Франчук В.М.

Захист інформаційних ресурсів:

криптографічні та стеганографічні методи захисту даних

Посібник для викладачів,
вчителів та студентів інформатичних спеціальностей

НПУ імені М.П.Драгоманова
Інститут інформатики

Київ 2012

Франчук В.М.

Захист інформаційних ресурсів:

криптографічні та стеганографічні методи захисту даних

Посібник для викладачів,
вчителів та студентів інформатичних спеціальностей

НПУ імені М.П.Драгоманова
Інститут інформатики

Київ 2012

УДК

Укладач:

В.М. Франчук, доцент, кандидат педагогічних наук (Національний педагогічний університет імені М.П. Драгоманова, Інститут інформатики)

Рецензенти:

Франчук В.М. Захист інформаційних ресурсів: криптографічні та стеганографічні методи захисту даних. Посібник для викладачів, вчителів та студентів інформатичних спеціальностей. – К.: НПУ імені М.П. Драгоманова, 2012. – 120 с.

Навчальний посібник призначено для використання в навчальному процесі при вивченні курсу «Захист інформаційних ресурсів» (ЗІР). У посібнику розглянуто питання, пов’язані з криптографічними та стеганографічними методи захисту даних.

Може бути використаний студентами, вчителями та викладачами і всіма хто цікавиться захистом інформаційних ресурсів.

Схвалено вченою радою Інституту інформатики НПУ імені М.П. Драгоманова
(протокол № 1 від 05 вересня 2012 р.)

Зміст

Перелік термінів та умовних скорочень

Терміни

- **CFS** – вільна криптографічна файлова система, створена для операційних систем Unix/Linux М. Блейзом (Blaze). CFS– це метод шифрування усієї файлової системи, за допомогою якої користувач може зберігати в ній зашифровані файли.
- **IPSec (Internet Protocol Security – IP-безпека)** – захищений протокол передавання даних від корпорації Cisco. Описаний в RFC 2401, 2402, 2406 та ін.
- **VPN (Virtual Private Network)** – віртуальна приватна мережа, що базується на об'єднанні віддалених вузлів за допомогою публічних каналів зв'язку (в абсолютній більшості випадків – в Інтернет), і в якій застосовуються процедури шифрування і аутентифікації. Технологія VPN передбачає створення між віддаленими вузлами спеціального тунелю (захищеного каналу зв'язку), через який і відбувається обмін даними.
- **Аутентифікація (authentication)** – процедура перевірки справжності даних, тобто того, що ці дані були створені легітимними (законними) учасниками процесу обміну даними.
- **Хеш (hash – суміш, мішанина)** – зашифрований вигляд. Синонімічні поняття, які використовуються, зокрема, для перевірки цілісності файлів/повідомлень: криптографічна контрольна сума (cryptographic checksum), «дайджест повідомлення» (message digest), «відбиток пальця» (fingerprint), цифровий підпис (digital signature). Створюється за рахунок застосування до файлу/повідомлення певного алгоритму, за яким генерується унікальне числове значення.

Умовні скорочення

- **C** – відкрите повідомлення.
- **K** – ключ криптосистеми.

- **S** – зашифроване повідомлення.
- **TCP/IP** – Transmission Control Protocol/Internet Protocol, протокол управління передаванням/протокол Інтернет.
- **ЕЦП** – електронний цифровий підпис.
- **КА** – криптоатака.
- **ОС** – операційна система.
- **ПК** – персональний комп'ютер.

Вступ

Зрозуміло, що в сучасному цивілізованому світі, в якому величезна кількість видів діяльності людей супроводжується комп'ютерною підтримкою, проблема безпеки комп'ютерних систем є надзвичайно актуальною. Врахування усіх недоліків захисних механізмів, передбачення можливих наслідків та загроз безпеки інформаційних ресурсів може убезпечити комп'ютерних користувачів від небажаних впливів різноманітних обставин і сторонніх людей на їхнє життя. Саме тому в наш час професіоналам потрібно володіти навиками використання апробованих методів і надійних засобів захисту комп'ютерних даних та розумітися у проблемі захисту інформаційних ресурсів в усій її багатогранності.

За всіх часів доступ до відомостей, що відносяться до певних сфер людської діяльності, дозволяв окремим особам чинити дії, що можуть нанести шкоду людям чи організаціям, у рамках яких протікає ця діяльність.

Працюючи із сучасними інформаційними системами користувачі повинні бути досить обережними, адже паралельно з виконанням процедур опрацювання даних в комп'ютерах здійснюється реєстрація усіх слідів діяльності людини, а це іноді може спричинити непередбачувані наслідки.

Перенесення чи не усіх негативних проявів характеру і кримінально зорієнтованих схильностей людей до віртуального світу спричинює ті сумні ситуації, в яких опиняються користувачі, подорожуючи в Інтернет. Тут зловмисники можуть розкрити конфіденційність користувача і скористатися з того у своїх злочинних цілях.

При цьому може статися викрадення цінних даних, у тому числі тих, що становлять таємницю. У деяких випадках не менш тяжкі наслідки може мати і модифікація (спотворення, фальсифікація) даних.

Активно користуючись мережевими ресурсами, в Інтернет можна натрапити на «крекерів», «кіберпанків», «фрікерів», «Інтернет-шахраїв», «мережевих шпигунів», «брейкерів», «кіберсквотерів», «крипто шантажистів» і т.п.

Інколи масштаби дій озброєних потужними комп'ютерами зловмисників, що добре знаються на різних технологіях програмування та вміють користуватися інструментами «соціальної інженерії», сягають рівня світових атак. Іноді потужні атаки навіть кваліфікують як «віртуальні війни».

У будь-якому випадку, і коли загрози втрати, фальсифікації вмісту комп'ютерних запам'ятовуючих пристроїв стосуються рядових користувачів, і коли це торкається високих сфер світової політики, фахівцям-комп'ютерникам потрібно вміти надійно захищати інформаційні ресурси і всіляко уникати дій різного роду зловмисників. Для цього, звісно, потрібно багатогранно досліджувати усі наявні технології та інструменти, до яких можуть вдатися хакери для доступу до вмісту запам'ятовуючих пристроїв комп'ютерів, а також знатися на гарантованих прийомах і методах захисту даних та протидії їх пошкодженню.

Розділ 1. Основні поняття захисту інформаційних ресурсів

1.1. Основні поняття з галузі захисту інформаційних ресурсів.

Щоб адекватно розумітися на проблемах захисту даних необхідно ознайомитись з термінологією, притаманною цій галузі інформатичної науки. Визначимо основні поняття з галузі захисту даних та наведемо поширену термінологію, яка стосується загроз безпеки (security threat) і деяких методів зламування комп'ютерних систем.

Під *інформаційною безпекою* розуміється захищеність інформаційних ресурсів і підтримуючої інфраструктури від випадкових та навмисних впливів природного або штучного характеру (неавторизованого доступу, руйнування, модифікації, розкривання, затримок у доступі і т. ін.), що можуть привезти до нанесення збитків власникам або користувачам інформаційних ресурсів і підтримуючої інфраструктури. Найчастіше поняття безпеки даних розуміють як захист даних і програм від несанкціонованого доступу до них.

Виділяють три цілі захисту даних: *цілісність даних, конфіденційність даних, доступність даних*.

Цілісність даних – це гарантованість того, що дані не були змінені, підмінені або знищені в результаті зловмисних дій або випадків. Під критичними даними розуміють дані, що потребують захисту через ймовірність нанесення (ризику) збитків в тому випадку, якщо відбудеться випадкове чи навмисне розкривання, зміна або руйнування даних.

Конфіденційність даних – це статус, наданий даним, чим визначається необхідний ступінь їх захисту. *Конфіденційні дані* – це дані, що потребують захисту. Конфіденційні дані повинні бути відомі тільки допущеним (і тим, що пройшли перевірку) авторизованим суб'єктам системи. До конфіденційних можна віднести, наприклад, такі дані: особисті дані користувачів, облікові записи (імена і паролі), дані про кредитні карти, дані про розробки та різні внутрішні документи, бухгалтерські відомості.

Витік даних є результатом дій порушника, внаслідок яких дані стають

відомими (доступними) суб'єктам, що не мають права доступу до них.

Розрізняють такі три види витоку даних:

- розголошення;
- несанкціонований доступ до даних;
- одержання захищуваних даних розвідками (як вітчизняними, так і іноземними).

Під *розголошенням даних* розуміється несанкціоноване доведення захищуваних даних до користувачів, які не мають права доступу до даних.

Під *захистом даних* розуміють заходи щодо контролю за доступом до цих даних.

Під *доступом до даних* розуміється ознайомлення з ними та їх опрацювання, зокрема, копіювання, модифікація або знищення.

За режимом доступу дані поділяються на *відкриті дані* та *дані з обмеженим доступом*, які бувають конфіденційними і таємними. *Конфіденційними даними* є відомості, які знаходяться у володінні, користуванні або розпорядженні окремих фізичних або юридичних осіб і поширюються за їх бажанням відповідно до передбачених ними умов.

До *таємних даних* належать дані, що містять відомості, які становлять державну та іншу, передбачену законом таємницю, розголошення якої завдає шкоди особі, суспільству і державі.

Під *порушенням режиму доступу* слід розуміти дії, які проявились у будь-якому порушенні порядку отримання, використання, поширення і зберігання даних, передбаченого правовими нормами та встановленого власником даних, доступ до яких обмежено, або уповноваженою ним особою. Найбільш поширеною формою такого порушення є несанкціонований доступ до даних.

Несанкціонований доступ – це доступ до даних, що порушує правила розмежування доступу з використанням засобів комп'ютерної техніки. Це незапланованe ознайомлення, опрацювання, копіювання, застосування різних вірусів, у тому числі програмних продуктів, що руйнують, а також

модифікують чи знищують дані з порушенням встановлених правил розмежування доступу. Здійснюється, як правило, з метою розкриття, зміни або руйнування даних.

Безпосереднім об'єктом злочину, наприклад, є право власності на дані, тобто порушене право власника на володіння, використання чи розпорядження даними.

Під *знищенням даних* слід розуміти їх втрату, коли дані в автоматизованій системі (в інформаційно-комунікаційній системі) перестають існувати для фізичних і юридичних осіб, які мають право власності на них в повному чи обмеженому обсязі.

Як *блокування даних* треба розглядати припинення доступу до даних. Такі дії можуть виражатись, наприклад, в електромагнітному, лазерному і іншому впливі на носії даних; в формуванні сигналів полів засобів і блоків програм, вплив яких на дані, їх носії і засоби технічного захисту викликає порушення цілісності даних, їх втрату чи модифікацію, у включенні до бібліотек програм спеціальних програмних блоків, зміни програмного забезпечення і інших подібних діях, що призводять до порушення цілісності даних.

Модифікація даних – це зміна їх змісту, порушення їх цілісності, в тому числі і часткове вилучення.

Фільтрація даних – це аналіз даних за сукупністю критеріїв.

Канал витоку даних – це сукупність джерел даних, матеріального носія або середовища поширення сигналу, що несе зазначені дані, і засоби виділення даних із сигналу або носія. Однією з основних властивостей каналу є місце розташування засобу виділення даних із сигналу або носія, що може розташовуватися в межах контрольованої зони, що охоплює систему, або поза нею.

Помилка – похибка у програмному коді даного програмного засобу. Можливі помилки, що не проявилися або ще не були використані зловмисниками.

Прорахунок – недолік програмного засобу, що визначається як його

програмним кодом, так і недоліком самого проекту чи способом застосування засобу.

Помилки і прорахунки являють собою *вразливості*.

Під *вразливістю* (vulnerability) інформаційної системи розуміється будь-яка її характеристика (властивість), використання якої порушником може призвести до реалізації загрози.

Загрозою (threat) комп'ютерним даним (інформаційній системі) називається потенційно можлива подія, дія, процес або явище, що може прямо чи опосередковано нанести збитки (матеріальні, моральні або інші) даним чи іншим ресурсам системи шляхом розкриття, модифікації або руйнування даних, або відмовлення в обслуговуванні критичних сервісів.

Прийнято виділяти такі узагальнені види загроз:

- загрози цілісності даних;
- загрози конфіденційності;
- загрози доступності даних.

Взагалі організації (та комп'ютерні користувачі) володіють різноманітними активами, засобами, капіталом, майном, які вони цінують і хочуть захищати. Все це – *ресурси (assets)* – може постраждати внаслідок вдалої реалізації загрози зловмисником. Ресурси містять у собі дані і підтримуючі засоби, що потрібні організації (користувачу) для ведення своєї діяльності (бізнесу, вирішення поставлених завдань тощо).

Прикладами ресурсів є:

- дані, тобто файли (з деталями платежів, звукові записи, файли зображень, відомості про якісь об'єкти, плани тощо);
- паперові документи (контракти, заповнені форми і т.п.);
- програмне забезпечення (системне програмне забезпечення, прикладне програмне забезпечення, засоби розробки програм, утиліти);
- фізичне обладнання (комп'ютери, комунікаційне обладнання, комп'ютерні носії даних, інше технічне обладнання);

- служби (обчислювальні, комунікаційні, утиліти);
- люди та їхні знання (технічні, операційні, маркетингові, юридичні, фінансові тощо);
- імідж і репутація організації.

Загрози можуть бути *природними* (повінь, пожежа, землетрус), *навмисними* або *випадковими*.

Результатом реалізації загрози може бути:

- руйнування об'єкта (засобів, даних, обладнання, зв'язку);
- ушкодження або модифікація об'єкта (даних, додатків);
- крадіжка, видалення або втрата об'єкта (обладнання, даних, додатків);
- розкриття об'єкта (даних);
- використання або впровадження нелегального об'єкта (обладнання, неліцензованого програмного забезпечення, фальшивих даних);
- переривання служби.

Під *випадковим* розуміється таке походження загроз, що обумовлюється спонтанними і незалежними від волі людей обставинами, що виникають у системі опрацювання даних у процесі її функціонування. Найбільш відомими подіями такого типу є відмовлення, збої, помилки, стихійні лиха і побічні впливи:

- *відмовлення* – вихід з ладу якого-небудь елемента системи, що приводить до неможливості його подальшого функціонування;
- *збій* – тимчасовий вихід з ладу якого-небудь елемента системи, наслідком чого може бути неправильне опрацювання даних;
- *помилка* – вихід з ладу (разовий або систематичний) елемента системи, що приводить до неправильного її функціонування;
- *побічний вплив* – негативний вплив на систему в цілому або на окремі її елементи, який чиниться якими-небудь явищами, що відбуваються всередині системи або в зовнішньому середовищі.

Навмисні загрози обумовлюються злочинними діями людей,

здійснюваними з метою реалізації одного або кількох видів загроз.

Відзначено два різновиди передумов появи загроз: *об'єктивні* (кількісна або якісна недостатність елементів системи) і *суб'єктивні* (діяльність розвідувальних служб іноземних держав, промислове шпигунство, діяльність кримінальних елементів і хуліганів, злочинні дії несумлінних співробітників системи).

Джерелами загроз є люди, технічні пристрої, програми й алгоритми, технологічні схеми опрацювання даних і зовнішнє середовище:

- люди – персонал, користувачі та сторонні особи, які можуть використовувати ресурси і дані організації безпосередньо з робочих місць, а також віддалено, використовуючи мережеві технології;
- технічні засоби – безпосередньо використовувані для опрацювання, зберігання і передавання даних (наприклад, засоби реєстрації даних, засоби введення і т.д.), та допоміжні (наприклад, засоби електроживлення, кондиціювання і т.д.);
- моделі, алгоритми і програми – цю групу джерел розглядають як недоліки проектування, реалізації і конфігурації (експлуатації) та називають недоліками програмного забезпечення (загального призначення, прикладного і допоміжного);
- технологічна схема опрацювання даних – розрізняють ручні, інтерактивні, внутрімашинні і мережеві технологічні схеми опрацювання;
- зовнішнє середовище – стан середовища (можливість пожеж, землетрусів і т.п.), побічні шуми (особливо небезпечні при передаванні даних) та побічні сигнали (наприклад, електромагнітне випромінювання апаратури).

Контроль безпеки – це практика, процедури і механізми, що можуть захистити об'єкти від загроз, зменшити вразливості або зменшити вплив небажаних подій.

Навмисну реалізацію загрози безпеки комп'ютерної системи називають *атакою*. Атака – це дії, що починаються зловмисником, проти комп'ютера (або

мережі) потенційної жертви. Це пошук і/або використання зловмисником тієї чи іншої вразливості системи.

Атаки бувають *зовнішніми*, *внутрішніми* та *сумісні* атаки (скоординовані атаки ззовні і зсередини).

Зовнішні атаки – група дій, що включає в себе збирання даних (непрямими методами, що не стосуються цільової системи і не можуть бути нею виявлені), спроби (збирання даних про цільову систему методами, що повинні бути виявлені цільовою системою), атаки (здійснення спроб вторгнення на основі зібраних даних та існуючих баз даних вразливостей), порушення функціонування служб міжмережевого екрану у випадку успіху атак.

Внутрішні атаки виконуються, виходячи з того, що атакуючий вже має доступ до комп'ютера внутрішньої мережі. При цьому можуть послідовно використовуватись різні рівні наявного доступу: користувача, розробника програмного забезпечення або адміністратора.

У випадку сумісних атак вважається, що зовнішній атакуючий знаходиться у змові з користувачем організації, щоб обійти систему захисту або передати дані через міжмережевий екран.

Атака може бути безуспішною (невдалою) і успішною (вдалою). Успішну атаку називають *вторгненням*.

Вторгнення – це несанкціонований вхід в інформаційну систему (у результаті дій, що порушують політику безпеки (Security Policy) або обходять систему захисту).

Найчастіше під *політикою безпеки* розуміють сукупність норм, правил і практичних рекомендацій, що регламентують роботу засобів захисту комп'ютерної системи від заданої множини загроз.

Під "*зламуванням системи*" розуміють навмисне проникнення в комп'ютерну систему з несанкціонованими параметрами входу, тобто іменем користувача і його паролем (паролями).

Зондування системи – це вид порушення політики безпеки комп'ютерної системи, який чиниться з метою знаходження конфіденційних даних для

подальшого їх пошкодження або знищення.

Проникнення в систему – це вид порушення політики безпеки комп'ютерної системи, який чиниться з метою доступу (читання, модифікації, знищення) до певних даних, впливу на справність системи, стеження за діями інших користувачів.

Під *прихованим каналом* мається на увазі шлях передавання даних, за допомогою якого можна використовувати дані у двох взаємопов'язаних процесах таким способом, який порушує політику безпеки комп'ютерної системи.

Готуючись до порушення політики безпеки деякої комп'ютерної системи, зловмисник для відомої йому вразливості розробляє або використовує готові (розроблені іншими) шаблони атак. *Шаблон атаки* – це варіант злому стосовно вразливого місця програмного забезпечення. У шаблоні атаки може бути передбачене використання декількох властивостей вразливих місць і повинні бути зазначені дані, необхідні для злому системи. Екземпляр шаблону атаки, створений для компрометації конкретного фрагменту коду програмного засобу, є програмою атаки, або *експлойтом* (інколи в спеціалізованій літературі також можна зустріти термін «сплойт»).

При проведенні атаки зловмисник використовує сценарій атаки, що передбачає використання різних шаблонів в залежності від поводження (реакції) атакованої системи.

Узагальнений сценарій атаки можна подати у вигляді наступних кроків:

- пасивна розвідка;
- активна розвідка;
- вибір (або розробка) експлойту;
- злом цільової системи;
- завантаження «корисного вантажу» (яким, як правило, є шкідлива програма);
- приховування слідів злому.

Звичайно, дана послідовність може бути порушена або можуть бути виключені окремі кроки даного сценарію.

Етап пасивної розвідки називається саме так, оскільки зловмисник не входить у безпосередній контакт з ціллю (комп'ютерною системою) або входить з дотриманням загальноприйнятих правил (наприклад, відвідуючи сайт компанії, куди входить цільова система). Метою даного етапу є збирання даних про ціль, тому часто його називають рекогносцировкою цілі. Збиранню даних сприяє використання певних баз даних (наприклад, ARIN – American Registry for Internet Numbers, APNIC– Asia-Pasific Network Information Center), що знаходяться в Інтернеті, а також спеціальних програм (наприклад, *whois),* що дозволяють одержати такі дані.

Етап проведення активної розвідки називають скануванням. Зловмисник намагається визначити структуру цікавлячої його мережі, точки доступу, доступні вузли і хости, розташування маршрутизаторів і міжмережевих екранів, встановлені операційні системи та їхні версії, відкриті порти і працюючі служби, версії встановлених програмних продуктів. На цьому етапі зловмисник входить у контакт з об'єктами цікавлячої його системи, тому його дії можуть бути виявлені засобами захисту цільової системи.

Для вирішення задач проведення активної розвідки (сканування) зловмисник може використовувати велику кількість різноманітних засобів, які здебільшого є загальнодоступними. Серед них такі програмні засоби, як *ping, traceroute, Nmap, SuperScan.* Поширеність таких засобів багато в чому обумовлена тим, що вони активно використовуються системними адміністраторами для вирішення поточних проблем управління і захисту мереж.

Існує безліч методик і засобів, що дозволяють визначити операційну систему, встановлену на хості, та її версію. Розроблено методи так званого прихованого (stealth) сканування, за допомогою яких приховується джерело, з якого проводиться сканування. Крім того, може використовуватися "сповільнене" сканування, коли зловмисник посилає окремі запити через великі

інтервали часу. У цьому випадку досить висока ймовірність того, що на цільовій системі не звернуть уваги на окремі запити.

На підставі отриманих даних зловмисник вибирає (модифікує наявний або розробляє новий) експлойт для проведення злому.

Відомі загрози реалізуються атаками, що базуються на використанні добре відомих вразливостей і скриптів атак (експлойтів). Такі загрози виходять від недостатньо компетентних хакерів (що в англомовній літературі називаються «script kiddies») або зовсім некомпетентних (називаються «newbies»).

Майже усі малодосвідчені зломщики починають злом на основі наявного експлойту. Тоді на етапі розвідки шукається система, яка має відповідні вразливості. При цьому можливий простий перебір адрес, щоб якимось чином (наприклад, випадково) вибрати вразливий хост, який і стане ціллю злому.

Існує безліч різних способів злому. До їхнього числа можна віднести одержання доступу до системи, розширення наявних або отриманих повноважень і відмовлення в обслуговуванні.

Злом практично завжди виконується за допомогою програмного забезпечення і спрямований на програмне забезпечення.

Успішна атака, як правило, складається з кількох послідовних дій. Серед методів злому можна виділити такі, як введення команд, використання каналів і портів, зміна прав доступу, використання властивостей файлової системи, маніпулювання змінними середовища, використання зовнішніх змінних, використання некоректних даних, добір параметрів, використання некоректного опрацювання помилок і т.д.

Як правило, атака проводиться не лише заради самого процесу злому. Зазвичай, зловмисник хоче мати можливість повернення у зламану систему або ж планує використовувати її в якості плацдарму для атаки інших систем. Крім того, його може цікавити одержання конфіденційних даних у зламаній системі. Для цього зловмисник у систему завантажує «корисний вантаж» (звичайно, корисний лише для себе).

До дій по завантаженню «корисного вантажу» атаки можна віднести

наступні:

- встановлення програм «прослуховування» мережевого трафіка локальної мережі, у якій знаходиться зламана система, для одержання даних, за допомогою можна здійснити злом мережевих сусідів;
- здійснення перенапрямлення портів, щоб забезпечити передавання пакетів на зламану систему, минаючи міжмережевий екран;
- встановлення «латок» на використану або наявні вразливості, щоб уникнути зламування системи іншими зловмисниками;
- створення фальшивих облікових записів, наділених привілеями суперкористувача;
- створення або встановлення готового потайного входу (або «люка») у зламану систему;
- встановлення «троянських коней» для збирання конфіденційних даних.

Ці та інші операції проводяться при нейтралізації аудиту у зламаній системі.

Таким чином, *атака* – це процес реалізації деякого сценарію атаки. У ході атаки зловмисник одержує дані (реакції атакованої системи), що свідчать про успіх (чи невдачі) застосування даного шаблону або служать підставою для застосування певного шаблону атаки. Опис кожної атаки може ґрунтуватись на використовуваних нею вразливостях атакованої системи.

1.2. Методи зламування захисту комп'ютерних інформаційних систем.

Деякі фахівці з комп'ютерної безпеки класифікують *методи зламування захисту комп'ютерних систем відповідно до трьох основних компонентів програмного забезпечення* будь-якої універсальної комп'ютерної системи – операційної системи, системи управління базами даних і мережевого програмного забезпечення. Тому всі спроби зламування захисту комп'ютерних систем відповідно до такого підходу можна поділити на три групи:

- атаки на рівні операційної системи;
- атаки на рівні систем управління базами даних;

- атаки на рівні мережевого програмного забезпечення.

При захисті операційної системи можна зіткнутись з такими типами атак:

- крадіжка пароля;
- підглядання за користувачем, коли той вводить пароль, що дає право на роботу з операційною системою;
- одержання пароля з файлу, в якому цей пароль був збережений користувачем, який не бажає утрудняти себе введенням пароля при під'єднанні до мережі (як правило, такий пароль зберігається у файлі в незашифрованому вигляді);
- пошук пароля, що користувачі записують на календарях, у записних книжках чи на зворотному боці комп'ютерних клавіатур;
- крадіжка зовнішнього носія з парольними даними (дискети чи електронного ключа, на яких зберігається пароль користувача, призначений для входу в операційну систему);
- повний перебір усіх можливих варіантів пароля;
- добирання пароля за частотою входження символів і біграм (дві літери), за допомогою словників найбільш часто застосовуваних паролів, із залученням знань про конкретного користувача – його імені, прізвища, номера телефону, дати народження і т.д., з використанням відомостей про існування еквівалентних паролів, при цьому з кожного класу випробовується всього один пароль, що може значно скоротити час перебирання;
- сканування жорстких дисків комп'ютера (хакер послідовно намагається звернутися до кожного файлу, збереженого на жорстких дисках комп'ютерної системи; якщо обсяг дискового простору досить великий, можна бути цілком впевненим, що при описі доступу до файлів і каталогів адміністратор допустив хоча б одну помилку, у результаті чого всі такі каталоги і файли будуть прочитані хакером; для приховування слідів хакер може організувати цю атаку під чужим ім'ям: наприклад під ім'ям

користувача, пароль якого відомий хакеру);

- так зване «збирання сміття» (якщо за допомогою засобів операційної системи можна відновлювати раніше вилучені об'єкти, хакер може скористатися цією можливістю, щоб одержати доступ до об'єктів, вилучених іншими користувачами: наприклад, переглянувши вміст «сміттєвих» кошиків або прочитавши дані з «хвостових кластерів»);
- перевищення повноважень (використовуючи помилки в програмному забезпеченні чи в адмініструванні операційної системи, хакер одержує повноваження, що перевищують повноваження, надані йому відповідно до діючої політики безпеки);
- запуск програми від імені користувача, що має необхідні повноваження, або в якості системної програми (драйвера, сервісу і т.д.); підміна бібліотеки, що динамічно завантажується, яка використовується при виконанні системних програм, або зміна середовищ, в яких описують шлях до таких бібліотек;
- модифікація коду або даних підсистем захисту самої операційної системи;
- відмова в обслуговуванні (метою цієї атаки є часткове або повне виведення з ладу операційної системи);
- захоплення ресурсів (за допомогою хакерської програми проводиться захоплення всіх ресурсів, що є в операційній системі);
- бомбардування запитами (за допомогою хакерської програми постійно направляється операційній системі запити, реакція на які вимагає залучення значних ресурсів комп'ютера);
- використання помилок у програмному забезпеченні чи в адмініструванні.

Небезпечною для користувачів може виявитись наявність у програмному забезпечені так званих «люків». «Люк» вставляється в програму в більшості випадків на етапі налагодження для полегшення роботи: даний модуль можна викликати в різних місцях програми, що дозволяє налагоджувати окремі його частини незалежно. Крім того, «люк» може вставлятись на етапі розробки для

подальшого зв'язку даного модуля з іншими модулями системи, але потім, внаслідок змінених умов, дана точка входження виявляється непотрібною.

Наявність «люка» дозволяє викликати програму нестандартним способом, що може серйозно відбитись на стані комп'ютерної системи захисту (невідомо, як у такому випадку за програмою будуть опрацьовуватись дані, як буде здійснюватись управління програмою середовищем системи тощо). Крім того, в таких ситуаціях не завжди можна прогнозувати хід виконання програми.

«Люк» належить до категорії загроз, що виникають внаслідок помилок реалізації якогось проекту (комп'ютерної системи в цілому, комплексу програм і т.д.). Оскільки використання «люків» може бути найрізноманітнішим і залежить від характеристик самої програми, то яким-небудь чином класифікувати дану загрозу важко.

«Люки» можуть з'явитись в програмах з таких причин:

- їх забули «усунути»;
- для використання при подальшому налагодженні;
- для забезпечення підтримки готової програми;
- для реалізації таємного контролю доступу до даної програми після її встановлення (такі «дірки» або «чорні ходи» інколи навмисно залишаються розробниками).

Перший із перелічених випадків – необміркований промах, який може призвести до «пролому» в системі захисту. Два наступних випадки – серйозні випробування системи безпеки, з якими вона може і не впоратись. Четвертий випадок може стати першим кроком навмисного проникнення в комп'ютерну систему з використанням даної програми.

Слід відзначити, що програмна помилка не є «люком». «Люк» – це механізм налагодження, який досить широко використовується для коригування і супроводу програм. «Люк» стає небезпечним, якщо він помічений, залишений і не здійснюється жодних заходів контролю за ним.

Велика небезпека «люків», особливо в програмах операційної системи,

компенсується значною складністю їх виявлення. Якщо не знати наперед, що дана програма вміщує «люк», необхідно опрацювати кілобайти (а іноді і мегабайти) даних, щоб знайти його. Тому в більшості випадків виявлення «люків» – це результат випадкового пошуку. Захист від них може бути лише одним – не допускати появи «люків» у програмі, а при отриманні програмних продуктів, розроблених третіми виробниками, – проводити аналіз початкових текстів програм з метою виявлення «люків».

Якщо в програмному забезпеченні комп'ютерної системи немає помилок і її адміністратор строго дотримується політики безпеки, що рекомендована розробниками операційної системи, то атаки всіх перерахованих типів малоефективні. Додаткові заходи, що повинні бути розпочаті для підвищення рівня безпеки, значною мірою залежать від конкретної операційної системи, що використовується для управління даною комп'ютерною системою. Проте повністю усунути загрозу зламування комп'ютерної системи на рівні операційної системи неможливо. Тому політика забезпечення безпеки повинна проводитися так, щоб, навіть подолавши захист, створюваний засобами операційної системи, хакер не зміг нанести серйозних збитків.

Серед атак на рівні систем управління базами даних найнебезпечнішими є атаки «салямі» (назва таких атак походить від назви ковбаси салямі, яка виготовляється з невеличких частин різних сортів м'яса. Так само і рахунок зловмисника поповнюється за рахунок різних вкладників), вони становлять найбільшу небезпеку для систем, що використовуються при опрацюванні грошових рахунків в банках. За методом «салямі» реалізуються крадіжки невеликих сум протягом тривалого часу в надії, що це не стане помітним. Принцип атак «салямі» побудований на тому факті, що при обробці рахунків використовуються цілі одиниці (центи, рублі, гривні, копійки), а при нарахуванні процентів майже завжди виходять дробові числа. Похибки обчислень, які дозволяють інтерпретувати правила округлення чисел в той чи інший бік, а також дуже великі обсяги обчислень, які необхідно виконувати при обробці рахунків клієнтури, роблять можливими атаки «салямі». При

заокруглені дробових чисел «програма-салямі» відділяє останні цифри числа, що містяться після другої десяткової цифри, і розміщує їх на окремих (нелегальних) рахунках.

Загрози розкриття комп'ютерних даних пов'язані з недосконалістю програмних засобів – прикладних, системних, комунікаційних. І цією недосконалістю часто користуються особи певного роду.

Цікавити певних осіб може навіть особисте листування, не пов'язане з поточною роботою. Такі люди зламують електронні поштові скриньки, ICQ, віртуальні світи користувачів і розміщують усе листування на Web-сторінках для загального огляду часто задля власної розваги.

Розкриття і оприлюднення даних, що зберігаються користувачами у віртуальних світах (подібних до організованого, наприклад, на поштовому сервері Mail.ru), на сторінках досить популярних в наш час соціальних мережевих сервісів (наприклад, Vkontakte.ru, Odnoklassniki.ru тощо), може приводити до цілком реальних повсякденних проблем.

Усі необережні зауваження в чатах, на форумах, у групах новин (на зразок, старовинної *Usenet),* повідомлення в ICQ, у відеоконференціях (наприклад, організованих на базі програм, подібних до *Skype),* на віртуальних дошках об'яв; перелік відвіданих web-сайтів, Інтернет-сервісів, на яких користувач зареєструвався під своїм справжнім ім'ям, можуть потрапити в базу даних якої-небудь компанії, зайнятої збиранням конфіденційних відомостей про різних людей.

Взагалі, доцільно змалечку привчати дітей до метафори про Інтернет, виголошеної одним відомим інформатиком: «Виходячи в Інтернет, людина ніби заявляє про себе, голосно кричучи у рупор». Саме так потрібно сприймати свої дії в комп'ютерних мережах.

Тому, перед тим, як починати працювати на комп'ютері, неважливо, вдома чи на роботі, а тим більше перед своєю появою у віртуальному просторі Інтернету, необхідно добре обміркувати свої вчинки, заздалегідь вирішити, в результаті яких дії, виконуваних на комп'ютері, можуть залишитися сліди чи

натяки на відомості, які вважаються конфіденційними.

Найбільш вразливими джерелами даних при роботі на комп'ютері є:

- набір програм, що використовується під час роботи, а також інтенсивність звертання до цих програм;
- передісторія відкривання документів за останній час;
- нелегально придбане програмне забезпечення, тобто піратські копії програм, а також легальні і скопійовані різними способами програми;
- особисте листування електронною поштою, навіть ті повідомлення, що були вилучені після прочитання, і ті, що були створені в автономному режимі;
- вся передісторія подорожей в Інтернеті (всі відвідані вузли, завантажені файли, ресурси Інтернету, що найбільшою мірою приваблюють);
- файли різних документів, з якими користувач працює в даний час, в тому числі такі, що містять конфіденційні дані – поточні проекти, фінансові звіти і т.п.;
- точні відомості про дату і час підготовки вказаних документів, що може стати джерелом відомостей про поточну діяльність організації;
- цілі документи, їх фрагменти, файли різних типів, що давним-давно вилучені за допомогою ОС – ці файли цілком можуть містити конфіденційні дані;
- паролі доступу до ресурсів Інтернет, в тому числі, до сервера провайдера Інтернет, до рахунків у банках і магазинах Інтернет, чи до різних платіжних систем, наприклад, WebMoney, CyberCash і т.д.;
- номери кредитних карток, що використовуються для онлайнових купівель.

Зрозуміло, що розкриття цих відомостей чи їх втрата несе в собі цілком реальну загрозу для користувачів і організацій, в яких вони працюють.

В залежності від характеру втрачених даних, наслідки можуть бути досить важкими і пов'язаними не стільки з вартістю комп'ютерного обладнання, скільки з розкриттям конфіденційності документів чи необхідністю

відновлення втрачених даних.

1.3. Основні принципи забезпечення безпеки інформаційних систем.

Міжнародним співтовариством встановлено дев'ять принципів забезпечення безпеки інформаційних систем та мереж:

- інформованість (awareness) – учасники повинні усвідомлювати необхідність безпеки інформаційних систем та мереж і те, що можна зробити для посилення безпеки;
- відповідальність (responsibility) – всі учасники відповідальні за безпеку інформаційних систем та мереж;
- реагування (response) – учасники повинні діяти одночасно і спільно для захисту, виявлення та реагування на інциденти безпеки;
- етика (ethics) – учасники повинні дотримуватись законних інтересів одне одного;
- демократія (democracy) – безпека інформаційних систем та мереж повинна бути сумісна з основними цінностями демократичного суспільства;
- оцінювання ризику (risk assessment) – учасники повинні проводити оцінювання ризику;
- планування (проектування) і застосування безпеки (security design and implementation) – учасники повинні включати механізми забезпечення безпеки в інформаційних системах та мережах;
- управління безпекою (security management) – учасники повинні використовувати вичерпний підхід до управління безпекою;
- переоцінювання (reassessment) – учасники повинні переглядати і переоцінювати безпеку інформаційних систем та мереж, а також проводити відповідні модифікації політики безпеки, практики оцінювання та інших необхідних процедур.

Незалежно від розмірів організації і специфіки її інформаційної системи робота із забезпечення режиму інформаційної безпеки у тому чи іншому вигляді повинна містити в собі наступні етапи:

- визначення політики інформаційної безпеки;
- визначення сфери (меж) системи управління інформаційною безпекою і конкретизація цілей її створення;
- управління ризиками;
- оцінювання ризиків;
- вибір контрзаходів, що забезпечують режим інформаційної безпеки;
- аудит системи управління інформаційної безпеки.

Політикою інформаційної безпеки визначається, яким чином компанія (організація, заклад) повинна забезпечувати безпеку комп'ютерних систем. Визначаються такі рівні комплексних заходів забезпечення безпеки:

- законодавчий (закони, нормативні акти, стандарти);
- адміністративний (дії загального характеру, що чиняться керівництвом організації);
- процедурний (заходи безпеки, що реалізуються персоналом);
- програмно-технічний (конкретні технічні заходи).

Останнім часом у сучасній комп'ютерній літературі багато говориться про таку практичну стратегію досягнення інформаційної гарантованості (information assurance) у мережевому обладнанні, що називається ешелонованою обороною (defense in depth). Стратегія ешелонованої оборони рекомендує застосування наступних принципів інформаційної гарантованості для технології:

- застосування захисту в багатьох місцях. Оскільки зловмисники можуть атакувати систему з безлічі місць, включаючи зовнішні і внутрішні, організація повинна застосовувати захисні механізми в різних точках, що повинно забезпечувати захист мереж та інфраструктури, захист меж мережі і території, а також захист комп'ютерного обладнання;
- застосування рівневого захисту передбачає встановлення захисних механізмів між потенційним зловмисником і об'єктом атаки;
- визначення стійкості безпеки досягається шляхом оцінюванням захисних

характеристик кожного компоненту інформаційної гарантованості;

- застосування інфраструктури виявлення атак і вторгнень, використання методів та засобів аналізу і кореляції одержуваних за допомогою даної інфраструктури результатів.

Концепція ешелонованої оборони в даний час є загальноприйнятою, тому виробники засобів захисту реалізують її випуском цілих лінійок захисних засобів, що функціонують спільно і управління якими здійснюється за допомогою, як правило, єдиного пристрою управління.

Оскільки не існує одного пристрою для забезпечення захисту від атак на цілісність, конфіденційність і доступність, використовуються комбінації компонентів захисту, що значно зменшує можливість успішної атаки. До основних компонентів безпеки відносяться:

- міжмережеві екрани;
- системи виявлення вторгнень;
- засоби контролю цілісності;
- засоби перевірки вмісту;
- сканери безпеки;
- засоби контролю конфігурацій;
- засоби контролю доступу й аутентифікації;
- віртуальні приватні мережі (VPN);
- вбудовані засоби безпеки (операційні системи, засоби аудиту, списки контролю доступу і т.д.).

Нарешті, останній етап роботи організації із забезпечення режиму інформаційної безпеки – аудит.

Під аудитом інформаційної системи розуміється системний процес оцінювання об'єктивних даних про поточний стан системи, події, що відбуваються в ній, що встановлює рівень їхньої відповідності визначеному критерію. Проведення аудиту дозволяє оцінити поточну безпеку функціонування інформаційної системи, оцінити ризики і управляти ними,

прогнозувати їх вплив на процеси організації, коректно й обґрунтовано підходити до питання забезпечення безпеки інформаційних активів організації, основними з яких є: ідеї, знання, проекти, результати внутрішніх обстежень.

1.4. Засоби забезпечення захисту комп'ютерних систем.

Проблема захисту комп'ютерних систем вимагає враховувати різні фактори – *правові*, *морально-етичні*, *адміністративні*, *фізичні* і *технічні*.

До *правових засобів* захисту даних належать діючі в країні закони, укази, стандарти та інші нормативні акти, які регламентують правила користування даними з обмеженим доступом і відповідальність за їх порушення.

До *морально-етичних засобів* забезпечення захисту інформаційних ресурсів належать всілякі норми поведінки, які традиційно склались або складаються з поширенням комп'ютерів в країні або в суспільстві. Ці норми здебільшого не є обов'язковими і законодавчо не затверджені, але їх невиконання в більшості випадків веде до падіння авторитету, престижу людини, групи осіб чи організації.

Морально-етичні норми бувають як неписані (наприклад, загальновизнані норми правдивості, патріотизму та ін.), так і оформлені в деякий статут правил і приписів.

Адміністративні засоби захисту інформаційних ресурсів – це засоби організаційного характеру, які регламентують процеси функціонування системи опрацювання даних, використання її ресурсів, діяльність персоналу, а також порядок роботи користувачів з системою таким чином, щоб найбільшою мірою ускладнити або виключити можливість реалізації загрози безпеці комп'ютерної системи. Вони включають:

- розробку правил опрацювання даних в комп'ютерній системі;
- заходи, що передбачаються при проектуванні, будівництві і облаштуванні обчислювальних центрів та інших об'єктів опрацювання даних за допомогою комп'ютерів (врахування впливу стихії, пожеж, охорона приміщень, організація захисту від встановлення підслуховуючої

апаратури і т.ін.);

- заходи, які здійснюються при доборі та підготовці персоналу (перевірка нових співробітників, ознайомлення їх з порядком роботи із конфіденційними даними, із ступенем відповідальності за порушення правил їх опрацювання;
- створення умов, за яких персоналу було б невигідно допускати зловживання і т.ін.); організацію надійного пропускного режиму;
- організацію обліку, зберігання, використання і знищення документів та носіїв з конфіденційними даними;
- розподілення реквізитів розмежування доступу (паролів, профілів повноважень тощо);
- організацію підготовки контролю за роботою користувачів і персоналу комп'ютерних систем;
- заходи, що здійснюються при проектуванні, розробці, ремонті і модифікації обладнання та програмного забезпечення (сертифікація всіх технічних і програмних засобів, які використовуються; суворе санкціонування, розгляд і затвердження всіх змін; перевірка їх на задоволення потреб захисту, документальне відображення змін і т.ін.).

Адміністративні засоби відіграють значну роль у забезпеченні безпеки комп'ютерних систем. Ці засоби необхідно використовувати тоді, коли інші методи і засоби захисту недоступні (відсутні або дуже дорогі). Але такі засоби потрібно скрізь, де лише можливо, заміняти надійнішими сучасними фізичними і технічними засобами.

Фізичні засоби захисту даних – це різного роду механічні, електро- або електронно-механічні пристрої і спорудження, спеціально призначені для створення фізичних перепон на можливих шляхах проникнення і доступу потенційних порушників до компонентів комп'ютерних систем та даних, що потребують захисту.

Фізична безпека комп'ютерів передбачає встановлення комп'ютерів на

металевих столах із спеціальними нішами для системних блоків або зі знімним металевим ковпаком. Така вимога, наприклад, корисна для запобігання можливого пошкодження вінчестера від випадкових поштовхів і коливань. Диски з цінними даними необхідно зберігати в сейфі, а не в ящику письмового столу. Конфіденційні дані повинні шифруватись. Прокладання мережевих кабелів бажано здійснювати в металевих коробах чи трубах, що зробить під'єднання до них більш складним. Там, де це доцільно, використовувати оптоволоконні кабелі. Повторювачі (репітери) і концентратори, а також комутатори, які останнім часом помітно потіснили їх, потрібно розміщувати у шафах, що замикаються і т.ін.

Технічними (апаратно-програмними) засобами захисту є різні електронні пристрої та спеціальні програми, які використовуються самостійно або в комплексі з іншими засобами, виконуючи функції захисту, тобто ідентифікації і аутентифікації користувачів, розмежування доступу до ресурсів, реєстрації подій, криптографічного захисту даних тощо.

Розділ 2. Криптографічні методи захисту даних

2.1. Основні поняття криптографії

Самим надійним захистом від несанкціонованого доступу до повідомлень, що передаються якимось чином, і програмних продуктів персонального комп'ютера є застосування різних методів шифрування (криптографічних методів захисту інформаційних ресурсів).

Криптологія (cryptology; від грецького κρυπτος – таємний, прихований, λογος – наука, слово, повідомлення) – наука про математичні методи захисту повідомлень шляхом їх перетворення; це галузь знань, де вивчаються тайнопис (криптографія) і методи її розкривання (криптоаналіз). Термін використовується для позначення всієї галузі секретного зв'язку. Криптологія поділяється на три напрямки: *криптографію* (cryptography), *стеганографію* (steganography) і *криптоаналіз* (cryptoanalysis).

Криптографія (cryptography, crypto; від kriptos – прихований, graphos – письмо) – галузь знань про захист повідомлень, де вивчаються методи перетворення повідомлень, що забезпечує їх конфіденційність, автентичність та виключає їх прочитання без знання ключа. Криптографія – це і мистецтво, і наука, і технологія.

Фахівців з криптографії, які займаються розробкою шифрів, називають *криптографами* (cryptographer).

Ключ (cryptographic key, cryptokey, іноді просто key) – послідовність символів, використовувана в криптографічних системах, необхідна для безперешкодного шифрування і дешифрування текстів. Таємність ключа забезпечує криптостійкість усієї системи.

Шифрування – це основний криптографічний метод захисту повідомлень, що забезпечує її конфіденційність.

Під конфіденційністю розуміють неможливість одержання даних з перетвореного масиву без знання додаткових даних (ключа). Автентичність даних полягає у вірогідності авторства і цілісності.

Шифрування даних – це процес перетворення відкритих даних в зашифровані і навпаки. Перша частина процесу називається *зашифровуванням*, друга – *розшифровуванням*. Також термін шифрування (у вузькому сенсі) використовується як синонім зашифровування. Однак неправильно в якості синоніму шифрування використовувати термін «кодування» (а замість «шифру» – «код»), оскільки під кодуванням зазвичай розуміють подання даних у вигляді відповідного набору знаків (наприклад, літер алфавіту). Шифрування – це частковий випадок кодування.

Зашифровування (encryption) – процес перетворення відкритих даних у зашифровані за допомогою шифру. Замість терміну «відкриті дані» часто використовують терміни відкритий текст і вихідний текст, а замість «зашифровані дані» – шифрований текст.

Розшифровування (decryption) – процес, зворотний зашифровуванню, тобто процес перетворення зашифрованих даних у відкриті за допомогою ключа. У деяких джерелах окремо виділяють термін дешифрування (deciphering), маючи на увазі під ним відновлення вихідного тексту на основі шифрованого без знання ключа, тобто методами криптоаналізу. Термін «дешифрування», як правило, застосовують стосовно процесу криптоаналізу шифротексту (криптоаналіз може полягати і в аналізі шифросистеми, а не лише зашифрованого відкритого повідомлення).

Математичні алгоритми, відповідно до яких виконується шифрування, іменуються *шифрами* (cipher, cypher, ciphercode).

Алгоритми шифрування – математичні правила, на основі яких створюються і зламуються секретні шифри. Основною характеристикою алгоритму шифрування є криптостійкість.

Криптостійкістю (cryptographic strength) називається характеристика шифру, за якою визначається його стійкість до дешифрування без знання ключа (тобто криптоаналізу). Зазвичай, ця характеристика визначається часовим інтервалом, необхідним для розкриття шифру.

Показник криптостійкості – основний параметр будь-якої криптосистеми.

В якості показника криптостійкості можна вибрати:

- кількість усіх можливих ключів або ймовірність визначення ключа за заданий час із заданими ресурсами;
- кількість операцій або час (із заданими ресурсами), необхідний для злому шифру із заданою ймовірністю;
- вартість обчислення ключових даних або вихідного тексту.

Криптоаналіз (cryptanalysis) – набір методик і алгоритмів дешифрування криптографічно захищених повідомлень, аналізу шифросистем. В ньому поєднуються математичні методи порушення конфіденційності й автентичності, а також розшифровування повідомлень без знання ключів.

Криптоаналітики (cryptanalyst) – це дешифрувальники.

Сучасний криптоаналіз спирається на такі математичні науки, як теорія ймовірностей і математична статистика, алгебра, теорія чисел, теорія алгоритмів та ряд інших. Усі методи криптоаналізу в цілому утворюють чотири напрямки.

1. Статистичний криптоаналіз. Досліджуються можливості злому криптосистем на основі вивчення статистичних закономірностей вихідних і зашифрованих повідомлень. Його застосування ускладнене тим, що в реальних криптосистемах повідомлення перед шифруванням піддаються стискуванню (перетворюючи вихідний текст у випадкову послідовність символів), або у випадку гамування використовуються псевдовипадкові послідовності великої довжини.
2. Алгебраїчний криптоаналіз – пошук математично слабких ланок криптоалгоритмів.
3. Диференціальний (або різницевий) криптоаналіз. Заснований на аналізі залежності зміни шифрованого тексту від зміни вихідного тексту.
4. Лінійний криптоаналіз. Метод, заснований на пошуку лінійної апроксимації[1]

[1] Апроксима́ція (лат. approximare – наближати) – наближене вираження одних математичних об'єктів іншими, простішими, напр. кривих ліній – ламаними, ірраціональних чисел – раціональними, неперервних

між вихідним і шифрованим текстом. Як і диференціальний аналіз, у реальних криптосистемах може бути застосований лише для аналізу окремих блоків криптоперетворень.

Спробу прочитання або підробки зашифрованого повідомлення, обчислення ключа за методами криптоаналізу називають *криптоатакою* або *атакою на шифр*. Вдалу криптоатаку називають *зломом*.

Прийнято розрізняти кілька рівнів криптоатаки в залежності від обсягу даних, доступних криптоаналітику. Можна виділити три рівні криптоатаки за зростанням складності.

- Атака на шифрований текст (рівень КА1). Порушникові доступні всі або деякі зашифровані повідомлення.
- Атака на пару «вихідний текст – шифрований текст» (рівень КА2). Порушникові доступні всі або деякі зашифровані повідомлення і відповідні їм вихідні повідомлення.
- Атака на обрану пару «вихідний текст – шифрований текст» (рівень КА3). Порушник має можливість вибирати вихідний текст, одержувати для нього шифрований текст і на основі аналізу залежностей між ними обчислювати ключ.

Усі сучасні криптосистеми є достатньо стійкі навіть до атак рівня КА3, тобто коли порушникові майже доступний шифруючий пристрій (або алгоритм шифрування).

Шифри, які взагалі неможливо розкрити, називаються *абсолютно* або *теоретично стійкими*. Існування подібних шифрів доводиться за теоремою Шеннона, однак ціною цієї стійкості є необхідність використання для шифрування кожного повідомлення ключа, не меншого за розміром, ніж саме повідомлення. В усіх випадках, за винятком ряду особливих, ця ціна є надмірною, тому на практиці в основному використовуються шифри, що не є абсолютно стійкими. Таким чином, найбільш поширені схеми шифрування

функцій – многочленами і т. д.

можуть бути розкриті за кінцевий час або, що точніше, за кінцеве число кроків, кожний з яких є деякою операцією над числами.

Важливе значення має поняття практичної стійкості, за якою визначаються практичні труднощі розкриття шифру. Кількісною мірою цих труднощів може служити число елементарних арифметичних і логічних операцій, які необхідно виконати, щоб розкрити шифр, тобто щоб для заданого шифротексту з ймовірністю, не меншою від заданої величини, визначити відповідний відкритий текст.

Усі сучасні криптосистеми побудовані за принципом Керкхоффа[2], тобто таємність зашифрованих повідомлень визначається таємністю ключа. Це значить, що навіть якщо сам алгоритм шифрування відомий криптоаналітику, той все одно не в змозі розшифрувати повідомлення, якщо не має у своєму розпорядженні відповідного ключа.

Отже, шифрування є процесом перетворення оригінального повідомлення (часто іменованого відкритим текстом) у кодований текст, що представляється випадковим набором символів. Прочитати зашифроване повідомлення зможе лише той, хто знає, як перетворити його у вихідне повідомлення (тобто декодувати).

Криптографічні методи захисту даних – це спеціальні методи шифрування, кодування або іншого перетворення повідомлень, в результаті чого їх зміст стає недоступним без пред'явлення ключа криптограми і зворотного перетворення.

Криптографічний метод захисту найбільш надійний, тому що охороняються безпосередньо самі дані, а не доступ до них (наприклад, зашифрований файл не можна прочитати навіть у випадку крадіжки носія).

[2] Принцип Керкхоффа (Kerchkoff) – принцип створення і поширення криптографічних алгоритмів, відповідно до якого в секреті тримається лише певний набір параметрів шифру (і в обов'язковому порядку криптографічний ключ), а все інше може бути відкритим без зниження криптостійкості алгоритму. Цей принцип був уперше сформульований у роботі «Військова криптографія» голландського криптографа Керкхоффа.

Даний метод захисту реалізується у вигляді програм або пакетів програм, що додаються до стандартної операційної системи. Захист на рівні операційної системи, найчастіше, повинен доповнюватися засобами захисту на рівні систем управління базами даних, що дозволяє реалізовувати складні процедури управління доступом.

2.2. Коротка історія криптографії

Історія криптографії нараховує кілька тисяч років. Потреба ховати написане з'явилася в людини майже відразу, як тільки вона навчилась писати. За деякими теоріями, сама мова виникла не тоді, коли з'явилася потреба спілкуватися, а тоді, коли древні люди захотіли приховати свої думки.

Ще в далекі часи питання про приватне збереження відомостей було дуже актуальним. Наприклад, в Єгипті, коли вмирали фараони, їхні тіла муміфікували, а потім їх ховали з папірусами. На них були написані секретні слова, за допомогою яких померлий міг потрапити на небеса. Вважається, що саме єгиптяни (близько 2000 років до н.е.) розробили основи криптографії. Рання криптографія була примітивна і часто полягає у перестановці символів, наприклад, «фараон» перетворювали у «нфоаар», алгоритм такого шифрування розгадати нескладно.

Історію криптографії умовно можна поділити на 4 етапи: наївна криптографія, формальна криптографія, наукова криптографія і комп'ютерна криптографія.

Для наївної криптографії (до поч. XVI століття) характерне використання будь-яких (як правило, примітивних) способів заплутування супротивника щодо змісту шифрованих текстів. На початковому етапі для захисту повідомлень використовувалися методи кодування і стеганографії (див. Розділ 3), що споріднені, але не тотожні криптографії.

Більшість з використовуваних шифрів зводилися до переставляння або моноалфавітної підстановки (шифрування заміною). *Перестановка* – нескладний метод криптографічного перетворення, що полягає в перестановці

місцями символів вихідного тексту за деяким правилом. Шифри перестановок в наш час не використовуються в чистому вигляді, оскільки їх криптостійкість недостатня. *Моноалфавітна підстановка* – це найбільш простий вид перетворення, що полягає в заміні символів вихідного тексту на інші (того ж алфавіту) за більш чи менш складним правилом. У випадку моноалфавітних підстановок кожний символ вихідного тексту перетворюється в символ шифрованого тексту за одним і тим самим правилом.

Одним з перших зафіксованих прикладів є шифр Ю. Цезаря (100-44 рр. до н.е.), який шифрував листи під час листування з Цицероном і своїми сенаторами. Цей шифр (див. питання 2.7.1.1) полягав у заміні кожної літери вихідного тексту іншою, віддаленою від неї в алфавіті на певне число позицій (з циклічним перенесенням, коли це необхідно). Інший шифр, полібіанський квадрат (див. питання 2.7.1.4), авторство якого приписується грецькому письменникові Полібію, є загальною моноалфавітною підстановкою, що проводиться за допомогою випадково заповненої алфавітом квадратної таблиці (для грецького алфавіту розмір становить 5x5).

Подібні криптографічні алгоритми називаються симетричними (із закритим ключем) , тому що для кодування/декодування використовується той самий секретний ключ. Знайти надійний спосіб передавання партнерові секретного ключа зовсім непросто. Тому в сучасних незахищених мережах, у яких потрібно провести обмін ключами, використовуються спеціальні протоколи управління ключами. У державних структур існують спеціальні служби розсилання ключів, причому досить дорогі. З іншого боку, шифри на базі секретних ключів швидко кодуються. Слід відмітити, що в урядових і військових системах зв'язку дуже довгий час використовувалися лише симетричні алгоритми.

Етап формальної криптографії (кін. XV століття – поч. XX століття) пов'язаний з появою формалізованих і досить стійких до ручного криптоаналізу шифрів. У європейських країнах це відбулося в епоху Відродження, коли розвиток науки і торгівлі викликав попит на надійні способи захисту

повідомлень. Важлива роль на цьому етапі належить Леона Баттиста Альберті (1404-1472 р.), італійському архітекторові, що одним з перших запропонував багатоалфавітну підстановку . Даний шифр, що одержав ім'я дипломата XVI століття Б. Віжінера (див. питання 2.7.3.1), полягав в послідовному «додаванні» літер вихідного тексту до ключа (процедуру можна полегшити за допомогою спеціальної таблиці). Його робота «Трактат про шифр» (1466) вважається першою науковою працею з криптології.

Однією з перших друкованих праць, у якій узагальнені і сформульовані відомі на той момент алгоритми шифрування, є праця «Поліграфія» (1508 р.) німецького абата І. Трисемуса. Йому належать два невеликих, але важливих відкриття: спосіб заповнення полібіанського квадрата (перші позиції заповнюються за допомогою ключового слова, що запам'ятовується легко, інші – літерами алфавіту, що залишилися) і шифрування пар літер (біграм).

Простим, але стійким способом багатоалфавітної заміни (підстановки біграм) є шифр Плейфера, що був відкритий на початку XIX століття Ч. Уітстоном (сам шифр названий на честь Л. Плейфера, що зробив його використання обов'язковим у міністерстві закордонних справ Великобританії). Уітстону належить і важливе удосконалення – шифрування «подвійним квадратом» (див. питання 2.7.1.7). Шифри Плейфера й Уітстона використовувалися аж до першої світової війни, оскільки погано піддавалися ручному криптоаналізу.

У XIX столітті голландець Керкхофф сформулював головну вимогу до криптографічних систем, що залишається актуальною і понині: таємність шифрів повинна бути заснована на таємності ключа, а не алгоритму.

Нарешті, останнім словом у донауковій криптографії, що забезпечило ще більш високу криптостійкість, а також дозволило автоматизувати (у значенні механізувати) процес шифрування, стали роторні криптосистеми.

Однією з перших подібних систем стала винайдена в 1790 році Т. Джефферсоном, майбутнім президентом США, механічна машина. Багатоалфавітна підстановка за допомогою роторної машини реалізується

варіацією взаємного положення обертових роторів, за кожним з яких здійснюється «прошита» у ньому підстановка.

Практичне поширення роторні машини одержали лише на початку XX століття. Однією з перших практично використовуваних машин стала німецька Enigma, розроблена в 1917 році Е. Хеберном і удосконалена А. Кірхом. Роторні машини активно використовувалися під час другої світової війни. Крім німецької машини Enigma використовувалися також пристрої Sigaba (США), Typex (Великобританія), Red, Orange і Purple (Японія). Роторні системи – вершина формальної криптографії, оскільки відносно просто реалізовували дуже стійкі шифри. Успішні криптоатаки на роторні системи стали можливими лише з появою ЕОМ на початку 40-х років минулого століття.

У 20 столітті широко застосовувалися «книжкові» шифри, в яких у якості ключа використовувалося яке-небудь масове друковане видання. Такі шифри досить легко розкривались. З теоретичної точки зору, «книжковий» шифр виглядає досить надійним, оскільки множина його ключів – множина усіх сторінок усіх доступних двом сторонам книг, перебрати які вручну неможливо. Однак при найменших апріорних відомостях різко звужується цей вибір.

Під час Великої Вітчизняної війни значну увагу в СРСР приділяли організації партизанського руху. Майже кожний загін у тилу ворога мав радіостанцію, а також те чи інше сполучення з «великою землею». Шифри, що були в партизанів, були вкрай нестійкими – німецькі дешифрувальники розкривали їх досить швидко. А це виливалося в бойові поразки і втрати. Партизани виявилися винахідливими й у цій галузі теж. Прийом був гранично простим. У вихідному тексті повідомлення робилась велика кількість граматичних помилок, наприклад, писали: "пройшсли трі єшшелоні з тиками". При правильному розшифровуванні для російської людини усе було зрозуміло. Але криптоаналітики супротивника перед подібним прийомом виявились безсилими: перебираючи можливі варіанти, вони зустрічали неможливе для російської мови сполучення «тнк» і відкидали даний варіант як абсолютно неправильний. Цей, здавалося б, доморощений прийом, насправді, дуже

ефективний і часто застосовується навіть тепер. У вихідний текст повідомлення підставляються випадкові послідовності символів, щоб ускладнити процес розшифровування за допомогою криптоаналітичних програм, що функціонують за методом перебирання, чи змінюються статистичні закономірності шифрограми, що також може додати роботи супротивникові. Але в цілому все-таки можна сказати, що довоєнна криптографія була вкрай слабка, і назвати її серйозною наукою складно.

При дешифруванні криптограм (тобто криптоаналізу) протягом століть був у нагоді аналіз частоти появи окремих символів і їх сполучень. Імовірності появи окремих літер у тексті сильно різняться (для російської і української мови, наприклад, літера «а» зустрічається в 30-40 разів частіше, ніж літера «ш»). Таким чином, досить знайти в підстановочній шифровці найчастіше використовувані коди символів, щоб з великою імовірністю стверджувати, що вони являють собою розповсюджені літери вітчизняного алфавіту «о», «а» чи «є». Причиною такого легкого дешифрування є значна надмірність (в інформаційному сенсі) тексту, поданого природною мовою.

Головна відмітна риса наукової криптографії (30-і – 60-і роки XX століття) – поява криптосистем зі строгим математичним обгрунтуванням криптостійкості.

У 1926 році інженер компанії AT&T Ж. Вернам запропонував шифр, що дійсно не розкривається. Його ідея полягає в тому, щоб у криптоалгоритмі циклічної підстановки для кожного наступного символу вибирати нове значення відступу (константу). Іншими словами, секретний ключ повинен використовуватися лише один раз. Якщо такий ключ вибирається випадковим чином, то, як це було строго доведено згодом, шифр є таким, що не розкривається.

Такий криптоалгоритм став теоретичним обгрунтуванням для використання так званих шифроблокнотів, широке застосування яких почалося в роки Другої світової війни. Шифроблокнот містить безліч ключів одноразового використання, що послідовно обираються при шифруванні

повідомлень. Пропозиція Вернама, однак, значно ускладнює задачі секретного зв'язку, адже щоразу до передавання зашифрованого повідомлення необхідно передати секретний ключ, рівний йому за довжиною, тобто такий, що містить стільки ж символів, скільки є у відкритому тексті.

До початку 30-х років остаточно сформувалися розділи математики, що є науковою основою криптології: теорія імовірностей, математична статистика, загальна алгебра і теорія чисел, почали активно розвиватися теорія алгоритмів, теорія інформації, кібернетика. Своєрідним водорозділом стала робота Клода Шеннона «Теорія зв'язку в секретних системах» (1949), де сформульовані теоретичні принципи криптографічного захисту повідомлень. Шеннон увів поняття «розсіювання» і «перемішування», обгрунтував можливість створення як завгодно стійких криптосистем.

У 60-х роках ведучі криптографічні школи підійшли до створення блокових шифрів, що є більш стійкими в порівнянні з роторними криптосистемами. Вони допускають практичну реалізацію лише у вигляді цифрових електронних пристроїв.

Блокові (або блочні) шифри найбільш розповсюджені на практиці. Вони являють собою сімейство зворотних перетворень блоків (частин фіксованої довжини) вихідного тексту. Фактично блочний шифр – це система підстановки на алфавіті блоків (вона може бути моно- чи багатоалфавітною в залежності від режиму блочного шифру). Одним з розповсюджених способів задания блочних шифрів є використання так званих мереж Фейстеля – методу перетворення довільної функції (що зазвичай називається F-функцією) в перестановку на множині блоків, створених Х. Фейстелем.

Відомі вчені Рівест (подекуди в літературі: Райвест, Rivest), Шамір (Shamir), Адлмен (Adleman), Ель-Гамаль (ElGamal), Меркль (Merkle), Діффі (Diffie), Хеллман (Hellman) та ін. є винахідниками багатьох криптоалгоритмів, які використовуються сьогодні.

З появою обчислювальних засобів з високою продуктивністю стала розвиватись комп'ютерна криптографія (з 70-х років XX століття).

Використання сучасних криптосистем забезпечує при великій швидкості шифрування на кілька порядків більш високу криптостійкість, ніж за допомогою «ручних» і «механічних» шифрів.

Першим класом криптосистем, практичне застосування яких стало можливим з появою потужних і компактних обчислювальних засобів, стали блокові шифри, що поділяються на два види: шифри переставляння (permutation, P-блоки) і шифри заміни (підстановки, substitution, S-блоки).

2.3. Принципи створення криптографічних алгоритмів

Довгий час криптографія була більш мистецтвом, чим наукою. Автори криптоалгоритмів діяли, як правило, «на удачу», оскільки у їх розпорядженні не було відповідної математичної теорії, за допомогою якої можна було б формалізувати криптографічні перетворення і перевести їх на мову науки.

Як уже зазначалося вище (див. питання 2.2), першою роботою, яка радикально змінила таке положення, прийнято вважати роботу американського інженера і математика Клода Шеннона «Теорія зв'язку у секретних системах», опубліковану у 1949 році. У цій роботі Шеннон запропонував два загальних принципи створення криптоалгоритмів (шифрів): *розсіювання і перемішування*.

Розсіюванням Шеннон назвав розповсюдження впливу одного знаку відкритого тексту на декілька знаків шифротексту. Узагальненням такого принципу є розповсюдження впливу одного знака ключа на декілька знаків шифротексту, що перешкоджає відновленню ключа по частинам.

Перемішуванням називають використовування таких криптографічних перетворень, які ускладнюють відновлення взаємозв'язку символів відкритого і зашифрованого тексту, а також ключа і зашифрованого тексту.

Алгоритм криптоперетворення повинен забезпечувати легкість зашифрування і розшифрування (за умовою, що секретний ключ є відомим). Розповсюдженим способом створення шифру є послідовне виконання нескладних зворотних перетворень, в яких найчастіше використовують операції додавання, множення, підстановки і перестановки. Підстановка задає

перемішування, а перестановка – розсіювання. Під час перестановки змінюється порядок послідовності символів відкритого тексту. Під час підстановки сукупність символів відкритого тексту заміняється сукупністю такої же кількості символів, а конкретний вигляд підстановки визначається секретним ключем.

При використанні багатократного чергування простих підстановок і перестановок можна отримати стійкий алгоритм шифрування (шифр) з хорошим загальним розсіюванням і перемішуванням.

2.4. Модель захищеної комп'ютерної системи

В будь-якому інформаційному обміну даними є два учасника взаємодії: відправник повідомлення і його одержувач (Рис. 2.1). Найчастіше відправник повідомлення бажає, щоб на усьому шляху проходження зміст повідомлення зберігався у таємниці, тобто щоб зловмисник – перехоплювач повідомлення не зміг зрозуміти його зміст. Тому відправник повідомлення за допомогою шифратора, у якому відкритий текст С піддається перетворенню f(C,K), отримує шифротекст S і саме його передає по відкритому каналу зв'язку. Після отримання повідомлення на основі ключа К вибирається зворотне перетворення f(S,K), яке виконується у дешифраторі. В результаті одержувач повідомлення отримує відкритий текст С.

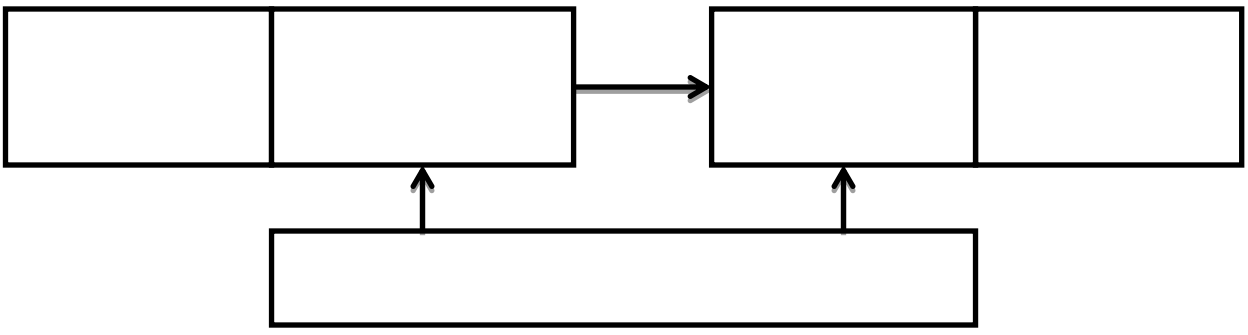


Рис. 2.1 Модель захищеної комп'ютерної системи

2.5. Криптографічні примітиви

До криптографічних примітивів відносять *алгоритми шифрування* і *генератори псевдовипадкових чисел*.

Алгоритми шифрування, по визначенню, повинні бути зворотними, оскільки у протилежному випадку відновити зашифровані дані буде неможливо. Тому будь-який алгоритм шифрування (або просто шифр) представляє собою дві зв'язані математичні функції, які використовують з метою прямого і зворотного перетворення даних (зашифрування і розшифрування).

Хороший алгоритм шифрування має наступні статистичні характеристики:

- відсутність статистичної залежності між відкритим текстом і шифротекстом;
- шифротекст за статистичними характеристиками не відрізняється від випадкової послідовності символів;
- зміна будь-якого символу (біта) у ключі шифрування при незмінному відкритому тексті призводить до зміни біля 50% символів (біт) шифротексту (для симетричних алгоритмів);
- зміна будь-якого символу (біта) в блоці відкритого тексту при незмінному ключі шифрування призводить до зміни біля 50% символів (біт) шифротексту (для блокових алгоритмів).

У сучасних шифрах згідно з принципом Керкхоффса їх секретність забезпечується секретністю змінного елементу – ключа шифрування, а не секретністю алгоритму. Таким чином, відкрите опублікування усіх деталей реалізації криптографічного алгоритму не повинно знижувати надійність шифру, якщо ключ шифрування зберігається у таємниці. Тому ключ шифрування повинен вибиратися випадковим чином. Однак у комп'ютерній техніці, у загальному випадку, не має хорошого джерела випадкових даних, за допомогою якого можна видавати значну кількість істинно випадкових даних. Тому у криптографії знаходять широке застосування генератори псевдовипадкових даних.

Псевдовипадкові дані не зовсім те саме, що істинно випадкові. У генераторах псевдовипадкових даних застосовується детермінований алгоритм

за допомогою якого видається послідовність значень в залежності від початкового значення, яке завантажено у генератор. Знаючи початкове значення, легко відтворити послідовність, яку видає генератор.

Криптографічні генератори псевдовипадкових чисел повинні задовольняти наступним вимогам:

- послідовність, яка видається за допомогою алгоритму генерації псевдовипадкових даних, повинна мати як можливо більший період;
- знаючи будь-який фрагмент послідовності, яка видається генератором, зловмисник не повинен мати ефективної можливості знайти початкове значення, яке у завантажується генератор;
- знаючи будь-який фрагмент послідовності, яка видається генератором, зловмисник не повинен мати можливості отримати достовірні дані про попередні чи наступні елементи послідовності.

У більшості мов програмування використовуються функції для генерації псевдовипадкових чисел. Однак, такі функції у більшості випадків не задовольняють перерахованим вимогам. Так, наприклад, функція rand стандартної бібліотеки мови С не задовольняє жодній перерахованій вище вимозі. Тому не варто використовувати для захисту даних генератори, які використовуються у мовах програмування, якщо достовірно не відома їх криптографічна стійкість.

2.6. Класифікація методів шифрування

В даний час не існує загальноприйнятої класифікації криптографічних методів захисту даних. Однак, можна виділити два основних типи криптографічних алгоритмів:

- *симетричні*, для яких ключ розшифрування або співпадає з ключем зашифрування, або може бути легко отриманий з нього;
- *асиметричні*, в яких для зашифрування і розшифрування використовуються два різних ключа. Асиметричні алгоритми також називають алгоритмами з відкритим ключем.

Задача забезпечення конфіденційності передавання даних з використанням симетричних криптосистем зводиться до забезпечення конфіденційності ключа шифрування. Звичайно ключ шифрування – це файл або масив даних, який зберігається на персональному носії.

Симетричне шифрування ідеально підходить для захисту даних «для себе», наприклад, з метою запобігання несанкціонованого доступу до даних за відсутності власника. Це може бути як архівне шифрування вибраних файлів, так і прозоре (автоматичне) шифрування цілих логічних чи фізичних носіїв даних.

Симетричне шифрування використовувати незручно тим, що перед початком обміну зашифрованими даними необхідно обмінятися секретними ключами з усіма адресатами. Передавання секретного ключа не може бути здійснено по загальнодоступним каналам зв'язку, секретний ключ від його джерела необхідно передавати відправнику і одержувачу по захищеному каналу розповсюдження ключів.

Крім того, використання одного ключа для всіх абонентів симетричної криптографічної мережі неприпустимо, оскільки у разі компрометації (втрати, крадіжки) ключа під загрозою буде знаходитись документообіг усіх абонентів. У такому випадку використовують матрицю ключів.

Матриця ключів представляє собою таблицю, яка містить ключі парного зв'язку абонентів. Кожен елемент таблиці K_{ij} призначений для зв'язку абонентів i та j і доступний тільки двом даним абонентам. Це означає, що для усіх елементів матриці ключів дотримується рівність $K_{ij}=K_{ji}$. Кожний i-й рядок матриці – це набір ключів конкретного абонента i для зв'язку з іншими абонентами. Набори ключів (мережеві набори) розповсюджуються між усіма абонентами криптографічної мережі. Слід зауважити, що мережеві набори повинні розповсюджуватись по захищеним каналам зв'язку або з рук у руки.

Основною перевагою асиметричних алгоритмів перед симетричними є те, що секретний ключ, за допомогою якого розшифровують усі отримані дані, відомо тільки одержувачу. Окрім того, первинне розповсюдження ключів у

системі не потребує передачі секретного ключа, який може бути перехоплено зловмисником. Несиметричні алгоритми отримали нову якість – на їх основі будуються протоколи цифрового підпису. Для аутентифікації з використанням симетричних алгоритмів часто необхідна участь довіреної третьої сторони, яка зберігає копії секретних ключів усіх користувачів. У такому випадку компрометація третьої сторони може привести до компрометації усієї системи аутентифікації. У системах з відкритим ключем така проблема усунута – кожен користувач відповідає за безпеку тільки свого секретного ключа.

Симетричні алгоритми теж мають свої переваги. Так, у разі виявлення в них будь-яких слабостей вони можуть бути дороблені шляхом внесення невеликих змін, а для несиметричних алгоритмів така можливість відсутня. Ще однією перевагою симетричних алгоритмів є їх швидкодія. Шифрування за допомогою симетричних алгоритмів відбувається набагато швидше, ніж за допомогою алгоритмів з відкритим ключем.

Симетричні алгоритми в свою чергу розділяють на дві категорії. До першої категорії відносяться алгоритми, за допомогою яких опрацьовують дані посимвольно (або побітово), і такі алгоритми називають *потоковими шифрами*. До другої категорії відносять алгоритми, за допомогою яких виконуються операції над групами символів (бітів). Такі групи символів (бітів) називають блоками, а алгоритми – *блоковими шифрами*.

2.7. Симетричні методи шифрування

2.7.1. Потокові криптографічні алгоритми

Шифрування відкритого повідомлення $C=\{c_1,c_2,\ldots,c_n\}$ для захисту від несанкціонованого доступу потоковими методами шифрування виконується посимвольно. У такому випадку символи повідомлення C залишаються на своїх позиціях і заміняються символами шифротексту $S=\{s_1,s_2,\ldots,s_n\}$.

До переваг потокових шифрів відносяться: відсутність розмноження помилок, проста реалізація, висока швидкість шифрування. Головним же недоліком є необхідність передачі даних синхронізації до заголовка

повідомлення, яка повинна бути прийнята до розшифрування будь-якого повідомлення.

2.7.1.1. Шифр Цезаря

Шифр Цезаря є прикладом шифру простої заміни. Його винайшов Юлій Цезар у 50-ті р. до н.е. У первинному вигляді у алгоритмі криптоперетворень передбачалось застосування незмінного ключа (*K=3*). З метою підвищення криптостійкості шифру зараз на значення ключа накладається лише одне обмеження: це повинно бути ціле додатне число.

Часто для зручності використання шифру Цезаря використовують два диски різного діаметру з намальованими на краях дисків алфавітами (Рис. 2.2). Спочатку диски повертають так, щоб напроти кожної букви алфавіту зовнішнього диска знаходилася та сама буква алфавіту внутрішнього диска. Якщо тепер повернути внутрішній диск на декілька символів, то ми отримаємо відповідність між символами зовнішнього диска і внутрішнього – шифр Цезаря. Цей диск можна використовувати як для зашифрування, так і для розшифрування повідомлень.

Рис. 2.2 Диск с шифром Цезаря.

Зашифрування – криптоперетворення *S=f(C,K).*

Для кожного символу повідомлення *C* відкритого тексту від першого символу до останнього необхідно виконати наступні операції:

- знайти *i(C)* – порядковий номер символу *C*;
- отримати *i(S)* – номер зашифрованого символу за формулою *i(S)= i(C)+K*;
- за номером *i(S)* знайти і записати символ шифротексту *S*.

Примітка. Якщо порядковий номер символу шифротексту *i(S)* перевищує довжину алфавіту, то його значення необхідно зменшити на довжину алфавіту або можна виконати «ділення за модулем» усіх порядкових номерів зашифрованих символів.

Розшифрування – зворотне криптоперетворення *C=f(S,K).*

Для кожного символу *S* зашифрованого повідомлення від першого символу до останнього необхідно виконати наступні операції:

- знайти *i(S)* – порядковий номер символу *S*;
- отримати номер розшифрованого символу *i(C)* за формулою *i(C)= i(S)-K*;
- за номером *i(C)* знайти і записати символ відкритого тексту *C*.

Примітка. Якщо порядковий номер символу відкритого повідомлення *i(C)* менше ніж 1, то до різниці додають довжину алфавіту.

Криптостійкість шифру слабка.

Приклад: Нехай ключ *K=3* і повідомленням для шифрування буде фраза «*i remember that september*». Будемо використовувати латинські букви із стандартним проходженням букв в алфавіті. Результати шифрування вказаної вище фрази показані нижче (див. Таблиця 2.1):

Таблиця 2.1

1	**i**		**r**	**e**	**m**	**e**	**m**	**b**	**e**	**r**		**t**	**h**	**a**	**t**		**s**	**e**	**p**	**t**	**e**	**m**	**b**	**e**	**r**
2	**9**	**0**	**18**	**5**	**13**	**5**	**13**	**2**	**5**	**18**	**0**	**20**	**8**	**1**	**20**	**0**	**19**	**5**	**16**	**20**	**5**	**13**	**2**	**5**	**18**
3	**12**	**3**	**21**	**8**	**16**	**8**	**16**	**5**	**8**	**21**	**3**	**23**	**11**	**4**	**23**	**3**	**22**	**8**	**19**	**23**	**8**	**16**	**5**	**8**	**21**
4	**12**	**3**	**21**	**8**	**16**	**8**	**16**	**5**	**8**	**21**	**3**	**23**	**11**	**4**	**23**	**3**	**22**	**8**	**19**	**23**	**8**	**16**	**5**	**8**	**21**
5	**l**	**c**	**u**	**h**	**p**	**h**	**p**	**e**	**h**	**u**	**c**	**w**	**k**	**d**	**w**	**c**	**v**	**h**	**s**	**w**	**h**	**p**	**e**	**h**	**u**

Пояснення до таблиці:

1-й рядок - фраза для шифрування;

2-й рядок - номери букв фрази для шифрування в латинському алфавіті;

3-й рядок - номери букв фрази для шифрування, збільшені на 3;

4-й рядок - результат «ділення по модулю 27» чисел 3-го рядка;

5-й рядок - зашифрована фраза.

Результатом шифрованої фрази буде: «*lcuhphpehucwkdwcvhswhpehu*».

2.7.1.2. Шифр Цезаря з ключовим словом

Шифр Цезаря з ключовим словом є одно алфавітною системою підстановки. Особливістю цієї системи є використання ключового слова для зсуву та зміни порядку символів в алфавіті підстановки. Виберемо деяке число *K*, від 0 до 25, і слово або коротку фразу в якості ключового слова. Бажано, щоб всі букви ключового слова були різними. Нехай слово DIPLOMAT буде ключовим і число *K=5*.

Ключове слово записується під літерами алфавіту, починаючи з букви, числове значення якої збігається з обраним числом *K* (див. Таблиця 2.2):

Таблиця 2.2

0	1	2	3	4	***5***	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
A	**B**	**C**	**D**	**E**	**F**	**Q**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**0**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
					D	***I***	***P***	***L***	***O***	***M***	***A***	***T***													

Решту букв алфавіту підстановки записують після ключового слова в алфавітному порядку (див. Таблиця 2.3):

Таблиця 2.3

					5																				
A	**B**	**C**	**D**	**E**	**F**	**Q**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**0**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
V	**W**	**X**	**Y**	**Z**	***D***	***I***	***P***	***L***	***O***	***M***	***A***	***T***	**B**	**C**	**E**	**F**	**G**	**H**	**J**	**K**	**N**	**Q**	**R**	**S**	**U**

Тепер для кожної букви довільного повідомлення маємо підстановку. Фраза «*i remember that september*» буде зашифровано як «*lgztztwzgjpvjhzejztwzg*».

Слід відмітити, що вимога про відмінність всіх букв ключового слова не є обов'язкова. Можна просто записати ключове слово (або фразу) без повторення однакових букв.

2.7.1.3. Шифр Трітеміуса

Шифр Трітеміуса[3] є модифікацією шифру Цезаря, а також ще називають шифром Віженера (див. питання 2.7.3.1). Ключове слово може бути будь-якої довжини (якщо воно буде складатися з одного символу, то отримаємо шифр Цезаря) і складатися з будь-яких символів алфавіту. У випадку, якщо ключове слово коротше за повідомлення, його повторюють циклічно.

Зашифрування і *розшифрування* повідомлень виконується за алгоритмами, які аналогічні алгоритмам шифру Цезаря, але порядковий номер символу шифротексту визначається за формулою $i(S)=i(C)+i(Kj)$, а порядковий номер символу відкритого повідомлення за формулою $i(C)=i(S)-i(Kj)$, де $i(Kj)$ – порядковий номер у алфавіті символу Kj ключового слова.

Ефективність криптоперетворень шифру «алфавітне додавання» залежить від розміру ключа. Чим довше ключове слово, тим вища його криптостійкість. Найкращий результат досягається, якщо довжина ключа дорівнює довжині повідомлення. При виборі такого ключа потрібно використовувати шифроблок або кодову книгу.

Криптостійкість шифру середня.

Приклад: Нехай ключовим словом буде «leonid» (латиниця) і повідомленням для шифрування фраза «I remember that September». Букви ключового слова мають наступні номери в латинському алфавіті: 12, 5, 15, 14, 9, 4. Шифрування за даним методом полягає в «зрушенні» першої букви вихідної фрази на 12 позицій, тобто в заміні букви «i» (9-а позиція) на букву «u», що знаходиться 9+12=21-й позиції, в заміні пропуску « » (другої букви кодованої фрази, 0-а позиція) на букву «e», що знаходиться в 0+5=5-й позиції і так далі. При «вичерпанні» букв ключового слова, останнє використовується знову і знову до тих пір, поки не будуть зашифровані всі букви вихідної фрази. Результати шифрування вказаної вище фрази показані нижче (див Таблиця 2.4):

[3] Шифр пов'язаний з маєтком вченого абата з Вюрцбурга Трітеміуса, якого до занять криптографією спонукало не тільки монастирське усамітнення, а й потреба зберігати від розголосу деякі духовні таємниці.

Таблиця 2.4

1	**i**		**r**	**e**	**m**	**e**	**m**	**b**	**e**	**r**		**t**	**h**	**a**	**t**		**s**	**e**	**p**	**t**	**e**	**m**	**b**	**e**	**r**
2	**9**	**0**	**18**	**5**	**13**	**5**	**13**	**2**	**5**	**18**	**0**	**20**	**8**	**1**	**20**	**0**	**19**	**5**	**16**	**20**	**5**	**13**	**2**	**5**	**18**
3	**l**	**e**	**o**	**n**	**i**	**d**	**l**	**e**	**o**	**n**	**i**	**d**	**l**	**e**	**o**	**n**	**i**	**d**	**l**	**e**	**o**	**n**	**i**	**d**	**l**
4	**12**	**5**	**15**	**14**	**9**	**4**	**12**	**5**	**15**	**14**	**9**	**4**	**12**	**5**	**15**	**14**	**9**	**4**	**12**	**5**	**15**	**14**	**9**	**4**	**12**
5	**21**	**5**	**33**	**19**	**22**	**9**	**25**	**7**	**20**	**32**	**9**	**24**	**20**	**6**	**35**	**14**	**28**	**9**	**28**	**25**	**20**	**27**	**11**	**9**	**30**
6	**21**	**5**	**6**	**19**	**22**	**9**	**25**	**7**	**20**	**5**	**9**	**24**	**20**	**6**	**8**	**14**	**1**	**9**	**1**	**25**	**20**	**0**	**11**	**9**	**3**
7	**u**	**e**	**f**	**s**	**v**	**i**	**y**	**g**	**t**	**e**	**i**	**x**	**t**	**f**	**h**	**n**	**a**	**i**	**a**	**y**	**t**		**k**	**i**	**c**

Пояснення до таблиці.

1-й рядок - фраза для шифрування;

2-й рядок - номери букв фрази для шифрування в латинському алфавіті;

3-й рядок - ключове слово з довжиною рівній довжині фрази;

4-й рядок - номери букв ключового слова в алфавіті;

5-й рядок - сума номерів 2-го і 4-го рядків у відповідних стовпцях;

6-й рядок - результат «ділення по модулю 27» чисел 5-го рядка;

7-й рядок - зашифрована фраза.

Таким чином, «шифровка» матиме вигляд: uefsviygteixtfhnaiayt kic.

Розшифрування «шифровки» проводимо «аналогічно» зашифруванню фрази.

2.7.1.4. Шифр Полібія

Шифр Полібія, також відомий як шахова дошка Полібія або квадрат Полібія - один з найдавніших методів шифрування, запропонований Полібієм[4]. Даний спосіб кодування спочатку застосовувався для грецької абетки, але потім поширився і на інші мови. Шифр Полібія є прикладом шифру складної заміни,

[4] Поліби́й (грец. Πολιβιος, лат. Polybius, близько 201 до н. е., Мегалополь, Аркадія – близько 120 до н. е.) – давньогрецький історик, державний і військовий діяч, автор «Загальної історії» («Історії») в 40 томах, що охоплюють події у Римі, Греції, Македонії, Малій Азії та в інших регіонах з 220 до н. е. по 146 до н. е. З книг «Історії» повністю збереглись тільки перші 5, інші дійшли в більш-менш детальних переказах. Всі інші праці Полібія не збереглись

оскільки один буквений символ відкритого повідомлення замінюється двома цифровими символами шифротексту.

До виконання криптоперетворень усі букви базового алфавіту розміщають у квадраті Полібія.

Зашифрування – криптоперетворення $S=f(C)$.

Кожний символ повідомлення C відкритого тексту замінюється двома цифровими символами шифротексту, перший з яких вказує на номер рядка, а другий – на номер стовпця, на перетині яких знаходиться символ відкритого повідомлення у квадраті Полібія.

На практиці шифротекст виводять групами символів, наприклад по чотири. Це роблять для того, щоб при «зломі» шифру криптоаналітик не мав можливості визначити кількість слів у відкритому повідомленні та довжину кожного слова.

Розшифрування – криптоперетворення $C=f(S)$.

Кожні два цифрових символи зашифрованого повідомлення замінюються одним буквеним символом відкритого повідомлення: перший з них вказує на номер рядка, а другий – на номер стовпця, де знаходиться буква-символ відкритого повідомлення.

Криптостійкість шифру слабка, оскільки у шифрі не використовується ключ.

Приклад: Візьмемо квадрат 5x5, стовпці і рядки якого нумерувалися від 1 до 5. У кожну клітинку цього квадрата записується одна буква латинського алфавіту (див. Таблиця 2.5).

Таблиця 2.5

	1	**2**	**3**	**4**	**5**
1	**A**	**B**	**C**	**D**	**E**
2	**F**	**G**	**H**	**I,J**	**K**
3	**L**	**M**	**N**	**O**	**P**
4	**Q**	**R**	**S**	**T**	**U**
5	**V**	**W**	**X**	**Y**	**Z**

Зашифруємо фразу Декарта «*Cogito, ergo sum*» («*Я мислю, отже, існую*»). Тоді результатом буде така послідовність (див. також Таблиця 2.6): *13 34 22 24 44 34 15 42 22 34 43 45 32*.

Таблиця 2.6

13	**34**	**22**	**24**	**44**	**34**	**15**	**42**	**22**	**34**	**43**	**45**	**32**
C	**O**	**G**	**I**	**T**	**O**	**E**	**R**	**G**	**O**	**S**	**U**	**M**

2.7.1.5. Шифр Полібія з ключовим словом

Шифр «модульна арифметика» є модифікацією шифру Полібія з ключовим словом. У якості ключа в даному шифрі вибирають слово, яке складається з унікальних символів. Таке слово розміщують на початку квадрата Полібія. Наступні за ключем елементи квадрата заповнюються буквами у порядку їх проходження в алфавіті з пропуском тих букв, які вже містяться у ключовому слові. Самі ж алгоритми криптоперетворень залишаються такими ж, як і у шифрі Полібія без ключа.

Слід зауважити, що ефективність шифрування залежить від ключового слова, оскільки саме від нього буде залежите якість перемішування символів у квадраті Полібія. Для більш якісного перемішування у ключове слово доцільно включати останні символи базового алфавіту.

Криптостійкість шифру середня.

Приклад: Візьмемо квадрат 5x5, стовпці і рядки якого нумерувалися від 1 до 5. Нехай ключове слово буде У кожну клітинку цього квадрата записується одна буква латинського алфавіту (див. Таблиця 2.5).

Таблиця 2.7

	1	**2**	**3**	**4**	**5**
1	**D**	**R**	**A**	**F**	**T**
2	**B**	**C**	**E**	**G**	**H**
3	**I,J**	**K**	**L**	**M**	**N**
4	**O**	**P**	**Q**	**S**	**U**
5	**V**	**W**	**X**	**Y**	**Z**

Зашифруємо фразу Декарта «*Cogito, ergo sum*» («*Я мислю, отже, існую*»). Тоді результатом буде така послідовність (див. також Таблиця 2.8): *22 41 24 31 15 41 23 12 24 41 44 45 34*.

Таблиця 2.8

22	**41**	**24**	**31**	**15**	**41**	**23**	**12**	**24**	**41**	**44**	**45**	**34**
C	**O**	**G**	**I**	**T**	**O**	**E**	**R**	**G**	**O**	**S**	**U**	**M**

2.7.1.6. Шифр Плейфера

Шифр Плейфера або квадрат Плейфера – один із методів шифрування, в якому вперше використана заміна біграм. Винайдений у 1854 році Чарльзом Уітстоном, але названий на честь Лорда Лайона Плейфера, який запровадив цей шифр в державні служби Великобританії. У шифрі передбачається шифрування пар символів (біграм). Для цього використовують матрицю 5x5 (для латинського алфавіту, для українського алфавіту необхідно збільшити розмір матриці до 4x8), що містить ключове слово або фразу. Для створення матриці і використання шифру досить запам'ятати ключове слово і чотири простих правила (див. нижче). Щоб скласти ключову матрицю, в першу чергу потрібно заповнити порожні клітинки матриці буквами ключового слова (не записуючи символи, які повторюються), потім заповнити наступні клітинки матриці символами алфавіту, які не зустрічаються в ключовому слові, за порядком (для англійського алфавіту зазвичай опускається символ «Q», щоб зменшити кількість символів алфавіту, в інших версіях «I» і «J» об'єднують в одну клітинку). Ключове слово може бути записано у верхньому рядку матриці зліва направо, або по спіралі з лівого верхнього кута до центру. Ключове слово, доповнене алфавітом складає матрицю 5x5 і є ключем шифру.

Зашифрування – криптоперетворення *S=f(C,K).*

Для того, щоб зашифрувати повідомлення необхідно розбити його на біграми (групи з двох символів), наприклад повідомлення «*Hello World*» стає «*HE LL OW OR LD*», і відшукати ці біграми в таблиці. Два символи біграми відповідають кутам прямокутника в ключовій матриці. Визначаємо положення

кутів цього прямокутника відносно один одного. Потім керуючись наступними 4-ма правилами зашифровуємо пари символів вихідного тексту:

1. Якщо два символи біграми збігаються, додаємо після першого символу «X», зашифровує нову пару символів і продовжуємо. В деяких варіантах шифру Плейфера замість символу «X» використовується символ «Q».
2. Якщо символи біграми вихідного тексту зустрічаються в одному рядку, то ці символи заміщаються на символи, розташовані в найближчих стовпцях справа від відповідних символів. Якщо символ є останнім у рядку, то він замінюється на перший символ цього ж рядка.
3. Якщо символи біграми вихідного тексту зустрічаються в одному стовпці, то вони перетворюються на символи того ж стовпчика, що знаходяться безпосередньо під ними. Якщо символ є нижнім в стовпці, то він замінюється на перший символ цього ж стовпця.
4. Якщо символи біграмми вихідного тексту знаходяться в різних стовпцях і різних рядках, то вони замінюються на символи, що знаходяться в тих же рядках, але у протилежних кутах прямокутника.

Наприклад потрібно зашифрувати біграму OR. Розглянемо чотири випадки:

1)

*	*	*	*	*
*	O	Y	R	Z
*	*	*	*	*
*	*	*	*	*
*	*	*	*	*

OR замінюється на YZ

2)

*	*	O	*	*
*	*	B	*	*
*	*	*	*	*
*	*	R	*	*
*	*	Y	*	*

OR замінюється на BY

3)

Z	*	*	O	*
*	*	*	*	*
*	*	*	*	*
R	*	*	X	*
*	*	*	*	*

OR заменяется на ZX

4)

*	*	*	*	*
*	*	*	*	*
Y	O	Z	*	R
*	*	*	*	*
*	*	*	*	*

OR заменяется на ZY

Розшифрування – криптоперетворення *C=f(S,K).*

Для розшифровки необхідно використовувати інверсію цих чотирьох правил, відкидаючи символи «X» (або «Q»), якщо вони не використовуються у вихідному повідомленні.

Криптостійкість шифру середня.

Приклад: Нехай ключове слово буде «playfair example», тоді матриця матиме наступний вигляд:

P	L	A	Y	F
I	R	E	X	M
B	C	D	G	H
J	K	N	O	S
T	U	V	W	Z

Зашифруємо повідомлення «*Hide the gold in the tree stump*». Утвориться послідовність біграм: *HI DE TH EG OL DI NT HE TR EX ES TU MP*.

- Біграма HI утворює прямокутник, замінюємо її на BM.
- Біграма DE розміщена в одній колонці, замінюємо її на ND.
- Біграма TH утворює прямокутник, замінюємо її на ZB.
- Біграма EG утворює прямокутник, замінюємо її на XD.
- Біграма OL утворює прямокутник, замінюємо її на KY.
- Біграма DI утворює прямокутник, замінюємо її на BE.
- Біграма NT утворює прямокутник, замінюємо її на JV.
- Біграма HE утворює прямокутник, замінюємо її на DM.
- Біграма TR утворює прямокутник, замінюємо її на UI.
- Біграма EX розміщена в одному рядочку, замінюємо її на XM.
- Біграма ES утворює прямокутник, замінюємо її на MN.
- Біграма TU розміщена в одному рядочку, замінюємо її на UV.
- Біграма MP утворює прямокутник, замінюємо її на IF.

Отримуємо зашифрований текст «*BM ND ZB XD KY BE JV DM UI XM MN UV IF*»

Таким чином повідомлення «*Hide the gold in the tree stump*» перетвориться на «*BMNDZBXDKYBEJVDMUIXMMNUVIF*».

2.7.1.7. Шифр «подвійний квадрат» Уітстона

Цей шифр було також розроблено Чарльзом Уітстоном у 1854 р. На відміну від шифру Плейфера, в «подвійному квадраті» використовується одночасно дві таблиці, які розташовані по горизонталі. Базовий алфавіт у таких таблицях розміщено випадково. Ключем є розташування символів алфавіту у двох таблицях.

Такий шифр відноситься до біграмних. Це означає, що шифрування і дешифрування виконують відразу над двома символами повідомлення одночасно.

Зашифрування – криптоперетворення $S=f(C,K)$.

При шифруванні відкрите повідомлення розділяють на біграми, тобто по два символи. Потім у лівій таблиці знаходять першу букву біграми, а у правій таблиці – другу букву. По сукупності двох таблиць подумки будують прямокутник таким чином, щоб букви біграми були розташовані у його протилежних кутах. В інших двох кутах такого прямокутника розміщені символи шифротексту. Перша буква біграми шифротексту береться з правої таблиці у стовпці, який відповідає другій букві біграми відкритого повідомлення. Друга буква біграми шифротексту береться з лівої таблиці у стовпці, який відповідає першій букві біграми відкритого повідомлення.

Примітка. Якщо обидві літери біграми повідомлення лежать в одному рядку, то і букви шифртексту беруть з цього ж рядка. Першу букву біграми шифртексту беруть з лівої таблиці в стовпці, відповідно другої літери біграми повідомлення. Друга ж буква біграми шифртексту береться з правої таблиці в стовпці, відповідному першій букві біграми повідомлення.

Розшифрування – криптоперетворення $C=f(S,K)$.

При дешифруванні шифротекст розділяють на біграми. Потім у правій таблиці знаходять перший символ шифротексту, а у лівій таблиці – другий символ. По сукупності двох таблиць подумки будують прямокутник, у двох протилежних кутах якого розташовані букви біграми. В інших двох кутах прямокутника розміщені символами відкритого повідомлення. Перша буква

відкритого повідомлення береться з лівої таблиці у стовпці, який відповідає другому символу біграми шифротексту. Друга буква відкритого повідомлення береться з правої таблиці у стовпці, який відповідає першому символу біграми шифротексту.

Криптостійкість шифру середня. Злом шифротексту потребує багато зусиль і довжини відкритого повідомлення більш ніж тридцять рядків.

Приклад: Нехай маємо дві таблиці з випадково розташованими в них латинськими алфавітами:

N	L	X	Y	F
I	R	A	E	M
B	C	D	G	H
J	K	U	O	W
P	T	V	S	Z

V	D	Z	X	T
E	U	I	W	G
R	B	C	M	H
N	K	O	S	J
A	L	Y	P	F

Зашифруємо повідомлення «*Hide the gold in the tree stump*». Утвориться послідовність біграм: *HI DE TH EG OL DI NT HE TR EX ES TU MP*.

- Біграма HI утворює прямокутник, замінюємо її на CM.
- Біграма DE розміщена в одній колонці, замінюємо її на RA.
- Біграма TH утворює прямокутник, замінюємо її на FC.
- Біграма EG утворює прямокутник, замінюємо її на WM.
- Біграма OL утворює прямокутник, замінюємо її на KS.
- Біграма DI утворює прямокутник, замінюємо її на CA.
- Біграма NT утворює прямокутник, замінюємо її на VF.
- Біграма HE утворює прямокутник, замінюємо її на RM.
- Біграма TR утворює прямокутник, замінюємо її на AC.
- Біграма EX розміщена в одному рядочку, замінюємо її на WY.
- Біграма ES утворює прямокутник, замінюємо її на WO.
- Біграма TU розміщена в одному рядочку, замінюємо її на LR.
- Біграма MP утворює прямокутник, замінюємо її на WZ.

Отримуємо зашифрований текст «*CM RA FC WM KS CA VF RM AC WY*

WO LR WZ»

Таким чином повідомлення «*Hide the gold in the tree stump*» перетвориться на «*CMRAFCWMKSCAVFRMACWYWOLRWZ*».

2.7.1.8. Шифр Вернама

Шифр Вернама (інша назва: англ. one-time pad – шифр одноразових блокнотів) - система симетричного шифрування, винайдена в 1917 році співробітниками фірми AT&T Мейджором Джозефом Моборном і Гільбертом Вернамом. Шифр Вернама є єдиною системою шифрування, для якої доведена абсолютна криптографічна стійкість.

Шифр одноразових блокнотів походить від того, що агент, який здійснював шифрування вручну, отримував свої копії ключів записаними у блокноті. Як тільки ключ використовувався, сторінка з ним знищувалась. Зрозуміло, що шифр просто реалізується і технічними засобами. Кажуть, що саме шифр одноразового блокноту використовувався для захисту від підслуховування встановленої під час холодної війни гарячої лінії зв'язку між Москвою і Вашингтоном.

Однак обмеженість сфери застосувань шифру очевидна, оскільки для його використання потрібен ключ такої ж довжини як саме повідомлення. З цією обставиною пов'язані дві проблеми. Перша полягає в генеруванні довгої послідовності випадкових бітів. Другою проблемою є необхідність у надійному каналі для регулярного обміну довгим ключем (на зразок дипломатичної пошти). У більшості ситуацій такого каналу або взагалі немає, або ж він не є достатньо швидким.

Зашифрування – криптоперетворення *S=f(C,K).*

Для зашифрування повідомлення використовується операція додавання бітів за модулем 2 (XOR)[5]. Операція позначається символом ⊕ і задається так:

$$0 \oplus 0 = 0, \quad 0 \oplus 1 = 1, \quad 1 \oplus 0 = 1, \quad 1 \oplus 1 = 0$$

[5] Виключна диз'юнкція (також операція XOR, додавання за модулем 2) - логічна операція, що приймає значення «істина» тоді і тільки тоді коли значення «істина» має рівно один з її операндів.

Перед зашифруванням повідомлення *C* записують у двійковій формі. Ключем *K* може бути довільне двійкове слово однакової довжини з повідомленням *C*. Криптотекст *S* отримують побітовим додаванням повідомлення і ключа, тобто $S = C \oplus K$.

Розшифрування – криптоперетворення *C=f(S,K).*

Для двійкових слів X і Y однакової довжини результат їх побітового додавання позначатимемо як $X \oplus Y$. Легко перевірити рівності $X \oplus Y = Y \oplus X$ та $(X \oplus Y) \oplus Z = X \oplus (Y \oplus Z)$, що справджуються для будь-яких двійкових слів X, Y і Z однакової довжини. Для фіксованого k позначимо через 0 двійкове слово 000...00, що складається із k нулів. Очевидно, що для будь-якого двійкового слова X довжини k виконуються рівності $X \oplus 0 = X$ і $X \oplus X = 0$.

Розшифрування у шифрі Вернама збігається із зашифруванням - щоб отримати вихідне повідомлення *C*, слід додати до криптотексту *S* той же ключ *K*. Це легко обгрунтувати: оскільки $S = C \oplus K$, то

$$S \oplus K = (C \oplus K) \oplus K = C \oplus (K \oplus K) = C \oplus 0 = C.$$

Криптостійкість шифру висока.

Приклад: Як видно з попередніх методів шифрування, у період класичної криптографії як правило не виникало потреби записувати відкритий текст та криптотекст якось інакше, ніж у звичайній абетці. Завдяки цьому криптограф-практик не потребував для роботи нічого, крім письмового приладдя свого часу, чого було достатньо і для шифрування, і для пересилання повідомлення. Але як тільки потрібно скористатися сучасними засобами зв'язку для передавання повідомлення, або шифрування повідомлення за допомогою комп'ютера, то виявилося, що у технічному відношенні традиційний текст не є найзручнішою формою для перетворення та передачі даних.

З цього погляду зручним є подання даних у цифровій формі. Ідея є зовсім простою - кожен символ тексту заміняємо його номером у алфавіті. Нумерацію будемо починати з 0. Для прикладу, слово «*банан*» буде подане як *01 00 17 00 17*. Кожна літера представлена своїм номером, записаним двома цифрами,

перша з яких може бути нулем. При потребі в алфавіт можна включити окрім букв також знаки пунктуації, пропуск, цифри тощо.

Номери букв можна записувати не в десятковій системі числення, а у двійковій. Для того ж слова «*банан*» матимемо запис *000001 000000 010001 000000 010001*, де кожен блок із шести цифр є номером відповідної букви у двійковому записі. Таку форму подання тексту називають *двійковою*.

Таким чином, довільний текст можна записати у двійковій формі, використовуючи всього лише два символи - 0 та 1. Ці два символи називаються *бітами*. Будь-яку послідовність бітів називають *двійковим словом*.

Застосуємо операцію XOR до двійкових слів однакової довжини. Додавання таких слів здійснюється побітово.

Для прикладу, зашифруємо слово «*банан*». Подамо це слово у двійковій формі (див. вище): *C* = 000001000000010001000000010001.

В якості ключа виберемо: K = 001101110101100010011000111010.

Виконуючи сумування цих двох двійкових послідовностей отримаємо криптотекст: S = 001100110101110011011000101011, або

$$\begin{array}{r} 000001000000010001000000010001 \\ \oplus\ 001101110101100010011000111010 \\ \hline 001100110101110011011000101011 \end{array}$$

Шифр Вернама є абсолютно надійним або, як ще кажуть, надійним у теоретико-інформаційному сенсі. Якщо суперник не знає ключа *K*, то з підслуханого криптотексту *S* він зовсім нічого не може довідатись про повідомлення *C*. Справді, двійкове слово *S* могло би бути криптотекстом для будь-якого повідомлення *C*`, якби шифрування здійснювалось з деяким іншим ключем *K*`, а саме *K*` = *C*` ⊕ *S*, в той час як для суперника всі ключі однаково вірогідні. Наприклад, за допомогою деякого ключа *K* який був використаний для шифрування повідомлення «*банан*», отримали криптотекст *S* 001100110101110011011000101011. Але такий же криптотекст можна отримати, якщо зашифрувати повідомлення «*груша*» з ключем *K*` = 001111100001100100000100101011.

2.7.2. Блокові криптографічні алгоритми

При блоковому шифруванні відкрите повідомлення $C=\{c_1,c_2,...,c_n\}$ для захисту від несанкціонованого доступу спочатку розбивається на рівні за довжиною блоки символів $C`=\{c`_1,c`_2,...,c`_k\}$, $C``=\{c``_1,c``_2,...,c``_k\}$, …, $C^n=\{c^n_1,c^n_2,...,c^n_k\}$. Потім до кожного блоку застосовують залежно від ключа функцію шифрування з метою перетворення їх у блоки секретного повідомлення такої самої довжини. У результаті отримують блоки шифротексту $S`=\{s`_1,s`_2,...,s`_k\}$, $S``=\{s``_1,s``_2,...,s``_k\}$, …, $S^n=\{s^n_1,s^n_2,...,s^n_k\}$, які об'єднують у шифротекст $S=\{s_1,s_2,...,s_n\}$.

Слід відмітити, що довжина відкритого повідомлення не завжди є кратною довжині блока. Тому виникає проблема доповнення, яка має кілька рішень. Можливо, наприклад, передавати у незашифрованому вигляді розмір корисної частини зашифрованих даних і після розшифрування зайві символи просто відкидати. На практиці частіше застосовують інший спосіб. Усі невикористані символи блока, окрім останнього, заповнюють будь-якими значеннями, а в замість останнього символу пишуть число, яке дорівнює кількості невикористаних байтів. Тоді, отримавши і розшифрувавши усі блоки, необхідно відкинути з кінця стільки символів, скільки вказано у останньому символі останнього блоку. Але у такому випадку виникає деяка складність. Якщо довжина відкритого повідомлення кратна довжині блоку, то необхідно додати 0 символів, і останній символ має містити число символів доповнення. Для розв'язання такої проблеми при шифруванні додається новий блок, останній символ якого містить розмір блоку, який буде цілком відкинуто при розшифруванні.

Основною перевагою блокового шифрування є те, що у добре сконструйованій системі невеликі зміни відкритого повідомлення або ключа викликають великі і непередбачені зміни у шифротексті. До недоліків блокового шифрування відноситься: невірне розшифрування всього блока при наявності похибки лише у одному символі прийнятого шифротексту, можливість криптоаналізу «зі словником».

2.7.2.1. Шифр скітала («палиця»)

Даний шифр є історично першим алгоритмом, в якому використано ідею блокового шифрування. Його використовували правителі Спарти ще у V столітті до н.е.. Шифрування виконувалось наступним чином. На палицю, яка мала форму циліндру, намотували по спіралі вузьку папірусну стрічку (без просвітів і перекривань), а потім на намотаній стрічці уздовж осі циліндру записували відкрите повідомлення (Рис. 2.3). Після розмотування стрічки виходило, що поперек її в «безладді» були написані якійсь букви. Одержувач повідомлення брав таку саму палицю і таким самим чином намотував на неї отриману стрічку, відновлюючи тим самим текст повідомлення. По назві палиці і отримав назву даний шифр (скітала – грец. σκυτάλη «палиця»).

Рис. 2.3 Шифр скітала

Оскільки довжина кожного блоку залежить від діаметру палиці, то алгоритми криптоперетворень у сучасному розумінні можливо інтерпретувати наступним чином.

Зашифрування – криптоперетворення *S=f(C,K).*

По ключу формується таблиця, у яку по стовпцям заносять відкрите повідомлення, а зчитування шифротексту виконується по її рядкам.

Примітка. Ключем є розмір таблиці, тобто кількість її стовпців і рядків.

Розшифрування – криптоперетворення *C=f(S,K).*

По ключу формується таблиця, у яку по рядкам заносять шифротекст, а зчитування відкритого повідомлення виконується по її стовбцям.

Криптостійкість шифру середня.

Приклад: Нехай потрібно зашифрувати повідомлення «*Це шифр палиця*».

Побудуємо таблицю з чотирьох стовпців (це є ключ) і впишемо по рядкам дане повідомлення.

ц	е	ш	и
ф	р	п	а
л	и	ц	я

Зчитуючи дане повідомлення за стовпцями, отримаємо зашифроване повідомлення: *цфл ери шпц иая*.

2.7.2.2. Шифр стандартної перестановки

Даний шифр відрізняється від попереднього методу шифрування тим, що стовпці таблиці переставляються по ключовому *слову, фразі* чи *набору чисел* довжиною у рядок таблиці.

Зашифрування – криптоперетворення $S=f(C,K)$.

По ключу формується таблиця, у якій кількість стовпців дорівнює кількості символів ключового слова. Кількість рядків таблиці визначається відношенням кількості символів відкритого повідомлення до довжини ключа.

Відкрите повідомлення заноситься у таблицю по рядкам і кожний рядок є окремим блоком. Якщо останній рядок таблиці, тобто останній блок містить меншу кількість символів повідомлення ніж усі інші блоки, його доповнюють символами алфавіту, починаючи з першого, потім другий, третій і т.д.. Після заповнення таблиці символами відкритого повідомлення до верхньої її частини додають ще два рядки, у верхній з яких заносять ключове слово, а у нижній – номери, визначені згідно стандартному розташуванні букв ключа у алфавіті. У випадку, якщо ключове слово містить однакові букви, то їх упорядковують зліва направо.

Далі стовпці таблиці переставляють таким чином, щоб номера символів ключового слова збільшувались зліва направо, починаючи з першого.

Шифротекст зчитується по рядкам і може бути розбито на групи.

Розшифрування – криптоперетворення $C=f(S,K)$.

Будують таблицю, кількість стовпців у якій дорівнює кількості символів ключового слова ($n_{стовп}=l_к$), а кількість рядків $n_{ряд}=(l_s/l_к)+1$, де l_s – кількість

символів шифротексту.

У ключовому слові переставляють символи згідно звичайному їх розташуванні у базовому алфавіті і кожному з них привласнюють порядковий номер. У верхній рядок таблиці заносять ключове слово з символами, які переставлені згідно звичайному розташуванню в алфавіті, у другий рядок – нумерацію символів, а в інші – шифротекст по рядкам. Потім стовпці таблиці переставляють таким чином, щоб у верхньому рядку отримати вихідне ключове слово. Відкрите повідомлення зчитують по рядкам.

Криптостійкість шифру висока.

Приклад: За допомогою методу стандартної перестановки зашифруємо повідомлення «*перестановка*» з ключовим словом «*шифр*». Отримаємо таблицю (див. Таблиця 2.9):

Таблиця 2.9

ш	**и**	**ф**	**р**
4	**1**	**3**	**2**
п	е	р	е
с	т	а	н
о	в	к	а

Після впорядкування стовпців за порядковими номерами букв ключового слова отримаємо наступну таблицю (див. Таблиця 2.10)

Таблиця 2.10

и	**р**	**ф**	**ш**
1	**2**	**3**	**4**
е	е	р	п
т	н	а	с
в	а	к	о

Зчитуючи дане повідомлення за рядками, отримаємо зашифроване повідомлення: *еерп тнас вако*.

2.7.2.3. Шифр вертикальної перестановки

Алгоритм криптоперетворень шифру вертикальної перестановки такий ж самий, як у шифрі стандартної перестановки, однак при шифруванні у даному

методі шифротекст виводять з таблиці перестановки по вертикалі, тобто по стовпцям, а при розшифруванні по вертикалі заповнюють шифротекстом вихідну таблицю.

Криптостійкість шифру висока.

Приклад: За допомогою методу вертикальної перестановки зашифруємо повідомлення «*перестановка*» з ключовим словом «*шифр*». Отримаємо таблицю (див. Таблиця 2.11):

Таблиця 2.11

ш	**и**	**ф**	**р**
4	**1**	**3**	**2**
п	е	р	е
с	т	а	н
о	в	к	а

Після впорядкування стовпців за порядковими номерами букв ключового слова отримаємо наступну таблицю (див. Таблиця 2.12)

Таблиця 2.12

и	**р**	**ф**	**ш**
1	**2**	**3**	**4**
е	е	р	п
т	н	а	с
в	а	к	о

Зчитуючи дане повідомлення за стопцями, отримаємо зашифроване повідомлення: *етв ена рак псо*.

2.7.2.4. Шифри з використанням комбінованих перестановок

Комбінована перестановка включає три рівня захисту:

1. Формування вихідної матриці перестановки.
2. Перестановка по ключу *К*.
3. Зчитування шифротексту з матриці перестановки.

Вихідну матрицю перестановки можна заповнювати по рядках звичайно, по рядках інверсно, по вертикалі звичайно, по вертикалі інверсно, по діагоналі звичайно, по діагоналі інверсно, по спіралі Архімеда чи будь-яким іншим

способом. Кожним з перелічених способів можна вводити і зчитувати шифротекст з матриці перестановки.

З метою додаткової криптостійкості можна повторно шифрувати повідомлення, яке вже було зашифровано. Такий спосіб отримав назву «подвійна перестановка». Використовувати подвійні перестановки стовпців і рядків таблиці. У такому випадку перестановки визначають окремо для стовпців і окремо для рядків. При шифруванні у вихідну матрицю перестановки заносять відкрите повідомлення і переставляють спочатку стовпці, а потім рядки. При дешифруванні порядок перестановок зворотній.

Криптостійкість шифру висока.

Приклад: За допомогою методу комбінованих перестановок зашифруємо повідомлення «*перестановка*» з ключами «*3, 1, 2*» і «*4, 1, 3, 2*».

Процес шифрування полягає в тому, що зліва від першого стовпчика таблиці записують перший ключ, довжина якого співпадає з кількістю її рядків, а над верхнім рядком таблиці записують другий ключ, довжина якого співпадає з кількістю її стовпчиків. Після цього здійснюють вписування відкритого тексту у таблицю по рядках, але не підряд, а у відповідності з першим ключем, який діє на рядки. Отримаємо таблицю (див. Таблиця 2.13):

Таблиця 2.13

	ш	**и**	**ф**	**р**
	4	**1**	**3**	**2**
3	о	в	к	а
1	п	е	р	е
2	с	т	а	н

Кріптограма утворюється шляхом прочитування по стовпчиках, причому стовпчики беруться не підряд, а у порядку, визначеному другим ключем, який діє на стовпчики. Таким чином отримуємо кріптограму «*вет аен кра опс*».

2.7.2.5. Шифри-трафарети. Квадрат та прямокутник Кардано

У шифрах Джероламо Кардано в якості ключа використовують маску,

розмір якої співпадає з розміром матриці перестановки, де четверта частина комірок прорізана таким чином, що за чотири оберти на 90^0 вони покривають усю матрицю. Тому такі шифри відносять до шифрів-трафаретів.

Криптоперетворення можливо виконувати, використовуючи два види решіток: квадратну та прямокутну.

- Квадратна решітка Кардано (*2n*2n*).

  Шифрування – криптоперетворення *S=f(C,K).*

Квадратним трафаретом називають нанесену на планшет квадратну матрицю з прорізаними віконцями. Планшет, як маску, накладають на папір того ж розміру, і у віконця вписують знаки повідомлення по порядку слідування рядків зліва направо. Після першого заповнення планшет повертають на 90° за годинниковою стрілкою і процедуру вписування повторюють. Таким способом вписування знаків повідомлення у віконця може бути здійснене чотири рази.

Квадратний трафарет цікавий тим, що після кожного повороту віконця знаходяться над незаповненими клітинками паперу. З цією метою розташування віконець на трафареті підбирають спеціально. Кількість віконець не повинна перевищувати четвертої частини від загальної кількості клітинок трафарету. Щоб описати трафарет, застосовують позначення: нуль – віконце відсутнє, одиниця – віконце є. Таким чином, весь трафарет можна представити у вигляді сукупності двійкових чисел, кожне з яких відповідає його рядку. Ці числа доцільно представити у десятковій системі числення. Зрозуміло, що сукупність цих чисел являє собою ключ шифру.

Якщо кількість віконець трафарету виявиться більшою кількості знаків повідомлення, то порожні віконця заповнюються випадковими знаками.

Наприклад, повідомлення «*перестановки*» при застосуванні квадратного трафарета розміром 4*4, що має чотири віконця, може бути перетворено у кріптограму «*сапобтвекрваеинг*» (див. Таблиця 2.14). Цікаво, що на останньому четвертому повороті трафарету знаки повідомлення вже вичерпались, і тому

порожні віконця були заповнені додатковими випадковими знаками «*абвг*». Таким чином, в наведеному прикладі після останнього внесення знаків у віконця на папері виявились заповненими всі клітинки (див. Таблиця 2.14).

Таблиця 2.14

		п	
			е
	р		
е			

0°

с			
	т		
			а
		н	

90°

			о
		в	
к			
	и		

180°

	а		
б			
		в	
			г

270°

с	а	п	о
б	т	в	е
к	р	в	а
е	и	н	г

Ключем цього шифру є розмір квадратного трафарету 4*4 та послідовність двійкових чисел «0010, 0001, 0100, 1000» або, що те ж саме, десяткових чисел «2, 1, 4, 8», за допомогою яких цей квадратний трафарет описується.

Розшифрування – криптоперетворення *C=f(S,K).*

Для розшифрування кріптограми потрібний такий же самий квадратний трафарет. В наведеному вище прикладі трафарет має чотири віконця, що дорівнює четвертій частині від загальної кількості клітинок 4*4=16.

Слід відмітити, що для підвищення кріптографічної стійкості кількість віконець трафарету можна зробити меншою. Розглянемо відповідній приклад, в якому квадратний трафарет розміром 4*4 має три віконця і описується послідовністю «0, 5, 0, 8» (див. Таблиця 2.15).

Таблиця 2.15

	п		е
р			

0°

е			
		с	
		т	

90°

			а
н		о	

180°

	в		
	к		
			и

270°

е	в		а
	п	с	е
н	к	о	
р		т	и

За рахунок зменшення кількості віконець при шифруванні повідомлення «*перестановки*» після внесення знаків на останньому четвертому повороті залишаються вільними ще чотири клітинки «*ев_а_псенко_р_ти*». Їх слід заповнити додатковими випадковими знаками «*абвг*» вже без трафарету. Таким чином, отримуємо криптограму «*еваабпсенковргти*».

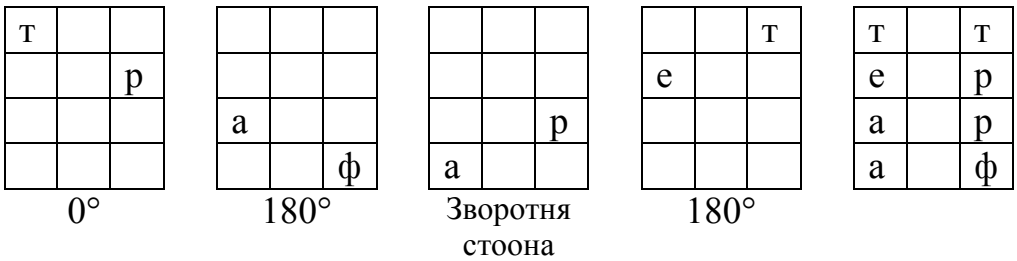
Шифр табличних перестановок з використанням квадратного трафарету є

блоковим. Розмір блоку можна визначити, якщо кількість віконець трафарету помножити на чотири (для першого прикладу розмір блоку становить 4*4=16 знаків, для другого прикладу – 3*4=12 знаків).

Кількість можливих квадратних трафаретів різко зростає зі збільшенням їх розміру: один для 2*2, 256 для 4*4, більше 100 тисяч – для 6*6. Ця кількість суттєво збільшується за рахунок використання зменшення кількості віконець.

- Прямокутна решітка Кардано (*2n*2m*).

Крім квадратних трафаретів існують також прямокутні трафарети. Для них застосовують повороти на 180°, а також перекладання на зворотню сторону. Розглянемо відповідній приклад, в якому прямокутний трафарет розміром 4*3 має два віконця. Цей трафарет описується послідовністю «4, 1, 0, 0».



За рахунок недостатньої кількості віконець при шифруванні повідомлення «*трафарет*» після внесення знаків на останньому четвертому положенні трафарету залишаються вільними ще чотири клітинки «*т_те_ра_ра_ф*». Їх слід заповнити додатковими випадковими знаками «*абвг*» вже без трафарету. Таким чином, отримуємо кріптограму «*татебраврагф*».

Криптостійкість шифрів висока.

2.7.3. Змішані симетричні криптосистеми

Крім потокових і блокових криптосистем, можливо також утворення змішаних систем, в яких використовують найкращі властивості кожної з них. У такому випадку відкрите повідомлення спочатку шифрують, як при звичайному потоковому шифруванні, а потім отриманий шифротекст розбивають на блоки фіксованого розміру і у кожному блоці додатково виконується перестановка за певним ключем. У результаті отримується шифр, який має додаткову властивість порівняно з потоковими шифрами, а саме: перехоплювач не знає,

якому біту (символу) відкритого повідомлення відповідає біт (символ) шифротексту. Завдяки такому підходу зашифроване повідомлення стає більш складнішим для розкриття.

Розглянутий підхід до створення змішаного (складного) шифру реалізовано, наприклад, у криптосистемі Віженера і криптосистемі DES (Data Encryption Standard).

Для реалізації криптоперетворень у шифрі Віженера використовують два ключа:

- ключ K_1 використовується для блокового шифрування, генерують квадрат Віженера (як генерується квадрат Віженера див. далі), за допомогою якого криптоперетворення кожного унікального символу виконується за зміненим базовим алфавіт використовуючи ключ K_1;
- ключ K_2 використовується для криптоперетворень повідомлень, використовуючи ідею потокового шифрування.

У криптосистемі DES при зашифруванні і розшифруванні повідомлень використовують також два ключа:

- ключ K_1 використовується для блокового шифрування, виконують початкову і зворотну перестановку;
- ключ K_2 використовується для визначення значення правого півблока на кожному з 16 раундів перестановки і заміни, використовуючи ідею потокового шифрування.

Крім того, у криптосистемі DES згідно з ідеєю блокового шифрування відкрите повідомлення розбивають на блоки однакової довжини і кожен з них піддають однаковому криптоперетворенню. Ідею блокового шифрування можливо простежити і при застосуванні перестановки правого півблоку вліво на кожному з 16 раундів перестановки і заміни.

Отже, в криптосистемі Віженера і криптосистемі DES в повній мірі використовуються властивості багатозначної заміни і властивості методів перестановки, тобто використовуються переваги потокових і блокових

криптоалгоритмів. Розглянемо більш детальніше цих два методи шифрування даних (цих дві криптосистеми).

2.7.3.1. Багатоалфавітна криптосистема Віженера

Дана криптосистема є першим шифром багатоалфавітної заміни, за допомогою якої шифрування стало дуже стійким до розкриття. Його створив Леон Баттіст Альберті у 1466 р.. У 1585 р. Блез де Віженер, який розвивав і удосконалював криптографічні системи, опублікував книгу про шифри, де вперше описав шифр багатоалфавітної підстановки на базі таблиці шифрування – квадрату Віженера. Надалі такий шифр стали називати його ім'ям.

Алгоритм генерації квадрату Віженера.

Квадрат Віженера складається з l_A стовпців і такої ж кількості рядків, де l_A – довжина базового алфавіту. У якості базового алфавіту візьмемо український алфавіт, який містить тільки прописні букви, тобто A={А,Б,В,Г,Ґ,Д,...,Ю,Я}. Це означає, що розмір квадрату буде 33*33.

Алгоритм генерації квадрату Віженера складається з наступних етапів:

1. У додатковий рядок і додатковий стовпець квадрату заноситься базовий алфавіт (виділено сірим кольором, див Таблиця 2.16).
2. Задати ключ К1. Візьмемо, наприклад, ключ К1=13572468.
3. До базового алфавіту застосовують початкову перестановку по ключу К1, у результаті чого отримують перший рядок квадрату (див. Таблиця 2.16).
4. Другий рядок квадрату отримується шляхом циклічного зсуву на один символ уліво першого рядка. Витиснутий перший символ переноситься у кінець рядка.
5. Третій рядок формується із другого аналогічно: виконується циклічний зсув уліво другого рядка на один символ, а витиснутий перший символ переноситься у кінець рядка. Так продовжують, доки не буде остаточно заповнено усі рядки квадрату.

Слід відмітити, що завдяки перестановці по ключу K_1 на третьому етапі алгоритму, квадрат Віженера може містити l_A різноманітних варіантів алфавітів

l_A*l_A, тобто 33*33 у нашому прикладі.

Квадрат Віженера, який побудовано на базі алфавіту A={А,Б,В,Г,Ґ,Д,...,Ю,Я} з використанням ключа K_1=13572468 згідно викладеному алгоритму, наведено нижче (див. Таблиця 2.16).

Зашифрування – криптоперетворення $S=f(C,K_1,K_2)$.

Зашифрування виконується посимвольно. Кожному символу відкритого повідомлення C вибирається рядок, а по символу ключового слова K_2 стовпець квадрату Віженера, на перетинанні яких усередині квадрату визначається символ шифротексту.

Таблиця 2.16

	А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
А	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
Б	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А
В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В
Г	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ
Ґ	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е
Д	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б
Е	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г
Є	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д
Ж	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є
З	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж
И	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З
І	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И
Ї	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І
Й	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї
К	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й
Л	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К
М	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л
Н	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М
О	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н
П	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О
Р	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П
С	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р
Т	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С
У	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т
Ф	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У
Х	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Ц	Ц	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х
Ч	Ч	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц
Ш	Ш	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч
Щ	Щ	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш
Ь	Ь	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Ю	Ю	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь
Я	Я	А	В	Ґ	Е	Б	Г	Д	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю

Ключове слово K_2 може бути будь-якої довжини і складатися з будь-яких символів алфавіту. Якщо ключове слово коротше за повідомлення, його повторюють циклічно.

Розшифрування – криптоперетворення *C=f(S,K1,K2).*

За ключем K_1 формують квадрат Віженера. Потім по символу ключового слова K_2 вибирається рядок, в якому усередині квадрату знаходимо символ шифротексту. В стовпці, який відповідає символу шифротексту, визначаємо символ відкритого повідомлення.

Криптостійкість шифру висока.

Приклад: Зашифруємо повідомлення *«криптосистема»* використовуючи ключ K_1=13572468 (див. Таблиця 2.16) та ключ K_2, якому надамо значення *«ключ»*. Відповідно до алгоритму зашифрування: на перетині рядка, в якому розміщена перша літера відкритого повідомлення *«к»* і стовпця, в якому міститься перший символ ключа K_2 (це символ *«к»*) отримаємо символ *«ш»*, який є першим символ зашифрованого повідомлення. На перетині рядка, в якому міститься друга літера відкритого повідомлення «*р*» і стовпця, в якому міститься другий символ ключа K_2 (це символ *«л»*) отримаємо символ «*ґ*», який є другим символом зашифрованого повідомлення, і т. д. Отримаємо зашифроване повідомлення «*шґжй йапб ґббu к*».

2.7.3.2. Багаторівнева криптосистема DES

Криптосистема DES (Data Encryption Standard) є одним з найкращих прикладів криптоалгоритму, який розроблено у відповідності з принципами розсіювання і перемішування, сформульованими Клодом Шенноном. Алгоритм шифру було розроблено співробітником фірми IBM Х. Фейстелем у 1976 р.. Після ряду змін і удосконалень, виконаних Національним агентством безпеки США (US NSA), криптосистему DES було прийнято у якості національного стандарту шифрування даних для ЕОМ і обчислювальних систем.

Зашифрування – криптоперетворення *S=f(C,K1,K2).*

Відкрите повідомлення розділяють на блоки таким чином, щоб довжина

кожного з них складала 64 біта. Потім кожен блок шифрують окремо, застосовуючи алгоритм шифрування.

Розглянемо основні етапи алгоритму шифрування DES:

1. До блоку відкритого тексту застосовують початкову фіксовану перестановку IP (initial permutation), яка задається ключем K_1.
2. 64-бітний блок розділяють на ліву L_0 і праву R_0 половини по 32 біта.
3. До отриманих половин застосовують 16 раундів перестановки і заміни по ключам K_i (i=1,2,...,16), які виробляються генератором псевдовипадкових чисел за заданим ключем K_2. На кожному i-му раунді компоненти правої і лівої половин блоку міняють місцями і до правої половини додається функція шифрування, яка залежить від значення правої половини блоку, отриманому на попередньому раунді, і значення ключа K_i, виробленому генератором псевдовипадкових чисел, тобто $f(R_{i-1},K_i)$. Звичайно така функція визначається як додавання компонентів по модулю 2 (mod 2).
4. До лівої L_{16} і правої R_{16} половин, які отримано після шістнадцятого раунду перестановки і заміни, застосовують операцію конкатенації і отримують 64-бітний блок.
5. До 64-бітного блоку застосовують зворотну перестановку IP^{-1}. Результатом є зашифрований блок тексту.

Викладений алгоритм шифрування можна представити у вигляді блок-схеми (Рис. 2.4 Блок-схема алгоритму шифрування DESРис. 2.4).

Дешифрування - криптоперетворення *C=f(S,K1,K2).*

Дешифрування виконується за допомогою ключа K_2, у якому змінено на протилежний спосіб адресації бітів. Перед початком криптоперетворення необхідно розділити шифротекст на блоки довжиною 64 біт, а потім до кожного блоку застосувати наступний алгоритм:

1. До 64-бітного блоку застосовують зворотну перестановку IP^{-1} за ключем K_1.
2. Блок розділяють на ліву L_{16} і праву R_{16} половини по 32 біта.
3. До отриманих половин застосовують 16 раундів перестановки і заміни по ключам K_i (i=16,15,...,1), які виробляються генератором псевдовипадкових

чисел за ключем K_2.

4. Ліву L_0 і праву R_0 половини, які отримано після шістнадцятого раунду перестановки і заміни, об'єднують у один блок довжиною 64 біта.
5. До отриманого блоку застосовують початкову перестановку IP. Результатом є блок відкритого повідомлення.

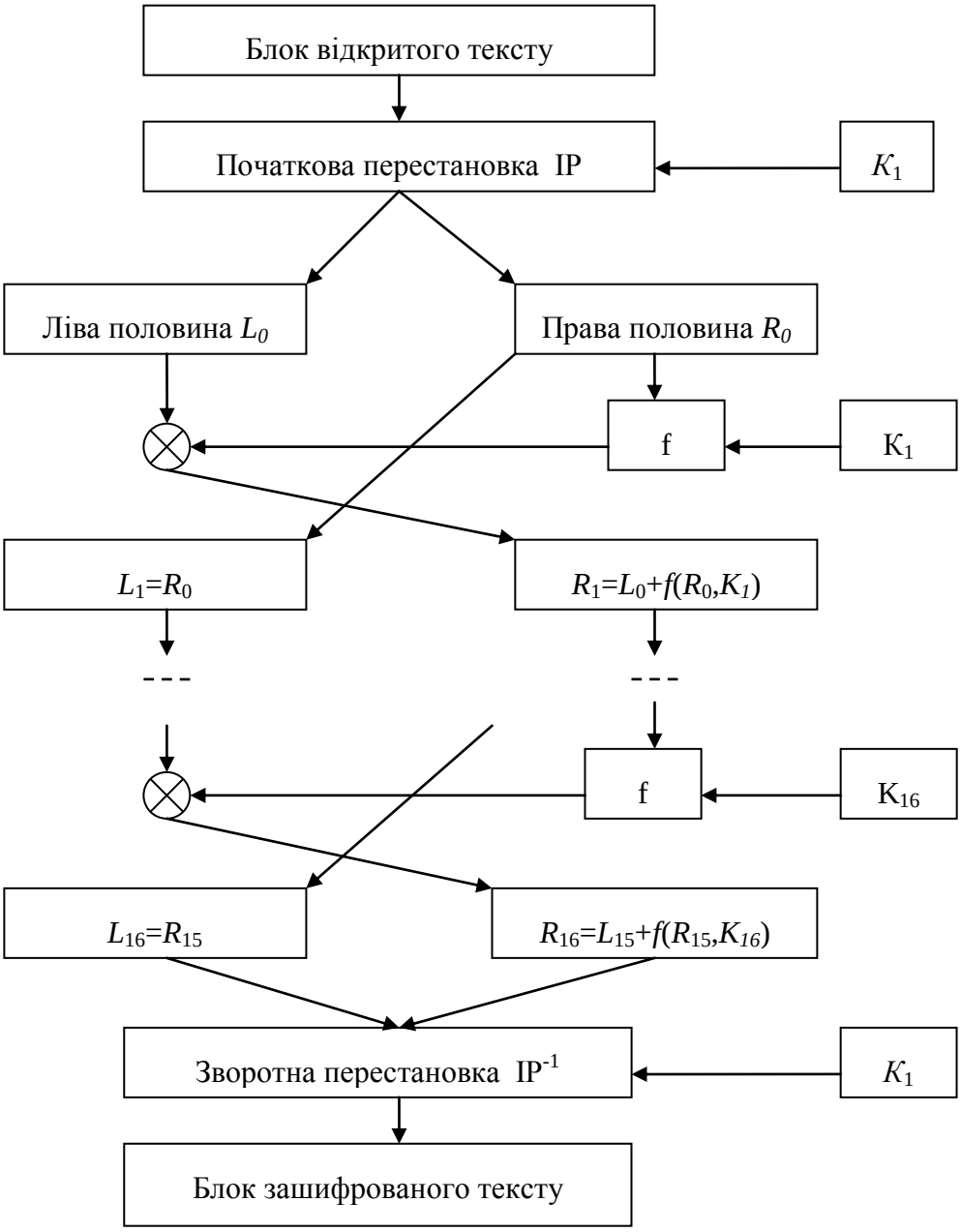


Рис. 2.4 Блок-схема алгоритму шифрування DES

Важливою особливістю криптосистеми DES є гнучкість при застосуванні у різноманітних додатках опрацювання даних. Оскільки кожен блок піддається

криптоперетворенню незалежно від усіх інших блоків, можна виконувати незалежну передачу даних і довільний доступ до них. У такому випадку для операцій шифрування і дешифрування не потрібна ні часова, ні позиційна синхронізація.

Криптостійкість шифру висока. Однак, вибір для нього ключів є дуже складною задачею. Наприклад звісно, що ключі, які складаються з 56 одиниць чи 56 нулів, зовсім не забезпечують захисту. Також слабкими виявляються ключі вигляду «28 одиниць та 28 нулями» і «28 нулів та 28 одиницями».

У випадках, коли надійність алгоритму DES недостатня, використовують його модифікацію – Triple DES (потрійний DES). Слід зауважити, що є кілька варіантів алгоритму Triple DES. Найчастіше використовують варіант шифрування на трьох ключах: відкрите повідомлення шифрується на першому ключі, отриманий шифротекст – на другому ключі, і наприкінці, дані, отримані після другого кроку, шифруються на третьому ключі. Усі три ключі вибираються незалежно один від одного.

Суттєвим недоліком алгоритму DES вважають малий розмір ключа, оскільки при сучасному рівні розвитку комп'ютерних засобів 56 значущих бітів означають високу вірогідність злому шляхом прямого перебирання значень ключа. Недоліком алгоритму також вважають і мале число раундів.

2.7.3.3. Багаторівнева криптосистема ГОСТ 28147-89

Широке впровадження наприкінці 80-х р.р. в СРСР ЕОМ привело до необхідності створення і впровадження алгоритму шифрування даних, який подібний до алгоритму DES. Такий алгоритм було прийнято в СРСР в якості державного стандарту в 1989 р., і він носить назву ГОСТ 28147-89. В алгоритмі ГОСТ вдалося уникнути недоліків алгоритму DES. Довжина ключа складає 256 біт, а число раундів збільшено до 32.

Зашифрування – криптоперетворення $S=f(C,K1,K2)$.

Відкрите повідомлення розділяють на блоки довжиною 64 біта. Потім

кожен блок шифрують окремо, застосовуючи алгоритм шифрування, основними операціями якого є додавання по модулю 2 і 2^{32}, підстановка та циклічний зсув. Ключ $К_2$, довжиною 256 біт, складається з восьми 32-бітних блоків X_i (i=0,1,...,7).

Алгоритм шифрування має наступний вигляд:

1. 64-бітний блок відкритого повідомлення розділяють на ліву L і праву R половини по 32 біта.
2. На першому раунді в суматорі *CM1* значення правої половини R додається по модулю 2^{32} (mod 2^{32}) до значення блоку X_0. Значення R при цьому зберігається.
3. У блоці K виконується 8 підстановок по ключу K_1, кожна з яких реалізує просту заміну (багатоалфавітну підстановку) на базі шістнадцятирічного алфавіту A_{16}.
4. В регістрі *Rg* виконується циклічний зсув на 11 у бік старших розрядів.
5. В суматорі *CM2* відбувається покоординатне додавання по модулю 2 (mod2) значень *Rg* і L. Отриманий результат заноситься у праву половину R, а його попереднє значення переноситься у ліву частину L.
6. Наступні цикли виконуються аналогічно. Вони відрізняються тільки блоком X_i ключа K_2, який визначається по наступному порядку зчитування:

$$X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_0, X_1, X_2, X_3, X_4, X_5, X_6, X_7, X_7, X_6, X_5, X_4,$$
$$X_3, X_2, X_1, X_0, X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0, X_7, X_6, X_5, X_4, X_3, X_2, X_1, X_0.$$

Завдяки такому вибору блоків підсумковий оператор шифрування не є статичним.

7. Після 32-го раунду результат із суматора CM2 заноситься у ліву частину L, а у правій частині R зберігається старе значення. Отримані частини об’єднують у один блок довжиною 64 біта. Результатом є блок зашифрованого тексту.

Викладений алгоритм шифрування можливо представити у вигляді блок-схеми (Рис. 2.5).

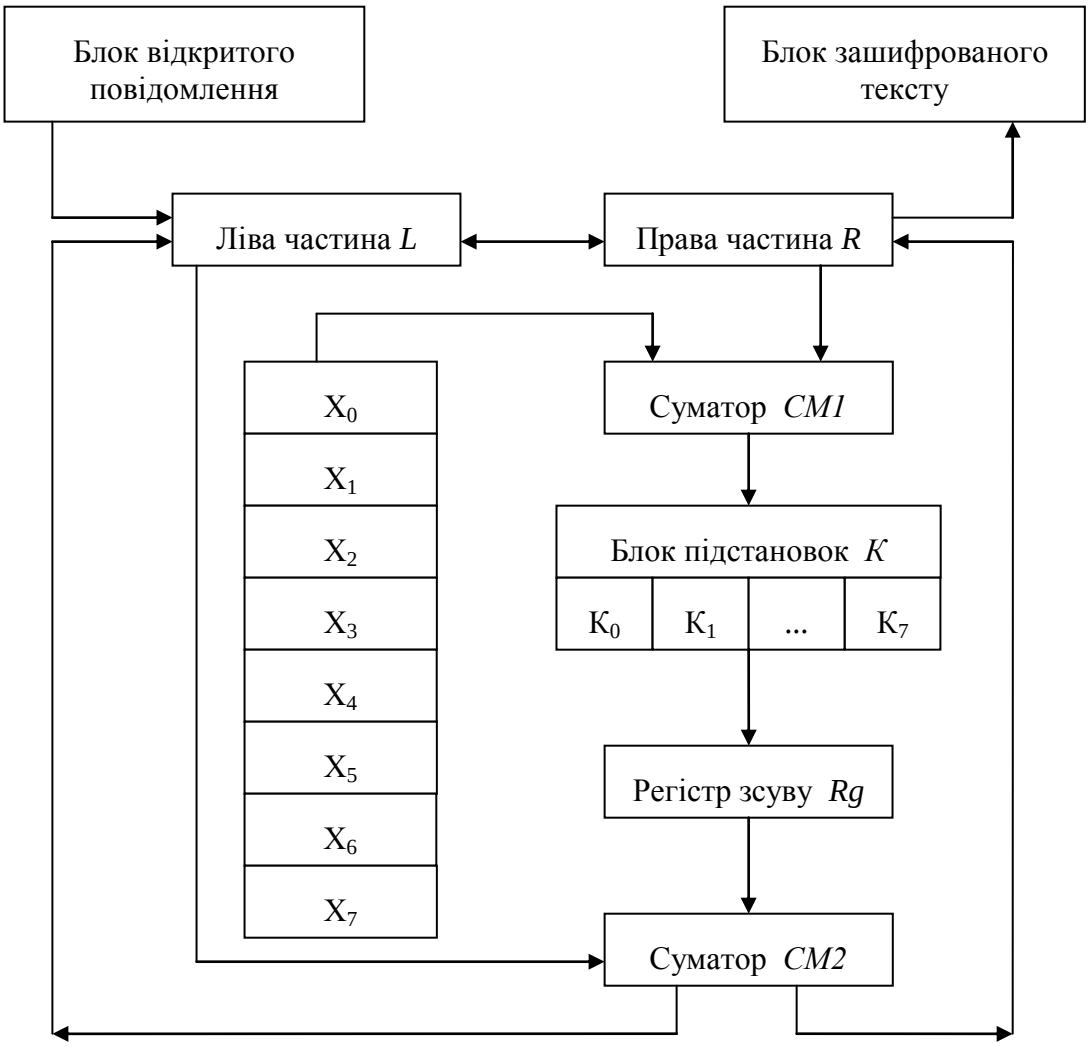


Рис. 2.5 Блок-схема алгоритму шифрування ГОСТ 28147-89 у режимі простої заміни

Дешифрування - криптоперетворення $C=f(S,K1,K2)$.

При виконанні зворотного криптоперетворення алгоритм шифрування, викладений вище, залишається незмінним. Змінюється тільки на зворотній порядок зчитування блоків X_i ключа K_2.

Криптостійкість шифру висока.

2.7.3.4. Новий стандарт симетричного шифрування

У 1997 році Американський інститут стандартизації NIST (National Institute of Standards & Technology) оголосив конкурс на новий стандарт симетричного криптоалгоритму, який було названо AES (Advanced Encryption Standard). До

його розробки було підключено найбільші центри криптології усього світу, оскільки переможник конкурсу повинен був стати світовим криптостандартом на найближчі 10-20 років.

До криптоалгоритмів-кандидатів на новий стандарт було пред'явлено наступні вимоги:

- алгоритм повинен бути симетричним;
- алгоритм повинен бути блочним шифром;
- алгоритм повинен мати довжину блоку 128 біт і підтримувати три довжини ключа: 128, 192 і 256 біт.

Додатково розробникам криптоалгоритмів рекомендувалося:

- використовувати операції, які легко реалізувати як апаратно (в мікрочіпах), так і програмно (на персональних комп'ютерах і серверах);
- орієнтуватися на 32-разрядні процесори;
- не ускладнювати без необхідності структуру шифру з тією метою, щоб усі зацікавлені сторони були у змозі самостійно провести незалежний криптоаналіз алгоритму і переконатися у тому, що в ньому не закладено будь-яких недокументованих функцій.

2 жовтня 2000 року NIST оголосив, що переможцем конкурсу став алгоритм Rijndael, який було розроблено двома фахівцями по криптології із Бельгії Дж. Дейменом (J. Daemen) і В. Райджменом (V. Rijmen).

Алгоритм Rijndael, який зараз називають алгоритмом AES – це блочний шифр з надійною математичною базою перетворень. В алгоритмі кожен блок даних, які шифруються, подаються у вигляді двомірного масиву байтів розміром 4x4, 4x6 чи 4x8 в залежності від встановленої довжини блоку. Далі на відповідних етапах проводяться перетворення або над незалежними стовпцями, або над незалежними рядками, або зовсім над окремими байтами у таблиці.

Криптостійкість шифру висока. Структура алгоритму дозволяє досягти високого ступеню оптимізації на персональних ЕОМ і серверах, і у той самий час не занадто сповільнює його виконання на електронних пристроях з

обмеженими ресурсами. З алгоритму AES знято усі патентні обмеження – його можливо використовувати в будь-якій криптопрограмі без відрахування коштів розробникам.

Усім системам симетричного шифрування властиві наступні недоліки:

- принциповою є вимога захищеності і надійності каналу передачі секретного ключа для обох учасників інформаційного обміну;
- пред'являються підвищені вимоги до служби генерації і розповсюдження ключів, оскільки для *n* абонентів при використанні схеми взаємодії «кожен з кожним» потрібно *n(n-1)/2* ключів, тобто залежність числа ключів від кількості абонентів є квадратичною; наприклад для *n=1000* абонентів потрібна кількість ключів складатиме *n(n-1)/2=499500*. Тому використання системи симетричного шифрування у такому вигляді в глобальній мережі Інтернет, де кількість користувачів вже перевищила за 100 мільйонів, практично неможливо без застосування додаткових методів і засобів.

2.8. Асиметричні і гібридні криптосистеми

Асиметричні алгоритми використовують для розв'язання двох основних задач: шифрування даних і цифрового підпису.

Принциповою відмінністю асиметричної криптосистеми від криптосистеми симетричного шифрування є те, що для шифрування даних і їх наступного розшифрування використовують різні ключі:

- відкритий ключ *K* використовують для зашифрування даних, і він визначається із секретного ключа *k*;
- секретний ключ *k* використовують для дешифрування даних, які зашифровані за допомогою парного йому відкритого ключа *K*.

Секретний і відкритий ключі генеруються попарно. Секретний (приватний) ключ повинен залишатися у його власника; його необхідно надійно захистити від несанкціонованого доступу (по аналогії з ключем шифрування в симетричних алгоритмах). Копія відкритого ключа (публічного) повинна

знаходитись у кожного абонента криптографічної мережі, з яким відбувається обмін даними власника секретного ключа.

Відзначимо характерні особливості асиметричних криптосистем:

1. Відкритий ключ *K* і шифротекст *S* можна відправляти по незахищеним каналам зв'язку, тобто зловмиснику вони можуть бути відомі.
2. Алгоритми зашифрування і дешифрування є відкритими.

У. Діффі і М. Хеллман сформулювали вимоги, виконання яких забезпечує високу криптостійкість асиметричної системи:

1. Визначення пари ключів (*K*, *k*) одержувачем на базі початкової умови повинно бути простим.
2. Відправник, знаючи відкритий ключ одержувача *K* і повідомлення *C*, може легко визначити криптограму *S*.
3. Одержувач, використовуючи свій секретний ключ *k* і криптограму *S*, може легко відновити початкове повідомлення *C*.
4. Зловмисник, знаючи відкритий ключ одержувача *K*, у разі спроби визначити секретний ключ одержувача *k* наштовхується на нездоланну обчислювальну проблему.
5. Зловмисник, знаючи відкритий ключ одержувача *K* і криптограму *S*, у разі спроби визначити початкове повідомлення *C* наштовхується на нездоланну обчислювальну проблему.

Однак далеко не всі стійкі алгоритми використовуються на практиці. Деякі з них потребують дуже великих ключів. Наприклад, розмір відкритого ключа криптосистеми HFE (Hidden Fields Equations) може досягати десятків мегабайт, що ускладнює розповсюдження таких ключів. При використанні деяких алгоритмів розмір шифротексту значно перевищує розмір відповідного йому відкритого тексту.

Концепція асиметричних криптографічних систем з відкритим ключем базується на використанні односпрямованих функцій. Односпрямованою називається функція *f(X)*, яка має наступні властивості:

- є алгоритм визначення значень функції $Y=f(X)$;
- немає алгоритму інвертування функції f, тобто не має розв'язання рівняння $f(X)=Y$ відносно X.

У якості прикладів односпрямованих функцій можливо вказати цілочисельне множення і цілочисельну модульну експоненту.

2.8.1. Асиметрична криптосистема RSA

Першим і самим розповсюдженим криптоалгоритмом асиметричного шифрування є алгоритм RSA, який отримав назву по першим буквам прізвищ його розробників: Рональда Райвеста (R. Rivest), Ади Шаміра (A. Shamir) і Леонардо Едельмана (L. Adleman). Алгоритм RSA, який було винайдено у 1977-1978 роках, став першим алгоритмом з відкритим ключем, який можна використовувати під час шифрування даних, так і для електронного цифрового підпису. У 1993 році метод RSA було прийнято в якості стандарту. Сьогодні RSA є складовою частиною багатьох стандартів, у тому числі International Standards Organization (ISO), Society for Worldwide Interbank Financial Telecommunications (SWIFT), ANSI X9.31, французького стандарту ETEBAC 5, австралійського AS2805.6.5.3 та ін..

Алгоритм RSA засновано на тому факті, що розкладання числа на прості множники досить складна операція. Згідно теоремі Ейлера, окремий випадок якої стверджує, що якщо число n представлено у вигляді добутку двох простих чисел p і g, то для будь-якого x має місце рівність $(x^{(p-1)(q-1)}) \bmod n = 1$.

Процес передачі зашифрованих даних у будь-якій асиметричній криптосистемі виконується наступним чином:

1. Підготовчий етап:
   - одержувач генерує пару ключів: секретний ключ k і відкритий ключ K;
   - відкритий ключ K відсилається відправнику повідомлення і іншим абонентам комп'ютерної мережі (або робиться доступним, наприклад, на спільному ресурсі).

2. Використання – обмін даними між абонентами:
   - відправник зашифровує відкрите повідомлення *C* за допомогою відкритого ключа одержувача *K* і відсилає шифротекст *S* по відкритому каналу зв'язку;
   - одержувач дешифрує повідомлення *S* за допомогою свого секретного ключа *k*. Ніхто інший (у тому числі відправник повідомлення) не може розшифрувати дане повідомлення, оскільки не має секретного ключа одержувача. Таким чином, захист даних в асиметричній криптосистемі засновано на секретності ключа *k* одержувача повідомлення.

Генерація сполучених ключів RSA.

Для алгоритму RSA етап створення ключів складається з наступних операцій:

1. Визначають два випадкових великих простих числа *p* і *q*. Для забезпечення максимальної безпеки *p* і *q* беруть рівної довжини і зберігають їх у таємниці.
2. Визначають модуль RSA: $n=p*q$.
3. Визначають функцію Ейлера $m=\varphi(n)=(p-1)(q-1)$. Така функція вказує на кількість позитивних чисел на інтервалі від 1 до *n*, які є взаємно простими за модулем *n*.
4. Вибирають число d, $0<d<m$, таке, що числа p, q, d є взаємно простими.
5. Визначають параметр *e* – мультипликативно зворотне числу *d*

$$e \equiv d^{-1} \bmod \varphi(n),$$

тобто $ed \equiv 1 \bmod m$.

Числа *e* і *d* називають секретним і відкритим показниками відповідно. Відкритим ключем є пара $\{d,n\}$, а секретним ключем – пара $\{e,n\}$.

Насправді, по домовленості, будь-який із згенерованих ключів можна оголосити секретним або відкритим. Однак, після оголошення секретного ключа необхідно забезпечити його конфіденційність, тобто зберігати у таємниці секретний показник і параметри *p*, *q*.

Вважається, що знаючи відкритий ключ $\{d,n\}$, складно визначити

секретний параметр *e*, однак, якщо вдасться розкласти *n* на множники *p, q*, то можливо дізнатися про значення *e*. Таким чином, загальна безпека системи RSA заснована на труднощі задачі розкладання. Математично така задача поки не розв'язна.

Зашифрування – криптоперетворення $S=f(C,K)$.

Відправник розділяє відкрите повідомлення на блоки, які дорівнюють $k=[\log_2(n)]$ біт, де дужки *[]* означають узяття цілої частини від дробового числа. Кожен такий блок може бути інтерпретовано як число з діапазону $(0;2^k-1)$.

Отримані блоки C_i зашифровують окремо один від одного, використовуючи формулу $S_i = C_i^d \pmod n$.

Блоки S_i і є зашифроване повідомлення. Їх можна передавати по відкритому каналу зв'язку. Це означає, що якщо зловмисник знає числа *d* і *n*, то по перехопленому значенню S_i прочитати початкове повідомлення він ніяк не зможе, окрім як повним перебором C_i.

Дешифрування - криптоперетворення $C=f(S,k)$.

Процедуру дешифрування може провести тільки власник секретного ключа *k*, використовуючи формулу $C_i=S_i^e \pmod n$.

Слід зауважити, що відкрите повідомлення *C*, шифротекст *S*, відкритий ключ *K* і секретний ключ *k* належать до множини цілих чисел $Z_n = \{0, 1, 2, \ldots, n-1\}$, де *n* – модуль RSA.

Криптостійкість алгоритму RSA всебічно досліджено і визнано стійким за умовою достатньої довжини ключів. Сьогодні довжина ключа 512 біт вважається недостатньою для забезпечення стійкості, а довжина 1024 біт – прийнятний варіант. Деякі автори стверджують, що з ростом потужності процесорів криптоалгоритм RSA втратить стійкість щодо атаки повним перебором. Однак не слід забувати, що збільшення потужності процесорів дозволить застосовувати більш довгі ключі, що підвисить стійкість алгоритму RSA.

Переваги. Використання асиметричних криптоалгоритмів дозволить здолати недоліки, які властиві системам симетричного шифрування:

- не потрібна секретна доставка ключів; асиметричне шифрування, на відміну від симетричного, має певну перевагу: динамічно передавати відкриті ключі, тоді як для симетричного шифрування до початку сеансу захищеного зв'язку необхідно обмінятися секретними ключами;
- зникає квадратична залежність числа ключів від числа користувачів; нескладно побачити, що в асиметричній криптосистемі RSA кількість ключів, які використовуються під час обміну повідомленнями, зв'язано з кількістю абонентів лінійною залежністю (у системі з N користувачами необхідно створити $2N$ ключів), а не квадратичною, як в симетричних системах.

Недоліки. У асиметричних криптосистем, зокрема і у криптосистемі RSA, є недоліки:

- на сьогодні не має математичного доведення необоротності функцій, які використовуються в асиметричних алгоритмах;
- порівняно з симетричним шифруванням асиметричне значно повільніше, оскільки під час криптоперетворень використовуються дуже ресурсоємні операції (наприклад, в алгоритмі RSA це піднесення одного великого числа у ступінь, яка є другим великим числом). Тому реалізувати апаратний шифратор з асиметричним алгоритмом значно складніше, ніж для симетричного алгоритму;
- необхідно захищати відкриті ключі від підміни.

Останнє розглянемо більш детально. Будемо вважати, що на комп'ютері абонента A зберігається відкритий ключ K_B абонента B. Зловмисник X має доступ до відкритих ключів, які зберігаються у абонента A. Він генерує свою пару ключів K_X і k_X і підмінює у абонента A відкритий ключ K_B абонента B на свій відкритий ключ K_X. Щоб відправити деякі дані абоненту B, абонент A зашифровує їх за допомогою ключа K_X, гадаючи, що це ключ K_B. Відповідно, абонент B не зможе прочитати отримане повідомлення, але його легко розшифрує і прочитає абонент X. Від підміни відкритих ключів може врятувати

тільки процедура сертифікації ключів.

2.8.2. Алгоритм шифрування Ель Гамаля

Даний алгоритм шифрування було розроблено американцем арабського походження Тахером Ель Гамалем у 1984 році. Порівняно з алгоритмом RSA, в запропонованому алгоритмі пропонується узагальнення на випадок довільної кінцевої групи, наприклад, групи точок еліптичної кривої. Однак у такому випадку відкритий текст повинен бути елементом групи. Тому в алгоритмі шифрування потрібно передбачати стадію перетворення довільного тексту в елемент групи (під час зашифрування) і зворотно (під час дешифрування).

В алгоритмі шифрування Ель Гамаля відкритим ключем шифрування K є сукупність:

- простого числа p;
- утворюючої g групи F_p, порядок якої має великий простий дільник;
- експоненти $y = g^x \pmod p$.

У якості секретного ключа розшифрування k беруть ціле число x, $0<x<p-1$.

Зашифрування – криптоперетворення $S=f(C,K)$.

Для зашифрування відкритого повідомлення C, яке є елементом групи F_p, відправник виконує наступні дії:

1. Вибирає випадкове число v, $0<v<p-1$.
2. Визначає $r \equiv y^v \pmod p$.

Шифротекстом є пара (S_1, S_2), де $S_1 = g^v \pmod p$, $S_2 = rC \pmod p$.

Дешифрування - криптоперетворення $C=f(S,k)$.

Одержувач повідомлення виконує алгоритм розшифрування, який передбачає наступні дії:

1. Відтворює r за допомогою свого секретного ключа x, тобто підраховує $r = (S_1)^x \pmod p$.
2. Відтворює відкритий текст C за формулою $C = r^{-1} S_2 \pmod p$.

Криптостійкість алгоритму висока, оскільки його засновано на

складності дискретного логарифмування. Розв'язання такої задачі виявляється значно більш складнішою проблемою, ніж розв'язання задачі, заснованої на складності розкладання складного числа на прості множники, на яких базується криптосистема RSA і подібні до неї системи.

2.8.3. Порівняльний аналіз асиметричних криптосистем

В алгоритмі Ель Гамаля шифротекст у два рази довше, ніж початковий відкритий текст. Окрім того, в такому алгоритмі під час шифрування визначаються дві експоненти, а в алгоритмі RSA – тільки одна, тобто шифрування алгоритмом RSA виконується у два рази швидше. У алгоритмі Ель Гамаля, на відмінність від алгоритму RSA, однакові відкриті повідомлення перетворюються у різні шифротексти, що ускладнює дослідження їх статистичних властивостей.

Безпека алгоритму RSA заснована на складності розкладання складного числа, а безпека протоколу Ель Гамаля – на складності дискретного логарифмування у кінцевому полі.

У разі застосування в алгоритмі RSA малих показників шифрування, тобто параметру d, можливе розкриття шифротексту без ключа за умови, що відправник передає різним абонентам-одержувачам близькі відкриті повідомлення. Крім того, використання загального секретного ключа двома користувачами комп'ютерної мережі приводить до взаємного розкриття їх секретних ключів.

Для уникнення розкриття шифротексту без ключа в алгоритмі Ель Гамаля необхідно тільки, щоб випадкові числа v були непередбачуваними і з великою вірогідністю неповторюваними.

2.9. Комбіновані криптосистеми

Головною перевагою асиметричних криптосистем з відкритим ключем є їх потенційно висока безпека: не має необхідності ні передавати, ні повідомляти

будь-кого про значення секретних ключів, ні впевнюватись у їх справжності. Однак швидкодія асиметричних криптосистем з відкритим ключем звичайно у сотні і більше разів менше за швидкодію симетричних криптосистем з секретним ключем.

У свою чергу, швидкодіючі симетричні криптосистеми мають суттєві недоліки: секретний ключ, який постійно поновлюється, повинен регулярно передаватися партнерам по інформаційному обміну, а під час таких передач виникає загроза розкриття секретного ключа.

Тому виник ефективний метод комбінованого застосування симетричного і асиметричного шифрування. Комбіноване використання симетричного і асиметричного шифрування дозволяє усунути основні недоліки, які властиві обом методам. Використання комбінованого (гібридного) методу шифрування дозволяє об'єднати переваги високої секретності, які надають асиметричні криптосистеми з відкритим ключем, з перевагами високої швидкості роботи, які властиві симетричним криптосистемам з секретним ключем.

У такому випадку симетричну криптосистему застосовують для шифрування початкового відкритого повідомлення, а асиметричну криптосистему з відкритим ключем – тільки для шифрування секретного ключа симетричної криптосистеми. У результаті асиметрична криптосистема з відкритим ключем не замінює, а тільки доповнює симетричну криптосистему з секретним ключем, що дозволяє підвищити в цілому захищеність даних, які передаються по відкритим каналам зв'язку. Такий метод шифрування іноді називають схемою електронного цифрового конверту.

Будемо вважати, що користувач *A* бажає застосувати комбінований метод шифрування з метою захищеної передачі повідомлення *C* користувачу *B*. Тоді послідовність дій користувачів A і B буде наступною:

- Дії користувача *A*:

1. Створює (наприклад, генерує випадковим чином) симетричний сеансовий секретний ключ K_S.

2. Шифрує відкрите повідомлення C за допомогою симетричного сеансового ключа K_S.
3. Шифрує секретний сеансовий ключ K_S за допомогою відкритого ключа K_B користувача B (одержувача повідомлення).
4. Здійснює передачу по відкритому каналу зв'язку на адресу користувача B зашифрованого повідомлення S разом із зашифрованим сеансовим ключем K_S.

- Дії користувача B, який отримав електронний цифровий конверт - зашифроване повідомлення S і зашифрований сеансовий ключ K_S:

1. Розшифровує за допомогою власного секретного ключа k_B сеансовий ключ K_S.
2. За допомогою отриманого сеансового ключа K_S розшифровує прийняте повідомлення S.

Слід відмітити, що електронний цифровий конверт може розкрити тільки законний одержувач – користувач B. Тільки користувач B, який володіє власним секретним ключем k_B, зможе вірно розшифрувати секретний сеансовий ключ K_S і потім за допомогою цього ключа розшифрувати і прочитати отримане повідомлення.

При використанні метода цифрового конверту недоліки симетричного і асиметричного криптоалгоритмів компенсуються наступним чином:

- проблема розповсюдження ключів симетричного криптоалгоритму усувається тим, що сеансовий ключ K_S, за допомогою якого шифрують повідомлення, передається по відкритим каналам зв'язку у зашифрованому вигляді; для шифрування ключа K_S використовують асиметричний криптоалгоритм;
- проблеми повільної швидкості асиметричного шифрування у даному випадку практично не виникає, оскільки асиметричним криптоалгоритмом шифрується тільки короткий ключ K_S, а усі інші дані шифруються швидким симетричним криптоалгоритмом.

В результаті отримують швидке шифрування в сполученні зі зручним розповсюдженням ключів.

При застосуванні комбінованого методу шифрування використовують криптографічні ключі як симетричних, так і асиметричних криптосистем. Очевидно, вибір довжин ключів для криптосистеми кожного типу слід здійснювати таким чином, щоб зловмиснику було однаково складно атакувати будь-який механізм захисту комбінованої криптосистеми. Насправді, якщо використовується короткий сеансовий ключ (наприклад, 40-бітовий DES), то не має значення наскільки довгі будуть асиметричні ключі. Хакери будуть атакувати не їх, а сеансові ключі.

2.9.1. Розповсюдження ключів

Будь-яка криптографічна система організована на використанні криптографічних ключів, які необхідно розповсюджувати. До розповсюдження ключів виставляються наступні вимоги:

- оперативність і точність розповсюдження;
- конфіденційність і цілісність розповсюджених ключів.

Для розповсюдження ключів між користувачами комп'ютерної мережі використовуються наступні основні засоби:

1. Використання одного чи декількох центрів розповсюдження ключів.
2. Прямий обмін ключами між користувачами мережі.

Проблемою першого підходу є те, що в центрі розповсюдження ключів відомо, кому і які ключі розповсюджено, що дозволяє читати усі повідомлення, які передаються по мережі. Можливі зловживання можуть суттєво порушити безпеку мережі. При використанні другого підходу проблема полягає у тому, щоб надійно впевнитися у справжності суб'єктів мережі.

Задача розповсюдження ключів зводиться до створення такого протоколу розповсюдження ключів, який забезпечує:

- взаємне підтвердження справжності учасників сеансу;

- підтвердження достовірності сеансу;
- використання мінімального числа повідомлень під час обміну ключами.

Характерним прикладом реалізації першого підходу є *система аутентифікації і розповсюдження ключів Kerberos*[6]. У такій системі будь-який клієнт *C*, який бажає отримати доступ до ресурсу мережі, направляє запит серверу аутентифікації *AS*. Сервер *AS* ідентифікує користувача за допомогою його імені і паролю та відсилає клієнту мандат на доступ до сервера служби виділення мандатів *TGS* (Ticket Granting Service). З метою використання конкретного цільового сервера інформаційних ресурсів *RS* клієнт *C* запитує у *TGS* мандат на звернення до цільового серверу *RS*. Якщо все вірно, то на *TGS* дозволяється використання необхідних ресурсів мережі і відсилає відповідний мандат клієнту *C*.

Запропонована модель взаємодії клієнта з серверами може функціонувати тільки за умови забезпечення конфіденційності і цілісності даних. Без строгого забезпечення інформаційної безпеки клієнт *C* не зможе відправляти серверам *AS*, *TGS*, *RS* свої запити і отримувати дозвіл на доступ до обслуговування у мережі.

Розглянемо тепер другий підхід – прямий обмін ключами між користувачами мережі. При використанні з метою захищеного інформаційного обміну криптосистеми з симетричним секретним ключем два користувача, які бажають обмінятись криптографічно захищеними даними, повинні мати загальний секретний ключ. Такі користувачі повинні обмінятись загальним ключем по каналу зв’язку безпечним чином. Якщо користувачі змінюють ключ часто, то передавання ключа перетворюється у серйозну проблему.

Для розв’язання такої проблеми можливо застосувати два основних способи:

1. Використання асиметричної криптосистеми з відкритим ключем з метою

[6] Kerberos - мережевий протокол аутентифікації, за допомогою якого можна передавати дані через незахищені мережі для безпечної ідентифікації.

захисту секретного ключа симетричної криптосистеми.

2. Використання системи відкритого розповсюдження ключів Діффі-Хеллмана.

Реалізація першого способу здійснюється в рамках комбінованої криптосистеми з симетричними і асиметричними ключами (див. питання 2.9).

Другий спосіб безпечного розповсюдження секретних ключів засновано на використанні *алгоритму відкритого розповсюдження ключів Діффі-Хеллмана*, за допомогою якого користувачам можна здійснювати обмін ключами по незахищеним каналам зв'язку. Його безпека зумовлена складністю визначення дискретних логарифмів в кінцевому полі, на відміну від легкості розв'язання прямої задачі дискретного піднесення у ступінь в тому самому кінцевому полі.

Суть метода Діффі-Хеллмана полягає у наступному. Користувачі A і B, які беруть участь в обміні даними генерують незалежно один від одного власні випадкові секретні ключі k_A і k_B (ключі k_A і k_B – випадкові великі цілі числа, які зберігаються користувачами A і B у таємниці).

Потім користувачі A і B визначають на підставі власних секретних ключів k_A і k_B свої відкриті ключі:

$$K_A = g^{k_A} (\mathrm{mod}\, n)$$
$$K_B = g^{k_B} (\mathrm{mod}\, n),$$

де n, g – великі цілі прості числа. Вони можуть не зберігатися у таємниці. Звичайно їх значення є загальнодоступними для всіх користувачів мережі або системи.

Наприкінці користувачі A і B обмінюються своїми відкритими ключами K_A і K_B по незахищеному каналу і використовують їх для визначення загального ключа сесії K (розподіленого секрету):

- користувач A:

$$K = (K_B)^{k_A} (\mathrm{mod}\, n) = (g^{k_B})^{k_A} (\mathrm{mod}\, n),$$

- користувач B:

$$K^* = (K_A)^{k_B} (\mathrm{mod}\, n) = (g^{k_A})^{k_B} (\mathrm{mod}\, n),$$

причому $K=K^*$, оскільки

$$(g^{k_B})^{k_A} (\mathrm{mod}\, n) = (g^{k_A})^{k_B} (\mathrm{mod}\, n).$$

Таким чином, у результаті запропонованих дій користувачі *A* і *B* отримують загальний ключ сесії *K*, який є функцією обох секретних ключів k_A і k_B.

Слід зауважити, що за допомогою схеми Діффі-Хеллмана можна шифрувати дані при кожному сеансі зв'язку на нових ключах. Це дозволяє не зберігати секретні дані на переносних пристроях для збереження даних, бо, як звісно, подібне зберігання секретних даних збільшує вірогідність попадання їх у руки конкурентів чи зловмисників.

2.10. Електронний цифровий підпис

Під час обміну електронними документами по комп'ютерній мережі суттєво знижуються витрати на обробку і зберігання документів, прискорюється їх пошук. Але з другого боку виникає проблема аутентифікації автора електронного документа і самого документа, тобто встановлення дійсності автора і відсутності змін в отриманому електронному документі.

Метою аутентифікації електронних документів є їх захист від можливих видів зловмисних дій, до яких відносяться:

- *активний перехват* – порушник, який підключився до мережі, перехоплює документи (файли) і змінює їх;
- *підміна* – абонент *B* змінює чи формує новий документ і заявляє, що отримав його від абонента *A*;
- *ренегатство* – абонент *A* заявляє, що не відсилав повідомлення абоненту *B*, хоч насправді здійснив відправлення;
- *маскарад* – абонент *X* відсилає документ абоненту *B* від імені абонента *A*;
- *повтор* – абонент *X* повторює раніше відправлений документ, який абонент *A* відсилав абоненту *B*.

Перелічені види зловмисних дій можуть принести суттєві збитки

банківським і комерційним структурам, державним підприємствам і організаціям, приватним особам, які застосовують у своїй діяльності комп'ютерні інформаційні технології.

Проблему перевірки цілісності повідомлення і справжності автора повідомлення дозволяє ефективно розв'язати методологія електронного цифрового підпису.

2.10.1. Призначення цифрового підпису

Електронний цифровий підпис (ЕЦП) використовують з метою аутентифікації текстів, які передаються по телекомунікаційним каналам зв'язку. Функціонально цифровий підпис аналогічний до звичайного рукописного підпису з усіма його основними перевагами:

- засвідчує, що підписаний текст надходить від особи, яка підпис поставила;
- не дає такій особі можливості відмовитись від обов'язків, які пов'язані з підписаним текстом;
- гарантує цілісність підписаного тексту.

Таким чином, електронний цифровий підпис надає юридичну значимість електронному документу.

Електронний цифровий підпис представляє собою відносно невелику кількість додаткових даних, які передається разом з підписаним текстом. ЕЦП засновано на оборотності асиметричних шифрів, а також на взаємозв'язку змісту повідомлення, самого підпису і пари ключів. Зміна хоча б одного з перелічених елементів зробить неможливим підтвердження справжності цифрового підпису.

ЕЦП реалізується за допомогою асиметричних алгоритмів шифрування і хеш-функцій.

2.10.2. Функція хешування

Функція хешування (*хеш-функція*) представляє собою перетворення, до

входу якого подається повідомлення змінної довжини *C*, а на виході отримуємо рядок фіксованої довжини *h(C)*. Тобто хеш-функція приймає в якості аргументу повідомлення (документ) *C* довільної довжини і повертає хеш-значення (хеш) *H=h(C)* фіксованої довжини.

Хеш-значення h(C) – це дайджест повідомлення C (криптографічна контрольна сума), тобто стиснене двійкове представлення основного повідомлення *C* довільної довжини. За допомогою функції хешування можна стиснути документ *C*, який має бути підписано, до 128 чи 256 біт, у той час коли повідомлення *C* може бути розміром в мегабайт і більше. Слід відзначити, що значення хеш-функції *h(C)* залежить від повідомлення (документу) *C* і за допомогою неї не можна відновити сам документ *C*.

Функція хешування повинна мати наступні властивості:

1. Хеш-функція може бути застосована до аргументу будь-якої довжини.
2. Результуюче значення хеш-функції має фіксований розмір.
3. Хеш-функцію *h(C)* просто визначити для будь-якого повідомлення *C*. Швидкість визначення хеш-функції повинна бути такою, щоб швидкість створення і перевірки ЕЦП у разі використання хеш-функції була значно більша у випадку використання самого повідомлення.
4. Хеш-функція повинна бути залежною від будь-яких змін у повідомлені *C*, таким як вставки, вилучення, перестановки і т.п..
5. Хеш-функція повинна мати властивість необоротності, тобто задача підбору документа C^*, який мав би потрібне значення хеш-функції, повинна бути нерозв'язною математично задачею.
6. Вірогідність того, що значення хеш-функцій двох різних документів (незалежно від їх довжин) співпадуть, повинна бути дуже мала; тобто для будь-якого фіксованого *C* неможливо знайти $C^* \neq C$, таке *що* $h(C^*) = h(C)$.

Теоретично можливо, що два різних повідомлення може бути стиснено у одну і ту саму згортку (такий випадок називають колізія чи зіткнення). Тому для забезпечення стійкості функції хешування необхідно передбачити спосіб уникати зіткнень. Зовсім зіткнень уникнути неможливо, оскільки в загальному

випадку кількість можливих повідомлень перевищує кількість можливих вихідних значень функції хешування. Однак вірогідність зіткнень повинна бути низькою.

Таким чином, функція хешування може використовуватись з метою виявлення змін повідомлення, тобто вона може використовуватися для створення *криптографічної контрольної суми*, яку ще називають *кодом виявлення змін* чи *кодом аутентифікації повідомлень*. В такому випадку хеш-функція використовується для контролю цілісності повідомлення, під час створення і перевірки електронного цифрового підпису.

Найбільш популярними хеш-функціями є MD2, MD4, MD5, SHA.

MD2, MD4, MD5 – алгоритми визначення криптографічної контрольної суми повідомлень, які розроблено Р. Райвестом. За допомогою цих алгоритмів створюється 128-бітовий хеш-код. Алгоритм MD2 – самий повільний, MD4 – самий швидкий. Алгоритм MD5 є модифікацією алгоритму MD4, в якому «пожертвували» швидкістю заради збільшення безпеки. SHA (Secure Hash Algorithm) – алгоритм визначення криптографічної контрольної суми повідомлень, за допомогою якого створюється 160-бітовий хеш-код. Алгоритм SHA дещо надійніший за MD4 і MD5.

2.10.3. Основні процедури цифрового підпису

Технологія використання системи електронного цифрового підпису припускає наявність мережі абонентів, які посилають один одному підписані електронні документи. Для кожного абонента генерується пара ключів: секретний і відкритий. Секретний ключ зберігається абонентом у таємниці і використовується ним для формування ЕЦП. Відкритий ключ відомо усім іншим користувачам і призначено для перевірки ЕЦП одержувачем підписаного електронного документу.

Система ЕЦП включає дві основні процедури:

- процедуру формування цифрового підпису;

- процедуру перевірки цифрового підпису.

У процедурі формування підпису використовують секретний ключ відправника повідомлення, у процедурі перевірки підпису – відкритий ключ відправника.

Процедура формування цифрового підпису

На підготовчому етапі такої процедури абонент *A* – відправник повідомлення – генерує пару ключів: секретний ключ k_A і відкритий ключ K_A. Відкритий ключ K_A розсилається іншим абонентам комп'ютерної мережі з метою використання під час перевірки підпису.

Для формування цифрового підпису відправник *A* визначає значення хеш-функції *h(C)* для повідомлення *C*, яке буде ним підписано, тобто визначає криптографічну контрольну суму *c* повідомлення *C*. Далі відправник *A* шифрує криптографічну контрольну суму *c* власним секретним ключем k_A. Отримана пара чисел представляє собою цифровий підпис для даного повідомлення *C*. Повідомлення *C* разом з цифровим підписом відправляється на адресу одержувача.

Процедура перевірки цифрового підпису

Будь-який абонент комп'ютерної мережі, в тому числі і абонент *B*, може перевірити цифровий підпис отриманого повідомлення за допомогою відкритого ключа K_A відправника повідомлення.

Під час перевірки ЕЦП абонент *B* – одержувач повідомлення – розшифровує прийняту криптографічну контрольну суму *c* відкритим ключем K_A відправника *A*. Крім того, одержувач сам визначає за допомогою хеш-функції *h(C)* криптографічну контрольну суму c^* прийнятого повідомлення і порівнює його з розшифрованим. Якщо дві криптографічні контрольні суми *c* і c^* співпадають, то цифровий підпис є справжнім. В протилежному випадку або підпис підроблено, або змінено вміст повідомлення.

Принциповим моментом у системі ЕЦП є неможливість підробки ЕЦП користувача без знання його секретного ключа. Тому необхідно захистити

секретний ключ підписування від несанкціонованого доступу. Секретний ключ ЕЦП, по аналогії з ключем симетричного шифрування, рекомендується зберігати на персональному носії в захищеному вигляді.

2.10.4. Основні властивості ЕЦП

Електронний цифровий підпис представляє собою унікальне число, яке залежить від документа, який підписують, і секретного ключа абонента. У якості документа, який необхідно підписати, може бути використано будь-який файл. ЕЦП звичайно містить додаткові дані, які однозначно ідентифікують автора підписаного документу. Такі дані додається до документу до визначення ЕЦП, що забезпечує і їх цілісність. Кожний підпис містить наступні дані:

- дату підпису;
- термін закінчення дії ключа даного підпису;
- дані про особу, яка підписала файл (П.І.Б., посада, скорочене найменування установи);
- ідентифікатор автора підпису (ім'я відкритого ключа);
- власне цифровий підпис.

По аналогії з асиметричним шифруванням необхідно забезпечити неможливість підміни відкритого ключа, який використовується для перевірки ЕЦП. Якщо припустити, що зловмисник X має доступ до відкритих ключів, які зберігає на власному комп'ютері абонент B, у тому числі і до відкритого ключа K_A абонента A, то він може виконати наступні дії:

- прочитати з файлу, в якому міститься відкритий ключ K_A, ідентифікаційні дані про абонента A;
- згенерувати власну пару ключів k_X і K_X, записавши у них ідентифікаційні дані абонента A;
- підмінити відкритий ключ K_A, який зберігається у абонента B, своїм відкритим ключем K_X, який містить ідентифікаційні дані абонента A.

Після виконання таких дій зловмисник X може надсилати документи

абоненту B, які будуть підписано його секретним ключем k_X. Під час перевірки підпису таких документів абонент B буде вважати, що документи підписано абонентом A і їх ЕЦП справжній, тобто вони не були модифіковані будь-ким. До з'ясування відносин безпосередньо з абонентом A у абонента B може не з'явитися сумнівів у справжності отриманих документів. Тому відкриті ключі ЕЦП необхідно захищати від підміни за допомогою відповідних цифрових сертифікатів, які видаються по запитам користувачів спеціальними уповноваженими організаціями – центрами сертифікації (Certificate Authority).

Сертифікат представляє собою електронну форму, у якій містяться наступна дані:

- відкритий ключ власника даного сертифікату;
- відомості про власника сертифікату, наприклад ім'я, адреса електронної пошти, найменування організації, де він працює, і т.п.;
- найменування організації, яка видала даний сертифікат;
- електронний підпис організації сертифікації.

Слід відмітити тісний зв'язок відкритих ключів з сертифікатами. Сертифікат є не тільки посвідченням особи, але і посвідченням приналежності відкритого ключа. За допомогою цифрового сертифікату встановлюється і гарантується відповідність між відкритим ключем і його власником, що запобігає підміні відкритого ключа.

2.10.5. Схема цифрового підпису RSA

Першою і найбільш відомою у всьому світі системою ЕЦП стала система RSA, математичну схему якої було розроблено у 1977 році в Массачуссетському технологічному інституті.

Генерація ключів

Алгоритм генерації ключів було викладено під час опису криптосистеми RSA (див. питання 2.8.1).

Генерація підпису

Для створення підпису повідомлення C абонент A виконує наступні дії:

1. Визначає криптографічну контрольну суму $c=h(C)$ повідомлення C за допомогою хеш-функції (на практиці для підпису RSA використовують функції MD4 і MD5).
2. Шифрує отриману криптографічну контрольну суму c за допомогою власного секретного ключа $\{e,n\}$, тобто визначає значення $s \equiv c^e \pmod n$, яке і є цифровим електронним підписом.

Перевірка підпису

Для перевірки підпису абонента A абонент B виконує наступні дії:

1. Розшифровує підпис s за допомогою відкритого ключа $\{d,n\}$ абонента A, тобто визначає значення $c^* \equiv s^d \pmod n$ і, таким чином, відтворює передбачувану криптографічну контрольну суму c^* повідомлення C.
2. Визначає криптографічну контрольну суму $c=h(C)$ повідомлення C за допомогою тієї ж самої хеш-функції, яку використовував абонент A.
3. Порівнює отримані значення c і c^*. Якщо вони співпадають, то підпис вірний, абонент A дійсно є тим, за кого себе видає, і повідомлення не було змінено під час передавання.

Алгоритм цифрового підпису RSA має суттєвий недолік – за допомогою нього зловмиснику можна без знання секретного ключа формувати підписи під тими документами, в яких результат хешування можливо підрахувати як добуток результатів хешування вже підписаних документів.

2.10.6. Схема підпису Ель Гамаля (EGSA)

Порівняно з RSA алгоритм цифрового підпису Ель Гамаля є більш надійним і зручним для реалізації на персональних комп'ютерах. В розробленому Ель Гамалем (1984 р.) алгоритмі, який називають EGSA (назва походить від слів El Gamai Signature Algorithm (алгоритм цифрового підпису Ель Гамаля)), вдалося уникнути слабкості алгоритму цифрового підпису RSA, пов'язаної з можливістю підробки цифрового підпису під деякими

повідомленнями без знання секретного ключа. У 1991 році Національний інститут стандартів і технологій США (National Institute of Standards and Technology – NIST) обґрунтував вибір алгоритму цифрового підпису Ель Гамаля в якості основи для національного стандарту цифрового підпису.

Генерація ключів

Алгоритм генерації ключів було викладено під час опису криптосистеми Ель Гамаля (див. питання 2.8.2). Різниця тільки у тому, що випадкове ціле число x, яке є секретним ключем для створення підпису, вибирається на інтервалі [0,q].

Генерація підпису

Для створення підпису повідомлення C, $0<C<q$, абонент A виконує наступні дії:

1. Генерує випадкове число v, $0<v<q$.
2. Визначає $r \equiv g^v \pmod p$.
3. Визначає число s при розв'язанні рівняння $C \equiv xr + vs \pmod q$:

$$s \equiv (C - xr)\, v^{-1} \pmod q.$$

Оскільки числа v і q взаємно прості, то v зворотно по модулю q, тобто число s завжди є і воно єдине.

Підписом для повідомлення C є пара (r,s).

Перевірка підпису

Для перевірки підпису (r,s) на документі C абонента A абонент B виконує перевірку порівняння $s \equiv (C - xr)\, v^{-1} \pmod q$ для експонент:

$$g^C \equiv y^r\, r^s \pmod p.$$

При генерації підпису у протоколі Ель Гамаля можна використовувати попередні розрахунки. Так, числа v і r ніяк не пов'язані з повідомленням і їх можна визначити заздалегідь, що дозволяє прискорити процедуру генерації підпису. Перевірка підпису потребує значно більше часу, оскільки потребує визначення трьох експонент.

Порівняно з RSA перевірка підпису в протоколі Ель Гамаля виконується повільніше і розмір підпису більший. Однак, якщо використовувати різниці

складових підпису по модулю порядку групи, то підпис виявляється коротшим за RSA.

2.10.7. Чинний стандарт цифрового підпису

Алгоритм цифрового підпису DSA (Digital Signature Algorithm) було запропоновано в 1991 році Національним інститутом стандартів і технологій США. У 1993 році він став стандартом США. Алгоритм DSA є подальшим розвитком алгоритму цифрового підпису Ель Гамаля.

Генерація ключів

Відправник і одержувач електронного документу використовують великі цілі числа: g, p – прості числа, довжиною L біт кожне ($512 \le L \le 1024$); q – просте число довжиною 160 біт (дільник числа $(p-1)$). Числа g, p, q є відкритими і можуть бути загальнодоступними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число x, $1<x<q$. Число x є *секретним ключем* відправника, який використовується для створення ЕЦП.

Потім відправник визначає значення $y = g^x \pmod p$.

Число y є *відкритим ключем*, який використовують для перевірки підпису відправника. Число y передається усім одержувачам документів.

Генерація підпису

Алгоритм передбачається використання однобічної функції хешування h(.). В стандарті використовують алгоритм безпечного хешування SHA-1.

Щоб підписати повідомлення C абонент A виконує наступні дії:

1. Генерує випадкове число v, $1<v<q-1$.
2. Визначає $r \equiv g^v \ ((\text{mod } p) \text{ mod } q)$.
3. Визначає $v^{-1} \pmod q$.
4. Визначає число $s \equiv \{h(C)+xr)v^{-1} (\text{mod } q)$, де h – алгоритм хешування SHA-1.
5. У випадку якщо $s=0$, необхідно перейти до виконання п.1. (Якщо $s=0$, то $s^{-1} \pmod q$ не існує, а воно потрібне при виконанні п.2 процедури перевірки підпису.)

Підписом для повідомлення C є пара чисел (r,s).

Перевірка підпису

Для перевірки підпису (r,s) на документі C абонента A абонент B виконує наступні дії:

1. Отримує справжню копію відкритого ключа y абонента A.
2. Визначає $w = s^{-1} \pmod q$ і хеш-значення $h(C)$.
3. Визначає значення $u_1 = \mathrm{h}(C)w \pmod q$, $u_2 = rw \pmod q$.
4. Використовуючи відкритий ключ y, визначає значення
$$t = g^{u1} y^{u2} ((\mathrm{mod}\ p) \bmod q).$$
5. Вважає підпис (r,s) під документом C справжнім, якщо $t=r$.

Оскільки r, s є цілими числами, окрім того кожне з них менше за q, то підписи DSA мають довжину 320 біт. Безпека алгоритму цифрового підпису DSA базується на труднощі задачі дискретного логарифмування.

Розділ 3. Стеганографічні методи захисту даних

3.1. Основні поняття стеганографії

Стеганографія – це метод організації зв'язку, за яким ховається сама наявність зв'язку. На відміну від криптографії, де точно можна визначити чи є передане повідомлення зашифрованим текстом, за допомогою методів стеганографії секретні повідомлення вмонтовують в непоказні послання так, щоб неможливо було запідозрити існування вмонтованого таємного послання. Приховування повідомлення за методами стеганографії значно знижує ймовірність виявлення самого факту передавання повідомлення.

Стеганографія займає свою нішу в забезпеченні безпеки: вона не замінює, а доповнює криптографію. Історія стеганографії – це історія розвитку людства.

Місцем зародження стеганографії багато-хто називає Єгипет, хоч першими «стеганографічними повідомленнями» можна назвати і наскальні малюнки древніх людей.

Перше згадування про стеганографічні методи в літературі приписується Геродоту (5 ст. до н.е.), який описав випадок передавання повідомлення Демартом, який зіскрібав віск з дощечок, писав лист на дереві, а потім знову покривав дощечки воском.

Інший епізод, що відносять до тих же часів – передавання послання за допомогою раба. Для передавання таємного повідомлення тиран Гистій, захоплений перськими військами в Сузах, знаходячись у полоні в царя Дарія, захотів послати повідомлення своєму родичеві в Мілеєт, що в Анатолії. Для цього поголили голову раба, нанесли на шкіру татуювання, і коли волосся відросло, відправили його з посланням.

У Китаї листи писали на смужках шовку. Для приховування повідомлень смужки з текстом листа заверталися в кульки, покривалися воском і потім ковтались посильними.

У середньовіччя внаслідок посилення стеження за людьми криптографія і стеганографія почали інтенсивно розвивались. Саме в середні віки вперше було

застосовано спільне використання шифрів і стеганографічних методів.

У XV столітті чернець Тритеміус, що займався криптографією і стеганографією, описав багато різних методів прихованого передавання повідомлень. Пізніше, у 1499 році, ці записи були об'єднані в книгу «Steganographia», що у даний час, знаючи латинь, можна прочитати в Інтернеті.

XVII-XVIII століття відомі як ера «чорних кабінетів» – спеціальних державних органів з перехоплення, перлюстрації і дешифрування листування. У штат «чорних кабінетів» крім криптографів і дешифрувальників, входили й інші фахівці, у тому числі і хіміки. Наявність фахівців-хіміків була необхідна через активне використання так званого невидимого чорнила. Прикладом може служити цікавий історичний епізод: повсталими дворянами в Бордо був заарештований францисканський чернець Берто, який був агентом кардинала Мазаріні. Повсталі дозволили Берто написати листа знайомому священикові в місто Блей. Однак наприкінці цього листа релігійного змісту, чернець зробив приписку, на яку ніхто не звернув увагу: «Посилаю Вам очну мазь; натріть нею очі і Ви будете краще бачити». Так він зумів переслати не лише сховане повідомлення, але і вказав спосіб його виявлення. У результаті чернець Берто був врятований.

Стеганографічні методи активно використовувалися і в роки громадянської війни між жителями півдня та півночі США. Так, у 1779 році два агенти – жителі півночі Семюель Вудхулл і Роберт Тоунсенд передавали повідомлення Джорджу Вашингтону, використовуючи спеціальне чорнило.

Різні симпатичні чорнила (безколірні, такі, що стають видимими після спеціальної обробки написаного) використовували і російські революціонери на початку XX століття, що знайшло відображення в радянській літературі: В.І. Куканов у своїй повісті «У джерел прийдешнього» описує застосування молока в якості чорнила для написання таємних повідомлень. Утім, царська охорона теж знала про цей метод (в архіві зберігається документ, в якому описано спосіб використання симпатичного чорнила і наведено текст перехопленого таємного повідомлення революціонерів).

Особливе місце в історії стеганографії займають фотографічні мікрокрапки. Однак мікрокрапки з'явилися набагато раніше, відразу ж після винаходу Дагером фотографічного процесу, і вперше у військовій справі були використані в часи франко-прусської війни (у 1870 році). Існують також акровірші та інші мовні ігри.

Стеганографічна система або *стегосистема* – це сукупність засобів і методів, що використовуються для формування прихованого каналу передавання даних.

При побудові стегосистеми повинні враховуватися наступні положення:

- супротивник має повне уявлення про стеганографічну систему і деталі її реалізації. Єдині дані, що залишаються невідомі потенційному супротивнику, є ключ, за допомогою якого лише його власник може встановити факт наявності і зміст схованого повідомлення;
- якщо хтось якимось чином довідається про факт існування схованого повідомлення, то це не повинно дозволити йому отримати доступ до подібних повідомлень доти, поки ключ зберігається в таємниці;
- потенційний супротивник повинен бути позбавленим яких-небудь технічних і інших переваг у розпізнаванні чи розкриванні змісту таємних повідомлень.

За аналогією з криптографією, за типом стегоключа стегосистеми можна поділити на два типи: з секретним ключем і з відкритим ключем.

У стегосистемі із секретним ключем використовується один ключ, що повинен бути визначений або до початку обміну секретними повідомленнями, або переданий через захищений канал.

У стегосистемі з відкритим ключем для вмонтовування і добування повідомлення використовуються різні ключі, які відрізняються таким чином, що за допомогою обчислень неможливо вивести один ключ з іншого. Тому один ключ (відкритий) може передаватися вільно через незахищений канал зв'язку.

Під *комп'ютерною стеганографією* розуміється приховування одних цифрових повідомлень в інших. Причому приховування повинно реалізовуватися так, щоб, по-перше, не були втрачені властивості і цінність приховуваних повідомлень, а по-друге, неминуча модифікація інформаційного носія не тільки не знищувала значеннєві функції, але і на певному рівні абстракції навіть не змінювала їх.

Комп'ютерна стеганографія широко використовується при обміні секретними повідомленнями, написанні вірусних програм, у захисті авторських прав, для приховування даних від копіювання і т.д.

У наш час можна виділити три тісно пов'язані між собою напрямки застосування стеганографії, що мають також одні корені: приховування даних (повідомлень), нанесення на твори цифрових водяних знаків та введення заголовків.

Приховування вмонтованих даних, що в більшості випадків мають великий обсяг, висуває серйозні вимоги до контейнера: розмір контейнера повинен в кілька разів (як правило, в 8-10) перевищувати розмір даних, що вмонтовуються.

Файл, який необхідно приховати, називають інформаційним, а файл, що використовується для приховування даних, називають файлом-контейнером. Контейнером може бути текстовий, графічний чи музичний файл. Можна використовувати пробіли в документах (інколи також незначно спотворюється накреслення літер), аудіофайли (особливо формату MP3, із малопомітним спотворенням у звучанні або мові), картинки і фотографії (в яких змінюються значення бітів із надлишковими даними про колір пікселів, що викликає малопомітні зміни в кольорах). Таємні повідомлення можуть приховуватись майже скрізь, але найрозповсюдженішими носіями стали саме зображення. Наприклад, у звичайній картинці 640x480 формату BMP можна приховати близько 300 Кбайт текстових даних, а розміром 1024x768– до 2 Мбайт.

Цифрові водяні знаки використовуються для захисту авторських чи майнових прав на цифрові зображення, фотографії або інші оцифровані твори

мистецтва. Основними вимогами, що висуваються до таких вмонтованих даних, є надійність і стійкість до перекручувань.

Цифрові водяні знаки займають невеликий обсяг у файлі, однак для їх вмонтовування використовуються більш складні методи, ніж для вмонтовування звичайних повідомлень чи заголовків.

Заголовки використовуються, в основному, для маркирування зображень у великих електронних сховищах (бібліотеках) цифрових зображень, аудіо- і відеофайлів.

Стеганографічні методи використовуються не лише для впровадження ідентифікуючого заголовку, але й інших індивідуальних ознак файлу.

Впроваджувані заголовки займають невеликий обсяг, а вимоги, що висуваються до них, мінімальні: за їх допомогою повинні вноситись незначні спотворення. і вони повинні бути стійкими до основних геометричних перетворень.

Існує досить велика кількість методів приховування повідомлень (та їх варіантів).

Найбільш розповсюдженим, але найменш стійким є метод заміни найменших значущих біт або LSB-метод (LSB - Least Significant Bit). Він полягає у використанні похибки дискретизації, що завжди існує в оцифрованих зображеннях або аудіо- і відеофайлах. Дана похибка дорівнює найменшому значущому розряду числа, що визначає величину колірної складової елемента зображення (пікселя). Тому модифікація молодших біт у більшості випадків не викликає значної трансформації зображення і не виявляється візуально.

Іншим популярним методом вмонтовування повідомлень є використання особливостей форматів даних, де застосовується стискування із втратою даних (наприклад JPEG). Цей метод (на відміну від LSB) більш стійкий до геометричних перетворень і виявлення каналу передавання, тому що є можливість у широкому діапазоні варіювати якість стиснутого зображення, що унеможливлює визначення походження спотворення.

Справді, в цифрових фото-, аудіо- і відеофайлах зазвичай міститься багато

надлишкових даних. Так, наприклад, в графічних кольорових RGB-файлах (RGB - скорочено від англ. Red, Green, Blue - червоний, зелений, синій) кожна точка малюнка кодується трьома байтами, кожний з яких відповідає певній адитивній складовій кольору (червону, зелену і синю). Модифікація кожного з трьох молодших біт приводить до зміни менш 1% інтенсивності даної точки, що майже непомітно людському оку. Це дозволяє ховати в картинці обсягом 800 Кб біля 100 Кб даних.

Значна кількість мультимедійних форматів містять поля розширення, які можуть заповнюватися користувацькими даними, а можуть бути заповнені нулями – в останньому випадку їх також можна використовувати для зберігання і навіть передавання даних. Однак за цим способом не завжди забезпечується необхідний рівень таємності. До того ж користуючись ним, не вдається ховати значні обсяги даних.

Дані можна ховати й у звукових файлах: лише одна секунда оцифрованого аудіо з частотою дискретизації 44,1 кГц і розрядністю відліку стереорежимі дозволяє без сприйманої людським вухом втрати якості сховати до 12 Кб даних. Це реалізується за рахунок заміни найменш значимих молодших розрядів звукового відліку на приховувані дані. При цьому відбувається незначне (менш 1%) перекручування звуку (насамперед басового діапазону), що середньостатистична людина без спеціального обладнання не зафіксує.

Також є можливість «ховати» дані у системних структурах.

Специфіка файлових систем така, що при зберіганні файл завжди займає ціле число блоків – це зроблено для зручності адресації. Через це зберігання маленьких файлів реалізується вкрай нераціонально – у системі FAT32 файл розміром 1 байт займає на диску 4 Кб. При цьому очевидно, що всі 4 Кб, за винятком 1 байта, ніяк не використовуються. Це саме ті ділянки, які використовують для приховування даних. Такий метод широко використовується для довгострокового прихованого зберігання невеликих обсягів даних – однак він вкрай небезпечний, його застосування досить легко виявити. Також існує можливість запису даних просто на невикористовувані

ділянки накопичувачів (наприклад, на нульову доріжку).

Можна ховати дані й у виконуваних файлах. Справа в тому, що в операційній системі Windows виконувані файли представлені у форматі PE (Portable Executable), що передбачає наявність в exe-модулі не лише виконуваного коду, але й багатьох інших даних. Це піктограми, різні системні дані, дані про імпортовані й експортовані функції і т д. Причому для зберігання всіх цих об'єктів файл PE ділиться на ряд секцій, що мають фіксований розмір, і кожен об'єкт починається з нової секції. При цьому утворюються порожні місця, в яких знов-таки можна розміщувати власні дані. На використання цієї ідеї, зокрема, спирався розробник вірусу WinCIH (приховування вірусу WinCIH реалізовувалось у виконуваних файлах). Автор іншого, менш відомого вірусу W32/Perrun приховував його в графічних файлах формату JPG (тіло вірусу дописувалось в кінець файлу картинки).

Для вмонтовування цифрових водяних знаків, що можуть сприяти захисту авторських прав, використовуються більш складні методи. У сучасних системах формування цифрових водяних знаків використовується принцип вмонтовування мітки, що є вузькосмужним сигналом у широкому діапазоні частот маркованого зображення. Зазначений метод реалізується за допомогою двох різних алгоритмів та їх можливих модифікацій. У першому випадку дані приховуються шляхом фазової модуляції інформаційного сигналу із псевдовипадковою послідовністю чисел. У другому – наявний діапазон частот поділяється на кілька каналів і передавання відбувається між цими каналами. Щодо вихідного зображення, мітка є деяким додатковим шумом, але оскільки шум у сигналі наявний завжди, його незначне зростання за рахунок вмонтовування мітки не дає помітних на око спотворень. Крім того, мітка розсіюється на усьому вихідному зображенні, в результаті чого стає більш стійкою до вирізання.

В даний час комп'ютерна стеганографія продовжує розвиватись: формується теоретична база, ведеться розробка нових, більш стійких методів вмонтовування повідомлень. Серед основних причин зростання зацікавленості

стеганографією можна виділити прийняті в ряді країн обмеження на використання сильної криптографії, а також проблему захисту авторських прав на художні твори в цифрових глобальних мережах.

3.2. Короткий огляд стеганографічних програмних засобів

На сьогодні існує значна кількість стеганографічних програм, що використовуються для шифрування даних в зображенні та звуці. З їх допомогою можна приховувати текстові повідомлення у файлах .BMP, .GIF, .WAV і т.ін. Вони призначені для тих випадків, коли користувач не хоче, щоб у кого-небудь створилося враження, що він щось приховує від сторонніх очей.

Одним з класичних (і простих) представників такого класу програм є засіб *Stegozaurus*, за допомогою якого можна приховувати будь-які дані в BMP-картинці. Якість зображення при цьому не змінюється, а картинка стає носієм даних.

Інша програма шифрування *NDEC* – також проста, але в той же час надзвичайно надійна. В ній застосовані оригінальні алгоритми багатоступінчастого поліморфного кодування з використанням двох ключів. Це означає що, наприклад, той самий файл, зашифрований тим самим ключем, щоразу буде мати новий, відмінний в усіх байтах вигляд. Користувачам NDEC при шифруванні/розшифруванні даних необхідно вводити два паролі. Довжина кожного паролю не може бути більшою 256 символів.

За допомогою утиліти *BDV DataHider* можна підтримувати роботу з графічними форматами jpg, .jpeg, .bmp, ico, .emf, .wmf, однак кодований файл зберігається лише у форматі .bmp. У програмі можна змінювати розміри зображення, глибину шифрування (від 1 до 8 біт), а також встановлювати пароль на дешифрування зображення, що також відбувається за допомогою BDV DataHider.

А ось за допомогою програми *Masker* (www.softpuls.com) можна використовувати в якості контейнерів графічні файли (BMP, JPG, TIF, PCX,

PNG), виконувані файли (EXE, DLL, OCX), звукові файли (WAV, MID, SND, MP3), відеофайли (AVI, MOV, MPG, ASF).

Слід відмітити, що приховувати дані у JPG-файлах непросто, оскільки за форматом JPEG використовується стискання із втратами. Найпершою (і найпростішою) програмою такого типу була *JPSTEG*, яка функціонувала в консольному режимі. Різні версії програми *JPHIDE* (або *JPHS*) призначені для функціонування з Windows-орієнтованим інтерфейсом та з командного рядка.

WbStego (wbstego.wbailer.com) – за допомогою платних і безкоштовних версій цієї програми можна приховувати дані в файлах формату BMP, TXT, HTM і PDF.

Hermetic Stego (www.hermetic.ch/hst/hst.htm) – комерційна програма, використовуючи яку можна приховувати окремі файли в рисунках формату GIF, JPG і PNG.

Encrypt Text in Picture (www.acezip.net) – безкоштовна програма, за якою можна реалізовувати приховування даних в рисунках формату BMP, JPG і JPEG.

Програмами, в яких одночасно застосовуються прийоми криптографії і стеганографії, також дуже зручно користуватись. Найдавніший (розробка 1996 року) приклад подібної програми– *S-Tools*, за допомогою якої можна було приховувати файли, зашифровані за алгоритмами DES, TDES , IDEA, у GIF, BMP-зображеннях, а також у нестиснених звукових WAV-записах.

Існують також інші професійні криптографічні системи такі, як *FET XP* (www.fetxp.com), в яких використовуються блочні шифри. Така програма відповідає світовому стандарту захисту даних AES (Advanced Encryption Standard), а також ряду інших стандартів, в числі яких DES, SHS (Secure Hash Standard) і т.д. Програма побудована на основі сучасних відкритих криптоалгортмів, стійкість яких обгрунтована математичними доведеннями і підтверджена численними дослідженнями. Серед основних реалізованих в утиліті функцій варто виділити наступні: по-перше, шифрування із застосуванням відкритого ключа дозволяє безпечно обмінюватись даними з

мережі Інтернет. По-друге, у програмі передбачено функцію повного (без можливості відновлення) вилучення файлів і папок з найрізноманітніших носіїв, будь-то жорсткі диски, дискети, ZIP-диски та ін. По-третє, у FET XP є вбудовані засоби перевірки електронного підпису і SFX-архіватор. Користувачі можуть захистити як окремо вибраний файл, так і цілу папку і навіть весь диск. Крім того, за допомогою утиліти підтримується ASCII-формат, протокол обміну ключами Діффі-Хеллмана і ряд криптографічних алгоритмів: Rijndael, Twofish, RC6, TripleDES, HAVAL, SHA-1, RSA, MD4, MD5. Для функціонування програми потрібен комп'ютер з тактовою частотою від 600 Мгц, 64 МБ оперативної пам'яті (рекомендується 128 МБ) і біля десяти мегабайт на жорсткому диску. FET XP має російськомовний інтерфейс і потужну довідкову систему.

За допомогою пакету *Steganos Security Suite* (www.steganos.com), що складається із семи утиліт різного призначення, можна створювати віртуальні диски, на яких дані шифруються з використанням алгоритму AES і 128-бітового ключа. Також можна кодувати електронні листи, вилучати з дисків дані без можливості відновлення. Використовуючи утиліту також можна приховувати дані будь-яких типів в файлах .BMP і .WAV. При цьому застосовується доволі складний алгоритм, що змінює LSB одиниці кодів повідомлень. При цьому з кожних 16 біт оригінального файлу змінюється лише 1 біт, а на одному компакт-диску можна сховати до 40 Мбайт даних. При цьому розробники потурбувались про те, щоб використовувана технологія не впливала на якість відтворення WAV та відображення BMP-файлів.

Steganography Premium (www.clickok.co.uk/steg/index.html) – комерційна програма для приховування окремих файлів в рисунках формату BMP, JPG та файлах ZIP або DBF, призначена також для шифрування файлів за допомогою одного з трьох алгоритмів (Arcfour, Blowfish або Rijndael).

Puffer (www.briggsoft.com/puffer.htm) – ще одна комерційна програма, за допомогою якої можна приховувати файли в рисунках формату BMP та PNG, для шифрування використовується алгоритм AES. Реалізовані функції шредера

і архіватора.

4t HIT Mail Privacy LITE (www.4t-niagara.com/hitmail.html) – безкоштовна програма, призначена для приховування коротких записок в рисунках форматів JPG, GIF, TIFF та BMP; за її допомогою шифруються дані з використанням ключа довжиною 128 біт (розробниками не вказано назви використовуваного алгоритму шифрування).

ImageHide (www.dancemammal.com/imagehide.htm)– безкоштовна програма для приховування коротких записок в рисунках форматів BMP і PNG, а також для їх шифрування за алгоритмом RC4.

SecurEngine Professional – безкоштовна програма, за якою пропонується на вибір 4 алгоритми шифрування AES, Blowfish, 3DES, GOST. З її допомогою можна шифрувати і приховувати файли в рисунках форматів BMP, GIF, PNG та в HTML-файлах.

Використовуючи freeware-продукт STCLite Stego можна забезпечити захист електронної кореспонденції. Використовуваний алгоритм шифрування виконаний у відповідності з ГОСТ 28147-89. Ніяких тимчасових копій даних за програмою не створюється – весь процес шифрування відбувається в оперативній пам'яті (у безкоштовній версії для шифрування пропонується парольний доступ, у комерційній – використовуються ключі на спеціальних носіях). Зашифровані дані архівуються, що дозволяє зекономити трафік. За допомогою програми можна створювати і пересилати приховані вкладення у листах HTML-формату.

Цей перелік існуючих (і корисних) стеганографічних програм далеко не повний. Якщо звернутись за адресою *www.petitcolas.net/fabien/steganography*, то можна знайти цілий архів програм, також пов'язаних зі стеганографією (Gif-It-Up, Hide and Seek, Steghide , MP3Stego тощо).

Великою перевагою окремих стеганографічних програм є те, що за ними можна наперед визначити зміни, які мають відбутися у зовнішньому вигляді вихідного файлу (наприклад, сполучення кольорів у фотографії) при введенні шифрованих даних. І якщо зміни будуть дуже великими, з'явиться

попередження про це, також буде запропоновано вибрати файл-контейнер більшого розміру.

Крім програм, в яких використовуються прийоми стеганографії за своїм прямим призначенням (тобто для приховування шифрованих даних у файлах-контейнерах), існує кілька програм, за якими приєднуються зашифровані файли до файла-контейнера (в цьому випадку не можна говорити про стеганографію у чистому вигляді). Це, наприклад, продукти *Steganography, Stealth Files* (www.froebis.com/english/sf40.shtml). При реалізації такої схеми відсутні обмеження на співвідношення об'ємів даних, що мають приховатись, і файла, що має бути для неї прикриттям.

Література

1. Вербіцький О.В. Вступ до криптології. – Львів: Видавництво науково-технічної літератури, 1998. – 249 с.
2. Вікіпедія [Електронний ресурс] – Режим доступу: http://uk.wikipedia.org.
3. Жельников В. Криптография от папируса до компьютера. – М.: ABF, 1996 – 336с.
4. Остапов С.Е., Валь Л.О. Основи криптографії: Навчальний посібник. – Чернівці: Книга – XXI, 2008. – 188 с.
5. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф.. Защита информации в компьютерных системах и сетях./ под ред. В.Ф.Шаньгина – М.: Радио и свіязь, 2001. – 376с.
6. Смалько О.А. Захист інформаційних ресурсів: Монографія. - Кам'янець-Подільський: ПП Буйницький О А, 2011. - 704 с.
7. Яковлев А.В., Безбогов А.А., Родин В.В., Шамкин В.Н.. Криптографическая защита информации. – Тамбов:Из-во Тамб. Гос. Тех.. ун-та, 2006. – 140 с.
8. Моргун О.М. Кріптографічні методи захисту інформації. [Електронний ресурс] – Режим доступу: http://a-morgun.narod.ru/a10-01/LK02.pdf.