МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

КАФЕДРА КІБЕРБЕЗПЕКИ

ОПОРНИЙ КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

«КРИПТОГРАФІЯ»

для студентів галузі знань 12 – «Інформаційні технології»
спеціальності 125 – «Кібербезпека»
ступеня вищої освіти «бакалавр»

ТЕРНОПІЛЬ -2020

Опорний конспект лекцій з дисципліни «Криптографія» для студентів галузі знань 12 – «Інформаційні технології» спеціальності 125 – «Кібербезпека» ступеня вищої освіти «бакалавр» / Тернопільський національний економічний університет; Уклад. М.М. Касянчук, І.З. Якименко–Тернопіль,ФО-П Шпак В., 2020. – 64 с.

Опорний конспект лекцій складається з розділів, що рекомендовані стандартом вищої освіти підготовки бакалавра галузі знань 12 – «Інформаційні технології» спеціальності – 125 «Кібербезпека».

Укладачі: Касянчук Михайло Миколайович, к.ф.-м.н., доцент, доцент кафедри кібербезпеки;

Якименко Ігор Зіновійович, к.т.н., доцент, доцент кафедри кібербезпеки;

Свистун Михайло Юрійович, старший інспектор управління стратегічних розслідувань у Тернопільській області Департаменту стратегічних розслідувань Національної поліції України, викладач кафедри кібербезпеки.

Рецензенти: Шевчук Р.П., к.т.н., доцент, доцент кафедри комп'ютерних наук Тернопільського національного економічного університету;

Кінах Я.І, к.т.н., доцент, доцент кафедри програмної інженерії Тернопільського національного технічного університету ім. І.Пулюя.

Затверджено на засіданні кафедри кібербезпеки (протокол №6 від 18.02.2020).

Розглянуто та схвалено групою забезпечення з кібербезпеки (протокол №3 від 18.02.2020).

Розглянуто та схвалено вченою радою факультету комп'ютерних інформаційних технологій, протокол №5від 26 лютого 2020 р.

Зміст

ВСТУП

Проблема захисту інформації шляхом її перетворення, що унеможливлює прочитання цієї інформації сторонньою особою, ще декілька десятиліть тому стосувалася в основному військових операцій. Бурхливе використання комп'ютерних мереж та поява нових потужних обчислювальних засобів спричинили швидкий розвиток криптографії.

Криптогра́фія (від грецького kryptós — прихований і gráphein — писати) — наука про математичні методи забезпечення конфіденційності (неможливості прочитання інформації стороннім) і автентичності (цілісності і справжності авторства) інформації. Для математичного аналізу криптографія використовує інструментарій абстрактної алгебри.

Забезпечити захист даних від доступу до них невтаємничених осіб можна за допомогою як симетричних, так і несиметричних алгоритмів. Вибір криптоалгоритму зумовлений передусім вимогами до швидкості роботи криптографічної системи та сфери його застосування. Тому дуже важливо вміти застосовувати основні криптосистеми на практиці.

Даний опорний конспект лекцій з дисципліни «Криптографія» призначений для вивчення студентами найпростіших криптосистем, симетричних та асиметричних алгоритмів шифрування даних, ознайомлення з протоколами електронного цифрового підпису, криптографією на еліптичних кривих, квантовою криптографією та основами стеганографії.

Розділ 1. Класичні симетричні криптосистеми.

Тема 1. Вступ. Задачі криптографії.

Для позначення всієї області таємного зв'язку використовується термін криптологія, який походить від грецьких коренів «cryptos» – таємний та «logos» – повідомлення. Криптологія поділяється на дві області: криптографію та криптоаналіз. Завдання криптографа – забезпечити конфіденційність та аутентичність повідомлень, які передаються по каналах зв'язку. Завдання криптоаналітика – зламати систему захисту, розроблену криптографами без знання ключа. Ключ – це певний секретний стан деяких параметрів алгоритму криптографічного перетворення даних, які забезпечують вибір тільки одного варіанту із всіх можливих варіантів для даного алгоритму. На даний час розрізняється два класи криптосистем: симетричні одноключові криптосистеми (з секретним ключем) та асиметричні двохключові криптосистеми (з відкритим ключем). В симетричних криптосистемах один і той самий ключ використовується як для шифрування, так і для розшифровування даних. В асиметричних криптосистемах для шифрування і розшифровування використовуються різні, але взаємопов'язані, ключі, причому визначити один ключ, знаючи інший, практично неможливо.

До шифрів, які використовуються для криптографічного захисту інформації, представляється ряд певних вимог:

1. Достатня криптостійкість.
2. Простота шифрування та розшифровування.
3. Незначна надлишковість інформації в зв'язку з шифруванням.
4. Нечуттєвість до незначних помилок шифрування.

Криптологія є частиною такої науки, яка називається захистом інформації, яка охоплює, крім теоретичних основ, також технічні засоби, юридичні аспекти тощо. Тайнопис також є ширшим поняттям, оскільки охоплює також приховування самого факту існування повідомлень.

Будь–яка спроба зі сторони зловмисника розшифрувати шифртекст для отримання відкритого тексту або зашифрувати свій власний текст для отримання правдоподібного шифртексту, не знаючи істинного ключа, називається криптоаналітичною атакою. Фундаментальне правило криптоаналізу, вперше сформульоване у 19 ст. голандцем А.Керкхоффом полягає в тому, стійкість шифру або криптосистеми повинна визначатися тільки секретністю ключа. Іншими словами, це означає, що весь алгоритм шифрування, крім секретного ключа, відомий криптоаналітику зловмисника. Це пояснюється тим, що криптосистема, яка являє собою сукупність апаратних і програмних засобів, яку можна змінити тільки при значних затратах часу і засобів, тоді як ключ змінюється дуже легко.

Існують такі основні типи криптоаналітичних атак:

1. Криптоаналітична атака при наявності тільки відомого шифртексту.
2. Криптоаналітична атака при наявності відомого відкритого тексту.
3. Криптоаналітична атака при можливості вибору відкритого тексту.
4. Криптоаналітична атака з адаптивним вибором відкритого тексту.
5. Криптоаналітична атака з використанням вибраного шифртексту.

6. Криптоаналітична атака методом повного перебору всіх можливих ключів (брутальна атака).

Тема 2. Шифри перестановки та простої заміни.

При шифруванні перестановкою символи відкритого тексту переставляються за визначеним правилом в межах блоку цього тексту. Шифри перестановки є найпростішими та найдревнішими шифрами.

Найдревніший шифр – шифр скитала використовувався в 5 ст. до нашої ери правителями Спарти. На циліндричний стержень спіраллю намотувалась стрічка пергаменту і вздовж стержня писали повідомлення. Тді пергамент знімали і отримували хаотично розміщені букви. Ключем був діаметр валика.

Пізніше почали використовувати шифр частоколу. Наприклад:

р п о р ф я
к и т г а і

Отримується шифртекст: рпосфякитгаі. Ключем є висота частоколу. Зокрема, для частоколу висотою 3 маємо (ліворуч):

и г і и о а я
р п о р ф я р т р і
к т а к п г ф

Отримується шифртекст: игірпорфякта. Це складний частокіл. При використанні простого частоколу отримуємо: иоаяртрікпгф. аналогічно можна використати частокіл і більшої висоти.

З кінця 14 ст. виникли шифруючі таблиці. Наприклад, відкритий текст записується в таблицю по стовбцях, а читається по рядках. Ключем є розмір таблиці. Їх удосконаленням стали шифруючі таблиці з ключовими словами, коли стовбці та рядки переставляються у відповідності до цих ключових слів.

		л	і	т	о
		2	1	4	3
з	2	п	р	и	л
и	3	і	т	а	ю
м	4	в	о	с	ь
а	1	м	о	г	о

		і	л	о	т
		1	2	3	4
з	2	р	п	л	и
и	3	т	і	ю	а
м	4	о	в	ь	с
а	1	о	м	о	г

1	о	м	о	г
2	р	п	л	и
3	т	і	ю	а
4	о	в	ь	с

В середні віки використовувалося також шифрування за допомогою магічних квадратів. Це квадратні таблиці з вписаними в клітинки послідовностями натуральних чисел, починаючи з 1, щоб їх сума по стовбцях, рядках та діагоналях дорівнювала одному і тому самому числу.

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

о	и	р	м
і	о	с	ю
в	т	а	ь
л	г	о	п

Якщо не враховувати повороти, то існує тільки один квадрат розміром 3x3, 880 – розміром 4x4, біля 250000 – розміром 5x5.

Шифр Кардано являє квадрат з клітинками, частина яких вирізані. Цей квадрат накладають на такий самий суцільний квадрат і у вирізані клітинки вписують текст. Потім верхній квадрат повертають на 90 градусів і відкриваються нові клітинки. Таким чином заповнюється вся таблиця. Ключем у шифрі Кардано є розміщення вирізаних клітин. У кожному рядку і стовбці може бути вирізана тільки одна клітинка і кількість клітинок у рядку і стовбці має бути парною.

До шифрів простої заміни відноситься шифр Цезаря, шифр Цезаря з ключовим словом, полібіанський квадрат, шифруючі таблиці Трисемуса.

Тема 3. Шифри складної заміни. Біграмні шифри.

Шифри складної заміни називають багатоалфавітними, оскільки для шифрування кожного символа вихідного повідомлення використовують свій шифр простої заміни.

Шифр Гронсфельда являє собою модифікацію шифра Цезаря з числовим ключем. Під буквами вихідного повідомлення записують цифри ключового слова. Якщо ключ коротший, то його запис циклічно повторюють. Шифртекст отримують аналогічно шифру зсуву, але кожен символ зсувають на ту кількість знаків, яка записана під символом. Потрібно відмітити, що шифр Гронсфельда зламується досить легко, але його можна вдосконалити, зокрема, подвійним шифруванням різними ключами.

Біграмний шифр Плейфейра, винайдений у 1854 році, використовує прямокутну таблицю з хаотично або з ключовим словом вписаними буквами алфавіту. Відкритий текст розбивається на біграми. Він повинен мати парну кількість символів і не містити біграм з однаковими буквами. В таблиці шукаються букви біграми і вважається, що вони є вершинами прямокутника. У двох інших вершинах будуть лежати дві букви шифртексту. Якщо букви відкритого тексту потрапляють в один рядок чи стовбець, то вибираються букви, що лежать під ними, або, відповідно, ліворуч.

Для усунення такого недоліку використовується подвійний квадрат Уїтстона, в якому використовується дві прямокутних таблиці з розміщеними буквами алфавіту. Букви відкритого тексту шукаються в різних таблицях і аналогічно утворюються прямокутники. Тепер букви відкритого тексту не потраплять в один стовбець, але можуть потрапити в один рядок. Для усунення цього недоліку використовується шифр чотирьох квадратів, розміщених в квадратах. Букви відкритого тексту шукаються в діагонально протилежних квадратах, в інших квадратах шукаються букви шифртексту. Тепер ні в один рядок, ні в один стовбець букви відкритого тексту не потраплять.

Узагальненням цих шифрів є шифр Віженера. він представляється квадратною таблицею Віженера, розмірність якої відповідає кількості букв в алфавіті. По горизонталі розміщуються букви відкритого тексту, по вертикалі – букви ключа. На їх перетині отримуємо букви шифртексту. Його удосконаленням є шифр Віженера з автоключем.

Тема 4. Шифр одноразового блокноту.

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 необхідно скористатися модерними засобами зв'язку для передачі повідомлення, або доручити шифрування комп'ютерові, то виявляється, що у технічному відношенні традиційний текст не є найзручнішою формою для перетворення та передачі інформації.

З цього погляду вигіднішим є подання інформації у цифровій формі. Ідея є зовсім простою - кожен символ тексту заміняється його номером у алфавіті (таблиця 4.1). Нумерація, як правило, починається 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.

Таблиця 4.1 – Нумерація букв українського алфавіту.

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И
0	1	2	3	4	5	6	7	8	9	10
І	Ї	Й	К	Л	М	Н	О	П	Р	С
11	12	13	14	15	16	17	18	19	20	21
Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
22	23	24	25	26	27	28	29	30	31	32

Номери букв можна записувати не в десятковій системі числення, а у двійковій. Для того ж слова банан матиме місце запис 000001000000010001000000010001, де кожен блок із шести цифр є номером відповідної букви у двійковому записі. Така форма подання тексту називається 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.

Шифр одноразового блокноту був винайдений у 1917 році Гілбертом Вернамом. Назва шифру 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.

Даний шифр використовує операцію додавання бітів за модулем 2. Ця операція позначається символом $\oplus$ і задається так:

$$0 \oplus 0 = 0,\ 0 \oplus 1 = 1,\ 1 \oplus 0 = 1,\ 1 \oplus 1 = 0.$$

Цю операцію можна поширити на двійкові слова однакової довжини зазначивши, що додавання таких слів здійснюється побітово. Наприклад,

$$\begin{array}{l} 000001000000010001000000010001 \\ \oplus \\ \underline{001101110101100010011000111010} \\ 001100110101110011011000101011 \end{array}$$

Для двійкових слів X і Y однакової довжини результат їх побітового додавання позначається 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. Нехай для фіксованого *k* через 0 позначається 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 $X \oplus X = 0$.

Перед шифруванням повідомлення *M* записують у двійковій формі. Ключем *K* служить довільні двійкове слово однакової з *M* довжини. Криптотекст *C* отримують побітовим додаванням повідомлення і ключа, тобто $C = M \oplus K$.

Для прикладу, нехай потрібно зашифрувати слово *банан*. Записуємо його у двійковій формі:

M = 00000100000001000100000010001.

В якості ключа виберемо

K = 00110111010110001001100011101O.

Сумування цих двох двійкових послідовностей вже проведене вище. Отож маємо криптотекст

C = 00110011010111001101100010101l.

Дешифрування у шифрі одноразового блокноту збігається із шифруванням, тобто щоб отримати вихідне повідомлення *M*, слід додати до криптотексту *C* той же ключ *K*. Це випливає з того, що оскільки $C = M \oplus K$, то

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

Шифр одноразового блокноту є абсолютно надійним або, як ще кажуть, надійним у теоретико-інформаційному сенсі. Якщо суперник не знає ключа *K*, то з підслуханого криптотексту *C* він зовсім нічого не може довідатись про повідомлення *M*. Справді, двійкове слово *C* могло би бути криптотекстом для будь-якого повідомлення *M'*, якби шифрування здійснювалось з деяким іншим ключем *K'*, а саме $K' = M' \oplus C$, в той час як для суперника всі ключі однаково вірогідні. Наприклад, за допомогою деякого ключа *K* щойно перетворене повідомлення *банан* у криптотекст

C = 00110011010111001101100010101l.

Але такий же криптотекст отримується при зашифруванні повідомлення *груша* з ключем

K' = 00111110000110010000010010101l.

Однак обмеженість сфери застосувань шифру очевидна, оскільки він вимагає ключа не меншої 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 нема, або ж він не є достатньо швидким.

Назва шифру походить від того, що агент, який здійснював шифрування вручну, отримував свої копії ключів, записаними в блокнот. Якщо ключ застосовувався, то сторінка з ним знищувалась.

Розділ 2. Сучасні симетричні криптосистеми.

Тема 5. Алгоритм DES.

Стандарт шифрування даних DES (Data Encryption Standard) був розроблений у 70-х роках фахівцями IBM і у 1976 році був прийнятий у якості федерального стандарту Сполучених Штатів для захисту комерційної та урядової інформації, не пов'язаної з національною безпекою.

Як і шифр одноразового блокноту, DES оперує з інформацією поданою у двійковій формі, а довжина блоку і довжина ключа вибрані рівними 64. Іншими словами, двійкове повідомлення *M* розбивається на блоки по 64 біти і шифрується кожен блок окремо, використовуючи один і той же двійковий ключ *K* довжини 64. Таким чином, повідомлення $M = M_1 M_2 M_3 ...$ перетворюється у криптотекст $C = C_1 C_2 C_3 ...$, де $C_i = E_K(M_i)$. У стандарті DES кожен блок криптотексту C, також є двійковою послідовністю довжини 64.

Алгоритм шифрування *E* повинен задовольняти три умови:

1) можливість дешифрування: для будь-якого ключа *K* різним блокам повідомлення *M*' і *M*" відповідають різні блоки криптотексту $E_K(M')$ і $E_K(M'')$, інакше кажучи, алгоритм E_K з будь-яким ключем здійснює перестановку двійкових послідовностей довжини 64;

2) ефективність: і шифрування, і дешифрування відбуваються швидко;

3) надійність: якщо ключ невідомий, то немає способу розкриття шифру.

Схема алгоритму DES працює з 64-бітовим блоком відкритого тексту. Після початкової перестановки блок розбивається на праву і ліву половини довжиною по 32 біта. Потім виконується 16 етапів однакових дій, функцією *f* у яких дані поєднуються з ключем. Після шістнадцятого етапу права і ліва половини поєднуються й алгоритм завершується заключною перестановкою (зворотною відносно початкової) (рисунок 5.1).

На кожному етапі біти ключа зсуваються і потім з 56 бітів ключа вибираються 48 бітів. Права половина даних збільшується до 48 бітів за допомогою перестановки з розширенням, поєднується за допомогою XOR з 48 бітами зміщеного і переставленого ключа, проходить через 8 S-блоків, утворюючи 32 нових біта, і переставляється знову. Ці чотири операції і виконуються функцією *f* (див. рисунок 5.2). Потім результат функції *f* поєднується з лівою половиною за допомогою іншого XOR. У підсумку цих дій з'являється нова права половина, а вихідна права половина стає нової лівою. Ці дії повторюються 16 разів, тобто здійснюється 16 етапів DES.

Якщо B_i - це результат *i*-ої ітерації, L_i і R_i - ліва і права половини B_i, K_i - 48-бітовий ключ для етапу *i*, а *f* - це функція, що виконує усі підстановки, перестановки і XOR із ключем, то етап можна представити так:

$$L_i = R_{i-1}, \; R_i = L_{i-1} \oplus f(R_{i-1}, K_i).$$

Початкова перестановка виконується ще до етапу 1, при цьому 58-й біт переставляється у бітову позицію 1, 50-й біт - у бітову позицію 2, 42-й біт - у бітову позицію 3, і так далі (таблиця 5.1).

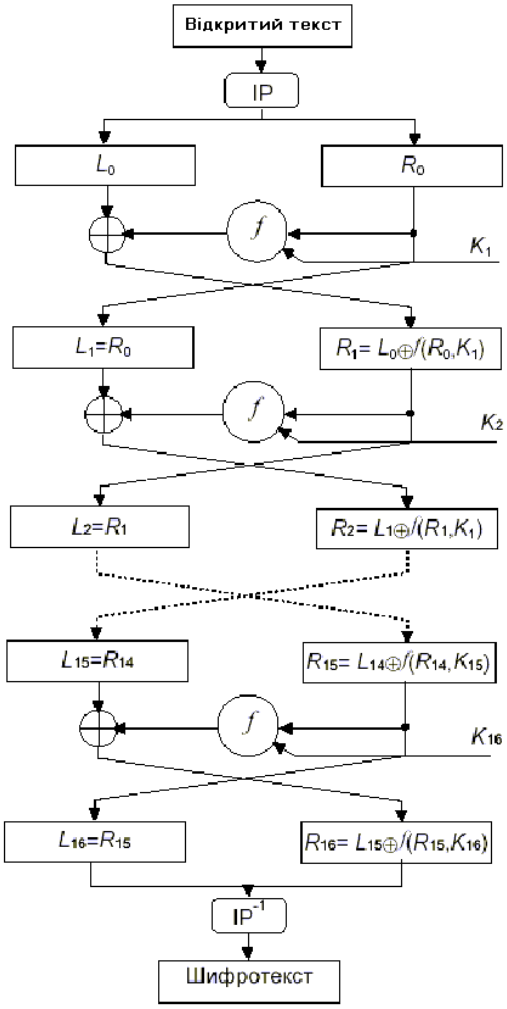


Рисунок 5.1 – Алгоритм DES

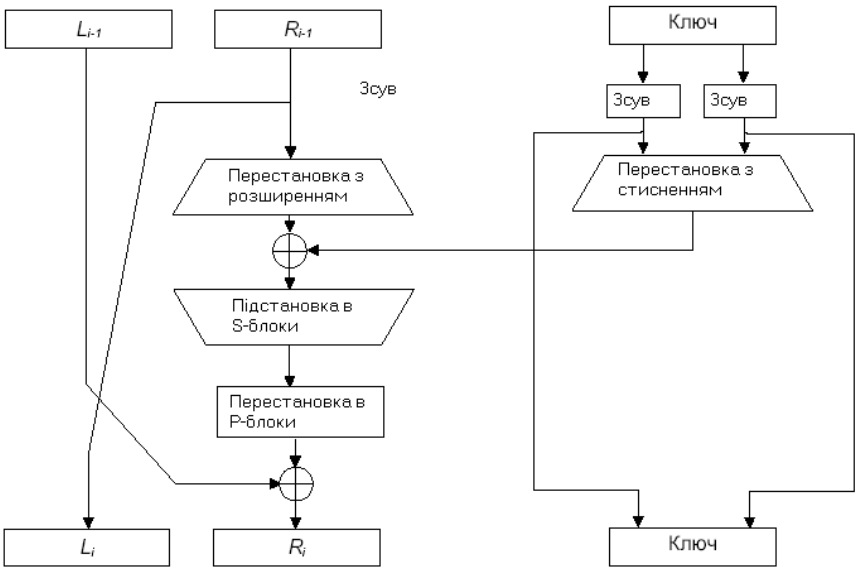


Рисунок 5.2 - Один етап DES

Початкова перестановка і відповідна завершальна перестановка не впливають на безпеку DES. Оскільки програмна реалізація цієї багатобітової перестановки нелегка (на відміну від тривіальної апаратної), у багатьох

програмних реалізаціях DES початкова і заключна перестановка не використовуються. Хоча такий новий алгоритм не відповідає стандарту DES.

Таблиця 5.1 - Початкова перестановка

58,	50,	42,	34,	26,	18,	10,	2,	60,	52,	44,	36,	28,	20,	12,	4,
62,	54,	46,	38,	30,	22,	14,	6,	64,	56,	48,	40,	32,	24,	16,	8,
57,	49,	41,	33,	25,	17,	9,	1,	59,	51,	43,	35,	27,	19,	11,	3,
61,	53,	45,	37,	29,	21,	13,	5,	63,	55,	47,	39,	31,	23,	15,	7

64-бітовий ключ DES спочатку зменшується до 56-бітового шляхом відкидання кожного восьмого біта (таблиця 5.2). Ці біти використовуються тільки для контролю парності, дозволяючи перевіряти правильність ключа.

Таблиця 5.2 - Перестановка ключа

57,	49,	41,	33,	25,	17,	9,	1,	58,	50,	42,	34,	26,	18,
10,	2,	59,	51,	43,	35,	27,	19,	11,	3,	60,	52,	44,	36,
63,	55,	47,	39,	31,	23,	15,	7,	62,	54,	46,	38,	30,	22,
14,	6,	61,	53,	45,	37,	29,	21,	13,	5,	28,	20,	12,	4

Після отримання 56-бітового ключа для кожного з 16 етапів DES генерується новий 48-бітовий підключ. Ці підключі K_i, визначаються в такий спосіб. Спочатку 56-бітовий ключ поділяється на дві 28-бітові половинки, які циклічно зсуваються ліворуч на один чи два біти в залежності від етапу (таблиця 5.3).

Таблиця 5.3 - Число бітів зсуву в залежності від етапу

Этап	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після зсуву вибирається 48 з 56 бітів. Оскільки при цьому не тільки вибирається підмножина бітів, але і змінюється їхній порядок, ця операція називається перестановка із стисненням (таблиця 5.4). Її результатом є набір з 48 бітів. Наприклад, біт зсунутого ключа в позиції 33 переміщається в позицію 35 результату, а 18-й біт зсунутого ключа відкидається.

Таблиця 5.4 - Перестановка зі стисненням

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

Через зсув для кожного підключа використовується інша підмножина бітів ключа. Кожен біт використовується приблизно в 14 з 16 підключів.

Права половина даних R_i розширюється від 32 до 48 бітів. Оскільки при цьому не просто повторюються визначені біти, але і змінюється їхній порядок, ця операція називається перестановкою з розширенням. Іноді вона називається *E*-блоком. Для кожного 4-бітового вхідного блоку перший і четвертий біт являють собою два біти вихідного блоку, а другий і третій біти - один біт вихідного блоку. Наприклад, біт вхідного блоку в позиції 3 переміститься в позицію 4 вихідного блоку, а біт вхідного блоку з позиції 21 - у позиції 30 і 32 вихідного блоку (рисунок 5.3).

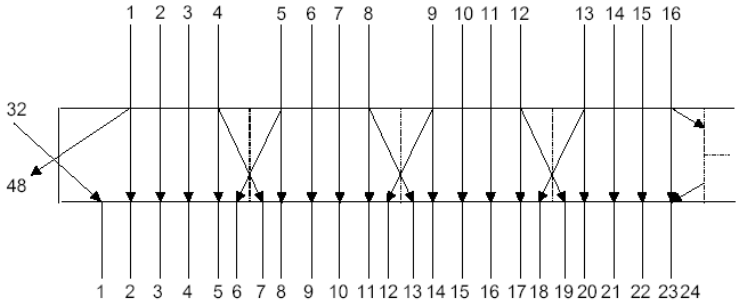


Рисунок 5.3 - Перестановка з розширенням

Хоча вихідний блок більше вхідного, кожен вхідний блок генерує унікальний вихідний блок (таблиця 5.5).

Таблиця 5.5 - Перестановка з розширенням

32,	1,	2,	3,	4,	5,	4,	5,	6,	7,	8,	9,
8,	9,	10,	11,	12.,	13,	12,	13,	14,	15,	16,	17,
16,	17,	18,	19,	20,	21,	20,	21,	22,	23,	24,	25,
24,	25,	26,	27,	28,	29,	28,	29,	30,	31,	32,	1

Після об'єднання стиснутого блоку з розширеним за допомогою XOR над 48-бітовим результатом виконується операція підстановки. Підстановки здійснюються у восьми блоках підстановки (S-блоках). У кожного S-блоку 6-бітовий вхід і 4-бітовий вихід. 48 бітів поділяються на вісім 6-бітових підблоків. Кожен окремий підблок обробляється окремим S-блоком: перший підблок - S-блоком 1, другий - S-блоком 2, і так далі (рисунок 5.4).

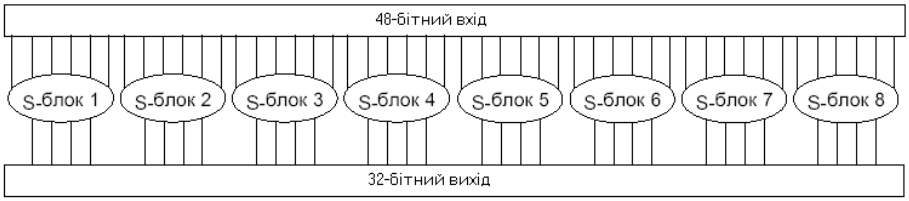


Рисунок 5.4 - Підстановка - S-блоки.

Кожен S-блок являє собою таблицю з 2 рядків і 16 стовпців. Кожен елемент у блоці є 4-бітовим числом. По 6 вхідних бітах S-блоку визначається, під якими номерами стовпців і рядків шукати вихідне значення (таблиця 5.6).

Таблиця 5.6 - S-блоки

S-блок 1:															
14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12.,	5,	9,	0,	7,
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12.,	11,	9,	5,	3,	8,
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13,
S-блок 2:															
15,	1,	8,	14,	6,	11,	3,	4,	9,	7,	2,	13,	12,	0,	5,	10,
3,	13,	4,	7,	15,	2,	8,	14,	12,	0,	1,	10,	6,	9,	11,	5,
0,	14,	7,	11,	10,	4,	13,	1,	5,	8,	12,	6,	9,	3,	2,	15,
13,	8,	10,	1,	3,	15,	4,	2,	11,	6,	7,	12,	0,	5,	14,	9,
S-блок 3:															
10,	0,	9,	14,	6,	3,	15,	5,	1,	13,	12,	7,	11,	4,	2,	8,
13,	7,	0,	9,	3,	4,	6,	10,	2,	8,	5,	14,	12,	11,	15,	1,
13,	6,	4,	9,	8,	15,	3,	0,	11,	1,	2,	12,	5,	10,	14,	7,
1,	10,	13,	0,	6,	9,	8,	7,	4,	15,	14,	3,	11,	5,	2,	12,
S-блок 4:															
7,	13,	14,	3,	0,	6,	9,	10,	1,	2,	8,	5,	11,	12,	4,	15,
13,	8,	11,	5,	6.	15,	0,	3,	4,	7,	2,	12,	1,	10,	14,	9,
10,	6,	9,	0,	12,	11,	7,	13,	15,	1,	3,	14,	5,	2,	8,	4,
3,	15,	0,	6,	10,	1,	13,	8,	9,	4,	5,	11,	12,	7,	2,	14,
S-блок 5:															
2,	12,	4,	1,	7,	10,	11,	6,	8,	5,	3,	15,	13,	0,	14,	9,
14,	11,	2,	12,	4,	7,	13,	1,	5,	0,	15,	10,	3,	9,	8,	6,
4,	2,	1,	11,	10,	13,	7,	8,	15,	9,	12,	5,	6,	3,	0,	14,
11,	8,	12,	7,	1,	14,	2,	13,	6,	15,	0,	9,	10,	4,	5,	3,
S-блок 6:															
12,	1,	10,	15,	9,	2,	6,	8,	0,	13,	3,	4,	14,	7,	5,	11,
10,	15,	4,	2,	7,	12,	9,	5,	6,	1,	13,	14,	0,	11,	3,	8,
9,	14,	15,	5,	2,	8,	12,	3,	7,	0,	4,	10,	1,	13,	11,	6,
4,	3,	2,	12,	9,	5,	15,	10,	11,	14,	1,	7,	6,	0,	8,	13,
S-блок 7:															
4,	11,	2,	14,	15,	0,	8,	13,	3,	12,	9,	7,	5,	10,	6,	1,
13,	0,	11,	7,	4,	9,	1,	10,	14,	3,	5,	12,	2,	15,	8,	6,
1,	4,	11,	13,	12,	3,	7,	14,	10,	15,	6,	8,	0,	5,	9,	2,
6,	11,	13,	8,	1,	4,	10,	7,	9,	5,	0,	15,	14,	2,	3,	12,
S-блок 8:															
13,	2,	8,	4,	6,	15,	11,	1,	10,	9,	3,	14,	5,	0,	12,	7,
1,	15,	13,	8,	10,	3,	7,	4,	12,	5,	6,	11,	0,	14,	9,	2,
7,	11,	4,	1,	9,	12,	14,	2,	0,	6,	10,	13,	15,	3,	5,	8,
2,	1,	14,	7,	4,	10,	8,	13,	15,	12,	9,	0,	3,	5,	6,	11

Вхідні біти певним чином визначають елемент S-блоку. Розглянемо 6-бітовий вхід S-блоку: b_1, b_2, b_3, b_4, b_5 і b_6. Біти b_1 і b_6 поєднуються, утворюючи 2-

бітове число від 0 до 3, що відповідає рядку таблиці. Середні 4 біти з b_2 по b_5 об'єднуються утворюючи 4-бітове число від 0 до 15, що відповідає стовпцю таблиці. Наприклад, нехай на вхід шостого S-блоку (тобто біти функції XOR з 31 по 36) подаються як 110011. Перший і останній біт об'єднуючись утворять 11, що відповідає рядку 3 шостого S-блоку. Середні 4 біти утворять 1001, що відповідає стовпцю 9 того ж S-блоку. Елемент S-блоку 6, що знаходиться на перетині рядка 3 і стовпця 9, - це 14. (рядки і стовпці нумеруються з 0, а не з 1.) Замість 110011 на вихід подається 1110.

Звичайно ж набагато легше реалізувати S-блоки програмно у вигляді масивів з 64 елементами.

Підстановка за допомогою S-блоків є ключовим етапом DES. Інші дії алгоритму лінійні і легко піддаються аналізу. S-блоки нелінійні і саме вони в більшому ступені ніж всі інше, забезпечують безпеку DES.

32-бітовий вихід підстановки за допомогою S-блоків (вісім 4-бітових блоків), перетасовуються відповідно до P-блоку. Ця перестановка переміщує кожен вхідний біт в іншу позицію. Жоден біт не використовується двічі, і жоден біт не ігнорується. Цей процес називається прямою чи простою перестановкою (таблиця 5.7). Наприклад, біт 21 переміщається в позицію 4, а біт 4 - у позицію 31.

Таблиця 5.7 - Перестановка за допомогою P-блоків

16,	7,	20,	21,	29,	12,	28,	17,	1,	15,	23,	26,	5,	18,	31,	10,
2,	8,	24,	14,	32,	27,	3,	9,	19,	13,	30,	6,	22,	11,	4,	25

Нарешті результат перестановки за допомогою P-блоків об'єднується за допомогою XOR з лівою половиною початкового 64-бітового блоку. Потім ліва і права половини міняються місцями.

Кінцева перестановка є зворотною стосовно початкової перестановки. Ліва і права половини не міняються місцями після останнього етапу DES, замість цього об'єднаний блок $R_{16}L_{16}$ використовується як вхід заключної перестановки (таблиця 5.8). Перестановка половинок з наступним циклічним зсувом привела б до точно такого ж результату. Це зроблено для того, щоб алгоритм можна було використовувати як для шифрування, так і для дешифрування.

Таблиця 5.8 - Заключна перестановка

40,	8,	48,	16,	56,	24,	64,	32,	39,	7,	47,	15,	55,	23,	63,	31,
38,	6,	46,	14,	54,	22,	62,	30,	37,	5,	45,	13,	53,	21,	61,	29,
36,	4,	44,	12,	52,	20,	60,	28,	35,	3,	43,	11,	51,	19,	59,	27,
34,	2,	42,	10,	50,	18,	58,	26,	33,	1,	41,	9,	49,	17,	57,	25

Для шифрування і дешифрування використовується той самий алгоритм. DES дозволяє використовувати для шифрування і дешифрування блоку ту саму функцію. Єдина відмінність полягає в тому, що ключі повинні використовуватися в зворотному порядку. Тобто, якщо на етапах шифрування використовувалися ключі $K_1, K_2, K_3, ..., K_{16}$, то ключами дешифрування будуть $K_{16}, K_{15}, K_{14}, ..., K_1$.

Алгоритм, що створює ключ для кожного етапу також циклічний. Ключ зсувається праворуч, а число позицій зсуву дорівнює 0, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

Існує 4 режими роботи алгоритму DES: електронна кодова книга (ECB), зчеплення блоків шифру (CBC), зворотній зв'язок по шифртексту (CFB), зворотній зв'язок по виходу (OFB).

Тема 6. Алгоритми IDEA та ГОСТ28147–89.

Алгоритм IDEA (International Data Encryption Algorithm) є блоковим шифром. Він оперує 64–бітовими блоками відкритого тексту. Його ключ має 128 біт. Один і той же алгоритм використовується і для шифрування, і для розшифровування. В алгоритмі використовуються такі математичні операції: побітове додавання по модулю 2; додавання беззнакових цілих по модулю 2^{16}; множення цілих по модулю (2^{16} +1). Всі операції виконуються над 16–бітовими підблоками. Ці три операції несумісні в тому, що ніяка пара з цих трьох операцій не задовольняє асоціативному та дистрибутивному законам. Всього виконується 8 циклів.

Комбінування цих трьох операцій забезпечує комплексне перетворення входу, істотно затрудняючи криптоаналіз IDEA в порівнянні з DES, який базується тільки на побітовому додаванні по модулю 2.

Алгоритм IDEA використовує 52 підключі (по шість для кожного з 8 циклів і чотири для перетворення виходу). Розшифрування здійскнюється в зворотньому порядку.

Алгоритм IDEA може працювати в тих же режимах, що і DES, однак має ряд переваг. Він значно безпечніший DES, оскільки має вдвічі більший ключ. Його внутрішня структура забезпечує кращу стійкість до криптоаналізу. Програмні реалізації IDEA приблизно вдвічі швидші, ніж DES.

Стандарт шифрування ГОСТ 28147–89 являє собою 64–бітовий блочний алгоритм з 256–бітовим ключем. Використовуються такі операції: побітове додавання по модулю 2; операція додавання по модулю 2^{32} двох 32–розрядних двійкових чисел; операція додавання двох 32–розрядних чисел за модулем 2^{32}-1.

Алгоритм передбачає чотири режими роботи:

1. Шифрування даних в режимі простої заміни.
2. Шифрування даних в режимі гамування.
3. Шифрування даних в режимі гамування із зворотним зв'язком.
4. Вироблення імітовставки.

Тема 7. Сімейство алгоритмів RC.

RC–4 являє собою потоковий шифр із змінною довжиною ключа. Алгоритм працює в режимі зворотнього зв'язку по виходу (OFB). Ключова послідовність не залежить від вихідного тексту. Структура алгоритму включає блок заміни розмірністю 8x8, який являє собою залежну від ключа перестановку чисел 0, ..., 255 змінної довжини. Є два лічильники i та j, початкове значення яких дорівнює 0. На початку генерується псевдовипадковий байт, який потім додається по модулю 2 з байтом вихідного тексту для отримання шифртексту. Ініціалізація блоку заміни виконується за допомогою ключа. За програмною реалізацією

алгоритм RC–4 приблизно в 10 разів швидший від DES. Можливі узагальнення алгоритму на більшу довжину слів і розмір блоку заміни. Можна побудувати шифр з блоком заміни розмірністю 16x16 (потрібно 128 Кбайт пам'яті) і довжиною слова 16 біт. Етап ініціалізації буде відбуватися значно повільніше, але в результаті алгоритм все одно буде швидшим.

В алгоритмі RC–6 передбачено використання чотирьох робочих регістрів, а також введена операція цілочисельного множення, яка дозволяє збільшити збурення, створене кожним циклом шифрування, що приводить до збільшення стійкості та можливості зменшити число циклів.

RC–6 є повністю параметризованим алгоритмом шифрування. Конкретна версія RC–6 позначається як RC–6–w/r/b, w позначає довжину слова в бітах, r – ненульова кількість ітераційних циклів шифрування, b – довжина ключа в байтах. У всіх варіантах RC–6–w/r/b працює з чотирма w – бітовими словами, використовуючи шість базових операцій, які позначаються таким чином:

a+b – цілочисельне додавання по модулю 2^w;

a-b – цілочисельне віднімання по модулю 2^w;

a⊕b – побітове виключаюче або w–бітових слів;

a×b – цілочисельне множення по модулю 2^w;

a<<b – циклічний зсув w–бітового слова ліворуч на величину, задану $\log_2 w$ молодшими бітами b;

a>>b – циклічний зсув w–бітового слова праворуч на величину, задану $\log_2 w$ молодшими бітами b.

Алгоритм обчислення ключів виглядає таким чином. Користувач задає ключ довжиною b байтів. Достатня кількість ненульових байтів дописується в кінець, щоб отримати ціле число слів. Потім ці байти записуються, починаючи з молодшого, в масив з c слів.

Структура шифру RC–6 є узагальненням мережі Фейстела. Блок тексту розбивається не на 2, а на 4 підблоки і на кожній ітерації змінюються два підблоки з чотирьох. При цьому в кінці ітерації шифрування відбувається циклічний зсув підблоків ліворуч (при розшифруванні відповідно праворуч). Це узагальнення привело до того, що була втрачена властивість інваріантності блоків шифрування і розшифровки, хоча це не є визначальним в оцінці даного алгоритму.

Розділ 3. Сучасні асиметричні криптосистеми.

Тема 8. Арифметика асиметричних криптосистем.

Для будь-якого цілого a і натурального b однозначно визначені цілі числа q і r такі, що a = $bq + r$ і $0 < r < b$. Число q називається *часткою,* а r – *остачею* від ділення a на b Наприклад, рівність –20= (–1)·67 + 47 означає, що –20 при діленні на 67 дає частку –1 і остачу 47. Для остачі будемо вживати таке позначення r = a mod b. Число r будемо також називати *(зведеним) лишком* числа a за модулем b. Числа a і b називаються *взаємно простими,* якщо НСД (a, b) = 1. *Алгоритм Евкліда* (3 ст. до н.е.) для знаходження НСД двох натуральних чисел a і b грунтується на співвідношеннях

НСД (a, b) = НСД (a, a mod b) для a > b

НСД(a,0) = a

Продемонструємо ідею цього алгоритму на прикладі.

Приклад. Щоб знайти НСД (211, 79), застосовуємо алгоритм Евкліда. Робота алгоритму зводиться до кількаразового ділення з остачею:

$$211 = 79 \cdot 2 + 53$$
$$79 = 53 \cdot 1 + 26$$
$$53 = 26 \cdot 2 + 1$$
$$26 = 1 \cdot 26 + 0$$

Маємо НСД (211, 79)=НСД (79, 53)=НСД (53, 26)=НСД (26,1)=НСД(1,0)=1.

Алгоритм Евкліда дає такий наслідок.

Твердження 2.2. *Для кожної пари взаємно простих чисел a і b можна знайти такі цілі u і v, що* $ua + vb = 1$.

Приклад. Нехай a = 211, b = 79. Протокол роботи алгоритму Евкліда виписаний у попередньому прикладі. Рухаючись знизу вверх, отримуємо

$1 = 1 \cdot 53 + (-2) \cdot 26 = 1 \cdot 53 + (-2) \cdot (79 - 1 \cdot 53) = (-2) \cdot 79 + 3 \cdot 53 = (-2) \cdot 79 + 3 \cdot (211 - 2 \cdot 79) = 3 \cdot 211 + (-8) \cdot 79.$

Отже, u = 3 і v= – 8.

Елемент, обернений до x за модулем n відносно множення, будемо позначати через x^{-1} mod n або просто x^{-1}. Це означає, що $x \cdot x^{-1} \bmod n = 1$.

Приклад. Нехай ми хочемо знайти елемент, обернений до 79 за модулем 211. Вище була отримана рівність 1 = 3·211 + (– 8·79. З неї випливає, що (–8)·79 = 1(mod211). Отже, $79^{-1} \bmod 211 = (-5) \bmod 211 = 203$.

Функцією Ейлера від числа n є кількість натуральних чисел, взаємно простих з n і позначається $\phi(n)$.

Теорема Ейлера (1763). *Для взаємно простих цілого x і натурального n справедлива конгруенція* $x^{\phi(n)} = 1 \pmod n$.

Мала теорема Ферма (1640). *Якщо ціле x не ділиться на просте p, то* $x^{p-1} = 1 \pmod p$.

Китайська теорема про остачі (I ст. до н.е.). *Для будь-якої пари взаємно простих натуральних чисел* n_1 *і* n_2 *та для будь-якої пари цілих чисел* x_1 і x_2, *можна знайти таке ціле x, що* $x = x_1 \pmod{n_1}$ і $x = x_2 \pmod{n_2}$.

Приклад. Нехай ми хочемо знайти ціле x, яке при діленні на 211 давало б остачу 100, а при діленні на 79 остачу 10. Вище була отримана рівність 1=3·211+(-8)·79.

Отже, в якості x можна взяти: 10·3·211+ 100·(-8)·79 = -56870. Зрозуміло, що це число можна замінити його остачею від ділення на 211·79 = 16669. В результаті отримуємо 9806.

Тема 9. Афінні шифри

Афінні шифри – підклас шифрів заміни, що включає як частковий випадок шифр Віженера і шифр перестановки з фіксованим періодом.

Шифр зсуву використовує в якості ключа s таке, що $0 \le s < n$, де n - кількість букв у алфавіті відкритого тексту.

В процесі шифрування у повідомленні кожна буква x заміщується буквою $E(x) = (x + s) \bmod n$.

Для здійснення дешифрування у криптотексті кожна буква x' заміщується буквою $D(x') = (x' + s') \bmod n$, де $s' = n - s$. Величину зворотнього зсуву s' називається дешифруючим ключем.

За аналогією можна побудувати *лінійний шифр*. Ключем в даному шифрі є a таке, що $0 < a < n$ і $НСД(a,n) = 1$. У повідомленні кожна буква x заміщується буквою $E(x) = (ax) \bmod n$.

В процесі дешифрування у криптотексті кожна буква x' заміщується буквою $D(x') = (a'x') \bmod n$, де $a' = a^{-1} \bmod n$ - дешифруючий ключ.

Узагальненням і шифру зсуву, і лінійного шифру є *афінний шифр*.

Ключами в даному шифрі використовуються a, s такі, що $0 \le s < n$, $0 < a < n$ і $НСД(a,n) = 1$. У повідомленні кожна буква x заміщується буквою $E(x) = (ax + s) \bmod n$.

В процесі дешифрування у криптотексті кожна буква x' заміщується буквою $D(x') = (a'x' + s') \bmod n$, де пара $a' = a^{-1} \bmod n$ і $s' = (-a's) \bmod n$ є дешифруючим ключем.

Як і кожен шифр простої заміни, афінний шифр піддається частотному аналізові.

Для означення афінних шифрів вищих порядків спочатку необхідно ввести операцію додавання в Z_n^k. Сумою векторів $X = (x_1,...,x_k)$ і $S = (s_1,...,s_k)$ з Z_n^k є вектор. Z_n^k з операцією додавання є групою. Вектор $-S = (n - s_1,...,n - s_k)$ є оберненим до вектора $S = (s_1,...,s_k)$.

Шифр зсуву k-го порядку (шифр Віженера з періодом k) в якості ключа використовує $S \in Z_n^k$. Повідомлення розбивається на k-грами. Кожна k -грама X заміщується k -грамою $E(x) = X + S$.

При дешифруванні кожна k -грама X' криптотексту заміщується k-грамою $D(X') = X' + S'$, де $S' = -S$ є дешифруючим ключем.

Перед тим як перейти до лінійного шифру необхідно зазначити, що через $M_k(Z_n)$ позначається множина матриць розміру $k \times k$ з коефіцієнтами з кільця Z_n, а через $GL_k(Z_n)$ - підмножину оборотних матриць. Для $A \in GL_k(Z_n)$ обернена до неї матриця позначається через A^{-1}. Добутком AX матриці $A = (a_{ij})$ з $M_k(Z_n)$ на вектор-стовпчик $X = (x_1,...,x_k)$ з Z_n^k є вектор-стовпчик

$$\begin{pmatrix} a_{11}a_{12}...a_{1k} \\ a_{21}a_{22}...a_{2k} \\ ..... \\ a_{k1}a_{k2}...a_{kk} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix} = \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + ... + a_{1k}x_k \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2k}x_k \\ ..... \\ a_{k1}x_1 + a_{k2}x_2 + ... + a_{kk}x_k \end{pmatrix}.$$

Лінійний шифр k-го порядку використовує ключ $A \in GL_k(Z_n)$. Повідомлення розбивається на *k*-грами. Кожна *k* -грама *X* заміщується *k* -грамою $E(X) = AX$.

В процесі дешифрування кожна *k*-грама X' криптотексту заміщується *k*-грамою $D(X') = A'X'$, де $A' = A^{-1}$ - дешифруючий ключ.

Афінний шифр k –го порядку. Ключами в даному випадку є $A \in GL_k(Z_n)$ і $S \in Z_n^k$. Повідомлення розбивається на *k* -грами. Кожна *k*-грама *X* заміщується *k*-грамою $E(X) = AX + S$.

Під час дешифрування кожна *k*-грама X' криптотексту заміщується *k*-грамою $D(X') = A'X' + S'$, де $A' = A^{-1}$ і $S' = -A'S$ - дешифруючий ключ.

Тема 10. Криптосистема RSA.

Алгоритм RSA запропонували в 1978 р. троє авторів: Р.Райвест (Rivest), А.Шамір (Shamir) і А.Адлеман (Adleman). Алгоритм одержав свою назву за першими буквами прізвищ його авторів. Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, що може працювати як у режимі шифрування даних, так і в режимі електронного цифрового підпису.

Надійність алгоритму ґрунтується на складності задач факторизації великих чисел та обчислення дискретних логарифмів.

У криптосистемі RSA відкритий ключ *K*, секретний ключ *k*, повідомлення *M* і криптограма *C* належать множині цілих чисел $Z_N = \{0,1,2,...,N-1\}$, де $N = P \cdot Q$ - модуль (*P і Q*- випадкові великі прості числа). Для забезпечення максимальної безпеки вибирають *P* і *Q* рівної довжини і зберігають у секреті.

Відкритий ключ *K* вибирають випадковим чином так, щоб виконувалися умови:

$$1 < K \le \varphi(N),\ НСД(K, \varphi(N)) = 1,\ \varphi(N) = (P-1)(Q-1),$$

де $\varphi(N)$ - функція Ейлера, *НСД* - найбільший спільний дільник.

Функція Ейлера $\varphi(N)$ вказує кількість додатних цілих чисел в інтервалі від 1 до *N*, що взаємно прості з *N*. Друга із зазначених вище умов означає, що відкритий ключ *K* і функція Ейлера $\varphi(N)$ повинні бути взаємно простими.

Далі, використовуючи розширений алгоритм Евкліда, обчислюють секретний ключ *k*, такий, що

$$k \cdot K \equiv 1 (\mathrm{mod}\ \varphi(N))$$

або

$$k = K^{-1} (\mathrm{mod}(P-1)(Q-1)).$$

Це нескладно здійснити, оскільки відома пара простих чисел (P,Q) і можна легко знайти $\varphi(N)$. Слід зазначити, що *k* і *N* повинні бути взаємно простими.

Відкритий ключ *K* використовують для шифрування даних, а секретний ключ *k* - для дешифрування.

Процедура шифрування визначає криптограму *C* через пару (відкритий ключ *K*, повідомлення *M*) у відповідності з наступною формулою:

$$C = E_K(M) = M^K (\text{mod } N).$$

Як алгоритм швидкого обчислення значення C використовують ряд послідовних піднесень до квадрату цілого *M* і множень на *M* приведенням за модулем *N*.

Зворотня операція, тобто визначення значення *M* за відомим значенням C, *K* і *N*, практично не здійсненна при $N \approx 2^{512}$.

Однак обернену задачу, тобто задачу дешифрування криптограми C, можна вирішити, використовуючи пару (секретний ключ *k*, криптограма C) за наступною формулою:

$$M = D_k(C) = C^k (\text{mod } N).$$

Таким чином, якщо криптограму

$$C = M^k (\text{mod } N)$$

піднести до степеня *k*, то в результаті відновлюється вихідний відкритий текст *M*, тому

$$(M^K)^k = M^{Kk} = M^{n\cdot\varphi(N)+1} \equiv M (\text{mod } N).$$

Таким чином, одержувач, що створює криптосистему, захищає два параметри: секретний ключ *k* і пару чисел (*P, Q*), добуток яких дає значення модуля *N*. З іншого боку, одержувач відкриває значення модуля *N* і відкритий ключ *K*.

Супротивнику відомі лише значення *K* і *N*. Якби він зміг розкласти число *N* на множники *P* і *Q* (задача факторизації), то він довідався б "таємний хід" - трійку чисел $\{P,Q,K\}$, обчислив значення функції Ейлера $\varphi(N) = (P-1)(Q-1)$ і визначив значення секретного ключа *k*.

Однак, як уже відзначалося, розкладання дуже великого *N* на множники не здійсненно за реальний час (за умови, що довжини обраних *P* і *Q* складають не менш 100 десяткових знаків).

Процедура шифрування і дешифрування в криптосистемі RSA.

Припустимо, що користувач *A* хоче передати користувачу *B* повідомлення в зашифрованому виді, використовуючи криптосистему RSA.

У такому випадку користувач *A* виступає в ролі відправника повідомлення, а користувач *B* - у ролі одержувача. Як відзначалося вище, криптосистему RSA повинен сформувати одержувач повідомлення, тобто користувач *B*. Розглянемо послідовність дій користувача *B* і користувача A.

1. Користувач *B* вибирає два довільних великих простих числа *P* і *Q*.

2. Користувач *B* обчислює значення модуля $N = P \cdot Q$.

3. Користувач *B* обчислює функцію Ейлера $\varphi(N) = (P-1)(Q-1)$ і вибирає випадковим чином значення відкритого ключа *K* з урахуванням виконання умов:

$$1 < K \le \varphi(N),\ ÍÑÄ\ (K, \varphi(N)) = 1.$$

4. Користувач *B* обчислює значення секретного ключа *k*, використовуючи розширений алгоритм Евкліда $k \equiv K^{-1} (\text{mod } \varphi(N))$.

5. Користувач *B* пересилає користувачу *A* пари чисел (*N*,*K*) по незахищеному каналі.

Якщо користувач *A* хоче передати користувачу *B* повідомлення *M,* він виконує наступні кроки.

6. Користувач *A* розбиває вихідний відкритий текст *M* на блоки, кожний з яких може бути представлений у вигляді числа

$$M_i = 0,1,2,...,N-1.$$

7. Користувач *A* шифрує текст, представлений у вигляді послідовності чисел *M,* за формулою

$$C_i = M_i^K \pmod N$$

і відправляє криптограму

$$C_1, C_2, C_3, ..., C_i, ...$$

користувачу *B.*

8. Користувач *B* розшифровує прийняту криптограму, використовуючи секретний ключ *k,* за формулою

$$M_i = C_i^k \pmod N$$

У результаті буде отримана послідовність чисел *M*, що формують собою вихідне повідомлення *M.* Щоб алгоритм RSA мав практичну цінність, необхідно мати можливість без істотних витрат генерувати великі прості числа, вміти оперативно обчислювати значення ключів *K* і *k.*

ПРИКЛАД Нехай p = 53 і *q* = 67. Тоді *n* = 3551 і φ(*n)* = 3432. Візьмемо e = 1021 — за допомогою розширеного алгоритму Евкліда легко перевірити, що НСД (1021,3432) = 1. Одночасно обчислюємо $d=1021^{-1} \bmod 3432 = 1237$. Ключі вибрано.

Відкритий ключ e = 1021 і n = 3551 оприлюднюємо. Тепер будь-хто може послати нам зашифроване повідомлення. Припустимо, один із ділових партнерів вирішив послати нам вказівку ПРОДАЙ. Спочатку він перетворює своє повідомлення у цифрову форму, замінюючи кожну літеру її двоцифровим десятковим номером в алфавіті: 1920 1805 0013. Видно, що з нашим модулем *n* цифрове повідомлення варто розбивати на блоки по 4 цифри, як це і зроблено. При шифруванні перший блок 1920 перетворюється у $1920^{1021} \bmod 3551 = 2393$. Таким же чином шифруються наступні два блоки, і в результаті виходить криптотекст 2393 17S8 2188.

Отримавши цей криптотекст, проводимо дешифрування піднесенням кожного блоку до степеня *d* = 1237 за модулем *n* = 3551. Можна переконатись, що $2393^{1237} \bmod 3551 = 1920$ і т.д.

Тема 11. Криптосистема Рабіна.

Генерування ключів. Вибирають два великі прості числа *p* і *q*. Обчислюють їх добуток *n* = *pq*. Покладають

Відкритий ключ: n .

Таємний ключ: p, q.

Шифрування відбувається блоками подібно до системи RSA, згідно з формулою

$$E\{M) = M^2 \bmod л.$$

Алгоритм дешифрування складніший, тому розглянемо його на прикладі.

Нехай таємний ключ вибрано так: *p*= 53 і q = 67. Тоді відкритим ключем буде *n* = 3551.

Розглянемо шифрування повідомлення ПРОДАЙ. Спочатку повідомлення записується у цифровій формі і розбивається на блоки по чотири цифри: 1920 1805 0013. Поший блок 1920 перетворюється у $1920^2 \bmod 3551 = 0462$. Подібно шифруються наступні два блоки, і в результаті виходить криптотекст: 0462 1758 0169.

Припустимо тепер, що ми отримали криптотекст 1497. Для шифрування слід з нього добути квадратні корені за модулем 3551. З цією метою добуваємо корені за простими модулями 53 і 67 із лишків 1497 mod 53 = 13 і 1497 mod 67 = 23, відповідно. Знаходимо $\sqrt{13} \bmod 53 = 15, 38$ і $\sqrt{23} \bmod 67 = 31, 36$. За допомогою алгоритму з Китайської теореми про остачі визначаємо чотири корені з 1497 за модулем 3551: (15,31) = 0969, (15,36) = 1711, (38,31) = 1840, (38,36) = 2582. Як зразу видно, лише другий корінь є числовим еквівалентом тексту в українській абетці, а саме повідомлення НІ.

Тема 12. Криптосистема Ель–Гамаля.

Схема шифрування Ель-Гамаля була запропонована в 1985 році і може використовуватися як в режимі шифрування даних, так і в режимі електронного цифрового підпису. Надійність алгоритму базується на складності обчислення дискретних логарифмів.

Даний алгоритм відноситься до класу ймовірнісних алгоритмів криптозахисту інформації. Особливість алгоритмів даного класу полягає у тому, що повідомленню M відповідає не один криптотекст C, а деяка родина криптотекстів C_M, причому кожен член $C \in C_M$ цієї родини вибирається з певною ймовірністю. Іншими словами, для кожного повідомлення M результат роботи алгоритму $E(M)$ є випадковою величиною, розподіленою на множині C_M. Щоб дешифрування було однозначним для будь-якої пари різних повідомлень C_1 та C_2 відповідні їм множини криптотекстів C_{M_1} та C_{M_2} не повинні перетинатися. Більше того, має бути ефективний алгоритм дешифрування, який використовує таємний ключ і за будь-яким $C \in C_M$ визначає M.

Таку криптосистему вважають надійною, якщо для будь-якої пари повідомлень M_1 та M_2 однакової довжини l, випадкові величини $E(M_1)$ та $E(M_2)$ неможливо відрізнити одну від одної ніяким ймовірнісним поліноміальним алгоритмом із ймовірністю, вищою за $1/l$. Тобто лише з незначною ймовірністю за двома криптотекстами можна визначити, відповідають вони одному й тому ж повідомленню чи різним. Такий рівень надійності, в принципі, недосяжний для детермінованих систем, у випадку яких пара різних криптотекстів C_1 та C_2 завжди відповідає різним відкритим текстам M_1 та M_2.

Розглянемо роботу алгоритму Ель-Гамаля. Для того, щоб генерувати пару ключів (відкритий ключ та секретний ключ), спочатку вибирають деяке велике просте число P і велике ціле число G, причому $G < P$. Числа P і G можуть бути поширені серед групи користувачів.

Потім вибирають випадкове ціле число X, причому $X < P$. Число X являється секретним ключем і повинне зберігатися в секреті.

Далі обчислюють $Y = G^X \bmod P$. Число Y є відкритим ключем.

Для того, щоб зашифрувати повідомлення *M*, вибирають випадкове ціле число K $(1 \le K < P-1)$ таке, що числа K і $(P\text{-}1)$ є взаємно простими. Потім обчислюють числа

$$a = G^K \bmod P,$$
$$b = Y^K M \bmod P.$$

Пара чисел (a,b) є шифротекстом. Слід відмітити, що довжина шифротексту вдвічі більша довжини вихідного відкритого тексту *M*. Для того щоб дешифрувати шифротекст (a,b) обчислюють:

$$M = b / a^X \bmod P.$$

Приклад. Нехай p=23, g=5, a=6. Обчислюємо $h=5^6 \bmod 23=8$. Відкритий і таємний ключ сформовано.

Припустимо, що шифрується числова інформація, і потрібно зашифрувати повідомлення $M = 7$. Нехай вибрано r = 10. Тоді $c_1 = 5^{10} \bmod 23 = 9$ і $c_2 = (7 \cdot 8^{10}) \bmod 23 = 21$. Отримуємо криптотекст $C = (9,21)$. Що стосується дешифрування, то легко перевірити, що справді $D(9,21) = 21 \cdot (9^6)^{-1} \bmod 23 = 7$.

Тема 13. Електронний цифровий підпис.

Першою та найвідомішою у світі системою ЕЦП стала система RSA, математична схема якої була розроблена у 1977 році у США.

В даній системі спочатку необхідно визначити пару ключів (відкритий та таємний). Для цього відправник електронного документу обчислює два великих простих числа P та Q, а потім знаходить їх добуток $N = P \cdot Q$ і значення функції $\varphi(N) = (P-1) \cdot (Q-1)$. Після цього відправник обчислює число E з умов $E \le \varphi(N)$ та $НСД(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E \cdot D \equiv 1 (\bmod \varphi(N))$.

Пара чисел (E,N) є відкритим ключем. Цю пару чисел відправник передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається у відправника як ключ для підпису.

Загальна схема формування та перевірки цифрового підпису показана на рисунку 13.1.

Припустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискається за допомогою хеш-функції $h(M)$ в ціле число $m = h(M)$. Після цього обчислюють цифровий підпис S під електронним документом M, використовуючи хеш-значення m та таємний ключ D:

$$S = m^D (\bmod N).$$

Пара (M,S) передається партнеру-отримувачу як електронний документ M, підписаний електронним підписом S, причому він сформований власником таємного ключа D.

Після отримання пари (M,S) одержувач обчислює хеш-значення повідомлення M двома способами. Перш за все, він встановлює значення m, застосовуючи криптографічні перетворення підпису S з використанням відкритого ключа E:

$$m = S^E (\bmod N).$$

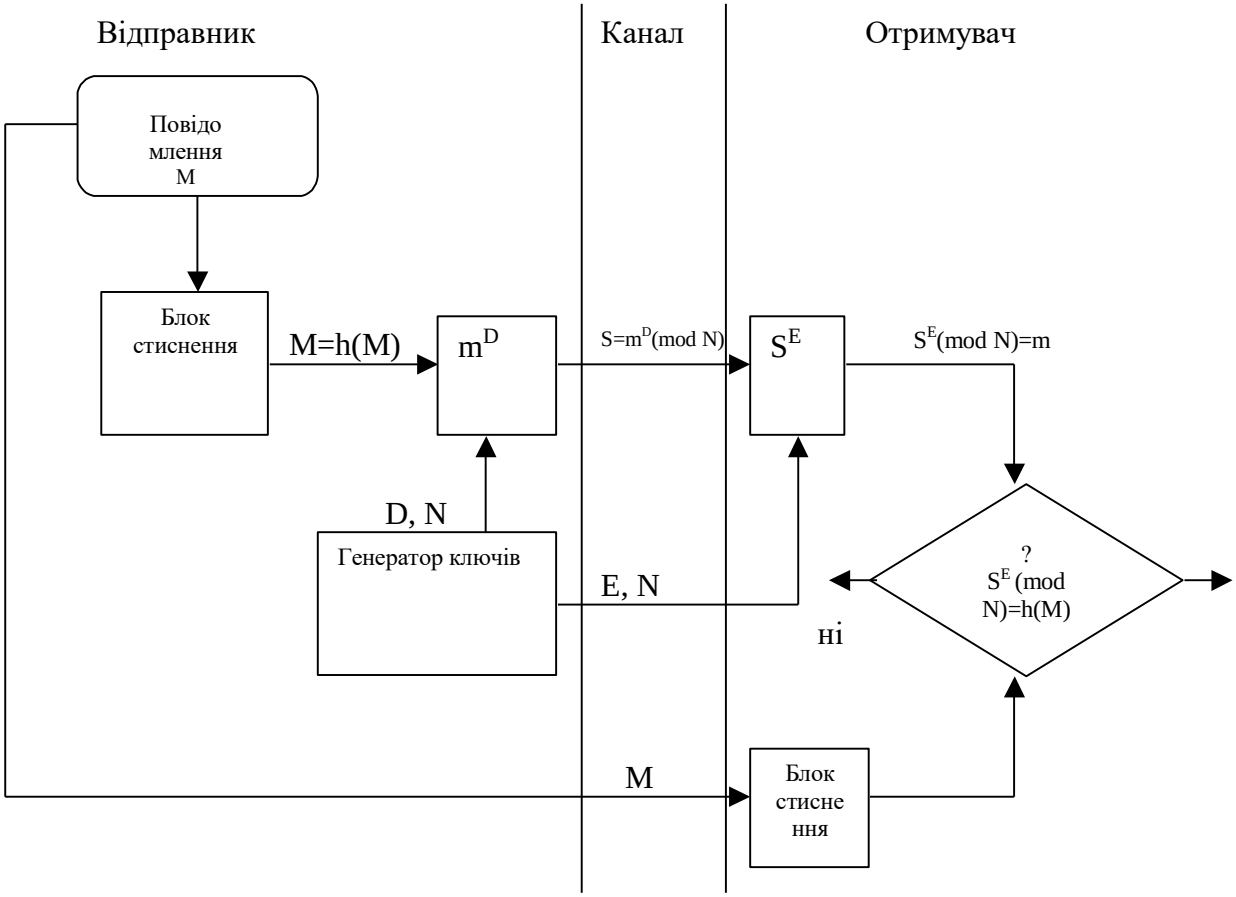


Рисунок 13.1 – Загальна схема цифрового підпису RSA.

Крім того, він знаходить результат хешування прийнятого повідомлення M за допомогою такої ж хеш-фукції $h(M)$:

$$m = h(M).$$

Якщо зберігається рівність обчислених значень, то отримувач признає пару (M, S) справжньою.

Очевидно, що дана схема дозволяє захиститись від декількох видів порушень:

- відправник не може відмовитись від свого повідомлення, якщо отримувач визнає, що таємний ключ відомий лише йому;
- порушник без знання таємного ключа не може ні сформувати, ні зробити осмислену зміну повідомлення, переданого по лінії зв'язку;
- дана схема дозволяє при вирішенні багатьох конфліктних ситуацій обходитись без посередників.

Інколи немає необхідності зашифровувати повідомлення, що передається, але потрібно його скріпити електронним підписом. В такому випадку текст шифрується закритим ключем відправника і отриманий ланцюжок символів прикріплюється до документа. Отримувач за допомогою відкритого ключа відправника розшифровує підпис і звіряє його з текстом.

Підпис у системі Ель–Гамаля. Для генерування ключів у системі Ель–Гамаля вибирають велике просте p, а також просте число g, $1 < g < p–1$. Числа p і g не є таємницею і перебувають в загальному користуванні. Кожен абонент вибирає собі випадкове число a у проміжку від 1 до p–1, і обчислює $h = g^a \bmod p$.

Відкритий клюй: p, g, h. Таємний клюй: a.

Підписування. Абонент А виробляє свій підпис *S* на повідомленні *M* таким чином:

- вибирає випадкове число 1<r<p-1;
- обчислює $s_1 = g^r \bmod p$;
- обчислює $r' = r^{-1} \bmod (p - 1)$;
- обчислює $s_2 = (M — as_1)r' \bmod (p — 1)$;
- покладає $S = (s_1, s_2)$.

Підтвердження підпису.

- Абонент Б перевіряє, чи $g^M = h^{S1}s_1^{S2} (\text{mod} p)$.

Алгоритм електронного цифрового підпису DSA. Він запропонований у 1991 році.

Генерування ключів. Вибирають велике просте число p таке, що p–1 має досить великий простий дільник *q*. Стандарт вимагає, щоб $2^{512}<p<2^{1024}$ і $q > 2^{160}$. Далі вибирають довільний елемент *h* порядку *q*. Параметри *p, q, h* не становлять таємниці і є спільними для всіх абонентів мережі.

Абонент А вибирає випадкове число *a* в діапазоні від 0 до $q – 1$ і обчислює число $b = h^a \text{ mod} p$. Його ключі формуються так.

Відкритий КЛЮЧ: b таке, що $b = h^a \text{ mod} p$.

Таємний ключ: a.

Підписування. Алгоритм підписування використовує вкорочуючу функцію f з довжиною вкорочення 160 бітів. Щоб виробити свій підпис *S* для повідомлення M, абонент А:

- вибирає випадкове число r в діапазоні від 0 до q-1;
- обчислює $r' = r^{-1} \bmod q$;
- обчислює $s_1 = (h^r \bmod p) \bmod q$;
- обчислює $s_2 = (r'(f(M) + as_1)) \bmod q$;
- формує підпис $S = (s_1, s_2)$.

Підтвердження підпису. Отримавши повідомлення *M* із підписом $S = (s_1,.s_2)$, абонент Б:

- обчислює $s' = S_2^{-1} \bmod q$;
- обчислює $u_1 = (f(M)s') \bmod q$;
- обчислює $u_2 = (s_1 \cdot s') \bmod q$;
- обчислює $t = (h^{U1} \cdot b^{U2} \bmod p) \bmod q$;
- перевіряє рівність $t = s_1$.

Розділ 4. Хеш-функції.

Тема 14. Шифрування з паролем.

Одним з найпростіших способів ідентифікації є парольна ідентифікація, яка здійснюється шляхом порівняння введеного імені та паролю, із тими які зберігаються у файлі паролів (рисунок 14.1).

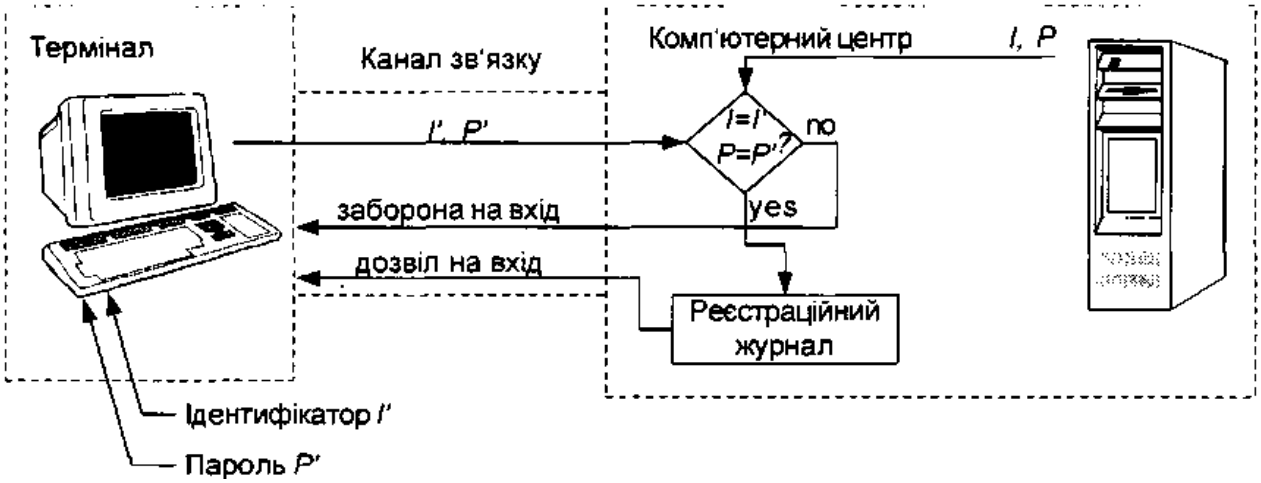


Рисунок 14.1 Класичний спосіб парольної ідентифікації

Проте такий підхід, не захищає КС від програмних та апаратних засобів сканування клавіатури та ліній передачі, а отже може призвести до витоку конфіденційної інформації. Тому, як правило, в сучасних КС пароль не передається в явній формі по лініях передачі, а натомість в якості паролю використовують якесь його відображення. У якості такого відображення використовуються важкооборотні однонапрямлені функції, застосування яких гарантує неможливість розкриття пароля за його відображенням за розумний час.

Як показує статистика більшість користувачів відчувають певний дискомфорт при створенні паролів. З однієї сторони занадто простий пароль використовувати небезпечно з іншої – складний пароль важко запам’ятати. Користувачі, які мають слабке уявлення про функціонування систем захисту інформації допускаються серйозних помилок при створенні паролів.

Надто простий пароль:

a. використання дати народження в якості пароля;

b. використання прізвища, імені, по батькові або ж їх комбінацій в якості пароля;

c. використання типових паролів таких як "qwerty", "123456", "password", "******" (шість зірочок) тощо;

d.використання назви обладнання на робочому столі в якості паролів.

Способи формування та реєстрації паролів. «Слабкі» паролі.

Правила формування пароля:

- пароль не може містити ім'я облікового запису користувача або будь-яку його частину.
- пароль повинен складатися не менше ніж з 8 символів.
- у паролі повинні бути присутніми символи трьох категорій з числа наступних чотирьох:

а) великі літери англійського алфавіту від A до Z;

б) малі літери англійського алфавіту від a до z;

в) десяткові цифри (від 0 до 9);

г) символи, які не належать алфавітно-цифровому набору (наприклад,!, $, #,%).

- забороняється використовувати в якості пароля ім'я входу в систему, прості паролі типу «123», «111», «qwerty» і їм подібні, а так само імена і дати народження своєї особистості і своїх родичів, клички домашніх тварин, номери автомобілів, телефонів та інші паролі, які можна вгадати, ґрунтуючись на інформації про користувача;
- забороняється використовувати в якості пароля один і той же символ, що повторюється або повторювану комбінацію з декількох символів;
- забороняється використовувати в якості пароля комбінацію символів, що набираються в закономірному порядку на клавіатурі (наприклад, 1234567 тощо);
- забороняється вибирати паролі, які вже використовувалися раніше.

Правила введення пароля:

- введення пароля повинен здійснюватися з урахуванням регістра, в якому пароль було встановлено;
- під час введення паролів необхідно виключити можливість його підглядання сторонніми особами або технічними засобами (відеокамери та ін.).

Правила зберігання пароля:

- забороняється записувати паролі на папері, у файлі, електронної записнику та інших носіях інформації, в тому числі на предметах;
- забороняється повідомляти іншим користувачам особистий пароль і реєструвати їх в системі під своїм паролем.

Особи, які використовують паролювання, зобов'язані:

- чітко знати і суворо виконувати вимоги цієї інструкції та інших керівних документів з паролювання.
- своєчасно повідомляти Адміністратору інформаційної безпеки про втрату, компрометації, несанкціонованому зміну паролів і несанкціонованому зміну термінів дії паролів.

Розглянемо основні параметри паролів, які визначають стійкість парольної системи до атаки ”brute-force” – атаки, що базується на звичайному переборі усіх можливих варіантів.

– потужність алфавіту паролів, тобто множина знаків, які можуть використовува- тися при введенні паролю; A

– довжина паролю; L

– потужність простору паролів, тобто множина усіх можливих паролів у системі; S

– швидкість підбору паролю; V

– термін дію паролю; T

– ймовірність підбору паролю на протязі терміну його дії. P

Для обчислення потужності простору паролів у залежності від алфавіту паролів та їх довжини використовують співвідношення:

$$S = A^L$$

Ймовірність підбору паролю на протязі терміну його життя обчислюють за формулою:

$$P = \frac{VT}{S}$$

Дві основні характеристики паролю – кількість його символів (довжина) і кількість варіантів символу у кожній позиції (алфавіт).

Очевидно, що із збільшенням алфавіту (основи степеневої функції) ймовірність підбору паролю знижується, але ще сильніше вона буде знижуватися із збільшенням довжини паролю (показника степеневої функції).

Тема 15. Поняття хеш-функції

Хеш-функція — функція, що перетворює вхідні дані будь-якого (як правило, великого) розміру в дані фіксованого розміру. Хешування (іноді гешування, англ. hashing) — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями або функціями згортки, а їх результати називають хешем, хеш-кодом або дайджестом повідомлення (англ. message digest).

Хешування застосовується для порівняння даних: якщо у двох масивів хеш-коди різні, масиви гарантовано розрізняються; якщо однакові - масиви, швидше за все, однакові. У загальному випадку однозначної відповідності між вихідними даними і хеш-кодом немає в силу того, що кількість значень хеш-функцій менше, ніж варіантів вхідного масиву; існує безліч масивів, що дають однакові хеш-коди - так звані колізії. Ймовірність виникнення колізій відіграє важливу роль в оцінці якості хеш-функцій. Існує безліч алгоритмів хешування з різними характеристиками (розрядність, обчислювальна складність, крипостійкість тощо). Вибір тієї чи іншої хеш-функції визначається специфікою розв'язуваної задачі. Найпростішими прикладами хеш-функцій можуть служити контрольна сума або CRC.

Криптографічна хеш-функція повинна забезпечувати:

- стійкість до колізій (два різні набори даних повинні мати різні результати перетворення);
- необоротність (неможливість обчислити вхідні дані за результатом перетворення).

Прискорення пошуку даних

Хеш-функції також використовуються в деяких структурах даних — хеш-таблицях і декартових деревах. Вимоги до хеш-функції в цьому разі інші:

- добра перемішуваність даних;
- швидкий алгоритм обчислення.

Наприклад, під час запису текстових полів в базі даних може розраховуватися їх хеш-код і дані можуть поміщатися в розділ, що відповідає цьому хеш-коду. Тоді при пошуку даних треба буде спочатку обчислити хеш-код тексту і відразу стане відомо, в якому розділі їх треба шукати, тобто, шукати

треба буде не по всій базі, а тільки по одному її розділу (це сильно прискорює пошук).

Побутовим аналогом хешування в даному випадку може служити розміщення слів у словнику за алфавітом. Перша буква слова є його хеш-кодом, і при пошуку ми переглядаємо не весь словник, а тільки потрібну букву.

Хеш-таблиця — структура даних, що реалізує інтерфейс асоціативного масиву, а саме, вона дозволяє зберігати пари (ключ, значення) і здійснювати три операції: операцію додавання нової пари, операцію пошуку і операцію видалення за ключем.

Існує два основних варіанта хеш-таблиць: з ланцюжками і з відкритою адресацією. Хеш-таблиця містить в собі деякий масив ***H***, елементи якого є пари (хеш-таблиця з відкритою адресацією) або списки пар (хеш-таблиця з ланцюжками).

Виконання операцій в хеш-таблиці починається з обчислення хеш-функції від ключа. Отримане хеш-значення i=hash(key) відіграє роль індексу в масиві ***H***. Після цього операція (додавання, видалення, пошук) перенаправляється об'єктові, який зберігається у відповідній комірці масиву ***H*[*i*]**.

Ситуація, коли для різних ключів отримується одне й те саме хеш-значення, називається колізією. Такі події непоодинокі — наприклад, при додаванні в хеш-таблицю розміром 365 комірок, усього лише 23-х елементів, ймовірність колізії вже перевищує 50 відсотків (якщо кожний елемент може з однаковою ймовірністю потрапити в будь-яку комірку) — див. парадокс днів народження. Через це механізм розв'язання колізій — важлива складова будь-якої хеш-таблиці.

В деяких особливих випадках вдається взагалі уникнути колізій. Наприклад, якщо всі ключі елементів відомі заздалегідь (або дуже рідко змінюються), тоді для них можна знайти деяку *досконалу хеш-функцію*, яка розподілить їх за комірками хеш-таблиці без колізій. Хеш-таблиці, які використовують подібні хеш-функції, не потребують механізму розв'язання колізій, і називаються хеш-таблицями з *прямою адресацією*.

Тема 16. Генерування та види хеш-функцій

Хеш-функція — функція, що перетворює вхідні дані будь-якого (як правило, великого) розміру в дані фіксованого розміру. Хешування (іноді гешування, англ. hashing) — перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями або функціями згортки, а їх результати називають хешем, хеш-кодом або дайджестом повідомлення (англ. message digest).

Хешування застосовується для порівняння даних: якщо у двох масивів хеш-коди різні, масиви гарантовано розрізняються; якщо однакові - масиви, швидше за все, однакові. У загальному випадку однозначної відповідності між вихідними даними і хеш-кодом немає в силу того, що кількість значень хеш-функцій менше, ніж варіантів вхідного масиву; існує безліч масивів, що дають однакові хеш-коди - так звані колізії. Ймовірність виникнення колізій відіграє важливу роль в оцінці якості хеш-функцій. Існує безліч алгоритмів хешування з різними характеристиками (розрядність, обчислювальна складність, крипостійкість тощо). Вибір тієї чи іншої хеш-функції визначається специфікою розв'язуваної задачі.

Найпростішими прикладами хеш-функцій можуть служити контрольна сума або CRC.

Криптографічна хеш-функція повинна забезпечувати:

- стійкість до колізій (два різні набори даних повинні мати різні результати перетворення);
- необоротність (неможливість обчислити вхідні дані за результатом перетворення).

Прискорення пошуку даних

Хеш-функції також використовуються в деяких структурах даних — хеш-таблицях і декартових деревах. Вимоги до хеш-функції в цьому разі інші:

- добра перемішуваність даних;
- швидкий алгоритм обчислення.

Наприклад, під час запису текстових полів в базі даних може розраховуватися їх хеш-код і дані можуть поміщатися в розділ, що відповідає цьому хеш-коду. Тоді при пошуку даних треба буде спочатку обчислити хеш-код тексту і відразу стане відомо, в якому розділі їх треба шукати, тобто, шукати треба буде не по всій базі, а тільки по одному її розділу (це сильно прискорює пошук).

Побутовим аналогом хешування в даному випадку може служити розміщення слів у словнику за алфавітом. Перша буква слова є його хеш-кодом, і при пошуку ми переглядаємо не весь словник, а тільки потрібну букву.

Хеш-таблиця — структура даних, що реалізує інтерфейс асоціативного масиву, а саме, вона дозволяє зберігати пари (ключ, значення) і здійснювати три операції: операцію додавання нової пари, операцію пошуку і операцію видалення за ключем.

Існує два основних варіанта хеш-таблиць: з ланцюжками і з відкритою адресацією. Хеш-таблиця містить в собі деякий масив $\boldsymbol{H}$, елементи якого є пари (хеш-таблиця з відкритою адресацією) або списки пар (хеш-таблиця з ланцюжками).

Виконання операцій в хеш-таблиці починається з обчислення хеш-функції від ключа. Отримане хеш-значення $i=hash(key)$ відіграє роль індексу в масиві $\boldsymbol{H}$. Після цього операція (додавання, видалення, пошук) перенаправляється об'єктові, який зберігається у відповідній комірці масиву $\boldsymbol{H[i]}$.

Ситуація, коли для різних ключів отримується одне й те саме хеш-значення, називається колізією. Такі події непоодинокі — наприклад, при додаванні в хеш-таблицю розміром 365 комірок, усього лише 23-х елементів, ймовірність колізії вже перевищує 50 відсотків (якщо кожний елемент може з однаковою ймовірністю потрапити в будь-яку комірку) — див. парадокс днів народження. Через це механізм розв'язання колізій — важлива складова будь-якої хеш-таблиці.

В деяких особливих випадках вдається взагалі уникнути колізій. Наприклад, якщо всі ключі елементів відомі заздалегідь (або дуже рідко змінюються), тоді для них можна знайти деяку *досконалу хеш-функцію*, яка розподілить їх за комірками хеш-таблиці без колізій. Хеш-таблиці, які використовують подібні хеш-функції, не потребують механізму розв'язання колізій, і називаються хеш-таблицями з *прямою адресацією*.

Список алгоритмів: HAVAL, MD2, MD4, MD5, N-Hash, RIPEMD-160, SHA, Snefru, Tiger, Whirlpool

3. Алгоритми MD-5, SHA, ГОСТ 34.11-94

MD5 *(Message Digest 5)* — 128-бітний алгоритм хешування, розроблений професором Рональдом Л. Рівестом в 1991 році. Призначений для створення «відбитків» або «дайджестів» повідомлень довільної довжини. Прийшов на зміну MD4, що був недосконалим. Описаний в RFC 1321.

У 2004 році китайські дослідники Сяоюнь Ван (Xiaoyun Wang), Денгуо Фен (Dengguo Feng), Сюецзя Лай (Xuejia Lai) і Хонбо Ю (Hongbo Yu) повідомили про знаходження ними вразливості в алгоритмі, що дозволяє за невеликий час (1 годину на кластері IBM p690) знаходити колізії хеш-функцій. На жаль, автори так і не відкрили свій секрет широкій публіці.

У 2006 році чеський дослідник Властимил Клима опублікував алгоритм, що дозволяє знаходити колізії на звичайному комп'ютері з довільним початковим вектором (A,B,C,D), за допомогою методу, що був названий: "тунелювання".

Алгоритм MD5

Початковий етап підготовки

- Вхідні дані вирівнюються так, щоб їхній розмір можна було порівняти з 448 по модулю з 512. Спочатку дописують одиничний біт (навіть якщо довжина порівняна з 448), далі необхідна кількість нульових бітів .
- Дописування 64-бітного представлення довжини даних по вирівнюванню. Якщо довжина перевищує 2^{64}-1, то дописують молодші біти.

Допоміжні таблиці та функції

- Ініціалізуть 4 змінних розміром по 32 біта:
  - A = 01 23 45 67;
  - B = 89 AB CD EF;
  - C = FE DC BA 98;
  - D = 76 54 32 10.

Вирівнювані дані розбиваються на блоки по 32 біта, і кожен проходить 4 раунда з 16 операторів. Всі оператори однотипні і мають вигляд: [abcd k s i], визначений як a=b+((a+Fun(b, c, d)+X[k]+T<i>)<<<s) , де X - блок даних, а T[1..64] - 64х елементна таблиця побудована наступним чином: T[i]=int(4294967296*|sin(i)|), s - циклічний зсув вліво на s біт отриманого 32-бітного аргументу.

- В першому раунді Fun F(X, Y, Z) = XY v (not X)Z
- В другому раунді Fun G(X, Y, Z) = XZ v (not Z)Y.
- В третьому раунді Fun H(X, Y, Z) = X xor Y xor Z.
- В четвертому раунді Fun I(X, Y, Z) = Y xor (X v (not Z)).

Циклічна процедура обчислення

Саме обчислення проходить наступним чином:

- Зберігаються значення A, B, C і D, що залишились після операцій з попередніми блоками(або їх початкові значення якщо блок перший)

AA = A

BB = B

CC = C

DD = D

Раунд 1

/*[abcd k s i] a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]

[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]

[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]

[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]

Раунд 2

/*[abcd k s i] a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]

[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]

[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]

[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]

Раунд 3

/*[abcd k s i] a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]

[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]

[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]

[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

Раунд 4

/*[abcd k s i] a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). */

[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]

[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]

[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]

[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

Проміжний результат

Виконати наступні операції

A = AA + A

B = BB + B

C = CC + C

D = DD + D

Після цього перевірити, чи є ще блоки, якщо є, то повторюють циклічну процедуру обчислення для наступного 32-х бітового блоку.

Результат

Після обчислення для всіх блоків даних, отримуємо кінцевий хеш у регістрах A B C D. Якщо вивести слова у зворотному порядку DCBA, то отримаємо MD5 хеш.

Тема 17. Генерація ЕЦП на основі хеш-функцій.

Аутентифікація повідомлень являє собою процедуру перевірки того, що отримані повідомлення прийшли від вказаного джерела і не були змінені на шляху прямування.

У аутентифікації повідомлень можуть бути виділені два основних рівня:

1. На нижчому рівні повинна виконуватися деяка функція, що породжує аутентифікатор (посвідчення, яке використовується для підтвердження автентичності суб'єкта).

2. Аутентифікатор потім використовується як примітив у протоколі аутентифікації вищого рівня, що дає одержувачу повідомлення можливість перевірити достовірність повідомлення.

Функції, які можуть служити для створення аутентифікатора, можна розділити на три класи:

1. Шифрування повідомлення. В якості аутентифікатора використовується шифрований текст всього повідомлення.

2. Обчислення коду автентичності повідомлення. Як аутентифікатора виступає значення фіксованої довжини, що генерується деякої відкритої функцією повідомлення і секретним ключем.

3. Обчислення функції хешування (хеш-код). В якості аутентифікатора використовується значення фіксованої довжини, що генерується деякої відкритої функцією, противопоставляющей будь-якого повідомлення довільної довжини значення фіксованої довжини, зване значенням хеш-функції.

Електронний підпис (ЕП) - інформація в електронній формі, приєднана до іншої інформації в електронній формі (електронний документ) або іншим чином пов'язана з такою інформацією. Використовується для визначення особи, яка підписала інформацію (електронний документ) [1].

По своїй істоті електронний підпис являє собою реквізит електронного документа, що дозволяє встановити відсутність спотворення інформації в електронному документі з моменту формування ЕП і перевірити приналежність підпису власнику сертифіката ключа ЕП. Значення реквізиту виходить в результаті криптографічного перетворення інформації з використанням *закритого ключа ЕП.*

Електронний підпис призначена для ідентифікації особи, яка підписала електронний документ і є повноцінною заміною (аналогом) власноручного підпису у випадках, передбачених законом.

Використання електронного підпису дозволяє здійснити:

- Контроль цілісності переданого документа: при будь-якому випадковому або навмисному зміні документа підпис стане недійсним, тому що обчислена вона на підставі вихідного стану документа і відповідає лише йому.
- Захист від змін (підроблення) документа: гарантія виявлення підробки при контролі цілісності робить підроблення недоцільним у більшості випадків.
- Неможливість відмови від авторства. Так як створити коректну підпис можна, лише знаючи закритий ключ, а він повинен бути відомим тільки власнику, то власник не може відмовитися від свого підпису під документом.
- Доказове підтвердження авторства документа: Так як створити коректну підпис можна, лише знаючи закритий ключ, а він повинен бути відомим тільки власнику, то власник пари ключів може довести своє авторство підпису під документом. Залежно від деталей визначення документа можуть бути підписані такі поля, як "автор", "внесені зміни", "мітка часу" і т. д.

DSA *(Digital Signature Algorithm)* - алгоритм з використанням відкритого ключа для створення електронного підпису, але не для шифрування (на відміну від RSA і схеми Ель-Гамаля). Підпис створюється таємно, але може бути публічно перевірена. Це означає, що тільки один суб'єкт може створити підпис

повідомлення, але будь-хто може перевірити її коректність. Алгоритм заснований на обчислювальній складності узяття логарифмів в кінцевих полях.

Для побудови системи цифрового підпису потрібно виконати наступні кроки:

1. Вибір криптографічного хеш-функції H (x).
2. Вибір великого простого числа *q*, розмірність якого *N* в бітах збігається з розмірністю в бітах значень хеш-функції H (x).
3. Вибір простого числа *p*, такого, що *(p-1)* ділиться на *q*. Бітова довжина *p* позначається *L*.
4. Вибір числа *g* такого, що його мультиплікативний порядок по модулю *p* дорівнює *q*.

Як згадано вище, а також у DSS (Digital Signature Standard), першочерговим параметром схеми цифрового підпису є використовувана криптографічна хеш-функція, необхідна для перетворення тексту повідомлення в число, яке власне і буде підписана. Важливою характеристикою цієї функції є бітова довжина вихідний послідовності, що позначається далі *N* (160 для функції SHA-1). У першій версії стандарту DSS рекомендована функція SHA-1 і, відповідно, бітова довжина підписуваного числа 160 біт. Зараз SHA-1 вже не є достатньо безпечною. У стандарті вказані наступні можливі пари значень чисел *L* і *N:*

1. L = 1024, N = 160
2. L = 2048, N = 224
3. L = 2048, N = 256
4. L = 3072, N = 256

У відповідності з цим рекомендовані хеш-функції сімейства SHA-2. Урядові організації повинні використовувати один з цих варіантів, але всі інші вільні вибирати. Проектує систему може вибрати будь-яку хеш-функцію. Тому далі не буде загострюватися увагу на використанні конкретної хеш-функції. Стійкість криптосистеми на основі DSA не перевершує стійкість використовуваної хеш-функції і стійкість пари (L, N), чия стійкість не більше стійкості кожного з чисел окремо. Раніше рекомендувалася довжина *p* L = 1024 біта. В даний момент для систем, які повинні бути стійкими до 2010 (2030) року, рекомендується довжина в 2048 (3072) біти.

Реалізація алгоритму

Оскільки NIST зробив алгоритм вільним до використання, то алгоритм можна вільно реалізовувати програмним, апаратним або будь-яким іншим чином. NIST розробив програму перевірки відповідності реалізації алгоритму стандарту. Реалізації систем цифрового підпису повинні використовувати криптографічні алгоритми, алгоритми генерації криптографічних ключів та способи розподілу ключів, безпека яких так чи інакше підтверджена. До таких алгоритмам і методам ставляться алгоритми та методи:

1. описані в документах FIPS (Federal Information Processing Standard)
2. прийняті в рекомендаціях FIPS або NIST
3. зазначені в списку функцій з підтвердженою безпекою в документі FIPS 140-2.

При формуванні електронного підпису на практиці часто підписується не саме повідомлення, а його хеш-образ.

У процесі цифрового підписання система ЕЦП обробляє вихідне повідомлення (прообраз) криптографічно стійким однобічним хеш-алгоритмом. Ця операція призводить до генерації *дайджесту повідомлення.*

Потім система зашифровує отриманий дайджест закритим ключем відправника, створюючи «електронний підпис», і прикріплює її до прообразу. Після отримання повідомлення, адресат за допомогою системи ЕЦП заново обчислює дайджест підписаних даних, розшифровує ЕЦП відкритим ключем відправника, тим самим звіряючи, відповідно, цілісність даних та їх джерело; якщо обчислений адресатом і отриманий з повідомленням дайджести збігаються, значить інформація після підписання не була змінена.

У системах ЕЦП можуть використовуватися різні хеш-функції, але вони повинні задовольняти ряду умов:

1. *хеш-функція* повинна бути чутлива до всіляких змін в тексті, таким як вставки, викиди, перестановки тощо;
2. хеш-функція повинна мати властивість незворотності, то є завдання підбору документа , Який володів би необхідним значенням хеш-функції, повинна бути обчислювально нерозв'язна;
3. ймовірність колізій, тобо того, що значення хеш-функцій двох різних документів (незалежно від їх довжин) співпадуть, повинна бути мізерно мала.

Розділ 5. Еліптична та квантова криптографія, стеганографія.

Тема 18. Поняття еліптичної криптографії

ЕК E над полем $E(F_p)$ описується наступним рівнянням:

$$y^2 \equiv x^3 + ax + b(\text{mod } p), \quad p \in F_p, \qquad (18.1)$$

$$\Delta = -(4a^3 + 27b^2) \neq 0, \qquad a, b \in F_p.$$

де $(x, y) \in E(F_p) \cup O$, O - нескінченно віддалена точка.

Найважливішим аспектом теорії ЕК є введення операції додавання точок P= (x_1, y_1) і Q = (x_2, y_2) як основи для побудови абелевої групи. Їх сумою буде третя точка R = P + Q = (x_3, y_3), яка належить ЕК. Координати цієї точки обчислюються з наступної системи рівнянь:

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, P \neq Q; \\ y_3 = -y_1 - \lambda(x_3 - x_1), \lambda = \dfrac{y_2 - y_1}{x_2 - x_1} \end{cases} \qquad (18.2)$$

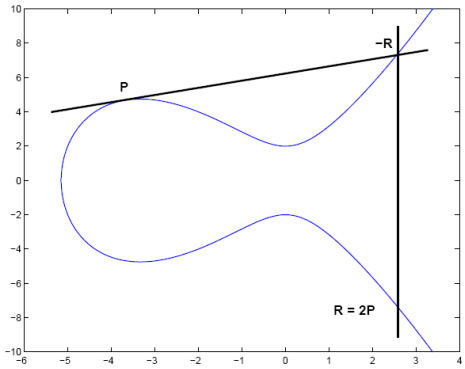
де

$$\lambda = \frac{-y_3 - y_1}{x_3 - x_1}. \qquad (18.3)$$

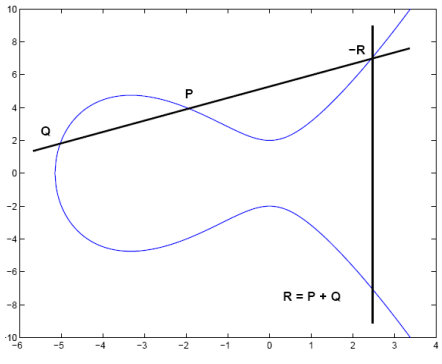
Якщо P=Q, R=2P=(x_3,y_3), тоді координати точки R=(x_3,y_3) знаходиться згідно співвідношення:

$$\begin{cases} x_3 = \nu^2 - 2x_1, P = Q; \\ y_3 = -y_1 - \nu(x_3 - x_1), \nu = \dfrac{3x_1^2 + a}{2y_1} \cdot \end{cases} \qquad (18.4)$$

Графічне представлення додавання двох точок та знаходження кратних точок на ЕК зображено на рис.18.1.



а



б

Рисунок 18.1 - Геометрична інтерпретація подвоєння точки P (а) і додавання точок P и Q (б)

На сьогодні ЕК застосовують для реалізації різноманітних класів систем захисту інформації, зокрема їх можна використовувати для побудови симетричних, асиметричних криптосистем та систем електронного цифрового підпису (ЕЦП). На рис. 18.2 зображено класифікацію сучасних методів

криптографії, що базуються на застосуванні еліптичних кривих. Їх аналіз показує, що математичний апарат еліптичних кривих можна застосовувати:

1) в асиметричних системах захисту інформації;
2) в симетричних криптосистемах;
3) для реалізації електоронно цифрового підпису;
4) для електронних платежів;
5) генератори для побудови псевдовипадкових послідовностей.

Слід зауважити, що стосовно симетричних криптосистем, в літературі показано лише теоретичну можливість побудови засобів такого класу, щодо ж практичної реалізації необхідно відзначити, що продуктивність таких систем є нижчою в порівнянні з традиційними.

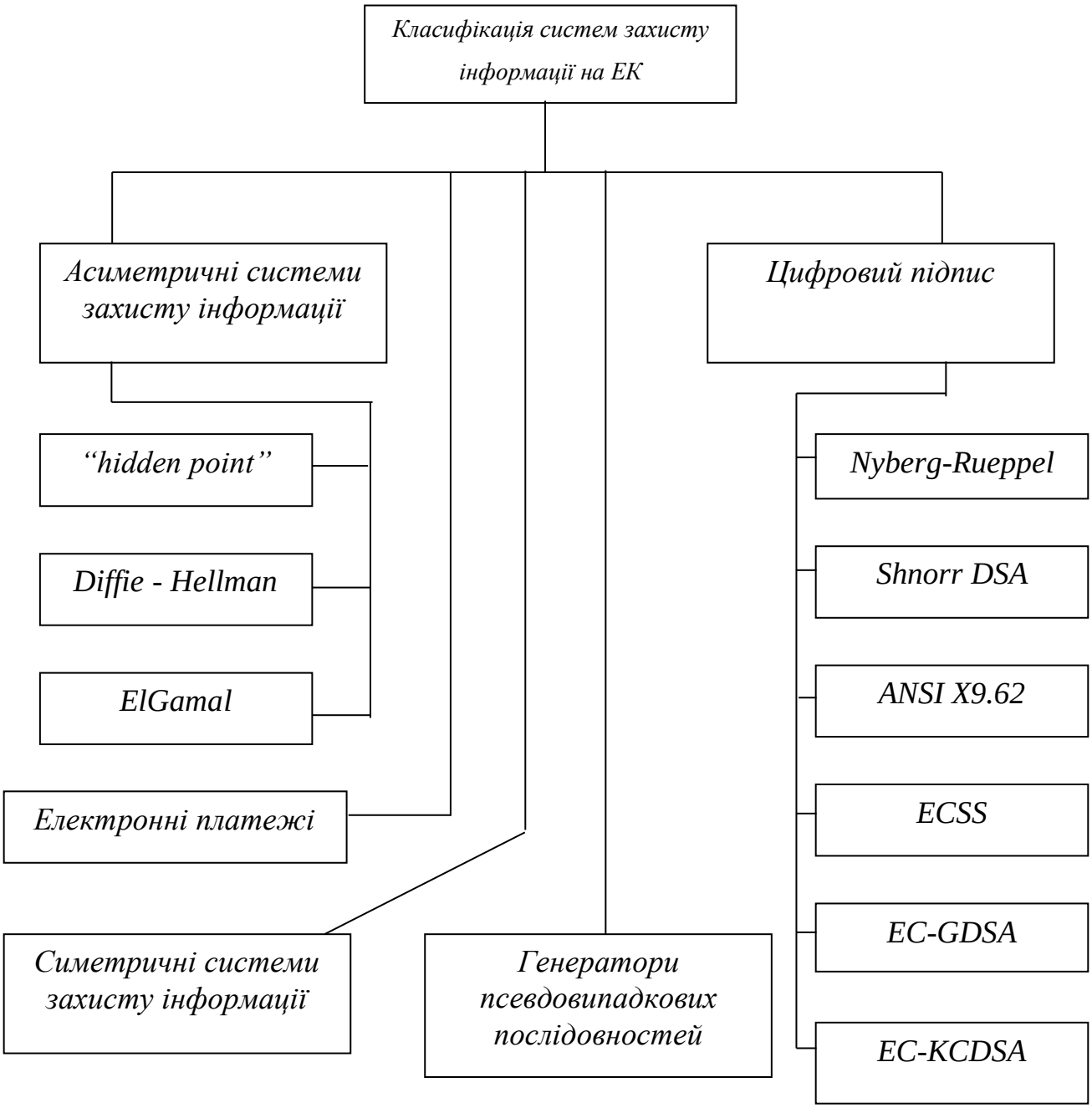


Рисунок 18.2 - Класифікація застосування еліптичних кривих для задач захисту інформаційних потоків.

Незважаючи на вагомі переваги застосування ЕК, поряд з цим існують певні проблеми та труднощі. Зокрема, виділяють такі класи задач: генерування параметрів еліптичної кривої; обчислення порядку еліптичної кривої; дискретне логарифмування.

Ці три класи задач взаємоповязані і є ключовими в системах захисту інформаційних потоків з використанням ЕК: перших два – для шифрування і аналізу стійкості, третій – дешифрування.

Проведений аналіз показує, що для розв'язання задачі знаходження кількості точок на ЕК можна застосувати наступні алгоритми: «крок гіганта-крок малюка», Шуфа, Еткіна та Еліза.

Часова складність першого алгоритму оцінюється як $O\left(q^{\frac{1}{4}+\varepsilon}\right)$, зокрема для обмеженого поля малої потужності, та ґрунтується на теорії Шанкса. Алгоритм приречений на невдачу дуже рідко, тоді і тільки тоді, коли для кожної точки еліптичної кривої P є більш, ніж одне число m в інтервалі $\left(p+1-2\sqrt{p}, p+1+2\sqrt{p}\right)$, для якої $mP=0$. Це стається тоді, коли показник степеня групи $E(F_p)$ дуже малий, тому в цьому випадку доцільно скористатися алгоритмами Шуфа, Еткіна та Еліза. Часова складність першого алгоритму ґрунтується на обчисленні високих степенів $x^p, y^p, x^{p^2}, y^{p^2}$ по модулю $f_l(x)$, має часову складність $O((\log p)M(n))$ і $O((\log p^2)M(n))$ відповідно бітових операцій, яке зводиться до декількох додавань і множень в кільці. Оскільки елементи кільця мають розмір $l^2 \log p$, то часова складність становить $O(\log p(l^2 \log(p)^2)$ і $O(l(l^2 \log p)^2)$ відповідно. Тут прийнято, що множення двох елементів розмірності n біт відбувається за час, який пропорційний до n^2. Беручи до уваги, що $l = O(\log p)$, і, здійснивши обчислення для кожного l, остаточно отримуємо вираз для роботи, яка витрачається на повне обчислення: $O(\log^8 p)$.

Удосконалення алгоритму зводиться до обчислення X_p і Y_p в кільці:

$$F_p[X,Y]/(F(X), Y^2 - X^3 - AX - B). \tag{18.5}$$

Беручи до уваги, що для F(X) притаманний порядок $(l-1)/2$, а не $(l^2-1)/2$, то для обчислення беруть тільки дії $O(l^2 \log^3 p + l^3 \log^2 p) = O(\log^5 p)$. Це і є суттєвим щодо значного зменшення часу виконання $O(\log^7 p)$ відповідної дробової частини алгоритму .

Слід зазначити, що підхід Елкіза справджується та є ефективним лише для простих чисел l, для яких φ має власні значення в F_l, тобто для половини простих чисел l, тих, що розпадаються в полі $Q(\sqrt{t^2-4p}$. При цьому потрібно обчислити коефіцієнти многочлена $F(X) \in F_p[X]$.

Тема 19. Реалізація шифрування на еліптичних кривих.

Показники оцінки ефективності функціонування алгоритмів шифрування на ЕК розділені на чотири категорії:

– показники оцінки стійкості алгоритмів на ЕК;

– програмно-реалізаційні показники;
– апаратно-реалізаційні показники;
– конструктивно-технологічні показники.

Система оцінки ефективності функціонування паралельних алгоритмів шифрування на ЕК $R_{пр}$ описується наступним чином:

$$\begin{cases} R_{пр} = \alpha_1 \cdot П_{ст} + \alpha_2 \cdot П_{прр} + \alpha_3 \cdot П_{ар} + \alpha_4 \cdot П_{кт} \\ \sum\limits_{i=1}^{4} \alpha_i = 1 \\ П_{ст} = \left\langle R_{дл}\left\langle R_{гбт}, R_{гп}\right\rangle, R_{сасв}\left\langle R_{го}, R_{ро}\right\rangle\right\rangle \\ П_{прр} = \left\langle R_{овп}, R_{шші}, R_{квао}\right\rangle \\ П_{ар} = \left\langle R_{аз}\left\langle R_{ле}, R_{еп}, R_{вв}\right\rangle, R_{отч}, R_{оп}\right\rangle \\ П_{кт} = \left\langle R_{пк}, R_{ппа}, R_{п}, R_{зс}\right\rangle \end{cases} \quad (19.1)$$

де $П_{ст}$ - показники стійкості алгоритмів шифрування на ЕК, які характеризують стійкість до аналітичних методів криптоаналізу. Ці показники є домінуючими тому, що будь-яке криптографічне перетворення інформації ґрунтується насамперед, на оцінці стійкості алгоритмів шифрування до відомих на сьогодні атак, а також атак спеціального виду. Усі показники визначаються на основі експертних оцінок в залежності від пріоритетного напрямку застосування алгоритмів шифрування в реальних прикладних задачах захисту інформаційних ресурсів.

Класи показників оцінки ефективності функціонування алгоритмів шифрування на ЕК наведені на рисунку 19.1.

На цьому етапі дослідження створюється математична модель, що визначає параметри комп'ютерної системи. Тому доцільно скористатися класифікацією атак на ЕК (рисунок 19.2). Для забезпечення необхідного рівня стійкості сучасних систем захисту інформації доцільно використовувати нові методи та алгоритми опрацювання інформаційних потоків для реалізації алгоритму Шуфа, генерування параметрів ЕК. Щоб модель була адекватною, потрібно врахувати основні види впливу атак на роботу системи:

1) програмні атаки – стосуються електронного-цифрового підпису;
2) математичні атаки – використовуються дискретного логарифму;
3) атаки на засоби реалізації криптоалгоритмів:
4) аналіз часу виконання окремих операцій;
5) аналіз енергоспоживання (звичайний – SPA та диференціальний – DPA);
6) диференційний аналіз помилок;
7) аналіз електромагнітного випромінювання;
8) аналіз радіоактивного випромінювання.

В рисунок 19.2 включено критерії стійкості алгоритмів шифрування у КМ згідно до атак на ЕК. Спираючись на результати аналізу сучасного стану застосування ЕК, слід скористатися наступними показниками оцінки стійкості:

— показники стійкості до програмних атак;
— показники стійкості до математичних атак;
— показники стійкості до апаратних атак.

Рисунок 19.1 – Показники оцінки ефективності функціонування алгоритмів шифрування на ЕК.

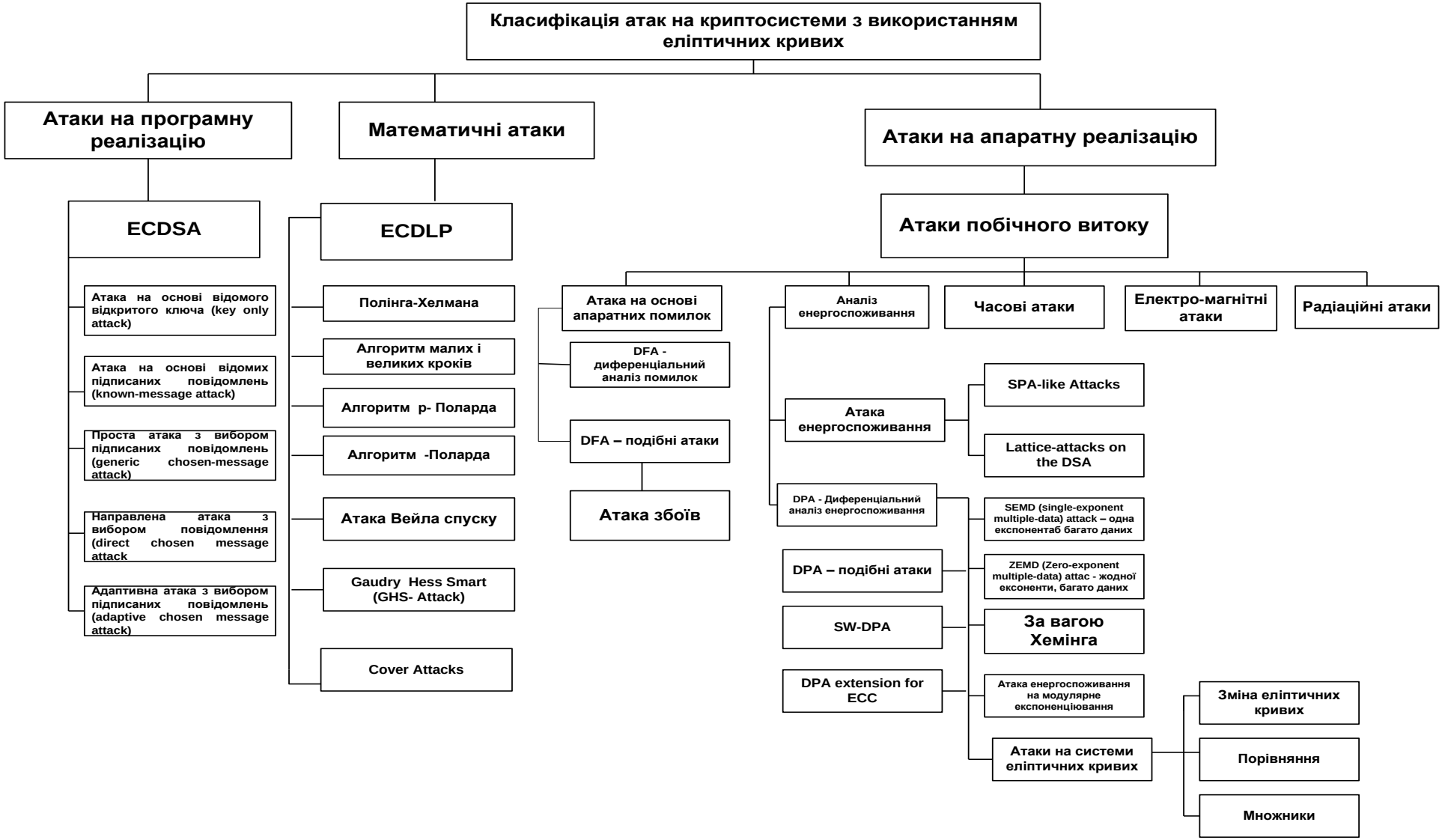


Рисунок 19.2 – Класифікація атак на ЕК.

Формалізовану модель оцінки стійкості до атак на ЕК M_{pr} створено на основі використання методу вагових коефіцієнтів з нормованими аргументами:

$$\begin{cases} M_{pr} = \alpha_1 \cdot P_{rpra} + \alpha_2 \cdot P_{rma} + \alpha_3 \cdot P_{rha} \\ \sum_{i=1}^{3} \alpha_i = 1 \\ P_{rpra} = \left\langle M_{dsa} \left\langle M_{koa,} M_{kma,} M_{gcma,} M_{dcma,} M_{acma} \right\rangle \right\rangle \\ P_{rma} = \left\langle M_{dlp} \left\langle M_{ph,} M_{bgs,} M_{\rho-P,} M_{\lambda-P,} M_{wa,} M_{ghsa,} M_{ca} \right\rangle \right\rangle \\ P_{rha} = \left\langle M_{sca} \left\langle M_{fa} \left\langle M_{dfa,} M_{dfa-like} \left\langle M_{ga} \right\rangle \right\rangle, M_{pa} \left\langle M_{spa,} M_{dpa} \right\rangle, M_{ta,} M_{ema,} M_{ra} \right\rangle \right\rangle \end{cases} \quad (19.2)$$

де P_{rpra} —показник стійкості алгоритмів шифрування на ЕК до програмних атак.

Основними складовими показника стійкості є :

– M_{dsa} — показник стійкості, зумовлений атакою на ЕЦП;

– M_{dsa} M_{koa} — показник стійкості, зумовлений атакою на основі відкритого ключа;

– M_{dsa} M_{kma} — показник стійкості, зумовлений атакою на основі підписаних повідомлень;

– M_{dsa} M_{gcma} — показник стійкості, зумовлений простою атакою з вибором підписаних повідомлень;

– M_{dsa} M_{dcma} — показник стійкості, зумовлений направленою атакою з вибором повідомлень;

– M_{dsa} M_{acma} — показник стійкості, зумовлений адаптивною атакою з вибором підписаних повідомлень;

– P_{rma} — показник стійкості до математичних атак.

– Він включає наступні критерії:

– M_{dlp} — показник стійкості, зумовлений атакою на дискретний логарифм;

– M_{ph} — показник стійкості, зумовлений атакою з використанням алгоритму Полінга-Хелмана;

– M_{bgs} — показник стійкості, зумовлений атакою з використанням алгоритму малих і великих кроків;

– $M_{\rho-P}$ — показник стійкості, зумовлений атакою з використанням алгоритму ρ-Поларда;

– $M_{\rho-P}$ $M_{\lambda-P}$ — показник стійкості, зумовлений атакою з використанням алгоритму λ-Поларда;

– $M_{\rho-P}$ M_{wa} — показник стійкості, зумовлений атакою спуску Вейла; M_{ghsa} — показник стійкості, зумовлений атакою Gaudry Hess Smart;

– $M_{\rho-P}$ M_{ca} — показник стійкості, зумовлений cover атакою.

– P_{rha} — показник стійкості до апаратних атак.

Включає наступні критерії:

– M_{sca} — показник стійкості, зумовлений атакою побічного витоку;

– M_{fa} — показник стійкості, зумовлений атакою на основі апаратних помилок;

– M_{dfa} — показник стійкості, зумовлений атакою на основі диференціального аналізу помилок, $M_{dfa-like}$ — DFA-подібні атаки, M_{ga} — показник стійкості, зумовлений атакою збоїв, M_{pa} — показник стійкості, зумовлений атакою енергоспоживання, M_{spa} — показник стійкості, зумовлений простою атакою енергоспоживання, M_{dpa} — показник стійкості, зумовлений атакою диференціального аналізу енергоспоживання, M_{ta} — часова атака, M_{ema} — показник стійкості, зумовлений електро-магнітною атакою, M_{ra} — показник стійкості, зумовлений радіаційною атакою. α_i — вагові коефіцієнти, що визначають міру впливу кожного з показників на результуюче значення стійкості алгоритму шифрування.

Запропонована класифікація сучасних методів криптоаналізу та формалізована модель дозволяють ефективно оцінювати рівень захисту ІП в КС.

Тема 20. ЕЦП на основі еліптичних кривих

ECDSA (Elliptic Curve Digital Signature Algorithm) - алгоритм з відкритим ключем для створення цифрового підпису, аналогічний за своєю будовою DSA, але певний на відміну від нього не над полем цілих чисел, а в групі точок еліптичної кривої.

Стійкість алгоритму шифрування грунтується на проблемі дискретного логарифма в групі точок еліптичної кривої. На відміну від проблеми простого дискретного логарифма і проблеми факторизації цілого числа, не існує суб-експонеціального алгоритму для проблеми дискретного логарифма в групі точок еліптичної кривої. З цієї причини "сила на один біт ключа" істотно вище в алгоритмі, який використовує еліптичні криві. [1]

Д. Брауном (Daniel RL Brown) було доведено, що алгоритм ECDSA не є більш безпечним, ніж DSA. Їм було сформульовано обмеження безпеки для ECDSA, яке призвело до наступного висновку:

"Якщо група еліптичної кривої може бути змодельована основною групою і її хеш-функція задовольняє певним обгрунтованого припущенням, то ECDSA стійка до chosen-message атаці з існуючою фальсифікацією." [2]

Алгоритм ECDSA в 1999 р. був прийнятий, як стандарт ANSI, в 2000 р. - як стандарт IEEE і NIST. Також у 1998 р. алгоритм був прийнятий стандартом ISO. Незважаючи на те, що стандарти ЕЦП створені зовсім недавно і перебувають на етапі вдосконалення, одним найбільш перспективних з них на сьогоднішній день є ANSI X9.62 ECDSA від 1999 - DSA для еліптичних кривих.

Для підписування повідомлень необхідна пара ключів - відкритий і закритий. При цьому закритий ключ повинен бути відомий тільки тому, хто підписує повідомлення, а відкритий - будь-якому охочому перевірити

достовірність повідомлення. Також загальнодоступними є параметри самого алгоритму.

Параметри алгоритму

1. Вибір хеш-функції H(x). Для використання алгоритму необхідно, щоб повідомлення було числом. Хеш-функція повинна перетворити будь-яке повідомлення в послідовність бітів, які можна потім перетворити в число.
2. Вибір великого простого числа q.

Зауваження: Якщо розмірність цього числа в бітах менше розмірності в бітах значень хеш-функції H(x), то використовуються тільки ліві біти значення хеш-функції.

3. Простим числом p позначається характеристика поля координат F_p.

Генерування ключів ECDSA

Для простоти будемо розглядати еліптичні криві над полем F_p, де F_p - кінцеве просте поле. Причому, якщо необхідно, конструкцію можна легко адаптувати для еліптичних кривих над іншим полем.

DSA *(Digital Signature Algorithm)* - алгоритм з використанням відкритого ключа для створення електронного підпису, але не для шифрування (на відміну від RSA і схеми Ель-Гамаля). Підпис створюється таємно, але може бути публічно перевірена. Це означає, що тільки один суб'єкт може створити підпис повідомлення, але будь-хто може перевірити її коректність. Алгоритм заснований на обчислювальній складності узяття логарифмів в кінцевих полях.

Для побудови системи цифрового підпису потрібно виконати наступні кроки:

1. Вибір криптографічної хеш-функції H (x).
2. Вибір великого простого числа *q,* розмірність якого *N* в бітах збігається з розмірністю в бітах значень хеш-функції H (x).
3. Вибір простого числа *p,* такого, що *(p-1)* ділиться на *q*. Бітова довжина *p* позначається *L*.
4. Вибір числа *g* такого, що його мультиплікативний порядок по модулю *p* дорівнює *q*.

Як згадано вище, а також у DSS (Digital Signature Standard), першочерговим параметром схеми цифрового підпису є використовувана криптографічна хеш-функція, необхідна для перетворення тексту повідомлення в число, яке власне і буде підписана. Важливою характеристикою цієї функції є бітова довжина вихідний послідовності, що позначається далі *N* (160 для функції SHA-1). У першій версії стандарту DSS рекомендована функція SHA-1 і, відповідно, бітова довжина підписуваного числа 160 біт. Зараз SHA-1 вже не є достатньо безпечною. У стандарті вказані наступні можливі пари значень чисел *L* і *N:*

1. L = 1024, N = 160
2. L = 2048, N = 224
3. L = 2048, N = 256
4. L = 3072, N = 256

У відповідності з цим рекомендовані хеш-функції сімейства SHA-2. Урядові організації повинні використовувати один з цих варіантів, але всі інші вільні вибирати. Проектувальник системи може вибрати будь-яку хеш-функцію. Тому

далі не буде загострюватися увага на використанні конкретної хеш-функції. Стійкість криптосистеми на основі DSA не перевершує стійкість використовуваної хеш-функції і стійкість пари (L, N), чия стійкість не більше стійкості кожного з чисел окремо. Раніше рекомендувалася довжина *p* L = 1024 біта. В даний момент для систем, які повинні бути стійкими, рекомендується довжина в 2048 (3072) біта.

Оскільки NIST зробив алгоритм вільним до використання, то алгоритм можна вільно реалізовувати програмним, апаратним або будь-яким іншим чином. NIST розробив програму перевірки відповідності реалізації алгоритму стандарту. Інформація про систему доступна, приклади роботи теж. Реалізації систем цифрового підпису повинні використовувати криптографічні алгоритми, алгоритми генерації криптографічних ключів та способи розподілу ключів, безпека яких так чи інакше підтверджена.

Тема 21. Криптографічні протоколи

Обмін ключем

Нехай абоненти А і Б, які, спілкуючись через канал, що ймовірно прослуховується, хочуть домовитися про спільний таємний ключ. Тоді:

- абонент А вибирає велике просте число p та просте 1<g<p-1 і відкрито, не роблячи з цього жодної таємниці, посилає p і g абонентові Б;
- абонент А вибирає випадкове число *a* в межах від 1 до p – 1, а абонент Б – випадкове число *b* в тих же межах;
- абонент А обчислює g^a mod p і посилає це значення абонентові Б, який обчислює g^b mod p і теж посилає абоненту А;
- обидва абоненти обчислюють одне і теж число

$$(g^b)^a \bmod p = (g^a)^b \bmod p = g^{ab} \bmod p,$$

яке і приймають в якості ключа.

ПРИКЛАД. Нехай p = 97, а g = 5. Припустимо, що абонент А вибрав число a = 12, а абонент Б вибрав b = 63. Тоді абонент А посилає абоненту Б $5^{12} \bmod 97 = 42$, абонент Б абоненту А $5^{63} \bmod 97 = 75$, і обоє обчислюють $75^{12} \bmod 97 = 42^{63} \bmod 97 = 21$.

Жереб по телефону

- абонент А вибирає своє число *a* і пару ключів *(K,K')*. Після цього зашифровує *a* і результат *c = Ek(a)* разом з ключем К посилає абоненту Б;
- абонент Б вибирає число b і посилає його абоненту А;
- абонент А посилає абоненту Б дешифруючий ключ К’;
- абонент Б перевіряє, що *(K,K')* справді є парою шифруючого та дешифруючого ключів, і обчислює a = *DK'(C)*.
- Обоє обчислюють R = a ⊕b.

Гра в карти заочно

- абоненти А і Б досягають згоди про кодування карт словами M_1,..., M_{52}, і домовляються, яка саме комутативна криптосистема буде використовуватись;
- Обоє таємно один від другого вибирають собі шифруючий та дешифруючий ключі;
- абонент А зашифровує повідомлення M_1,..., M_{52}, перемішує випадковим чином

крнптотексти $E_A(M_1)$, ..., $E_A\{M_{52}\}$ і посилає їх абоненту Б;

- абонент Б вибирає випадкові п'ять криптотекстів, і посилає їх назад абоненту А. Це карти, якими буде грати абонент А;
- із карт, що залишилися, абонент Б вибирає ще п'ять для себе;
- абонент Б зашифровує відібрані карти за допомогою власного ключа і отримані криптотексти посилає абоненту А;
- абонент А дешифрує отримані криптотексти і повертає абоненту Б результат;
- абонент Б дешифрує надіслані абонентом А криптотексти і отримує свою п'ятірку карт;
- в кінці гри абоненти обмінюються ключами і перевіряють, чи ніхто з них не хитрував.

Розподіл таємниці

Нехай натуральне число *s* є цінною секретною інформацією (номер рахунку у швейцарському банку, код команди на запуск балістичної ракети тощо). Завданням протоколу є так подрібнити секрет s на частини, по одній для кожного із *n* учасників, щоб будь-які *k* учасників могли відновити s, поєднавши свої частинки, але щоб ніяка група з *k* – 1 учасника цього зробити не могла.

Вибирають досить велике просте число p, більше за s. Покладають a_0 = s, і вибирають випадковим чином числа a_1,...., a_{k-1}. Нехай $f(x) = \Sigma_{0\le i<k} a_i x^i$ – многочлен від змінної *x*. і-ий учасник протоколу, де 1 ≤i≤ n, отримує значення s_i=f(i).

Якщо відомі довільні *к* значень $f(i_1)$, . . ., $f(i_k)$, то многочлен f можна реконструювати за інтерполяційною формулою Лагранжа:

$$f(x) = \sum_{i=1}^{k} f(i_1) \prod_{j \neq 1} \frac{x - i_j}{i_i - i_j}.$$

Після цього легко знаходиться секрет $s = a_0$ = f(0).

Знання лише *к* - 1 значення функцій f не дає *жодної* інформації про секрет.

Тема 22. Приклади сучасних комп'ютерних криптографічних систем.

Один з підходів до вирішення проблеми безпеки в Інтернеті був запропонований компанією Netscape Communications. Нею був розроблений протокол SSL (Secure Sockets Layer) захищеного обміну інформацією між клієнтом і сервером. SSL вимагає застосування надійного транспортного протоколу (наприклад, TCP).

Фірма Netscape почала займатися захистом Web–транзакцій з тих пір, як з'явилися перші браузери. Використовуючи попередній досвід в даній області, Netscape розробила протокол SSL 1.0 (secure socket layer).

Найбільш поширеним пакетом програм для підтримки SSL є SSLeay. Остання версія (SSLeay v. 0.9.8) підтримує SSLv3. Ця версія доступна у вихідних текстах. Даний пакет призначений для створення і управління різного роду сертифікатами. Так само до його складу входить і бібліотека для підтримки SSL різними програмами. Ця бібліотека необхідна, наприклад, для модуля SSL в поширеному HTTP сервері Apache.

Протокол SSL (secure socket layer) був розроблений фірмою Netscape, як протокол, що забезпечує захист даних між найбільш відомими сервісними протоколами (такими, як HTTP, NNTP, FTP і т.д.), а також транспортними протоколами (TCP/IP).Часто для нього використовується абревіатура HTTPS.

Протокол SSL надає "безпечний канал", який має три основні властивості:

– канал є приватним. Шифрування використовується для всіх повідомлень після простого діалогу, який служить для визначення секретного ключа. Для шифрування застосовуються: RC4_128, RC4_40, RC2_128, RC2_40, DES40 та ін;

– канал аутентифікований. Серверна сторона діалогу завжди автентифікується;

– канал надійний. Транспортування повідомлень включає в себе перевірку цілісності (із залученням Message Autentification Code MAC, що обчислюються за допомогою хеш–функцій MD5).

Не секрет, що можна без особливих технічних зусиль переглядати дані, якими обмінюються між собою клієнти і сервери. Був навіть придуманий спеціальний термін для цього – sniffer. А у зв'язку зі збільшенням обсягу використання Інтернету в комерційних цілях неминуче поставало питання про захист переданих даних. І користувачі не дуже були б раді, якщо номер їх кредитної картки був би перехоплений яким–небудь заповзятливим хакером по дорозі до віртуального магазину. І, загалом, поява такого протоколу, як SSL, було цілком закономірним явищем. З одного боку, залишаються всі можливості сервісних протоколів (для програм–серверів), плюс до цього, всі дані передаються у зашифрованому вигляді. І декодувати їх досить важко. Слід зазначити, що SSL не тільки забезпечує захист даних в Інтернеті, але так само виробляє впізнання сервера і клієнта (server/client authentication). Протокол SSL прийнятий W3 консорціумом (W3 Consortium) як основний захисний протокол для клієнтів і серверів (WWW browsers and servers) в мережі Інтернет. Частіше за все цей протокол використовується в складі будь–якого Інтернет–ресурсу, що здійснює маніпуляції з особистими або фінансовими даними користувачів Інтернету, які його відвідують. Найчастіше це банки, Інтернет–магазини або будь–які інші віртуальні місця, до яких приходять по своїх справах користувачі, змушені передавати свої особисті, і часто, секретні дані. Цього може вимагати і проста реєстрація, і процедура оплати будь–якого товару, або будь–яка інша процедура, при якій користувачі змушені чесно видавати свої паспортні дані, PIN–и і паролі. Використовуючи звичайний HTTP–протокол, ми передаємо і отримуємо інформацію в чистому, не зашифрованому вигляді. Таким чином, передана нами інформація може бути легко перехоплена, і використана сторонньою людиною.

Отже, з'являються два досить вагомих докази: по–перше, передану інформацію треба шифрувати, і, по–друге, ми повинні бути впевнені, що передаємо інформацію саме туди, куди потрібно. Саме для вирішення цих двох питань і використовується SSL. Схема використання протокола SSL показана на рисунку 22.1.

Протокол SSL реалізується у вигляді двошарового (багатошарового) середовища, спеціально призначеного для безпечного переносу секретної

інформації через незасекречені канали зв'язку. В якості першого шару в такому середовищі використовується деякий надійний транспортний протокол, наприклад, TCP. По слову "транспортний" неважко здогадатися, що TCP бере на себе функції "несучого", і надалі стає візником для всіх шарів (протоколів), що лежать вище Другим за рахунком шаром, що накладається на TCP, є протокол записів SSL (Record Protocol). Разом ці два шари, TCP і SSL Record Protocol, формують своєрідне ядро SSL. В подальшому це ядро стає первинною герметизуючою оболонкою для всіх наступних більш складних протокольних інфраструктур. В якості однієї з таких структур використовується протокол узгодження SSL (Handshake Protocol). Він дозволяє серверу і клієнту ідентифікувати один одного і узгоджувати криптографічні алгоритми і ключі, перед тим як додатки, що працюють на серверній і клієнтській стороні, зможуть почати передачу або прийом інформаційних байтів в захищеному режимі.

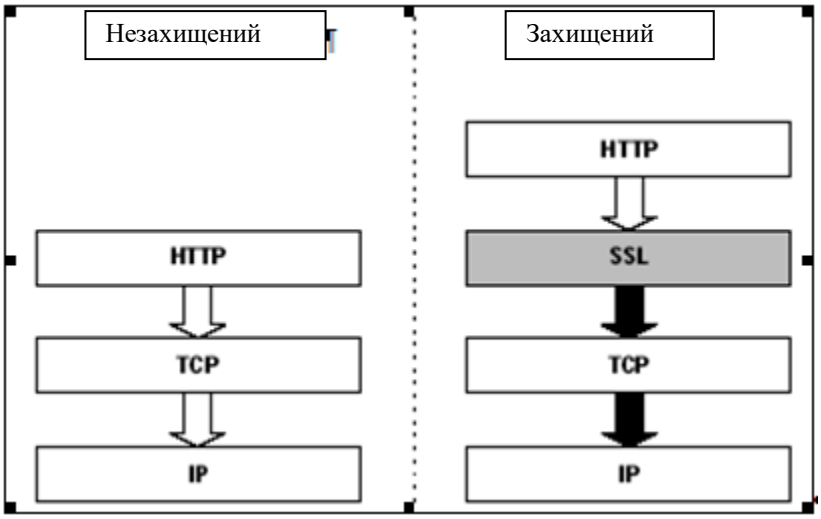


Рисунок 22.1 – Використання SSL

Однією з важливих переваг SSL є його повна програмно–платформенна незалежність. Протокол розроблено на принципах переносимості, і ідеологія його побудови не залежить від тих програм, у складі яких він використовується. Крім цього, важливо і те, що поверх протоколу SSL, можуть прозоро накладатися й інші протоколи; або для ще більшого збільшення ступеня захисту цільових інформаційних потоків, або для адаптації криптографічних здібностей SSL під яке–небудь інше, цілком певне завдання. Протоколи верхнього рівня можуть розміщуватися поверх протоколу SSL прозорим чином. Рішення про те, як ініціалізувати SSL–діалог і як інтерпретувати сертифікати аутентифікації, залишається на розсуд розробників протоколів і програм, які працюють поверх SSL.

Починається використання SSL в той момент, коли ввести в адресному рядку браузера URL як початок абревіатуру HTTPS. В результаті відбувається підключення до порту за номером 443, який для SSL зазвичай використовується за умовчанням; для стандартного HTTP з'єднання найчастіше використовується порт 80. У процесі підключення браузер користувача (надалі клієнт), посилає

серверу повідомлення узгодження (hello message). У свою чергу, сервер також повинен посилати клієнтові своє вітальне повідомлення. Повідомлення узгодження є первинними, ініціалізувалися повідомленнями і містять інформацію, використовувану при подальшій настройці секретного каналу, що відкривається. У загальному випадку вітальне повідомлення встановлює чотири основних параметри: версія протоколу, ідентифікатор сесії, спосіб шифрування, метод компресії, а також два спеціально згенерованих випадкових числа. Сервер і клієнт генерують такі числа незалежно один від одного, а потім, просто обмінюються ними один з одним.

Після отримання повідомлення узгодження від клієнта сервер відсилає свій сертифікат, якщо такий у нього є. Також, при необхідності, сервер може послати і якесь ключове повідомлення, наприклад у разі відсутності сертифіката. Якщо сервер авторизовано (тобто має відповідний сертифікат), він може зажадати і клієнтський сертифікат, якщо того зажадає обраний спосіб шифрування даних. Після цього проводиться ще ряд проміжних обмінних операцій, в процесі яких проводиться остаточне уточнення обраного алгоритму шифрування, ключів і секретів, і далі, сервер посилає клієнтові якесь фінальне повідомлення, після чого обидві сторони приступають до обміну зашифрованою інформацією.

На практиці процес обміну ключами та сертифікатами іноді може займати відносно багато часу. З цією метою, часто передбачається можливість повторного використання одних і тих же ідентифікаційних даних. Бувають ситуації, коли після встановлення з'єднання з SSL–сервером у користувача з'являється бажання відкрити ще одне вікно браузера, і через нього здійснити ще одне підключення до того ж SSL–сервера.В цьому випадку, щоб не повторювати весь цикл попередніх обмінних операцій, браузер може відправити серверу ідентифікатор сесії попереднього з'єднання, і, якщо сервер прийме цей ідентифікатор, весь набір шифровочних і компресійних параметрів буде взято від попереднього з'єднання. Браузери від Netscape також можуть здійснювати і так званий "keep alive" запит. При цьому по завершенню передачі зашифрованих даних встановлене SSL–з'єднання закривається не відразу, а лише після закінчення деякого часу.

Тепер розглянемо, яким чином все–таки працює SSL. Уявимо собі, що є дві людини, які спілкуються за допомогою Інтернету і відповідно не бачать один одного. Вони позбавлені можливості дізнатися про те, хто ж його абонент. Їх імена – Аліса і Боб.Припустимо Алісі треба дізнатися, чи справді вона розмовляє з Бобом, чи ні. У цьому випадку діалог може виглядати наступним чином: Аліса відправляє Бобу випадкове повідомлення. Боб шифрує його з допомогою свого приватного ключа і відправляє його Алісі. Аліса дешифрує це повідомлення (з допомогою публічного ключа Боба). І, порівнявши це повідомлення з уже посланим, може переконатися в тому, що його дійсно послав Боб. Але насправді з боку Боба не дуже вдала ідея шифрувати повідомлення від Аліси з допомогою свого приватного ключа і повертати його. Це аналогічно підпису документа, про який Боб мало що знає. З такої позиції Боб має сам придумати повідомлення і послати його Алісі у двох примірниках. У першому повідомлення передається відкритим текстом, а друге повідомлення зашифровано за допомогою приватного

ключа Боба. Таке повідомлення називається message digest. А спосіб шифрування повідомлення за допомогою свого приватного ключа називається цифровим підписом (digital signature).

Тепер закономірно постає питання про те, яким чином поширювати свої публічні ключі. Для цього (і не тільки) була придумана спеціальна форма – сертифікат (certificate).Сертифікат складається з наступних частин: ім'я людини / організації, яка випускає сертифікат / для кого був випущений даний сертифікат (суб'єкт сертифіката) / публічний ключ суб'єкта / деякі тимчасові параметри (термін дії сертифіката тощо). Сертифікат підписується приватним ключем особи (або організації), яка випускає сертифікати. Організації, які виробляють подібні операції, називаються Certificate authority (CA). Якщо в стандартному Web-клієнті (web-browser), який підтримує SSL, зайти в розділ security, то там можна побачити список відомих організацій, які підписують сертифікати.

Цілями протоколу SSL/TLS в порядку пріоритетності є:

– криптографічна безпека. SSL/TLS повинен використовуватися для встановлення безпечного з'єднання між двома партнерами;

– сумісність. Незалежні програмісти повинні розробляти програми, що використовують SSL/TLS, які будуть здатні успішно обмінюватися криптографічними параметрами без знання особливостей програм один одного;

– розширюваність. SSL/TLS шукає спосіб, як при необхідності вбудувати в систему нові ключі і методи шифрування. Тут є дві побічні мети: виключити необхідність створення нового протоколу (що може бути пов'язане з введенням нових слабких місць) і зробити непотрібним впровадження нової бібліотеки, що забезпечує безпеку;

– відносна ефективність. Криптографічні операції вимагають великих потужностей, особливо цим славляться алгоритми з відкритими ключами. З цієї причини, протокол SSL / TLS має опційну схему кешування сесії, що дозволяє зменшити число з’єднань, які встановлюються з використанням нових тимчасових буферів. Було вжито заходів, щоб зменшити мережеву активність.

Протокол TLS базується на специфікації протоколу SSL 3.0, опублікованого Netscape. Різниця між цим протоколом і SSL 3.0 незначна, TLS 1.0 має механізм, за допомогою якого додатки можуть підтримувати SSL 3.0.

Виходячи з вищесказаного, в даній дипломній роботі поставлена задача побудови моделі загроз та методів захисту для Web-транзакцій, в яких використовуються стандартні протоколи TLS\SSL. Метою нашої роботи є аналіз існуючих атак на протокол SSL/TLS і розробка рекомендацій та методів боротьби з такими атаками.

У SSL усі дані пересилаються у вигляді рекордів (записів), об'єктів, які складаються з заголовка і деякої кількості даних. Кожен заголовок рекорду містить два або три байта коду довжини. Якщо старший біт в першому байті коду довжини рекорду дорівнює 1, тоді рекорд немає заповнювача і повна довжина заголовка дорівнює 2 байтам, в іншому випадку рекорд містить заповнювач і повна довжина заголовка дорівнює 3 байтам. Передача завжди починається з заголовка.

Зауважимо, що в разі довгого заголовка (3 байти), другий за старшинством біт першого байта має спеціальне значення. Коли він дорівнює нулю, то рекорд, що посилається, є інформаційним. При рівності 1, рекорд, що посилається, є security escape (в даний час не визначено жодного значення security escapes; це зарезервовано для майбутніх версій протоколу).

Тема рекорду визначає значення, яке відоме під назвою PADDING. Значення PADDING специфікує число байтів, доданих відправником до вихідного рекорду. Дані заповнювача використовуються для того, щоб зробити довжину рекорду кратну розміру блоку шифру, якщо застосовано блочний шифр.

Відправник "заповненого" рекорду додає заповнювач після наявних даних, а потім шифрує все це, тому що довжина цього масиву кратна розміру блоку використовуваного шифру. Вміст заповнювача не грає ролі. Так як обсяг переданих даних відомий, заголовок повідомлення може бути коректно сформований з урахуванням обсягу PADDING.

Одержувач цього рекорду дешифрує все поле даних і отримує вихідну інформацію. Після цього проводиться обчислення істинного значення RECORD-LENGTH (з урахуванням наявності опційного PADDING), при цьому заповнювач з поля даних видаляється.

Для двобайтового заголовку максимальна довжина рекорду дорівнює 32767 байтів. Для трьохбайтового заголовку максимальна довжина рекорду дорівнює 16383 байтів. Повідомлення протоколу діалогу SSL повинні відповідати поодиноким рекордам протоколу SSL (Record Protocol). Повідомлення прикладного протоколу можуть займати кілька рекордів SSL.

Перш, ніж послати перший рекорд SSL, всі порядкові номери обнуляються. При передачі повідомлення порядковий номер інкрементується, починаючи з повідомлень CLIENT–HELLO і SERVER–HELLO.

Обробка помилок в протоколі з'єднань SSL дуже проста. Коли помилка детектована, елемент, що виявив її посилає своєму партнеру повідомлення. Помилки, які є неусувними, вимагають від клієнта і сервера розриву з'єднання. Сервери і клієнт повинні "забути" всі ідентифікатори сесії, пов'язані з розірваним з'єднанням. Протокол діалогу SSL визначає такі помилки:

– NO–CIPHER–ERROR. Ця помилка надсилається клієнтом серверу, коли він не може знайти шифр або розмір ключа, який підтримується також і сервером. Ця помилка фатальна;

– NO–CERTIFICATE–ERROR. Коли надіслано повідомлення REQUEST–CERTIFICATE, ця помилка може бути надіслана, якщо клієнт немає сертифікату. Ця помилка усувна;

– BAD–CERTIFICATE–ERROR. Такий відгук надсилається, коли сертифікат з якоїсь причини вважається приймаючою стороною поганим. Поганий означає, що або некоректний підпис сертифіката, або некоректне його значення (наприклад, ім'я в сертифікаті не відповідає очікуваному). Ця помилка усувна (тільки для аутентифікації клієнта);

– UNSUPPORTED–CERTIFICATE–TYPE–ERROR. Цей відгук надсилається, коли клієнт/сервер отримує тип сертифіката, який він не підтримує. Ця помилка усувна (тільки для аутентифікації клієнта).

Тема 23. Стеганографія.

Стеганогра́фія — (з грец. στεγανός — прихований + γράφω — пишу) — тайнопис, при якому повідомлення, закодоване таким чином, що не виглядає як повідомлення — на відміну від криптографії. Таким чином непосвячена людина принципово не може розшифрувати повідомлення — бо не знає про факт його існування.

Якщо криптографія приховує зміст повідомлення, то стеганографія приховує сам факт існування повідомлення.

Перший запис про використання стеганографії зустрічається в трактаті Геродота «Історія», що відноситься до 440 року до н. е. У трактаті були описані два методи приховування інформації. Демарат відправив попередження про майбутній напад на Грецію, записавши його на дерев'яну підкладку воскової таблички до нанесення воску. Другий спосіб полягав у наступному: на поголену голову раба записувалося необхідне повідомлення, а коли його волосся відростало, він вирушав до адресата, який знову голив його голову і зчитував доставлене повідомлення.

У своєму творі «Про перенесення облоги» Еней запропонував робити малопомітні дірки поруч з літерами в книзі або іншому документі (книжковий шифр Енея).

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

У XV столітті чернець Трітеміус (1462–1516), який займався криптографією і стеганографією, описав багато різних методів прихованої передачі повідомлень. Пізніше, в 1499 році, ці записи були об'єднані в книгу «Steganographia».

Метод приховування інформації за допомогою мікроточки з'явився відразу ж після винаходу Даґером фотографічного процесу, і вперше у військовій справі були використані в часи франко-пруської війни (в 1870 році), але широкого застосування до Другої світової війни цей метод не мав.

В березні 2000 року 17-річна американська школярка Вівіана Риска (Viviana Risca) створила алгоритм, який може «ховати» повідомлення в генну послідовність ДНК. На конкурсі молодих вчених компанії Intel Science Talent Search[en] вона продемонструвала технологію впровадження комп'ютерних повідомлень в генну послідовність молекули.

Розглядаючи програмні засоби захисту, доцільно спинитись на стеганографічних методах. Слово «стеганографія» означає приховане письмо, яке не дає можливості сторонній особі дізнатися про його існування. Одна з перших згадок про застосування тайнопису датується V століттям до н. е. Сучасним прикладом є випадок роздрукування на ЕОМ контрактів з малопомітними

викривленнями обрисів окремих символів тексту — так вносилась шифрована інформація про умови складання контракту.

Комп'ютерна стеганографія базується на двох принципах. По-перше, аудіо- і відеофайли, а також файли з оцифрованими зображеннями можна деякою мірою змінити без втрати функціональності. По-друге, можливості людини розрізняти дрібні зміни кольору або звуку обмежені. Методи стеганографії дають можливість замінити несуттєві частки даних на конфіденційну інформацію. Сімейна цифрова фотографія може містити комерційну інформацію, а файл із записом сонати Гайдна — приватний лист.

Але найчастіше стеганографія використовується для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення — цифрові водяні знаки записуються як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Щодо впровадження засобів програмно-технічного захисту в ІС, розрізняють два основні його способи:

- додатковий захист — засоби захисту є доповненням до основних програмних і апаратних засобів комп'ютерної системи;
- вбудований захист — механізми захисту реалізуються у вигляді окремих компонентів ІС або розподілені за іншими компонентами системи.

Перший спосіб є гнучкішим, його механізми можна додавати і вилучати за потребою, але під час його реалізації можуть постати проблеми забезпечення сумісності засобів захисту між собою та з програмно-технічним комплексом ІС. Вмонтований захист вважається більш надійним і оптимальним, але є жорстким, оскільки в нього важко внести зміни. Таким доповненням характеристик способів захисту зумовлюється те, що в реальній системі їх комбінують.

Наприкінці 90-х років виділилося кілька напрямків стеганографії:

- Класична стеганографія
- Комп'ютерна стеганографія[4]
- Цифрова стеганографія

Одним з найпоширеніших методів класичної стеганографії є використання симпатичних чорнил (невидимих). Зазвичай процес запису здійснюється наступним чином: перший шар — наноситься важливий запис невидимим чорнилом, другий шар — запис видимими чорнилом, що нічого не значить.

Текст, записаний такими чорнилом, проявляється лише за певних умов (нагрівання, освітлення, хімічний проявник і т. д.).

Винайдені ще в I столітті н. е. Філоном Александрійським, ними користувались як в середньовіччі, так і в новітній час, наприклад, у листах революціонерів з російських в'язниць. Написаний звичайним молоком текст на папері між рядків видимого тексту, проявляється при нагріванні над полум'ям (зазвичай свічки).

Існує також чорнило з хімічно нестабільним пігментом. Написане цими чорнилами виглядає як написане звичайною ручкою, але через певний час нестабільний пігмент розкладається, і від тексту не залишається і сліду. Хоча при використанні звичайної кулькової ручки текст можливо відновити по деформації паперу, цей недолік можна усунути за допомогою м'якого пишучого вузла, на зразок фломастера.

Симпатичними чорнилами можуть слугувати найрізноманітніші речовини: лимонна кислота, віск, яблучний сік, молоко, сік цибулі, слина, пральний порошок, аспірин, крохмаль з різними хімічними чи фізичними «декодерами»: температура, сода, йод, солі, заліза, ультрафіолетове світло, для воску навіть крейда чи зубний порошок.

Комп'ютерна стеганографія — напрям класичної стеганографії, заснований на особливостях комп'ютерної платформи. Приклади — стеганографічна файлова система StegFS для Linux, приховування даних в невикористовуваних областях форматів файлів, підміна символів в назвах файлів, текстова стеганографія і т. д. Наведемо деякі приклади:

- Використання зарезервованих полів комп'ютерних форматів файлів — суть методу полягає в тому, що частина поля розширень, не заповнена інформацією про розширення, за замовчуванням заповнюється нулями. Відповідно ми можемо використовувати цю «нульову» частину для запису своїх даних. Недоліком цього методу є низька ступінь скритності і малий обсяг переданої інформації.
- Метод приховування інформації в невикористовуваних місцях гнучких дисків — при використанні цього методу інформація записується в невживані частини диска, наприклад, на нульову доріжку. Недоліки: маленька продуктивність, передача невеликих за обсягом повідомлень.
- Метод використання особливих властивостей полів форматів, які не відображаються на екрані — цей метод ґрунтується на спеціальних «невидимих» полях для отримання виносок, покажчиків. До прикладу, написання чорним шрифтом на чорному тлі. Недоліки: маленька продуктивність, невеликий обсяг переданої інформації.
- Використання особливостей файлових систем — при зберіганні на жорсткому диску файл завжди (не рахуючи деяких ФС, наприклад, ReiserFS) займає ціле число кластерів (мінімальних адресуються обсягів інформації). До прикладу, в раніше широко використовуваної файлової системи FAT32 (використовувалася в Windows98/Me/2000) стандартний розмір кластера — 4 Кб. Відповідно для зберігання 1 Кб інформації на диску виділяється 4 Кб інформації, з яких 1Кб потрібен для зберігання файлу,, а інші 3 ні на що не використовуються — відповідно їх можна використовувати для зберігання інформації. Недолік даного методу: легкість виявлення.

Розвиток засобів цифрової обчислювальної техніки дав поштовх для розвитку комп'ютерної стеганографії, яка ґрунтується на вбудовуванні секретного повідомлення в цифрові дані, що, як правило, мають аналогову природу (аудіозаписи, зображення, відео). Можливе також вбудовування інформації в текстові та скомпресовані файли.

Цифрова стеганографія — напрям класичної стеганографії, заснований на захованні або впровадженні додаткової інформації в цифрові об'єкти, викликаючи при цьому деякі спотворення цих об'єктів. Але, як правило, дані об'єкти є мультимедіа-об'єктами (зображення, відео, аудіо, текстури 3D-об'єктів) та внесення спотворень, які знаходяться нижче межі чутливості середньостатистичної людини, не призводить до помітних змін цих об'єктів.

Крім того, в оцифрованих об'єктах, тобто таких, що спочатку мають аналогову природу, завжди присутній шум квантування; також, при відтворенні цих об'єктів з'являється додатковий аналоговий шум і нелінійні спотворення апаратури, все це сприяє більшій непомітності прихованої інформації.

Останнім часом набули популярності методи, коли прихована інформація передається через комп'ютерні мережі з використанням особливостей роботи протоколів передачі даних. Такі методи одержали назву «мережна стеганографія». Цей термін вперше ввів Кжиштоф Щипьорский (Krzysztof Szczypiorski) в 2003 році. Типові методи мережевої стеганографії включають зміну властивостей одного з мережевих протоколів. Крім того, може використовуватися взаємозв'язок між двома або більше різними протоколами з метою більш надійного приховування передачі секретного повідомлення. Мережева стеганографія охоплює широкий спектр методів, зокрема:

- WLAN стеганографія ґрунтується на методах, які використовуються для передачі стеганограм в бездротових локальних мережах (Wireless Local Area Networks). Практичний приклад WLAN стеганографії — система HICCUPS (Hidden Communication System for Corrupted Networks).
- LACK стеганографія — приховування повідомлень під час розмов з використанням IP-телефонії. Наприклад: використання пакетів, що затримуються, або навмисно пошкоджуються та ігноруються приймачем (цей метод називають LACK — Lost Audio Packets Steganography), або приховування інформації в полях заголовку, які не використовуються.
- DAB- і DVB стеганографія — стеганографічна обробка інформації, що циркулює у мережах цифрового звукового і/або телевізійного мовлення[5][6].

VoIP (англ. *voice over IP*) — технологія передачі медіа даних в реальному часі за допомогою сімейства протоколів TCP/IP. IP-телефонія — система зв'язку, при якій аналоговий звуковий сигнал від одного абонента дискретизується (кодується в цифровий вигляд), компресується і пересилається по цифрових каналах зв'язку до іншого абонента, де проводиться зворотна операція — декомпресія, декодування і відтворення. Розмова відбувається у формі аудіо-потоків за допомогою протоколів RTP (Real-Time Transport Protocol)

LACK — це метод стеганографії для IP-телефонії, який модифікує пакети з голосовим потоком. Він використовує те, що в типових мультимедійних комунікаційних протоколах, таких як RTP, надмірно затримані пакети вважаються приймачем марними і відкидаються.

Принцип функціонування LACK виглядає наступним чином. Передавач (Аліса) вибирає один з пакетів з голосового потоку і його корисне навантаження замінює бітами таємного повідомлення — стеганограмою, яка вбудовується в пакет. Потім обраний пакет навмисно затримується. Кожного разу, коли надмірно

затриманий пакет досягає отримувача, незнайомого з стеганографічною процедурою, він відкидається. Однак, якщо отримувач (Боб) знає про прихований зв'язок, то замість видалення отриманих RTP пакетів, він вилучає приховану інформацію[7].

Існуючі алгоритми вбудовування таємної інформації можна поділити на декілька підгруп:

- Працюючі з самим цифровим сигналом. Наприклад, метод LSB (Least Significant Bit);
- «Впаювання» прихованої інформації. В даному випадку відбувається накладення приховуваного зображення (звуку, іноді тексту) поверх оригіналу. Часто використовується для вбудовування ЦВЗ (цифровий водяний знак).
- Використання особливостей форматів файлів. Сюди можна віднести запис інформації в метадані або в різні інші не використовувані зарезервовані поля файлу.

За способом вбудовування інформації стегоалгоритми можна розділити на лінійні (адитивні: A17, A18, Ь18Б, A21, A25), нелінійні та інші.

LSB (Least Significant Bit, найменший значущий біт) — суть цього методу полягає в заміні останніх значущих бітів у контейнері (зображення[8], аудіо[9] або відеозапису) на біти приховуваного повідомлення. Різниця між порожнім і заповненим контейнерами повинна бути не відчутна для органів сприйняття людини.

Принцип цього методу полягає в наступному: Припустимо, є 8-бітне зображення в градаціях сірого. 00h (00000000b) позначає чорний колір, FFh (11111111b) — білий. Усього є 256 градацій (2 ^ 8). Також припустимо, що повідомлення складається з 1 байта — наприклад, 01101011b. При використанні 2 молодших біт в описах пікселів, нам буде потрібно 4 пікселя. Припустимо, вони чорного кольору. Тоді пікселі, що містять приховане повідомлення, будуть виглядати наступним чином: 00000001 00000010 00000010 00000011. Тоді колір пікселів зміниться: першого — на 1/255, другого і третього — на 2/255 і четвертого — на 3/255. Такі градації, мало того що непомітні для людини, можуть взагалі не відобразитися при використанні низькоякісних пристроїв виведення. В ролі базового контейнера пропонується використовувати файли BMP-зображень високої роздільності з глибиною кольору 24 та 32 біти, таємне зображення може мати розширення .BMP, .GIF, .PNG, .JPEG.

Недоліком методу LSB є чутливість до розміру зображення, тобто чим менший розмір зображення, тим більше будуть відрізнятися два сусідні пікселі, тому пропонується використовувати зображення з великою роздільністю. Також метод «видає себе» при побітовому перегляді зображення, де чітко видно області зображення в які «вбудовано» таємну інформацію. Попри це, метод запису Least Sagnificant Bit є досить популярним, стійким та простим в реалізації.

Підвиди LSB-алгоритмів для растрових зображень без палітри

BlindHide (приховування наосліп). Найпростіший алгоритм: дані записують, починаючи з верхнього лівого кута зображення до правого нижнього — піксел за пікселем. Приховані дані програма записує у наймолодших бітах кольорів пікселя. Приховані дані розподіляються у контейнері нерівномірно. Якщо

приховані дані не заповнять повністю контейнер, то лише верхня частина зображення буде засміченою.

HideSeek (заховати-знайти). Цей алгоритм у псевдовипадковий спосіб розподіляє приховане повідомлення у контейнері. Для генерації випадкової послідовності використовує пароль. Дещо «розумніший» алгоритм, але все ж не враховує особливостей зображення-контейнера.

FilterFirst (попередня фільтрація). Виконує фільтрацію зображення-контейнера — пошук пікселів, у які записуватиметься прихована інформація (для яких зміна наймолодших розрядів буде найменш помітною для ока людини).

BattleSteg (стеганографія морської битви). Найскладніший і найдосконаліший алгоритм. Спочатку виконує фільтрацію зображення-контейнера, після чого прихована інформація записується у «найкращі місця» контейнера у псевдовипадковий спосіб (подібно, як у HideSeek).

Інші методи приховування інформації в графічних файлах орієнтовані на формати файлів з втратою, наприклад, JPEG. На відміну від LSB вони більш стійкі до геометричних перетворень. Це виходить за рахунок варіювання в широкому діапазоні якості зображення, що призводить до неможливості визначення джерела зображення.

Цифрові водяні знаки (ЦВЗ)

Найчастіше стеганографію використовують для створення цифрових водяних знаків. На відміну від звичайних їх можна нанести і відшукати тільки за допомогою спеціального програмного забезпечення — цифрові водяні знаки записують як псевдовипадкові послідовності шумових сигналів, згенерованих на основі секретних ключів. Такі знаки можуть забезпечити автентичність або недоторканість документа, ідентифікувати автора або власника, перевірити права дистриб'ютора або користувача, навіть якщо файл був оброблений або спотворений.

Цифровий водяний знак (ЦВЗ) — технологія, створена для захисту авторських прав мультимедійних файлів та інтелектуальної власності контейнера (Intellectual Property). Зазвичай цифрові водяні знаки невидимі. Однак ЦВЗ можуть бути видимими на зображенні або відео. Зазвичай це інформація являє собою текст або логотип, який ідентифікує автора.

Стеганографія застосовує ЦВЗ, коли сторони обмінюються секретними повідомленнями, впровадженими в цифровий сигнал. Використовується як засіб захисту документів з фотографіями — паспортів, водійських посвідчень, кредитних карт з фотографіями.

ЦВЗ можна також використовувати для виявлення потенційних піратів: під час продажу в зображення вбудовують інформацію про час продажу та інформацію про покупця. Ключовою відмінністю ЦВЗ від звичайного приховання інформації є наявність активного противника. Наприклад, використовуючи ЦВЗ для захисту авторського права, активний противник намагатиметься видалити чи змінити вбудовані ЦВЗ. Тому основною вимогою є стійкість вбудованих даних до атак. Таємність не є настільки важливою, як у прихованій комунікації.

Тема 24. Квантова криптографія

Квантовий розподіл ключа - метод передачі ключа, який використовує квантові явища для гарантії безпечної зв'язку. Цей метод дозволяє двом сторонам, з'єднаним з відкритого каналу зв'язку, створити загальний випадковий ключ, який відомий тільки їм, і використовувати його для шифрування і розшифрування повідомлень.

Важливою і унікальною властивістю квантового розподілу ключа є можливість виявити присутність третьої сторони, яка намагається отримати інформацію про ключ. Тут використовується фундаментальний аспект квантової механіки: процес виміру квантової системи порушує її. Третя сторона, яка намагається отримати ключ, повинна виміряти надіслані через з'єднання квантові стани, що веде до їх зміни і появи аномалії. За допомогою квантової суперпозиції, квантової заплутаності і передачі даних в квантових станах можна здійснити канал зв'язку, який виявляє аномалії.

Якщо кількість аномалій нижче певного порогу, то буде створено, ключ що гарантує безпеку (третя сторона не має інформації про це), інакше секретний ключ не буде створено і зв'язок припиняється. Стан квантового об'єкта (тобто, грубо кажучи, об'єкта дуже малої маси і розмірів, наприклад, електрона або фотона) може бути визначено виміром. Однак відразу після виконання цього виміру квантовий об'єкт неминуче переходить в інший стан, причому передбачити цей стан неможливо.

Отже, якщо в якості носіїв інформації використовувати квантові частинки, то спроба перехопити повідомлення призведе до зміни стану частинок, що дозволить виявити порушення секретності передачі. Крім того, неможливо отримати повну інформацію про квантовий об'єкт, і отже, неможливо його скопіювати. Ці властивості квантових об'єктів роблять їх «невловимими».

Ідея використовувати квантові об'єкти для захисту інформації від підробки та несанкціонованого доступу вперше була висловлена Стефаном Вейснер (Stephen Weisner) в 1970 р Через 10 років Беннет і Брассард, які були знайомі з роботою Вейснер, запропонували використовувати квантові об'єкти для передачі секретного ключа. У 1984 р вони опублікували статтю, в якій описувався протокол квантового поширення ключа BB84. Носіями інформації в протоколі BB84 є фотони, поляризовані під кутами 0, 45, 90, 135 градусів. Відповідно до законів квантової фізики, за допомогою вимірювання можна розрізнити лише два ортогональних станів: якщо відомо, що фотон поляризований або вертикально, або горизонтально, то шляхом вимірювання, можна встановити - як саме; те ж саме можна стверджувати щодо поляризації під кутами 45 і 135 градусів. Однак точно відрізнити вертикально поляризований фотон від фотона, поляризованого під кутом 45 градусів, неможливо. Ці особливості поведінки квантових об'єктів лягли в основу протоколу квантового розподілу ключа.

Щоб обмінятися ключем, Аліса і Боб роблять такі дії: 1) Аліса посилає Бобу фотон в одному з поляризованих станів (0, 45, 90, 135 градусів) і записує кут поляризації. Відлік кутів ведеться від напрямку "вертикально вгору" за годинниковою стрілкою. В реальних же системах перед процесом передачі ключа обладнання спеціально юстирується для забезпечення однакового режиму відліку

на приймачі і передавачі (причому цю юстировку доводиться проводити періодично в процесі передачі), а "просторове розташування" початку відліку кута – несуттєво; 2) Боб в своєму розпорядженні має два аналізатори: один розпізнає вертикально-горизонтальну поляризацію, інший - діагональну.

Для кожного фотона Боб випадково вибирає один з аналізаторів і записує тип аналізатора і результат вимірювань; 3) По загальнодоступному каналу зв'язку Боб повідомляє Алісі, які аналізатори використовувалися, але не повідомляє, які результати були отримані; 4) Аліса по загальнодоступному каналу зв'язку повідомляє Бобу, які аналізатори він вибрав правильно. Ті фотони, для яких Боб невірно вибрав аналізатор, відкидаються.

Квантова передача включає шифрування інформації в квантові стани, або кубіти, на відміну від класичної передачі, що використовує біти. Як правило, використовуються фотони для квантових станів. Квантовий розподіл ключів використовує певні властивості квантових станів для забезпечення безпеки. Існує різні підходи квантового розподілу ключів, але вони можуть бути розділені на дві основні категорії, в залежності від властивостей, які вони використовують.

Протокол підготовки та вимірювання. На відміну від фізики, вимір є невід'ємною частиною квантової фізики. Вимірювання невідомого квантового стану змінює його в деякому роді. Це відомо як квантовий індетермінізм і лежить в основі результатів, таких як принцип невизначеності Гейзенберга і теореми про заборону клонування. Це може бути використано для того щоб виявити будь-які прослушки на зв'язку і, що більш важливо, для розрахунку кількості інформації, яка була перехоплена. Протоколи засновані на заплутаності. Квантові стани двох (або більше) окремих об'єктів можуть бути з'єднані таким чином, що вони будуть описуватися за допомогою комбінованого квантового стану, а не як індивідуальний об'єкт. Це називається заплутаністю і означає, що вимірювання на один об'єкт впливає і на інший. Якщо сплутана пара об'єктів є спільною між двома учасниками, то перехоплення будь-якого об'єкта змінює систему в цілому, розкриваючи присутність третіх осіб (і кількість інформації, яку вони отримали).

Розглянемо схему фізичної реалізації квантової криптографії. Зліва знаходиться відправник, праворуч - одержувач. Для того, щоб передавач мав можливість імпульсно варіювати поляризацію квантового потоку, а приймач міг аналізувати імпульси поляризації, використовуються комірки Поккельса. Передавачем формується одне з чотирьох можливих станів поляризації. На комірки дані надходять у вигляді керуючих сигналів.

Для організації каналу зв'язку зазвичай використовується волокно, а в якості джерела світла беруть лазер. На стороні одержувача після комірки Поккельса розташована кальцитова призма, яка повинна розщеплювати пучок на дві складові, що вловлюються двома фотодетекторами (ФЕП), а ті, в свою чергу, вимірюють ортогональні складові поляризації. Спочатку необхідно вирішити проблему інтенсивності переданих імпульсів квантів, що виникає при їх формуванні. Якщо в імпульсі міститься 1000 квантів, існує ймовірність того, що 100 з них будуть відведені криптоаналітиків на свій приймач. Після чого, проводячи аналіз відкритих переговорів, він зможе отримати всі необхідні йому

дані. З цього випливає, що ідеальний варіант, коли в імпульсі кількість квантів прагне до одного.

Тоді будь-яка спроба перехопити частину квантів неминуче змінить стан всієї системи і відповідно спровокує збільшення числа помилок у одержувача. У цій ситуації слід не розглядати прийняті дані, а заново повторити передачу. Однак, при спробах зробити канал більш надійним, чутливість приймача підвищується до максимуму, і перед фахівцями постає проблема «темнового» шуму. Це означає, що одержувач приймає сигнал, який не був відправлений адресантом. Щоб передача даних була надійною, логічні нулі і одиниці, з яких складається двійкове подання переданого повідомлення, представляються у вигляді не одного, а послідовності станів, що дозволяє виправляти одинарні і навіть кратні помилки.

Для подальшого збільшення відмовостійкості квантової криптосистеми використовується ефект Ейнштейна - Подільського - Розена, що виникає в тому випадку, якщо сферичним атомом були випроменені в протилежних напрямах два фотона. Початкова поляризація фотонів не визначена, але в силу симетрії їх поляризації завжди протилежні. Це визначає той факт, що поляризацію фотонів можна дізнатися тільки після вимірювання. Криптосхема на основі ефекту Ейнштейна - Подільського - Розена, що гарантує безпеку пересилання, була запропонована Екертом.

Відправником генерується кілька фотонних пар, після чого один фотон з кожної пари він відкладає собі, а другий пересилає адресату. Тоді якщо ефективність реєстрації близько одиниці і на руках у відправника фотон з поляризацією «1», то у одержувача буде фотон з поляризацією «0» і навпаки. Тобто легальні користувачі завжди мають можливість отримати однакові псевдовипадкові послідовності. Але на практиці виявляється, що ефективність реєстрації і вимірювання поляризації фотона дуже мала.

У 1989 році Беннет і Брассар в Дослідницькому центрі IBM побудували першу працюючу квантово-криптографічний систему. Вона складалася з квантового каналу, що містить передавач Аліси на одному кінці і приймач Боба на іншому, розміщені на оптичній лаві довжиною близько метра в світлонепроникному півтораметровому кожусі розміром 0,5 × 0,5 м. Власне квантовий канал був вільний повітряний канал довжиною близько 32 см. Макет управлявся від персонального комп'ютера, який містив програмне уявлення користувачів Аліси і Боба, а також зловмисника.

У тому ж році передача повідомлення за допомогою потоку фотонів через повітряне середовище на відстань 32 см з комп'ютера на комп'ютер завершилася успішно. Основна проблема при збільшенні відстані між приймачем і передавачем - збереження поляризації фотонів. На цьому заснована достовірність способу. Створена за участю Женевського університету компанія GAPOptique під керівництвом Ніколаса Гісіна поєднує теоретичні дослідження з практичною діяльністю. Першим результатом цих досліджень стала реалізація квантового каналу зв'язку за допомогою оптоволоконного кабелю довжиною 23 км, прокладеного по дну озера і з'єднує Женеву і Ніон. Тоді був згенерований секретний ключ, рівень помилок якого не перевищував 1,4%. Але все-таки

величезним недоліком цієї схеми була надзвичайно мала швидкість передачі інформації.

Пізніше фахівцям цієї фірми вдалося передати ключ на відстань 67 км з Женеви до Лозанни за допомогою майже промислового зразка апаратури. Але і цей рекорд був побитий корпорацією Mitsubishi Electric, яка передала квантовий ключ на відстань 87 км, правда, на швидкості в один байт в секунду. Активні дослідження в галузі квантової криптографії ведуть IBM, GAP-Optique, Mitsubishi, Toshiba, Національна лабораторія в ЛосАламосі, Каліфорнійський технологічний інститут, молода компанія MagiQ і холдинг QinetiQ, підтримуваний британським міністерством оборони.

Зокрема, в національній лабораторії Лос-Аламоса була розроблена і почала широко експлуатуватися досвідчена лінія зв'язку, довжиною близько 48 кілометрів. Де на основі принципів квантової криптографії відбувається розподіл ключа, і швидкість розподілу може досягати кілька десятків кбіт / с. У 2001 році Ендрю Шилдс і його колеги з TREL і Кембріджського університету створили діод, здатний випускати одиночні фотони. В основі нового світлодіода лежить «квантова точка» - мініатюрний шматочок напівпровідникового матеріалу діаметром 15 нм і товщиною 5 нм, який може при подачі на нього струму захоплювати лише по одній парі електронів і дірок. Це дало можливість передавати поляризовані фотони на більшу відстань.

В ході експериментальної демонстрації вдалося передати зашифровані дані зі швидкістю 75 Кбіт / с - при тому, що більше половини фотонів втрачалося. В Оксфордському університеті ставляться завдання підвищення швидкості передачі даних. Створюються квантово-криптографічні схеми, в яких використовуються квантові підсилювачі. Їх застосування сприяє подоланню обмеження швидкості в квантовому каналі і, як наслідок, розширення сфери практичного застосування подібних систем. В Університеті Джона Хопкінса на квантовому каналі довжиною 1 км побудована обчислювальна мережа, в якій кожні 10 хвилин проводиться автоматичне підстроювання. В результаті цього, рівень помилки знижено до 0,5% при швидкості зв'язку 5 кбіт / с. Міністерством оборони Великобританії підтримується дослідницька корпорація QinetiQ, яка є частиною колишнього британського агентства DERA (Defence Evaluation and Research Agency), яка спеціалізується на неядерних оборонних дослідженнях і активно удосконалює технологію квантового шифрування. Дослідженнями в галузі квантової криптографії займається американська компанія Magiq Technologies з Нью-Йорка, що випустила прототип комерційної квантової кріптотехнологіі власної розробки. Основний продукт Magiq - засіб для розподілу ключа (quantum key distribution, QKD), яке названо Navajo (за назвою племені індіанців Навахо, мова яких під час Другої світової війни американці використовували для передачі секретних повідомлень, оскільки за межами США його ніхто не знав).

Navajo здатний в реальному часі генерувати і поширювати ключі засобами квантових технологій і призначений для забезпечення захисту від внутрішніх і зовнішніх зловмисників. У жовтні 2007 року на виборах в Швейцарії були повсюдно використані квантові мережі, починаючи виборчими дільницями і закінчуючи датацентрі ЦВК. Була використана техніка, яку ще в середині 90-х в

Університеті Женеви розробив професор Ніколя жизень. Також одним з учасників створення такої системи була компанія Id Quantique. У 2011 році в Токіо відбулася демонстрація проекту «Tokyo QKD Network», в ході якого розробляється квантове шифрування телекомунікаційних мереж. Була проведена пробна телеконференція на відстані в 45 км. Зв'язок в системі йде за звичайними оптоволоконними лініями. В майбутньому передбачається застосування для мобільного зв'язку.

В теорії квантова комунікація ідеально безпечна. На практиці ж виявляється, що зараз існують різні лазівки, якими «квантові хакери» можуть скористатися, що було показано дослідницькими групами ХойКвон Ло і Вадима Макарова [6]. Лазівки можуть виникнути і в джерелі, і в детекторі. Реальні джерела і детектори рідко повністю відповідають своїм ідеальним теоретичним прототипам, використовуваним при доказі безпеки квантової комунікації. Наприклад, в одному з видів детекторів gated detector ефективність детекції змінюється за допомогою зміщення напруги і, таким чином, залежить від часу. В ідеальній ситуації існує два детектора: один для нульового біта, а інший для одиниці, і важливо упевнитися в тому, що ефективність детектування в обох детекторів абсолютно однакова. Група Хой-Квон Ло показала, що маленьке розбіжність у часі зміщення напруги може спричинити за собою значне неспівпадання ефективності детектування. Такий принцип атаки за допомогою тимчасового зсуву. І це тільки один з прикладів. Лазівки в системах КРК можуть бути різноманітними.

Список використаних джерел

1. Hoffstein J., Pipher J., Silverman J. An Introduction to Mathematical Cryptography. Springer Science+Business Media, LLC, 2008. 524 p.
2. Jeffrey H., Jill P., Joseph H. An Introduction to Mathematical Cryptography. Berlin: Springer, 2008. 540 p.
3. Задірака В.К., Олексюк О.С. Комп'ютерна криптологія. Тернопіль, Київ, 2002. 504 с.
4. Фергюссон Н., Шнайер Б. Практическая криптография. М.: Вильямс, 2005. 424 с.
5. Вербіцький О.В. Вступ до криптології. Львів: ВНТЛ, 1998. 247 с.
6. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО, 2006. 336 с.
7. Shoup V. A Computational Introduction to Number Theory and Algebra. Cambridge University Press, 2005. 517 p.
8. Stillwell J. Elements of Number Theory. Springer, 2010. 256 p.
9. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях. М. : Радио и связь, 2001. 376 с.
10. Столлингс В. Криптография и защита сетей. Принципы и практика. М.: Вильямс. 2001. 669 с.
11. Diffie W., Hellman M. New directions in cryptography. *IEEE Transactions on Information Theory*. 1976. Vol. 22, №6. P. 644–654.
12. Rivest R., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*. 1978. Vol. 21, Iss. 2. P. 120–126.
13. Ян С. Криптоанализ RSA. Ижевск: РХД. 2011. 312 с.
14. Srivastava A., Mathur A. The Rabin cryptosystem and analysis in measure of chinese reminder theorem. *International Journal of Scientific and Research Publications*. 2013. Vol. 3 (6). P. 1-4.
15. Hayder R.H. H-Rabim Cryptosystem. *Journal of Mathematics and Statistics*. 2014. Vol. 10 (3). P. 304-308.
16. Коблиц Н. Курс теории чисел и криптографии. М.: ТВП. 2001. 254с.
17. ElGamal T. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*. 1985. Vol. 31 (4). P. 469–472.
18. Adki V., Hatkar S. A Survey on Cryptography Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*. 2016. Vol. 6 (6). P. 469-475.
19. Washington L. Elliptic Curves Number Theory and Cryptography. Series Discrete Mathematics and Its Applications, Chapman & Hall/CRC, 2008. 524 p.
20. Якименко І.З., Тимошенко Л.М., Касянчук М.М. Вибір параметрів еліптичних кривих у задачах шифрування інформаційних потоків. *Сучасна спеціальна техніка*. 2018. № 2. С. 63–71.
21. Kasianchuk M. Yakymenko I., Pazdriy I., Melnyk A., Ivasiev S. Rabin's modified method of encryption using various forms of system of residual classes. *The Experience of Designing and Application of CAD Systems in Microelectronics* (*CADSM-2017*): Proceedings of the XIV International Conference. Polyana-Svalyava. 2017. P.222-224.
22. Касянчук М., Карпінський М., Казмірчук С. Методологія опрацювання багаторозрядних чисел в асиметричних криптосистемах. *Захист інформації*. 2019. Т.21, №2. С. 65- 73.