методами можна знайти відмінності.

При шифруванні підстановкою замінюваними одиницями інформації є символи алфавіту, і, отже, шифруванню можуть піддаватися будь-які дані, для фіксації яких використовується вибраний алфавіт. При кодуванні заміні піддаються смислові елементи інформації, тому для кожного спеціального повідомлення в загальному випадку необхідно використовувати свою систему кодування. Правда, останнім часом розроблені спеціальні коди, що мають на меті скоротити об'єм інформації при її записі. Специфіка цих кодів полягає в тому, що для запису символів, що часто зустрічаються, використовуються короткі двійкові коди, а для запису тих, що рідко зустрічаються — довгі. Прикладом такого коду може служити код Хоффмана.

Двійковий код для букв алфавіту утворюється шляхом послідовного запису нулів і одиниць на маршруті від вершини графа до кінця гілки, відповідного даній букві. Якщо граф кодування зберігається в таємниці, то таке кодування має криптографічну стійкість на рівні шифрування простою заміною.

При смисловому кодуванні основної кодованої одиниці є смисловий елемент тексту. Для кодування складається спеціальна таблиця код, що містить перелік кодованих елементів і відповідних ним код.

Іноді код складається із списку слів і фраз разом з відповідними їм випадковими групами чисел і букв, званими *кодовими групами*. Оскільки кодові групи зазвичай коротше за вирази, які вони представляють, коди, крім секретності, забезпечують також і стиснення інформації.

Цифровий (електронна) підпис на основі криптосистеми RSA

Асиметрична криптографія дозволяє принципово вирішити задачу підтвердження істинності електронного документа. Ця можливість заснована на тому, що зашифрувати дані, використовуючи секретний ключ **d** замість відкритого ключа **e** може тільки той, кому секретний ключ відомий. При цьому існує можливість перевірки застосування секретного ключа до даним без його розкриття.

Дійсно, хай нам необхідно завірити блок **m** відкритого тексту. Сам відкритий текст не є секретним. Зашифруємо **m** використовуючи **d** замість **e**: **з = md(mod n)**. Відправимо повідомлення подвійної довжини виду **m||c**. Одержувач має можливість перевірити наш підпис, оскільки після зведення **з** в ступінь **e** повинне виходити значення **s = m** (при дійсному підписі) і значення **s ≠ m** інакше. Для нашого прикладу **m =(3, 1, 2)**, **з = (27, 1, 8)**, **m || з = (3, 1, 2, 27, 1, 8)**.

На практиці подвоєння довжини повідомлення, очевидно, є небажаним. Це є одній з причин, по яких замість **з = md(mod n)** використовуються дані вигляду $\mathbf{з = (h(m))^d (mod\ n)}$. Тут функція **h**, звана *хеш-функцією*, відображає повідомлення довільної довжини в короткі блоки фіксованої довжини, причому так, що окрім блоку **m** підібрати інший блок **z** з властивістю h(m)= **h(z)** практично неможливе.

Стандарт шифрування даних DES

Стандарт шифрування даних (DES — Data Encryption Standard) прийнятий в США в 1977 році як федеральний. У стандарт входить опис блокового шифру типу шифру Файстеля, а також різних режимів його роботи, як складової частини декількох процедур криптографічного перетворення даних. Зазвичай під абревіатурою DES розуміється саме *блоковий шифр*, який в стандарті відповідає процедурі шифрування в режимі електронної кодової книги (ECB —E1ectronic Codebook Mode). Назва викликана тим, що будь-який блоковий шифр є простим підстановлювальним шифром і в цьому відношенні подібний до кодової книги.

Закінчення лістингу 2

```
BN_a_mod_b(n4,a,n3);

 end;
 if (i<=50) then Exit;
 BN_a_sub_b(a,none,n2);
 i := 0;
 oldseed := RandSeed;
 for j := 0 to BIGNUM_DWORD do
 begin
  n4[j]:= Random(2);
  n4[j]:= Cardinal(RandSeed);
 end;
 BN_a_mod_b(n4,a,n1);
 BN_a_exp_b_mod_c(n1,n2,a,n3);
 BN_a_sub_b(n3,none,n4);
 BN_a_mod_b(n4,a,n3);
 while (i<=50) and (BN_a_cmp_b(n3,nzero)=0) do
 begin
  Inc(i);
  if (i>50) then Break;
  for j := 0 to BIGNUM_DWORD do
  begin
   n4[j]:= Random(2);
   n4[j]:= Cardinal(RandSeed);
  end;
  BN_a_mod_b(n4,a,n1);
  BN_a_exp_b_mod_c(n1,n2,a,n3);
  BN_a_sub_b(n3,none,n4);
  BN_a_mod_b(n4,a,n3);
 end;
 RandSeed := oldseed;
 if (i<=50) then Exit;
 Result := 1;
end;
end.
```


При правильному використанні коди набагато важче розкрити, чим інші класичні системи. Успіх їх використання пояснюється трьома причинами. Найбільш важливою із них є велика довжина використовуваного ключа. У типовій системі шифрування використовується ключ завдовжки максимум декілька сотень бітів. Наприклад, ключем шифру на основі простої підстановки є переставлений алфавіт, що представляє в середньому 90 бітів, тоді як кодова книга хорошого розміру може містити сотні тисяч і навіть мільйон бітів. Як показав Шенон, робота криптоаналітика важка, коли з повідомлення видаляється надмірність. Причому, коди працюють з відносно великими блоками відкритого тексту (словами і фразами) і, отже, приховують локальну інформацію, яка інакше могла б дати цінні "зачіпки" для криптоаналізу.

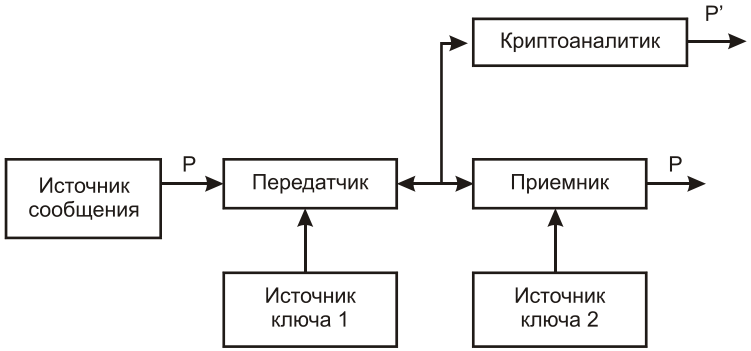
До *недоліків* слід віднести те, що ключ при кодуванні використовується недостатньо добре, оскільки при кодуванні окремого слова і фрази використовується лише дуже мала частина кодової книги. В результаті код при інтенсивному використанні піддається частковому аналізу і виявляється особливо чутливим до розтину за наявності відомого відкритого тексту. По цих причинах для забезпечення більшої надійності коди необхідно частіше міняти.

До інших видів криптографічного захисту віднесені розтин і стиснення даних. ***Розтин даних*** полягає в тому, що масив даних, що захищаються, розтинається на елементи, кожен з яких не дозволяє розкрити зміст інформації, що захищається, і виділені таким чином елементи розміщуються в різних зонах пам'яті. Зворотна процедура називається збіркою даних. Абсолютно очевидно, що алгоритм збірки даних повинен зберігатися в таємниці.

Шифрування з відкритим ключем

Одне з головних обмежень використання звичайних криптографічних систем пов'язане з трудністю розповсюдження ключів. Діффі і Хеллман, а також, незалежно від них, Меркль, показали, що можна виключити захищений канал передачі ключів і при цьому забезпечити захист при передачі повідомлень по незахищеному каналу без здійснення яких-небудь попередніх заходів. Як видно з мал. 6, між відправником і одержувачем допускається двосторонній обмін, але перехоплювач тут пасивний і лише слухає. На відміну від звичайних систем, в яких ключ

повинен зберігатися в секреті, системи, що допускають таку роботу, називаються ***системами з відкритим ключем***.



Мал. 6. Потік інформації в криптографічній системі з відкритим ключем

Для вирішення цієї проблеми пропонуються два підходи. При відкритому розповсюдженні ключів відправник і одержувач можуть домовитися про ключ, використовуваний в звичайній криптографічній системі. Не дивлячись на те, що супротивник слухає всі переговори, він не в змозі обчислити ключ і не може зрозуміти подальшого обміну повідомленнями. Другий підхід реалізує криптографічні системи з відкритими ключами, в яких для шифрування використовуються різні ключі.

Причина, по якій ключі в звичайних криптографічних системах повинні так ретельно захищатися, полягає в тому, що функції шифрування і дешифровки в ній нерозлучні. Будь-яка особа, що отримала ключ для шифрування повідомлень, може також дешифрувати повідомлення. Якщо засоби шифрування розділені, то секретність можна забезпечити без засекречування ключа шифрування, оскільки його не можна використовувати для розшифровки.

Взаємодіючи по відкритих каналах зв'язку, абоненти А і В вирішують наступні завдання:

- спочатку у А і В немає ніякої загальної секретної інформації, але в кінці процедури така загальна секретна інформація (загальний ключ) у А і В з'являється, тобто виробляється;
- супротивник, який перехоплює всі передачі і знає, що хоче отримати А і В, проте не може відновити вироблений загальний ключ А і В.

Продовження лістингу 2

```
 if (BN_a_cmp_b(a,nzero)=0) then Exit;
if (BN_a_cmp_b(a,none)=0) then begin Result := 1;
Exit; end;
 if (BN_a_cmp_b(a,nn) <=0) then
 begin
  i := 0;
  while (i<=53) and (Cardinal(primes[i]) <>a[0]) do
Inc(i);
  if (i>53) then Exit;
  Result := 1;
  Exit;
 end;
 Move(nzero,n1,sizeof(nzero));
 i := 0;
 n1[0]:= primes[i];
 BN_a_mod_b(a,n1,n2);
 while (i<=53) and (BN_a_cmp_b(n2,nzero) >0) do
 begin
  Inc(i);
  if (i>53) then Break;
  n1[0]:= primes[i];
  BN_a_mod_b(a,n1,n2);
 end;
 if (i<=53) then Exit;
 Move(nzero,n1,sizeof(nzero));
 BN_a_sub_b(a,none,n2);
 i := 0;
 n1[0]:= primes[i];
 BN_a_exp_b_mod_c(n1,n2,a,n3);
 BN_a_sub_b(n3,none,n4);
 BN_a_mod_b(n4,a,n3);
 while (i<=50) and (BN_a_cmp_b(n3,nzero)=0) do
 begin
  Inc(i);
if (i>50) then Break;
n1[0]:= primes[i];
  BN_a_exp_b_mod_c(n1,n2,a,n3);
  BN_a_sub_b(n3,none,n4);
```

Продовження лістингу 2

```
begin
 for i := 0 to BIGNUM_DWORD do res[i]:= 0;
 for i := 0 to BIGNUM_DWORD do nzero[i]:= 0;
 for i := 0 to BIGNUM_DWORD do none[i]:= 0; none[0]:=
1;
 BN_ab_GCD(a,b,n4);
 if (BN_a_cmp_b(n4,none) <>0) then Exit;
 Move(b,n1,sizeof(a));
 Move(a,n2,sizeof(a));
 Move(none,n7,sizeof(a));
 repeat
  BN_a_div_b(n1,n2,n3);
  BN_a_mod_b(n1,n2,n4);
  Move(n2,n1,sizeof(n2));
  Move(n4,n2,sizeof(n2));
  BN_a_mul_b(n3,n7,n5);
  BN_a_sub_b(res,n5,n6);
  Move(n7,res,sizeof(n7));
  Move(n6,n7,sizeof(n6));
 until (BN_a_cmp_b(n4,nzero)=0);
 if (res[BIGNUM_DWORD] and $80000000 <> 0) then
 begin
  BN_a_add_b(res,b,n7);
  Move(n7,res,sizeof(n6));
 end;
end;

function BN_PrimeTest(var a: TBigNum): Integer;
var i,j: Integer;
var oldseed: LongInt;
var nzero,none,nn: TBigNum;
var n1,n2,n3,n4: TBigNum;
begin
 Result := 0;
 for i := 0 to BIGNUM_DWORD do nzero[i]:= 0;
 for i := 0 to BIGNUM_DWORD do none[i]:= 0; none[0]:=
1;
 for i := 0 to BIGNUM_DWORD do nn[i]:= 0; nn[0]:=256;
```


Запропоновано вирішувати ці завдання за допомогою функції **F(x)= бx (mod p)**, де **p** — велике просте число, **x** — довільне натуральне число, **би** — деякий примітивний елемент поля **G F(p)**.

Примітивним називається такий елемент **бі** з **G F(p)**, що кожен елемент поля, може бути представлений у вигляді **б.** Доводиться, що примітивний елемент завжди існує.

Загальновизнано, що інвертування функції **бx (mod p)**, тобто дискретне логарифмування, є важким математичним завданням.

Саму ж процедуру або, як прийнято говорити, протокол вироблення загального ключа, можна описати таким чином.

Числа **p** і **б** вважаються загальнодоступними.

Абоненти А і В незалежно один від одного випадково вибирають поодинці натуральному числу — скажемо **xA** і **xB** і. Ці елементи вони тримають в секреті. Далі кожен з них обчислює новий елемент:

$$yA = б^{x_A} \pmod p \quad yB = б^{x_B} \pmod p$$

Потім вони обмінюються цими елементами по каналу зв'язку. Тепер абонент А, отримавши **yB** і знаючий свій секретний елемент **xA**, обчислює новий елемент:

$$y_B^{x_A} = (б^{x_B})^{x_A} \pmod p$$

Аналогічно поступає абонент В:

$$y_A^{x_B} = (б^{x_A})^{x_B} \pmod p$$

З властивостей поля виходить, що тим самим у А і В з'явиться загальний елемент, який і є загальним ключем А і В.

З опису протоколу видно, що супротивник знає **p**, **би**, $б^{x_A} б^{x_B}$ не знає **xA**, **xB** і хоче дізнатися $б^{x_A x_B}$. В даний час немає алгоритмів дій супротивника, ефективніших, ніж дискретне логарифмування, а це — важке математичне завдання.

Ці системи повинні розроблятися так, щоб полегшити генерацію випадкових пар інверсних ключів **Е** для шифрування і **Д** для дешифровки і роботу з цими ключами, але щоб обчислення **Д** по **Е** було обчислювальне таким, що не реалізовується.

Криптографічна система з відкритим ключем є парою сімейств алгоритмів $\{EK\}_{K\{K\}\in}$ і $\{ ДК\}_{K\{K\}\in}$, що визначають оборотні перетворення

ЕК:{M}→{m}
ДК:{M}→{m}⁻

на кінцевому просторі повідомлень {M} з наступними властивостями.

1. Для кожного **K{K} ∈Дk** назад до **ЕК**, тобто при будь-яких **До** і **М** справедливо **ДКЕК(М)= М.**
2. Для кожного **K{K} ∈**і **M{M}∈** неважко обчислити величини **ТЕНЬК(М)** і **ДК(М)**.
3. Для майже кожного **K{K}∈** неможливо в обчислювальному відношенні вивести з **ТЕНЬК** який-небудь легко здійснимий алгоритм, еквівалентний **ДК**.
4. По кожному заданому **K{K} ∈**можна отримати інверсну пару **ТЕНЬК** і **ДК**.

Властивість 3 дозволяє не засекречувати ключі шифрування користувача **ТЕНЬК** і при цьому не компроментувати секретність ключа дешифровки **ДК**. Отже, криптографичні системи розпадаються на дві частини (сімейство перетворень шифрування і сімейство перетворень дешифровки) таким чином, по даному членові одного сімейства неможливо визначити відповідний член іншого.

Властивість 4 гарантує наявність шляху обчислення відповідних пар зворотних перетворень, що реалізовуються, коли не накладено ніяких обмежень на те, яким повинне бути перетворення шифрування або дешифровки. На практиці криптографічне устаткування повинне містити генератор дійсних випадкових чисел для генерації **До**, а також що генерує пару **ЕК** і **ДК** по заданому **К**.

Система такого роду спрощує проблему розподілу ключів. Кожен користувач генерує пару взаємно зворотних перетворень **Е** і **Д.** Отримаємо перетворення дешифровки **Д** в секреті, а перетворення шифрування публікує у відкритому довіднику на зразок технічного довідника. Тепер будь-який охочий може шифрувати повідомлення і посилати їх користувачеві, але ніхто, окрім нього, не може дешифрувати призначені для нього повідомлення.

Замість приведених умов 1–4 безліч перетворень забезпечує, що для кожного **K{K}∈ ЕК** є зворотним **Дk**, тобто при будь-яких **До** і **М** справедливе затвердження **ЕКДК(М)= М**, то можливо, а часто і бажано здійснювати шифрування за допомогою ключа **Д**, а дешифровка — за

Продовження лістингу 2

```
Inc(i);

 end;
 Move(n2,res,sizeof(n2));
end;

procedure BN_ab_GCD(var a,b,res: TBigNum);
var i: Integer;
var n1,n2,n3,nzero: TBigNum;
begin
 res[0]:= 1;
 for i := 1 to BIGNUM_DWORD do res[i]:= 0;
 for i := 0 to BIGNUM_DWORD do nzero[i]:= 0;
 if (BN_a_cmp_b(a,nzero)=0) or
(BN_a_cmp_b(b,nzero)=0) then Exit;
 if (BN_a_cmp_b(a,b) >0) then
 begin
  Move(a,n1,sizeof(a));
  Move(b,n2,sizeof(a));
 end
 else
 begin
  Move(b,n1,sizeof(a));
  Move(a,n2,sizeof(a));
 end;
 while (BN_a_cmp_b(n2,nzero) <>0) do
 begin
  BN_a_mod_b(n1,n2,n3);
  Move(n2,n1,sizeof(n1));
  Move(n3,n2,sizeof(n3));
 end;
 Move(n1,res,sizeof(n1));
end;

procedure BN_a_modinv_b(var a,b,res: TBigNum);
var i: Integer;
var n1,n2,n3,n4,n5,n6,n7: TBigNum;
var nzero,none: TBigNum;
```

Продовження лістингу 2

```
 BN_a_shr_k(n2,1,n3);

Move(n3,n2,sizeof(n3));
   Dec(k);
  end;
  BN_a_sub_b(n1,n2,n3);
  Move(n3,n1,sizeof(n3));
  BN_a_setbit_k(res,k);
 end;
end;

procedure BN_a_exp_b_mod_c(var a,b,c,res: TBigNum);
var i,n: Integer;
var n1,n2,n3: TBigNum;
begin
 FillChar(n3,sizeof(n3),0);
 if (BN_a_cmp_b(c,n3)=0) then Exit;
 for i := 0 to BIGNUM_DWORD do res[i]:= 0;
 if (BN_a_cmp_b(b,n3)=0) then
 begin
  res[0]:= 1;
  Exit;
 end;
 Move(a,n1,sizeof(a));
 for i := 0 to BIGNUM_DWORD do n2[i]:= 0;
 n2[0]:= 1;
 n := BN_a_upbit(b)-1;
 i := 0;
 while (i<=n) do
 begin
  if ( (b[i shr 5] shr (i and 31)) and 1 = 1 ) then
  begin
   BN_a_mul_b(n2,n1,n3);
   BN_a_mod_b(n3,c,n2);
  end;
  BN_a_mul_b(n1,n1,n3);
  BN_a_mod_b(n3,c,n1);
```


допомогою ключа **Е. По** цій причині часто називають **ЕК** *відкритим ключем*, а **Дк** — *особистим ключем*.

За час, минувший після того, як була запропоновані ця система, розроблено декілька шляхів її реалізації.

Цифровий підпис

Ідея ***цифрового підпису*** (її ще називають ***електронним підписом***) була запропонована Діффі і Хеллманом. Суть її полягає у використанні односторонньої функції з секретом **FК**. В даний час ця ідея реалізована у великій кількості систем передачі даних. Повідомлення, підписане цифровим підписом, можна представити у вигляді пари **(x,y)**, де **x** — повідомлення, **FК: x → y — одностороння** функція, відома всім взаємодіючим абонентам, **y** — вирішення рівняння **FК(y)= x**. З визначення функції **FК** очевидні наступні достоїнства цифрового підпису.

1. Підписати повідомлення **x**, тобто вирішити рівняння **FK(y)= x**, може тільки абонент, що є володарем даного секрету **До**; іншими словами, підроблювати підпис неможливо.
3. Перевірити достовірність підпису може будь-який абонент, що знає відкритий ключ, тобто саму функцію **FK**.
4. При виникненні суперечок відмовитися від підпису неможливо.
5. Підписані повідомлення **(x,y)** можна, не побоюючись збитку, пересилати по будь-яких каналах зв'язку.

Саме перераховані достоїнства і зумовили широке застосування і розповсюдження систем цифрового підпису.

Як практично виглядає використання цифрового підпису? Розглянемо, як здійснюється робота банку з платіжними дорученнями своїх клієнтів. Всі абоненти цієї мережі знають односторонню функцію **FK**, і кожен клієнт має власний, нікому невідомий секрет **К.** Клієнт підписує платіжне доручення **x** за допомогою функції **FK** зі своїм секретом До і посилає підписане платіжне доручення в банк. Банк, отримавши повідомлення від клієнта і знаючи ключ, перевіряє достовірність підпису клієнта і лише після цього виконує його платіжне доручення. Через відмічені достоїнства цифрового підпису і банк, і клієнт упевнені, що їх інтереси не постраждають.

Широкий розвиток систем електронних платежів, електронної пошти і інших систем передачі даних зажадало великої різноманітності цифрових підписів. Це привело до розвитку теорії протоколів цифрового підпису,

який в даний час складає великий розділ теоретичної криптографії. В рамках цієї теорії систематизовані різні види взломів систем цифрового підпису, різні види успіхів, яких супротивник може досягти, різні види стійкості схем цифрового підпису. Вдалося також довести еквівалентність існування двох гіпотетичних об'єктів: односторонньої функції і стійкої схеми цифрового підпису.

Криптографічна система RSA

Як би не були складні і надійні класичні криптографічні системи, їх слабким місцем при практичній реалізації є проблема розподілу ключів. Для того, щоб був можливий обмін конфіденційною інформацією між двома абонентами, ключ повинен згенерувати одним з них, а потім яким-небудь чином переданий іншому в конфіденційному порядку. У загальному випадку для передачі ключа по каналах зв'язку потрібне використання ще однієї криптосистеми, для якої знов виникає проблема розподілу ключів і так далі

Для вирішення цієї і ряду інших проблем були запропоновані ***криптосистеми з відкритим ключем***, звані також ***асиметричними криптосистемами***.

Перед відправкою повідомлення адресата початковий текст шифрується *відкритим* (загальнодоступним) ключем адресата. Алгоритм шифрування побудований таким чином, що розшифровка повідомлення можлива тільки з використанням *особистого* (секретного) ключа адресата.

Вперше модель системи секретного зв'язку з відкритим ключем була запропонована Діффі і Хеллманом в 1976 році.

Суть цієї моделі полягає в тому, що ключ відомий повністю тільки одержувачеві повідомлення і є трійкою чисел **До = (e, d, n)**, де підключ **e** служить ключем шифрування, а ключ **d** — ключем розшифровки. При цьому тільки **d** є секретним (особистим) ключем. Стійкість системи забезпечується за рахунок особливих властивостей шифрперетворення, яке є так званою *односторонньою функцією з лазівкою*. Обчислення значення такої функції (від відкритого тексту і параметра **e**) повинне бути нескладним, в той же час її звернення повинне бути обчислювальне таким, що не реалізовується без знання секретної інформації, "лазівки", пов'язаної з секретним ключем **d**.

У криптосистемі з відкритим ключем повідомлення, призначене абонентові, зашифровується відправником за допомогою ключа **e** і розшифровується одержувачем за допомогою ключа **d**. Якщо

Продовження лістингу 2

```
procedure BN_a_mod_b(var a,b,res: TBigNum);
var до: Integer;
var n1,n2,n3: TBigNum;
begin
 FillChar(n3,sizeof(n3),0);
 if (BN_a_cmp_b(b,n3)=0) then Exit;
 Move(a,n1,sizeof(a));
 while (BN_a_cmp_b(n1,b) >=0) do
 begin
  до := BN_a_upbit(n1) - BN_a_upbit(b);
  BN_a_shl_k(b,k,n2);
  if (BN_a_cmp_b(n2,n1) >0) then
  begin
   BN_a_shr_k(n2,1,n3);
   Move(n3,n2,sizeof(n3));
  end;
  BN_a_sub_b(n1,n2,n3);
  Move(n3,n1,sizeof(n3));
 end;
 Move(n1,res,sizeof(n1));
end;

procedure BN_a_div_b(var a,b,res: TBigNum);
var до: Integer;
var n1,n2,n3: TBigNum;
begin
FillChar(res,sizeof(res),0);
 FillChar(n3,sizeof(n3),0);
 if (BN_a_cmp_b(b,n3)=0) then Exit;
 Move(a,n1,sizeof(a));
 while (BN_a_cmp_b(n1,b) >=0) do
 begin
  до := BN_a_upbit(n1) - BN_a_upbit(b);
  BN_a_shl_k(b,k,n2);
  if (BN_a_cmp_b(n2,n1) >0) then
  begin
```

Продовження лістингу 2

```
for j := 0 to BIGNUM_DWORD do res[j]:= 0;
i := до div 32;
 if (i>BIGNUM_DWORD) then Exit;
 for j := i to BIGNUM_DWORD do
  res[j-i]:= a[j];
 i := до mod 32;
 if (i=0) then Exit;
 u := 0;
 for j := BIGNUM_DWORD downto 0 do
 begin
  d := res[j] shl (32-i);
  res[j]:= (res[j] shr i) + u;
  u := d;
 end;
end;

function BN_a_upbit(var a: TBigNum): Integer;
var i,j: Integer;
begin
 i := BIGNUM_DWORD;
 while (i>=0) and (a[i]=0) do Dec(i);
Result := 0;
 if (i<0) then Exit;
 j := 31;
 while (j>0) and (a[i] and (1 shl j) = 0) do Dec(j);
 Result := i*32 + j + 1;
end;

procedure BN_a_setbit_k(var a: TBigNum; до: Integer);
begin
 if (k<0) or (k>32*BIGNUM_DWORD-1) then
 begin
  Exit;
 end;
 a[до shr 5]:= a[до shr 5] or (1 shl (до and 31));
end;
```


шифрперетворення дійсно є односторонньою функцією, то сам відправник не в змозі розшифрувати сформовану ним криптограму.

Широко відомим прикладом криптосистеми з відкритим ключем є криптосистема ***RSA***, що розроблена в 1977 році і отримала назву на честь її творців: Рівеста, Шаміра і Ейдельмана. Стійкість цієї системи ґрунтується *на складності оборотності статечної функції* в кільці вирахувань цілих чисел по складеному модулю **n** (при належному виборі модуля).

Необхідні відомості з елементарної теорії чисел

1. *Простим числом* називається натуральне число, що має тільки два нерівні натуральні дільники.
2. Кожне натуральне число єдиним чином, з точністю до порядку запису співмножників, представляється у вигляді *ступенів простих чисел.*
3. *Найбільшим загальним дільником* двох цілих чисел **НОД(a,b)** (або **(a,b)**) називається найбільше ціле, на яке без залишку ділиться як **a,** так і **b**.
4. Хай **a > b** і **d = (a,b)**. Тоді існують цілі **x** і **y**, рівняння, що є *рішенням,* **xa + yb = d**. Якщо **d = 1**, то **a** і **b** називаються *взаємно простими*.
5. Найбільшого загального дільника двох чисел можна знайти за допомогою алгоритму Евкліда. Для цього **a** ділиться із залишком на **b**, тобто **a = q1b + r1**. Далі замість **a** і **b**, розглядаємо відповідно **b** і **r1**: **b = q2r1+ r2**. На наступному кроці роль **b** і **r1**, грають **r1** і **r2**: **r1 = q3r2 + r3** і так далі Процес закінчується на деякому кроці **k+1**, для якого **rk+1= 0**. Тоді **НОД(a,b)= rk**.
6. Для вирішення рівняння **xa + yb = d** можна використовувати дані, отримані в кожному кроці алгоритму Евкліда, рухаючись від низу до верху, за допомогою виразу залишку через інші елементи, використовувані у відповідному кроці. Наприклад, з **r2 = q4r3 + r4** слідує **r4 = r2 +q4r3**. У останній рівності **r3** можна замінити, виходячи із співвідношення **r1 = q3r2 + r3**, тобто **r4 = r2 - q4(q3r2 - r1)**. Тому **r4 = (1 - q4q3)r2 + q4r1**. Таким чином, ми виразили **r4** у вигляді цілочисельної комбінації залишків з меншими номерами, які, у свою чергу, можуть бути виражені аналогічно. Просуваючись “від низу до верху”, врешті-решт, ми виразимо **r4**

через початкові числа **a** і **b**. Якби ми почали не з **r4**, а з **rk**, то отримали б **rk = xa + yb = d**. Число **a** порівняно з числом **b** по модулю **n**, якщо **a – b** ділиться на **n**. Запис даного твердження має наступний вигляд: **a = b(mod n)**. Найменше ненегативне число **a,** таке, що **a = A(mod n)** називається ***вирахуванням*** числа **A** по модулю **n**. Якщо **(a,n) = 1**, то існує **x**, таке, що **x = a-1 (mod n)**. Дійсно **(a,n) = 1 = d = ax + ny**, тому **ax = 1(mod n)**. Таке число **x** називається ***зворотнім*** до **a** по модулю **n** і записується у вигляді **a-1 (mod n)**.

7. Хай функція (n)**φ**, де **n** — натуральне число, рівна кількості натуральних чисел, менших **n**, для яких **(a,n)=1**. Така функція називається *функцією Ейлера*. Для чисел **n** виду $n = \prod_i p_i$ (**pi** — просте) функція Ейлера визначається як $ц(n) = \prod_i (p_i - 1)$.
8. ***Теорема Ейлера***. Хай **(a,n) = 1**. Тоді **aц(n)= 1(mod n)**.
   Наслідок. Якщо **ed = 1(mod ц(n))** і **(a, n) = 1**, то $(ae)^d = a(\text{mod } n)$.
9. Для більшості вирахувань по модулю **n = pq** показник ступеня в співвідношенні **aц(n)= 1(mod n)** може бути зменшений, але в цьому випадку він залежить від **a**. Найменший показник **до(a)**, для якого **ak(a)= 1(mod n)**, називається *порядком числа* **a** *по модулю* **n** і позначається як **ordn(a)**. Для будь-якого **a** значення **ordn(a)** є дільником значення функції Ейлера **ц(n)**.

Алгоритм RSA

Криптосистема RSA на кожному такті шифрування перетворює двійковий блок відкритого тексту **m** довжини **size(n)**, що розглядається як ціле число, відповідно до формули: **з = me(mod n)**.

При цьому **n = pq**, де **p** і **q** — випадкові прості числа великої розрядності, які знищуються після формування модуля і ключів. Відкритий ключ складається з пари чисел **e** і **n**. Підключ **e** вибирається як достатньо велике число з діапазону **1 < e < ц(n)**, з умовою: **НОД(e (n))= φ1**, де (n) — найменше загальне кратне чисел **p–1** і **q–1**. Далі, вирішуючи в цілих числах **x**, **y** рівняння **xe + yц(n)= 1**, вважається **d = x**, тобто **ed = 1((n))**. При цьому для всіх **m** виконується співвідношення **med = m(n)**, тому знання **d** дозволяє розшифровувати криптограми.

Продовження лістингу 2

```
add [ebx],eax

popad
  end;
 end;
end;

procedure BN_a_shl_k(var a: TBigNum; до: Integer; var
res: TBigNum);
var i,j: Integer;
var d,u: Cardinal;
begin
 for j := 0 to BIGNUM_DWORD do res[j]:= a[j];
 if (k<=0) then Exit;
 for j := 0 to BIGNUM_DWORD do res[j]:= 0;
 i := до div 32;
if (i>BIGNUM_DWORD) then Exit;
 for j := i to BIGNUM_DWORD do
  res[j]:= a[j-i];
 i := до mod 32;
 if (i=0) then Exit;
 d := 0;
 for j := 0 to BIGNUM_DWORD do
 begin
  u := res[j] shr (32-i);
  res[j]:= (res[j] shl i) + d;
  d := u;
 end;
end;

procedure BN_a_shr_k(var a: TBigNum; до: Integer;
                     var res: TBigNum);
var i,j: Integer;
var d,u: Cardinal;
begin
 for j := 0 to BIGNUM_DWORD do res[j]:= a[j];
 if (k<=0) then Exit;
```

Продовження лістингу 2

```
 for j := 0 to BIGNUM_DWORD do res[j]:= 0;


for i := 0 to BIGNUM_DWORD do
 begin
  j := i*4;
  asm
   pushad
   mov esi,[a]
   mov edi,[b]
   add edi,[j]
   mov ebx[res]
   add ebx,[j]
   mov ecx,BIGNUM_DWORD
   sub ecx,[i]
   cmp ecx,0
   je  @_mul_2
  @_mul_1:
   mov eax,[esi]
   mov edx,[edi]
   mul edx
   add [ebx],eax
   adc [ebx+4],edx
   pushfd
   cmp ecx,1
   je  @_mul_1_1
   popfd
   adc dword ptr[ebx+8],0
   pushfd
  @_mul_1_1:
   popfd
   add esi,4
   add ebx,4
   loop @_mul_1
  @_mul_2:
   mov eax,[esi]
   mov edx,[edi]
   mul edx
```


Щоб гарантувати надійний захист інформації, до систем з відкритим ключем пред'являються дві наступні вимоги.

1. Перетворення початкового тексту повинне виключати його відновлення на основі відкритого ключа.
2. Визначення закритого ключа на основі відкритого також повинне бути обчислювальне таким, що не реалізовується. При цьому бажана точна нижня оцінка складності (кількості операцій) розкриття шифру.

Алгоритми шифрування з відкритим ключем набули широкого поширення в сучасних інформаційних системах.

Розглянемо побудову криптосистеми RSA на простому прикладі.

1. Виберемо **p = 3** і **q = 11.**
2. Визначимо **n = 3 • 11 = 33.**
3. Знайдемо (n) = **(φp – 1)(q – 1) = 20.**
4. Виберемо **e**, взаємно просте з **20**, наприклад, **e = 7.**
5. Виберемо число **d**, задовольняюче **7d = 1(mod 20)**.

Легко побачити, що **d = 3(mod 20)**.

Представимо шифроване повідомлення як послідовність цілих чисел за допомогою відповідності: **A = 1**, **B = 2**, **3 = 3 ..., Z = 26**. Оскільки **size(n)= 6**, то наша криптосистема в змозі зашифровувати букви латинського алфавіту, що розглядаються як блоки, Опублікуємо відкритий ключ **(e, n) = (7, 33)** і запропонуємо іншим учасникам системи секретного зв'язку зашифровувати з його допомогою повідомлення, що направляються в нашу адресу. Хай таким повідомленням буде `CAB`, яке у вибраному нами кодуванню приймає вигляд **(3, 1, 2)**. Відправник повинен зашифрувати кожен блок і відправити зашифроване повідомлення до нашої адреси:

```
RSA(C)= RSA(3)= 37 = 2187 = 9(mod 33);
RSA(A)= RSA(1)= 17 = 1(mod 33);
RSA(B)= RSA(1)= 27 = 128 = 29(mod 33).
```

Отримавши зашифроване повідомлення **(9, 1, 29)**, ми зможемо його розшифрувати на основі секретного ключа **(d, n) = (3, 33)**, підносячи кожен блок до ступеня **d = 3**:

```
93 = 729 = 3(mod 33);
13 = 1(mod 33);
293 = 24389 = 2(mod 33).
```

Для нашого прикладу легко знайти секретний ключ перебором. На практиці це неможливо, оскільки для використання на практиці рекомендуються в даний час наступні значення **size(n)**:

- 512–768 бітів — для приватних осіб;
- 1024 бітів — для комерційної інформації;
- 2048 бітів — для секретної інформації.

Приклад реалізації алгоритму RSA представлений в лістингах 1 і 2 (компілятори — Delphi, FreePascal).

Лістинг 1. Приклад реалізації алгоритму RSA на мові Pascal

```
program Rsa;
$APPTYPE CONSOLE}$IFDEF FPC} {$MODE DELPHI}$ENDIF}

uses SysUtils, uBigNumber;

//Генератор випадкових чисел
var t: array[0..255] of Byte;
var pos: Integer;
var cbox: array[0..255] of Byte =
(237, 240, 161, 1, 130, 141, 205, 98, 27, 169, 181,
202, 173, 47, 114, 224, 35, 183, 79, 82, 153, 220,
172, 22, 17, 11, 200, 131, 14, 154, 167, 91, 250, 31,
213, 112, 126, 241, 236, 155, 198, 96, 87, 143, 244,
151, 134, 38, 129, 233, 186, 101, 41, 94, 231, 115,
113, 199, 51, 145, 229, 37, 69, 180, 85, 33, 207,
163, 102, 187, 4, 89, 7, 44, 75, 88, 81, 120, 10,
232, 221, 168, 230, 158, 247, 211, 216, 156, 95, 64,
242, 215, 77, 165, 122, 5, 15, 119, 100, 43, 34, 48,
30, 39, 195, 222, 184, 92, 78, 135, 103, 166, 147,
32, 60, 185, 26, 251, 214, 90, 139, 45, 73, 150, 97,
116, 136, 68, 219, 248, 191, 192, 16, 8, 243, 50,
132, 105, 62, 201, 204, 65, 0, 99, 182, 121, 194,
108, 160, 170, 56, 226, 206, 254, 117, 178, 9, 197,
234, 127, 58, 171, 40, 29, 177, 142, 3, 228, 188,
162, 212, 157, 49, 175, 174, 140, 70, 106, 123, 66,
```


Продовження лістингу 2

```
end;
end;

procedure BN_a_sub_b(var a,b,res: TBigNum);
begin
 asm
  pushad
  mov esi,[a]
  mov edi,[b]
  mov ebx,[res]
  mov ecx,BIGNUM_DWORD
  mov eax,[esi]
  sub eax,[edi]
  pushfd
  mov [ebx],eax
  add esi,4
  add edi,4
  add ebx,4
 @_sub_1:
  mov eax,[esi]
  popfd
  sbb eax,[edi]
  pushfd
  mov [ebx],eax
  add esi,4
  add edi,4
  add ebx,4
  loop @_sub_1
 @_sub_2:
  popfd
  popad
 end;
end;procedure BN_a_mul_b(var a,b,res: TBigNum);
var i,j: Integer;
begin
```

```
function  BN_a_cmp_b(var a,b: TBigNum): Integer;
var i: Integer;
```

Продовження лістингу 2

```
begin
 i := BIGNUM_DWORD;
 while (i>=0) and (a[i]=b[i]) do Dec(i);
 Result := 0;
 if (i>=0) and (a[i]>b[i]) then Result := 1;
 if (i>=0) and (a[i]<b[i]) then Result := -1;
end;

procedure BN_a_add_b(var a,b,res: TBigNum);
begin
 asm
  pushad
  mov esi,[a]
  mov edi,[b]
  mov ebx,[res]
  mov ecx,BIGNUM_DWORD
  mov eax,[esi]
  add eax,[edi]
  pushfd
  mov [ebx],eax
  add esi,4
  add edi,4
  add ebx,4
 @_add_1:
  mov eax,[esi]
  popfd
  adc eax,[edi]
  pushfd
  mov [ebx],eax
  add esi,4
  add edi,4
  add ebx,4
  loop @_add_1
 @_add_2:
  popfd
  popad
```


Продовження лістингу 1

```
196, 246, 179, 42, 218, 71, 217, 227, 18, 164, 24,
67, 159, 25, 111, 255, 193, 245, 2, 238, 133, 21,
137, 152, 109, 148, 63, 124, 203, 104, 54, 55, 223,
80, 107, 210, 225, 149, 252, 76, 12, 189, 93, 46, 23,
13, 36, 209, 61, 249, 110, 144, 86, 52, 253, 72, 28,
53, 57, 125, 59, 235, 84, 128, 208, 146, 20, 74, 6,
239, 190, 83, 19, 138, 118, 176);

procedure InicMyRandom;
var i: Integer;
var s: string;
begin
 WriteLn('Введіть який-небудь текст для ініціалізації
генератора випадкових чисел (до 256 символів):');
 ReadLn(s);
 i := 1;
 while (i<=255) and (i<=Length(s)) do
begin
  t[i]:= Ord(s[i]);
  Inc(i);
 end;
 pos := 0;
 WriteLn('OK');
 WriteLn;
end;

function MyRandom: Cardinal;
var i: Integer;
var l: Cardinal;
begin
 if (pos = 0) then
 begin
  for i := 1 to 255 do     t[i]:= cbox[(t[i-1]+t[i])
and 255];
  for i := 254 downto 0 do t[i]:= cbox[(t[i]+t[i+1])
and 255];
 end;
 l := 0;
 for i := 0 to 3 do l := l shl 8 +
```

Продовження лістингу 1

```
Cardinal(t[pos+i]);
 Result := l;
 pos := (pos+4) and 255;
end;

//-----------------------------------------------------
----------
//Головна програма
var i,j: Integer;
var maxbit: Integer;
var none,ntwo: TBigNum;
var n1,n2: TBigNum;
var p,q,z: TBigNum;
var n,e,d: TBigNum;
var s1,s2: string;

begin
 WriteLn;
 InicMyRandom();
 repeat
  Write('Введіть максимальний розмір простих чисел (p
i q) в бітах (8-257): ');
  ReadLn(maxbit);
until (maxbit>=8) and (maxbit<=257);
//p
 WriteLn('Введіть велике десяткове значення, яке буде
використано як перше просте число (Enter
          -> генерується програмою): ');
 ReadLn(s1);
 BN_dec_to_bignum(s1,p);
 BN_bignum_to_dec(p,s2);
 if (s1<>s2) then
 begin
  if (s1<>'') then WriteLn('Число задане невірно!');
  s1 := '0'; BN_dec_to_bignum(s1,p);
  for i := 0 to BIGNUM_DWORD do n1[i]:= MyRandom();
  BN_a_shr_k(n1(BIGNUM_DWORD+1)*32-maxbit,p);
  BN_bignum_to_dec(p,s2);
```


Продовження лістингу 2

```
s := '';

repeat

 BN_a_mod_b(n1,nten,n2);
  s := Chr(n2[0]+48)+s;
  BN_a_div_b(n1,nten,n2);
  Move(n2,n1,sizeof(n2));
 until (BN_a_cmp_b(n1,nzero)=0);
 while (Length(s) >1) and (s[1]='0') do
Delete(s,1,1);
end;

procedure BN_dec_to_bignum(var s: string; var a:
TBigNum);
var i,j,l: Integer;
var n1,n2,n3,n4: TBigNum;
var nten: TBigNum;
begin
 for i := 0 to BIGNUM_DWORD do a[i]:= 0;
 for i := 0 to BIGNUM_DWORD do nten[i]:= 0; nten[0]:=
10;
 for i := 0 to BIGNUM_DWORD do n1[i]:= 0; n1[0]:= 1;
 for i := 0 to BIGNUM_DWORD do n2[i]:= 0;
 l := Length(s);
 for i := l downto 1 do
 begin
  j := Ord(s[i])-48;
  if (j<0) or (j>9) then Exit;
  n2[0]:= Cardinal(j);
  BN_a_mul_b(n1,n2,n3);
  BN_a_add_b(a,n3,n4);
  Move(n4,a,sizeof(n4));
  BN_a_mul_b(n1,nten,n3);
  Move(n3,n1,sizeof(n3));
 end;
end;
```

Продовження лістингу 2

```
for i := 0 to BIGNUM_DWORD do a[i]:= 0;

for i := 0 to BIGNUM_DWORD do n16[i]:= 0; n16[0]:=
16;
 for i := 0 to BIGNUM_DWORD do n1[i]:= 0; n1[0]:= 1;
 for i := 0 to BIGNUM_DWORD do n2[i]:= 0;
 l := Length(s);
 for i := l downto 1 do
 begin
  j := Ord(UpCase(s[i]));
  case j of
   Ord('0')..Ord('9'): j := j - Ord('0');
   Ord('A')..Ord('F'): j := j - Ord('A')+ 10;
   else Exit;
  end;
  n2[0]:= Cardinal(j);
  BN_a_mul_b(n1,n2,n3);
  BN_a_add_b(a,n3,n4);
  Move(n4,a,sizeof(n4));
  BN_a_mul_b(n1,n16,n3);
  Move(n3,n1,sizeof(n3));
 end;
end;

procedure BN_bignum_to_dec(var a: TBigNum; var s:
string);
var i: Integer;
var n1,n2: TBigNum;
var nzero,nten: TBigNum;
begin
 for i := 0 to BIGNUM_DWORD do nzero[i]:= 0;
 for i := 0 to BIGNUM_DWORD do nten[i]:= 0; nten[0]:=
10;
 s := '0';
 if (BN_a_cmp_b(a,nzero)=0) then Exit;
 Move(a,n1,sizeof(a));
```


Продовження лістингу 1

```
WriteLn('Число, що згенерувало: ',s2);

end;
 WriteLn('Пошук першого простого числа...
Чекайте...');
 p[0]:= p[0] or 1;
 s1 := '2'; BN_dec_to_bignum(s1,ntwo);
 j := 0;
 while (BN_PrimeTest(p)=0) and (j<8192) do
 begin
  BN_a_add_b(p,ntwo,n1);
  Move(n1,p,sizeof(n1));
  Inc(j);
  Write('.');
 end;
 WriteLn;
 if (j>=8192) then
 begin
  WriteLn('На жаль, просте число не знайдене!');
  WriteLn('Натисніть Enter для виходу.'); ReadLn;
  Halt(1);
 end;
 BN_bignum_to_dec(p,s1);
 WriteLn('Перше просте число p = ',s1);
//q
 WriteLn('Введіть велике десяткове значення, яке буде
використано як друге просте число (Enter
          -> генерується програмою): ');
ReadLn(s1);
 BN_dec_to_bignum(s1,q);
 BN_bignum_to_dec(q,s2);
 if (s1<>s2) then
 begin
  if (s1<>'') then WriteLn('Число задане невірно!');
  s1 := '0'; BN_dec_to_bignum(s1,q);
  for i := 0 to BIGNUM_DWORD do n1[i]:= MyRandom();
  BN_a_shr_k(n1(BIGNUM_DWORD+1)*32-maxbit,q);
  BN_bignum_to_dec(q,s2);
```

Продовження лістингу 1

```
 WriteLn('Число, що згенерувало: ',s2);

end;
 WriteLn('Пошук першого простого числа...
Чекайте...');
 q[0]:= q[0] or 1;
 s1 := '2'; BN_dec_to_bignum(s1,ntwo);
 j := 0;
 while (BN_PrimeTest(q)=0) and (j<8192) do
 begin
  BN_a_add_b(q,ntwo,n1);
  Move(n1,q,sizeof(n1));
  Write('.');
 end;
 WriteLn;
 if (j>=8192) then
 begin
  WriteLn('На жаль, просте число не знайдене!');
  WriteLn('Натисніть Enter для виходу.'); ReadLn;
 end;
 BN_bignum_to_dec(q,s1);
 WriteLn('Друге просте число q = ',s1);
 WriteLn;
//n = p*q
 BN_a_mul_b(p,q,n);
 BN_a_div_b(n,q,n1);
 if (BN_a_cmp_b(p,n1) <>0) then
 begin
  WriteLn('На жаль, результат множення p*q дуже
великий!');
  WriteLn('Натисніть Enter для виходу.'); ReadLn;
  Halt(1);
 end;
 BN_bignum_to_dec(n,s1);
WriteLn('n = p*q = ',s1);
// z =(p-1)*(q-1)
 s1 := '1'; BN_dec_to_bignum(s1,none);
 BN_a_sub_b(p,none,n1);
```


Продовження лістингу 2

```

var primes: array[0..53] of Integer =
   (  2,   3,   5,   7,  11,  13,  17,  19,  23,  29,

31,  37,
     41,  43,  47,  53,  59,  61,  67,  71,  73,  79,
83,  89,
     97, 101, 103, 107, 109, 113, 127, 131, 137, 139,
149, 151,
    157, 163, 167, 173, 179, 181, 191, 193, 197, 199,
211, 223,
    227, 229, 233, 239, 241, 251);

procedure BN_bignum_to_hex(var a: TBigNum; var s:
string);
var i: Integer;
begin
 i := BIGNUM_DWORD;
 while (i>=0) and (a[i]=0) do Dec(i);
s := '0';
 if (i<0) then Exit;
 s := '';
 while (i>=0) do
 begin
  s := s + IntToHex(a[i],8);
  Dec(i);
 end;
 while (Length(s) >1) and (s[1]='0') do
Delete(s,1,1);
end;

procedure BN_hex_to_bignum(var s: string; var a:
TBigNum);
var i,j,l: Integer;
var n1,n2,n3,n4: TBigNum;
var n16: TBigNum;
begin
```

Лістинг 2. Допоміжний модуль uBigNumber

```
unit uBigNumber;
$IFDEF FPC} {$MODE DELPHI} {$ASMMODE INTEL}$ENDIF}
```

Продовження лістингу 2

```
interface

const BIGNUM_DWORD = 31;
type  TBigNum = array[0..BIGNUM_DWORD] of Cardinal;
procedure BN_bignum_to_hex(var a: TBigNum; var s:
string);
procedure BN_hex_to_bignum(var s: string; var a:
TBigNum);
procedure BN_bignum_to_dec(var a: TBigNum; var s:
string);
procedure BN_dec_to_bignum(var s: string; var a:
TBigNum);
function  BN_a_cmp_b(var a,b: TBigNum): Integer;
procedure BN_a_add_b(var a,b,res: TBigNum);
procedure BN_a_sub_b(var a,b,res: TBigNum);
procedure BN_a_mul_b(var a,b,res: TBigNum);
procedure BN_a_shl_k(var a: TBigNum; до: Integer;
                     var res: TBigNum);
procedure BN_a_shr_k(var a: TBigNum; до: Integer;
                     var res: TBigNum);
function BN_a_upbit(var a: TBigNum): Integer;
procedure BN_a_setbit_k(var a: TBigNum; до: Integer);
procedure BN_a_mod_b(var a,b,res: TBigNum);
procedure BN_a_div_b(var a,b,res: TBigNum);
procedure BN_a_exp_b_mod_c(var a,b,c,res: TBigNum);
procedure BN_ab_GCD(var a,b,res: TBigNum);
procedure BN_a_modinv_b(var a,b,res: TBigNum);
function BN_PrimeTest(var a: TBigNum): Integer;

implementation

uses SysUtils;
```


Продовження лістингу 1

```
 BN_a_sub_b(q,none,n2);

BN_a_mul_b(n1,n2,z);
 BN_bignum_to_dec(z,s1);
 WriteLn('z = (p-1)*(q-1) = ',s1);
// d
 WriteLn('Введіть велике десяткове значення, яке буде
          використано як відкритий ключ (Enter ->
          генерується програмою): ');
 ReadLn(s1);
 BN_dec_to_bignum(s1,d);
 BN_bignum_to_dec(d,s2);
 if (s1<>s2) then
 begin
  if (s1<>'') then WriteLn('Число задане невірно!');
  s1 := '0'; BN_dec_to_bignum(s1,n1);
  for i := 0 to BIGNUM_DWORD do n1[i]:= MyRandom();
  BN_a_mod_b(n1,z,d);
  BN_bignum_to_dec(d,s2);
  WriteLn('Число, що згенерувало: ',s2);
 end;
 WriteLn('Пошук відкритого ключа... Чекайте...');
 d[0]:= d[0] or 1;
 s1 := '1'; BN_dec_to_bignum(s1,none);
 s1 := '2'; BN_dec_to_bignum(s1,ntwo);
 j := 1;
 BN_ab_GCD(d,z,n1);
 while (BN_a_cmp_b(n1,none) <>0) and (j<1000) do
 begin
  BN_a_add_b(d,ntwo,n1);
  Move(n1,d,sizeof(n1));
  BN_ab_GCD(d,z,n1);
  j := j+1;
 end;
 BN_ab_GCD(d,z,n1);
 if (BN_a_cmp_b(n1,none) <>0) then
 begin
  WriteLn('На жаль, відповідного простого числа не
знайдено!');
```

Продовження лістингу 1

```
  WriteLn('Натисніть Enter для виходу.'); ReadLn;
  Halt(1);
 end;
 WriteLn;
 BN_bignum_to_dec(d,s1);
 WriteLn('Відкритий ключ d = ',s1);
 WriteLn;
// e
 WriteLn('Обчислення секретного ключа...');
 BN_a_modinv_b(d,z,e);
 BN_bignum_to_dec(e,s1);
 WriteLn('Секретний ключ e = ',s1);
 WriteLn;
//e*d mod z = 1 ?
 BN_a_mul_b(e,d,n1);
 BN_a_mod_b(n1,z,n2);
 if (BN_a_cmp_b(n2,none) <>0) then
 begin
  WriteLn('ЗБІЙ: e*d mod z <> 1!');
  WriteLn('Натисніть Enter для виходу.'); ReadLn;
  Halt(1);
 end;
 WriteLn('e*d mod z = 1');
 WriteLn;
//Перевірка ключів.
 WriteLn('Введіть велике значення для перевірки
ключів (Enter
          -> генерується програмою):');
 ReadLn(s1);
 BN_dec_to_bignum(s1,n1);
 BN_bignum_to_dec(n1,s2);
 if (s1<>s2) then
 begin
  if (s1<>'') then WriteLn('Число задане невірно!');
  s1 := '0'; BN_dec_to_bignum(s1,n1);
  for i := 0 to BIGNUM_DWORD do n1[i]:= MyRandom();
 end;
 n1[7]:= 0;
```


Закінчення лістингу 1

```
BN_a_mod_b(n1,n,n2);

 BN_bignum_to_hex(n2,s2);
WriteLn('Початкове значення   = 0x',s2);
 BN_a_exp_b_mod_c(n2,e,n,n1);
 BN_bignum_to_hex(n1,s1);
 WriteLn('Зашифроване значення = 0x',s1);
 BN_a_exp_b_mod_c(n1,d,n,n2);
 BN_bignum_to_hex(n2,s1);
 WriteLn('Розшифроване значення = 0x',s1);
 if (s1<>s2) then
 begin
  WriteLn('ЗБІЙ: розшифроване значення не співпадає
           з початковим!');
  WriteLn('Натисніть Enter для виходу.'); ReadLn;
  Halt(1);
 end;
 WriteLn('OK');
 WriteLn;
//Техническая інформація.
 WriteLn('-----------------------------------------
-------');
 BN_bignum_to_hex(e,s1);
 WriteLn(' e = 0x',s1'  (',BN_a_upbit(e),'bit)');
 BN_bignum_to_hex(d,s1);
 WriteLn(' d = 0x',s1'  (',BN_a_upbit(d),'bit)');
 BN_bignum_to_hex(n,s1);
 WriteLn(' n = 0x',s1'  (',BN_a_upbit(n),'bit)');
 WriteLn('-----------------------------------------
-------');
 WriteLn;
 WriteLn(' Розмір блоку початкового тексту:
',IntToStr(BN_a_upbit(n)-1),' бітів');
 WriteLn(' Розмір блоку зашифрованого тексту:
',IntToStr(BN_a_upbit(n)),' bit');
 WriteLn;
 WriteLn('Натисніть Enter для виходу.'); ReadLn;
end.
```