**Лабораторна робота № 1.3.**

**Лабораторна робота №3. Криптопакет KRYPTON**

**Мета роботи. О**знайомитись з принципами роботи  програми KRYPTON.

**Хiд роботи**

**KRYPTON** дає змогу шифрувати трьома різними способами, а саме:

**1) NORMAL,**

**2) KEYBOOK,**

**3) PASS PHRASE.**

Щоб зашифрувати файл (або групу файлів), необхідно його виділити й натиснути F2 і вибрати один із методів шифрування.

Метод **normal** шифрує файл і автоматично створює для нього ключ. Цей метод найпростіший.

Метод **keybook** для шифрування потребує вказати інший файл, з допомогою якого буде створений ключ.

У разі шифрування методом **pass phrase** треба ввести пароль (не більше як 4 символи).

**Опис параметрів командного рядка:**

Запускають програму з параметром. **Krypton a myenc** filel.txt file2.zz ШеЗ.

Команда шифрує **"filel.txt", "filel.zz", "file3"** звичайним методом **ОТР – One – Time – Pad** – одноразовий блокнот (ОТР) у файли **"myenc.enc"** і **"myenc.key"** (файли даних і ключа практично однаково захищені, однак в ".епс" файлі містяться ще і координатно – іменні незашифровані дані).

**Krypton akb myenc** mykblO filel.txt file2.zz fileS.

Те ж саме, але за методом **Кодова книжка** з використанням спецфайла **"mykblO.kyb",** який повинен бути в тій самій директорії. Після закінчення шифрування програма виведе число "Key Id#", без якого розшифрування неможливе.

Krypton gkb curkb.

Ця команда генерує нову кодову книжку **"curkb. kyb".** Програма запросить довжину одного ключа та  їхню загальну кількість.

Krypton a myenc.

Ця команда розшифрує всі файли з шифровки (ОТР) **"myenc.enc"** за ключем **"myenc.key".** Також  цими  параметрами  можуть бути конкретні імена  файлів.

***Алгоритм RSA.*** Незважаючи на досить велику кількість різних асиметричних шифросистем, найпопулярніша криптосис-тема RSA, яка розроблена в 1977 р. і набула назву на честь Рона Рівеста, Аді Шаміра і Леонарда Ейдельмана.

Систему побудовано на факті, що добуток великих простих чисел здійснюється легко, проте розкладання на множники добутку двох таких чисел практично неможливе. Доведено (теорема Рабіна), що розкриття шифру RSA еквівалентно такому розкладу. Тому для довільної довжини ключа дають нижню оцінку кількості операцій для розкриття шифра, а з урахуванням продуктивності сучасних комп’ютерів оцінити і необхідний для цього час.

Можливість гарантовано оцінити захищеність алгоритму RSA стала однією з причин популярності цієї криптосистеми на фоні десятків інших схем. Тому алгоритм RSA використовується в банківських комп’ютерних мережах, особливо для роботи з віддаленими клієнтами (обслуговування кредитних карток).

Сьогодні алгоритм RSA використовується в багатьох стандартах, серед яких SSL, S – HHTP, S – MIME, S/WAN, STT i PCT.

Розглянемо математичні результати, які лежать в основі цього алгоритму.

**Теорема 1.** (*Мала теорема Ферма*)*.*

Якщо *р*  –  просте число, то

*xр – 1= 1* (mod *p*)

          для будь – якого *х*, простого відносно *р*, і

*xр=х* (mod *p*)

          для будь – якого *х*.

**Визначення***.* Функцією Ейлера φ(*n*) називається число додатних цілих, менших вiд *n* і простих відносно *n*.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *N* | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| φ(*n*) | 1 | 2 | 2 | 3 | 2 | 6 | 4 | 6 | 4 | 10 | 4 |

**Теорема 2.**

Якщо *n=p× q* (*p* i *q* – відмінні одне від одного прості числа), то

*φ*(*n*)*=*(*p – 1*)(*q – 1*).

**Теорема 3.**

Якщо *n=p× q* (*p* i *q* – відмінні один від одного прості числа),  
*х* – просте відносно *p* i *q*, то

*xφ*(*n*)*=1* (mod *n*).

**Наслідок.**

Якщо *n=p× q*, (*p* i *q* – відмінні одне від одного прості числа),

*е* – просте відносно *φ*(*n*), то відображення

*Ee,n: x→xe* (mod *n*)

буде взаємно однозначним.

Очевидний і той факт, що коли *е* – просте відносно *φ*(*n*), то існує ціле *d* – таке, що

*ed= 1* (mod *φ*(*n*)).

На цих математичних фактах і заснований популярний алгоритм RSA.

Нехай *n=p× q*, де *p* i *q* – відмінні прості числа. Якщо *e* i *d* задовольняють рівняння *ed = 1* (mod *φ*(*n*)), то відображення *Ee,n* i *Ed,n* будуть інверсіями *Ed,n* i *Ee,n*, і легко обчислюються, якщо відомі *e,d,p,q*. Якщо відомі *e* i *n*, а *p* i *q* невідомі, то *Ee,n* є односторонньою функцією і знаходження *Ed,n* за заданим *n* рівнозначно розкладанню *n*. Якщо *p* i *q* – достатньо великі прості, то розкладання *n* практично нездійсненне. Це і закладено в основу системи шифрування RSA.

Користувач *і* вибирає пару різних простих *pi* i *qi* та розраховує пару цілих (*ei, di*), які є простими відносно *φ*(*n*), де *ni=piqi*. Довідкова таблиця містить ключі *{*(*ei,ni*)*}*. Припустимо, що вихідний текст

*x=*(*x0,x1,…,xn – 1*)*, x Є Zn, 0 ≤ i < n.*

Користувач *і* зашифровує текст при передачі його користувачу *j*, застосовуючи до *n* відображення *Edi,ni*:

*N → Edi,ni n=n’*.

Користувач *j* проводить дешифрування *n’*, застосувавши *Eеi,ni*:

*N’ → Eei,ni n’= Eеi,ni Edi,ni n=n*.

Самоочевидно, що для того, щоб знайти інверсію *Edi,ni* до *Eеi,ni* , потрібно знати множники *n=piqi*. Час виконання найкращих з відомих алгоритмів розкладу при *n*=10100 на сьогодні виходить за межі реальних технічних можливостей.

***Застосування алгоритму RSA*.**

**Приклад.** Зашифруємо повідомлення САВ. Для простоти використаємо маленькі числа (на практиці застосовують значно більші).

1. Виберемо *p*=3 i *q*=11.
2. Визначимо *n*=3×11=33.
3. Знайдемо (*p* – 1)(*q* – 1)=20. В якості *d* візьмемо взаємно просте з 20, наприклад *d*=3.
4. Виберемо число *е*. В ролі такого числа виступає будь – яке число, для якого задовольняється співвідношення (*e*×3) (mod 20)=1, наприклад 7.
5. Запишемо шифроване повідомлення як послідовність цілих чисел за допомогою відображення: А→1, В→2, С→3. Тоді повідомлення набуде вигляду (3,1,2). Зашифруємо за допомогою ключа {7,33}.

ШТ1=(37)(mod 33)=2187(mod 33)=9,

ШТ2=(17) (mod 33)=1(mod 33)=1,

ШТ3=(27) (mod 33)=128(mod 33)=29.

1. Розшифруємо одержане зашифроване повідомлення (9,1,29) на основі закритого ключа {3,33}:

ВТ1=(93)(mod 33)=729(mod33)=3,

ВТ2=(13)(mod 33)=1(mod33)=1,

ВТ1=(293)(mod 33)=24389(mod33)=2.

Отже, в реальних системах алгоритм RSA реалізується в такий спосіб: кожен користувач вибирає два великих простих числа *p* i *q* та, відповідно до описаного алгоритму, вибирає два простих числа *e* i *d*. Як результат добутку перших двох чисел (*p* i *q*) встановлюється *n*.

Відкритий ключ утворює {*e,n*}, а {*d,n*} – закритий (хоч можна і навпаки). Відкритий ключ публікується і доступний кожному бажаючому надіслати власнику ключа повідомлення, яке зашифроване вказаним алгоритмом. Після шифрування, повідомлення неможливо розкрити за допомогою відкритого ключа. Власник закритого ключа має можливість розшифрувати прийняте повідомлення.

***Система Ель – Гамаля.***

Ця система є альтернативою до RSA і при одинаковому значенні ключа забезпечує таку ж криптостійкість.

На відміну від RSA метод Ель – Гамаля заснований на проблемі дискретного логарифма. Цим він подібний до алгоритму Діффі – Хелмана. Якщо підносити число до ступеня в скінченному полі досить легко, то відновити аргумент за значенням (знайти логарифм) досить складно.

Основу системи становлять параметри *p* i *g* – числа, перше з яких – просте, а друге – ціле.

Далі Аліса генерує секретний ключ *a* і обчислює відкритий ключ *y=ga*mod *p*. Якщо Борис хоче надіслати послати Алісі повідомлення *m*, то він вибирає випадкове число *k*, менше, ніж *р,* і обчислює

*y1*=*gk* mod *p*    та

*y2*=*m*Å*yk*,

де Å означає побітове додавання за модулем 2. Потім Борис надсилає (*у1,у2*) Алісі.

Аліса, одержавши зашифроване повідомлення, відновлює його:

                            m=(y1*a* mod p) Åy2.

***Електронний підпис на основі алгоритму RSA****.*

Найпростішим і найпоширенішим інструментом електронного підпису є вже знайомий алгоритм RSA. Крім того, існують десятки інших схем цифрового підпису.

Припустимо, що *d, p, q* – секретні, а  *е, n = pq* – відкриті.

Нехай DATA – повідомлення, яке передає Аліса Борису.

Аліса підписує DATA для Бориса при передачі:

                   EeB,nB {EdA,nA {DATA}}.

При тому використовується:

-       закритий ключ EdA,nA Аліси,

-       відкритий ключ EeB,nB Бориса.

Борис може читати це підписане повідомлення спочатку за допомогою закритого ключа Бориса EdB,nB з метою одержання

EdA,nA{DATA}= EdB,nB{ EeB,nB { EdA,nA{DATA}}}

і потім  –  відкритого ключа EeА,nА Аліси для одержання

DATA= EeA,nA {EdA,nA {DATA}}.

Відтак, у Бориса з’явиться повідомлення DATA, надіслане йому Алісою.

Очевидно, що ця схема дає змогу захиститись від декількох видів порушень.

Аліса не може відмовитись від свого повідомлення, якщо вона визнає, що секретний ключ відомий тільки їй.

Порушник без знання секретного ключа не може ні сформувати, ні зробити осмислену зміну повідомлення, яке передається по лінії зв’язку.

Ця схема дає можливість у вирішенні багатьох конфліктних ситуацій обходитися без посередників.

Деколи немає необхідності зашифровувати передавальне повідомлення, але потрібно закріпити його електронним підписом. У такому випадку текст шифрують закритим ключем відправника і одержання ланцюжок символів прикріплюється до документа. Одержувач за допомогою відкритого ключа відправника розшифровує підпис і порівнює з текстом.

У 1991р. Національний інститут стандартів і технології (NIST) запропонував для алгоритму цифрового підпису DSA (Digital Signature Algorithm) стандарт DSS (Digital Signature Standard), в основу якого покладено алгоритми Ель – Гамаля і RSA.

**Цифрова сигнатура.**

Часто виникають ситуації, коли одержувач повинен вміти довести достовірність повідомлення зовнішній особі. Щоб мати таку можливість, до передавальних повідомлень мають бути приписані так звані цифрові сигнатури.

*Цифрова сигнатура* – це рядок символів, який залежить як від індентифікатора відправника, так і від змісту повідомлення.

Використання цифрової сигнатури передбачає застосування деяких функцій шифрування:

*S=H*(*k,T*),

де *S* – сигнатура, *k* – ключ, *Т* – вихідний текст.

Функція *H*(*k,T*) – є хеш – функція, якщо вона задовольняє таким умовам:

-       вихідний текст може бути довільної довжини;

-       саме значення *H*(*k,T*) має фіксовану довжину;

-       значення функції *H*(*k,T*) легко обчислюються для довільного аргументу;

-       відновити аргумент за значенням функції практично неможливо;

-       функція *H*(*k,T*) – однозначна.

З визначення випливає, що для будь – якої хеш – функції є тексти – близнюки, які мають однакове значення хеш – функції, оскільки потужність множини аргументів необмежено більша від потужності множини значень. Наявність такого факту набула назву *ефекту дня народження*.

Найвідоміші з хеш – функцій – MD2, MD4, MD5 i SHA.

Три алгоритми серії MD розроблені Рівестом у 1989, 1990, 1991 роках відповідно. Всі вони перетворюють текст будь – якої довжини в 128 – бітну сигнатуру.

Алгоритм MD2 передбачає:

-       доповнення тексту до довжини, кратної 128 бітам;

-       визначення 16 – бітної контрольної суми (старші розряди відкидаються);

-       додавання контрольної суми до тексту;

-       повторне визначення контрольної суми.

Алгоритм MD4 передбачає:

-       доповнення тексту до довжини, рівної 448 біт за модулем 512;

-       додавання довжини тексту в 64 – бітному вираженні;

-       512 – бітні блоки піддають процедурі Damgard – Merkle, для чого кожен блок бере участь у трьох різних циклах.

В алгоритмі MD4 доволі швидко були знайдені “дірки”, тому він був заміненим алгоритмом MD5, в якому кожний блок бере участь не в трьох, а в чотирьох різних циклах.

Алгоритм SHA (Secure Hash Algorithm), розроблений NIST (National Institute of Standard and Technology), повторює ідеї серії MD. B SHA використовуються тексти 264 біт, які закриваються сигнатурою завдовжки 160 біт. Такий алгоритм передбачається використовувати в програмі Capstone.