ЗМІСТ

ВСТУП

В даний час дуже широко використовується термін «комп'ютерна безпека». Насправді комп'ютер схильний лише кільком ризикам, якщо він по мережі не підключений до інших комп'ютерів. За останній час відсоток використання комп'ютерних мереж (особливо Інтернету) значно виріс, тому сьогодні термін «комп'ютерна безпека» використовується для опису проблем, пов'язаних з мережевим використанням комп'ютерів і їх ресурсів.

Основними технічними складовими комп'ютерної безпеки є:

1. Конфіденційність.
2. Цілісність.
3. Аутентифікація.
4. Доступність.

Конфіденційність, також відома як секретність, означає, що у неавторизованих користувачів не буде доступу до вашої інформації. Наслідки, які можуть бути викликані прогалинами в конфіденційності, можуть варіюватися від незначних до руйнівних.

Цілісність означає, що ваша інформація захищена від неавторизованих змін, що не відноситься до авторизованих користувачів. Загрозу цілісності баз даних і ресурсів, як правило, представляє хакерство.

Аутентифікація – це сервіс контролю доступу, який здійснює перевірку реєстраційної інформації користувача. Іншими словами, це означає, що користувач – це є насправді той, за кого він себе видає.

Доступність включає в себе розмежування прав користувачів за доступом до ресурсів автоматизованої системи і захист цієї системи та її елементів від несанкціонованого доступу. *Доступність* означає те, що ресурси доступні авторизованим користувачам.

Іншими важливими компонентами, яким велика увага приділяється професіоналами в області комп'ютерної безпеки, є контроль над доступом і суворе виконання зобов'язань. Контроль над доступом має на увазі не тільки факт, що користувач має доступ тільки до наявних ресурсів і послуг, але й той факт, що у нього є право доступу до ресурсів, які він законно очікує. Що стосується суворого виконання зобов'язань, то це має на увазі неможливість відмови користувачу того, що він відправив повідомлення і навпаки.

Концепція комп'ютерної безпеки дуже велика, тому до даних технічних аспектів є й інші додатки. Основними питаннями, пов'язаними з цим терміном, є комп'ютерний злочин (спроби запобігти, виявити атаки) та конфіденційність / анонімність в кіберпросторі.

Питання безпеки – важлива частина концепції впровадження нових інформаційних технологій у всі сфери життя суспільства. Тому широкомасштабне використання обчислювальної техніки та телекомунікаційних систем призводить до якісно нових можливостей несанкціонованого доступу до ресурсів та даних інформаційної системи, до їх високої вразливості. Таким чином, забезпечення цілісності, достовірності і доступності інформації – важливі складові успіху діяльності будь-якої організації.

Хоча конфіденційність, цілісність, аутентифікація є важливими компонентами комп'ютерної безпеки, для користувачів Інтернету найбільш важливою складовою є конфіденційність, тому що більшість користувачів думають, що їм нема чого приховувати, або інформація, яку вони надають при реєстрації на сайті, не є секретною.

В Інтернеті інформація дуже швидко поширюється серед компаній і потроху зібрана інформація з різних джерел може багато чого сказати про людину. Тому можливість контролю інформації, для чого вона збирається, хто і як може нею скористатися, є дуже серйозним і важливим питанням в контексті комп'ютерної безпеки.

Ефективність механізмів захисту інформації значною мірою залежить від реалізації ряду принципів. По-перше, механізми захисту треба проектувати одночасно з розробкою інформаційної системи, що дозволяє забезпечити їх безконфліктність, своєчасну інтеграцію в обчислювальне середовище та скорочення витрат. По-друге, питання захисту треба розглядати комплексно в рамках єдиної системи захисту інформації.

Спеціальне програмне забезпечення для захисту інформації ПК

Для захисту ПК використовуються різні програмні методи, які значно розширюють можливості по забезпеченню безпеки інформації, що зберігається. Серед стандартних захисних засобів ПК найбільше поширення отримали:

1. Засоби захисту обчислювальних ресурсів, що використовують *парольну ідентифікацію* і обмежують доступ несанкціонованого користувача.
2. Застосування різних *методів шифрування (криптографічних методів),* що не залежать від контексту інформації.
3. Засоби захисту від копіювання комерційних програмних продуктів.
4. Захист від комп'ютерних вірусів і створення архівів.

У найпростішому випадку завжди можна скористатися апаратними засобами встановлення пароля на запуск операційної системи ПК з допомогою установок в CMOS Setup. На жаль, використання подібної парольної ідентифікації не є надійним. Досить ввести універсальний пароль (AWARD_SW) або відключити акумуляторну батарею, розташовану на материнській платі, і комп'ютер забуде всі налаштування CMOS Setup.

Використання криптографії

Можливість використання ПК в локальних мережах (при сполученні їх з іншими ПК) або застосування «модемів» для обміну інформацією по телефонних дротах пред'являє більш жорсткі вимоги до програмного забезпечення щодо захисту інформації ПК. Споживачі ПК в різних організаціях для обміну інформацією все ширше використовують електронну пошту, яка без додаткових засобів захисту може стати надбанням сторонніх осіб.

Найнадійнішим захистом від несанкціонованого доступу до передаваної інформації та програмних продуктів ПК є застосування різних криптографічних методів захисту інформації (КМЗІ).

Криптографія – це розділ прикладної математики, який вивчає методи перетворення інформації з метою хования її змісту.

КМЗІ – це спеціальні методи шифрування, кодування або іншого перетворення інформації, в результаті якого її зміст стає недоступним без пред'явлення ключа криптограми і зворотного перетворення. Криптографічний метод захисту, безумовно, найбільш надійний метод захисту, оскільки охороняється безпосередньо сама інформація, а не доступ до неї (наприклад, зашифрований файл не можна прочитати навіть у випадку крадіжки носія). Даний метод захисту реалізується у вигляді програм або пакетів програм, що розширюють можливості стандартної операційної системи. Захист на рівні операційної системи, частіше за все, повинен доповнюватися засобами захисту на рівні систем управління базами даних, які дозволяють реалізовувати складні процедури управління доступом.

В даний час не існує загальноприйнятої класифікації КМЗІ. Проте, коли піддається перетворенню (шифровці) кожен символ передаваного повідомлення («симетричний» метод закриття інформації), можна умовно виділити чотири основні групи:

1. Підстановка – символи тексту, що шифрується, замінюються символами того ж або іншого алфавіту відповідно до заздалегідь зазначеного правила.

2. Перестановка – символи тексту, що шифрується, переставляються по деякому правилу в межах заданого блоку передаваного тексту.

3. Аналітичне перетворення – текст, що потрібно зашифрувати, перетворюється за деяким аналітичним правилом.

4. Комбіноване перетворення – вихідний текст шифрується двома або великим числом способів шифрування. Існує велика кількість програмних продуктів шифрування інформації, що розрізняються за ступенем надійності.

1 ЕЛЕМЕНТИ КЛАСИЧНОЇ КРИПТОГРАФІЇ

1.1 Основні поняття

Відомо, що передача інформації від об'єкту до адресата проводиться за допомогою сигналів. Для того, щоб сигнали були однозначно зрозумілі, їх необхідно складати за правилом, яке строго фіксоване протягом всього часу передачі даної групи повідомлень.

Визначення. *Повідомленням* називається скінченна або нескінченна послідовність букв в деякому алфавіті.

Повідомлення породжуються деякими джерелами. Розглядаються джерела двох типів – комбінаторні та імовірнісні.

Комбінаторне джерело здатне породити тільки деякі повідомлення. Скажімо, припустимо вважати, що речення української або англійської мови створені деяким комбінаторним джерелом. *Імовірнісні* джерела породжують в принципі будь-які повідомлення, але різні повідомлення мають різні ймовірності.

Комбінаторним джерелом називається деяка скінченна множина S. В якості таких джерел, як правило, будемо розглядати множини слів деякого алфавіту (словники).

Скінченним імовірнісним джерелом назвемо довільну скінченну множину S із визначеним на ній розподілом ймовірностей. Цю множину будемо називати алфавітом (абстрактним алфавітом), а його елементи — буквами.

В подальшому розглядатимемо імовірнісне джерело, що створює повідомлення, в яких послідовні букви з'являються незалежно із заданою ймовірністю.

Визначення. Правило (алгоритм), що зіставляє кожному конкретному повідомленню строго певну комбінацію різних символів (або відповідних їм сигналів), називається *кодом*, а процес перетворення повідомлення в послідовність кодів — *кодуванням*. Послідовність символів, яка в процесі кодування привласнюється кожному з множини повідомлень, що передаються, називається *кодовим словом*.

Символи, за допомогою яких записано повідомлення, що передається, складають *первинний алфавіт* $A=\{a_1, a_2, \ldots, a_n\}$, а символи, за допомогою яких повідомлення трансформується в кодові слова, – *вторинний алфавіт* $D=\{d_1, d_2, \ldots, d_m\}$.

Кодова система є спосіб коду з кожною буквою алфавіту, або, строгіше, функція, яка пов'язує код з кожною буквою алфавіту: f: A–>D.

Потужність d вторинного алфавіту D будемо називати *основою* кодової системи: d=|D|.

Кодовою системою будемо називати сукупність кодів, побудованих за правилом

f: A–>D, $f(a_i) = d_i$, $a_i \in A$, $d_i \in D$, з d.

У випадку вторинного алфавіту D={0, 1} *кодом* називають послідовність двійкових цифр у вторинному алфавіті, що складається з нуля і одиниці, кодова система f: A–>{0,1}, $f(a_i) = d_i$, $d_i \in \{0,1\}$, з d=2.

Кодова система *має властивість префікса*, якщо жоден код не є префіксом будь-якого іншого коду тієї ж кодової системи. Це полегшує розшифровку повідомлення, оскільки для того, щоб знайти першу букву повідомлення, що дешифрується, треба лише прочитати послідовність двійкових цифр, поки вона не співпаде з одним з кодів.

Кодова система Хаффмена є системою з мінімальною надмірністю, тобто серед всіх двійкових кодових систем змінної довжини, що мають властивість префікса, вона має найменшу вартість.

Визначення. *Розбиттям* A множини S (не обов'язково скінченної) називається сукупність попарно непересічних множин A_1, A_2, …, A_k, $k \geq 1$, які дають в об'єднанні множину S. Множини A_i, $1 \leq i \leq k$, називають *атомами* розбиття A.

Найменш інформативним є розбиття, єдиним атомом якого служить сама множина S. Найбільш інформативним є розбиття, атомами якого служать одноелементні множини, тобто множини $\{A_1\}$, $\{A_2\}$, …, $\{A_k\}$.

Ентропією H(S) імовірнісного джерела називається ентропія його найбільш інформативного розбиття:

$$H(S) = -\sum_{i=1}^{m} p(A_i) * \log_2 p(A_i) \quad , \qquad (1)$$

де A_i – елементи множини S, $S=\{A_1, ..., A_m\}$.

S – імовірнісне джерело, $S=\{a_1, a_2, ..., a_k\}$, $p(a_i)$ – ймовірність, з якою породжується буква a_i, i=1, ... , k.

Вартість C(S) визначається як середнє число цифр вторинного алфавіту на букву первинного повідомлення в припущенні, що повідомлення складене з букв, вибраних незалежно одна від одної, кожна із заданою ймовірністю.

Кодування f ставить у відповідність букві a_i слово $f(a_i)$ довжини $l_i = |f(a_i)|$, $l_i = L(a_i)$. При цьому *вартістю* кодування f на джерелі S називається середня довжина кодового слова, тобто

$$C(f, S) = \sum_{i=1}^{k} p(A_i) | f(A_i) | \qquad (2)$$

Надмірність R(f, S)= C(f, S) – H(S),

де C(f, S) — вартість кодування, H(S) — ентропія джерела.

Стиск інформації полягає в кодуванні з невеликою надмірністю. Представляє інтерес розгляд тих кодувань, що дешифруються, префіксних, взаємно однозначних кодувань.

Префіксною множиною кодової системи ми називатимемо множину всіх букв алфавіту, які мають коди, що починаються з даного префікса.

Якщо відома яка-небудь префіксна множина, то задачу кодування можна звести до задачі кодування для меншого алфавіту. В схемі кодування Хаффмена використовується алгоритм, згідно якому дві букви з найменшою ймовірністю повинні складати префіксну множину. Букви префіксної множини об'єднуються в одну, і утворюється нова буква, інформативність якої дорівнює інформативності об'єднаних букв, а ймовірність якої дорівнює сумі ймовірностей букв префіксної множини. Цей алгоритм застосовується багаторазово, поки не залишаться лише дві букви, і задача не буде розв'язана.

1.2 Кодова система з мінімальною надмірністю. Алгоритм побудови кодової системи Хаффмена

Число елементів множини повідомлень позначимо через N. Нехай *P(i)* – ймовірність і-го повідомлення. Тоді

$$\sum_{i=1}^{N} P(i) = 1$$

.

Довжина L(i) і-го повідомлення є число кодових цифр, що йому зіставляються. Тому середня довжина повідомлення рівна

$$L_{ср} = \sum_{i=1}^{N} P(i)L(i)$$

Накладемо наступні основні обмеження на кодову систему множини букв первинного алфавіту:

1) ніякі два коди повідомлень не складаються з однакових послідовностей кодових цифр;

2) кодова система має властивість префікса;

3) l(1)<=l(2)<=...<=l(N–1)=l(N);

4) принаймні два (і не більше потужності вторинного алфавіту) повідомлення довжини l(N) мають коди ідентичні, за виключенням своїх останніх цифр.

З обмеження 3) випливає, що два найменш ймовірних повідомлення повинні мати однакову довжину. Обмеження 4) зводиться до вимоги, щоб при потужності вторинного алфавіту, що дорівнює двом, було б тільки два повідомлення довжини l(N), коди яких однакові за виключенням їх останніх цифр. Останніми цифрами кодів цих повідомлень будуть 0 або 1. Необхідно зіставити ці два коди повідомлень N-му та (N–1)-му повідомленням, оскільки поки ще не відомо, чи існують інші коди повідомлень довжини l(N). Як тільки це виконано, два згаданих повідомлення стають еквівалентними одному складеному повідомленню. Його код (поки ще не визначений) буде загальним префіксом порядку l(N)–1 цих двох повідомлень. Імовірність складеного повідомлення дорівнює сумі ймовірностей двох повідомлень, що його складають.

Множина, що містить це складене повідомлення замість двох повідомлень, що його складають, буде називатися *першою допоміжною множиною* повідомлень. Ця щойно утворена множина містить на одне повідомлення менше, ніж початкова. Її елементи треба впорядкувати у відповідності до їх ймовірностей (за зменшенням). Отриману множину можна розглядати в точності так само, як і початкову. Необхідно, щоб коди двох найменш імовірних повідомлень цієї нової множини співпадали за виключенням їх останніх цифр; за ці цифри приймають 0 та 1, по одній для кожного повідомлення. Кожна нова допоміжна множина містить на одне повідомлення менше, ніж попередня.

Ця процедура повторюється до тих пір, поки число елементів в останній з утворених допоміжних множин не зменшиться до двох. Кожному з цих двох складених повідомлень зіставляється по одній двійковій цифрі. Потім ці повідомлення об’єднуються, утворюючи одне повідомлення, яке має імовірність, що дорівнює 1, і кодування завершено.

В таблиці 1.1 наведено *приклад* побудови кодової системи для алфавіту, що

містить 13 повідомлень. В лівій колонці містяться впорядковані за величиною ймовірності повідомлення множини, що кодується, з потужністю N=|A|=13. Оскільки при комбінуванні двох повідомлень (позначених дужкою) кожному з них зіставляється нова цифра, то повна кількість цифр, зіставлених кожному початковому повідомленню, дорівнює числу комбінацій, в які входить це повідомлення. Наприклад, повідомлення, помічене зірочкою, або складене повідомлення, частиною якого воно є, комбінується з іншими п’ять разів, і тому його код повинен мати довжину в п’ять цифр. Сам код складається з кодових знаків, що зіставляються об’єднаним ймовірностям, починаючи з найменшої, записаних у зворотному порядку.

В таблиці 1.2 наведено ймовірності повідомлень та відповідні коди побудованої кодової системи Хаффмена та їхні довжини.

1.3 Метод оптимального кодування

Накладемо ще одне (п’яте) обмеження на кодову систему множини повідомлень: 5) для будь-якої послідовності з L(N) –1 цифр, що не є префіксом кодів повідомлень, або вона сама, або один з її префіксів використовуються як код якого-небудь повідомлення. Узагальнимо описаний метод побудови кодової системи Хаффмена для вторинного алфавіту більшої потужності (d>2).

Оптимальне кодування множини повідомлень за допомогою трьох або більшого числа цифр аналогічно процедурі двійкового кодування. Потрібно користуватися таблицею допоміжної множини повідомлень, подібній таблиці 1.1. Дужки, що помічають повідомлення, які комбінуються в складені, використовуються таким же чином, як це робилося в таблиці 1. Проте для задоволення умови 5) потрібно зажадати, щоб всі ці дужки, окрім, мабуть, однієї комбінації найменше імовірних повідомлень початкової множини, об'єднували завжди D повідомлень (D – число символів вторинного алфавіту).

Необхідно відзначити, що остання допоміжна множина завжди має одне повідомлення з імовірністю, рівною 1. Число елементів в кожній попередній множині збільшується на D–1, поки ми не приходимо до першої допоміжної множини. Тому, якщо число повідомлень в першій допоміжній множині дорівнює N_1, то число $(N_1–1)/(D–1)$ повинно бути цілим числом. Проте $N_1=N – n_0+1$, де n_0 – число найменше імовірних повідомлень, що об'єднуються дужкою в початковій множині. Тому n_0, яке щонайменше дорівнює 2 і не перевищує D, повинно мати таке значення, щоб число $(N–n_0)/(D–1)$ було цілим.

У таблиці 1.3 розглянуто приклад множини з 8 повідомлень, яку потрібно закодувати за допомогою чотирьох цифр; при цьому n_0 виявилося рівним 2. Код, виписаний в таблиці, одержано зіставленням кожній дужці, окрім першої, чотирьох цифр 0, 1, 2, 3 по порядку.

Оскільки на першому кроці об’єднуються 2 найменш ймовірних повідомлення, їм зіставляються кодові цифри 0 та 1. Кодову систему, побудовану за методом оптимального кодування за меншу кількість кроків, називають *оптимальною нерівномірною кодовою системою*. Вона містить коди меншої довжини, ніж коди Хаффмена, і тому найменшу вартість серед усіх кодових систем для того ж алфавіту.

Таблиця 1.1

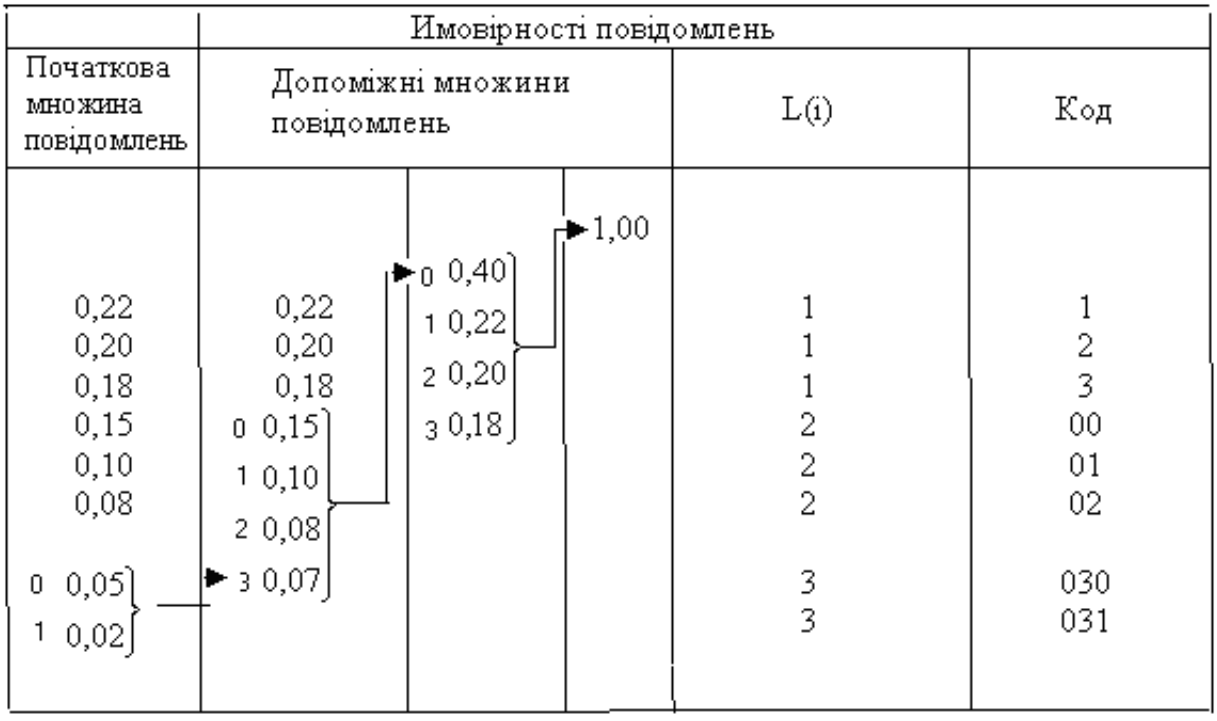
$p(A_i)$	1	2	3	4	5	6	7	8	9	10	11
											0→0,60 }
										→0,40	1 0,40 ⌋
									→0,36	0 0,36 }	
								→0,24	0,24	1 0,24 ⌋	
0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0,20	0 0,20 }		
							→0,20	0,20	1 0,20 ⌋		
0,18	0,18	0,18	0,18	0,18	0,18	0,18	0,18	0 0,18 }			
						→0,18	0,18	1 0,18 ⌋			
					→0,14	0,14	0 0,14 }				
0,10	0,10	0,10	0,10	0,10	0,10	0,10	1 0,10 ⌋				
0,10	0,10	0,10	0,10	0,10	0,10	0 0,10 }					
0,10	0,10	0,10	0,10	0,10	0,10	1 0,10 ⌋					
				→0,10	0 0,10 }						
		→ 0,08	0,08	0,08	1 0,08 ⌋						
			→ 0,08	0 0,08 }							
0,06	0,06	0,06	0,06	1 0,06 ⌋							
0,06	0,06	0,06	0 0,06 }								
0,04	0,04	0,04	1 0,04 ⌋								
0,04	0,04	0 0,04 }									
0,04 *	0,04	1 0,04 ⌋									
0,04	0 0,04 }										
	1 → 0,04 ⌋										
0 0.03 } 1 0.01 ⌋											

$$H=-\sum_{i=1}^{N} p(A_i)\log_2 p(A_i);\ C=\sum_{i=1}^{N} p(A_i)\,l(A_i)$$

Таблиця *1.2 Результати оптимального кодування*

i	P(i)	l(i)	P(i) l(i)	Код
1	0.20	2	0.40	10
2	0.18	3	0.54	000
3	0.10	3	0.30	011
4	0.10	3	0.30	110
5	0.10	3	0.30	111
6	0.06	4	0.24	0101
7	0.06	5	0.30	00100
8	0.04	5	0.20	00101
9	0.04	5	0.20	01000
10	0.04	5	0.20	01001
11	0.04	5	0.20	00110
12	0.03	6	0.18	001110
13	0.01	6	0.06	001111
			L_{cp}=3,42	

Таблиця *1.3 Процедура оптимального кодування для D=4*



1.4 Алфавітні кодові системи

Теорема 1. Нехай S – ймовірнісне джерело, що виробляє повідомлення, які можна упорядкувати (алфавітно), і нехай H(S) обчислена за допомогою рівності (1) (лабораторної роботи 1) з ймовірностей p_i видачі джерелом S різних блоків довжиною N знаків. Тоді існує алфавітна кодова система, що однозначно дешифрується, яка зв'язує блоки з N знаків джерела S з їх кодовими послідовностями, вартість якої C(S) задовольняє нерівності:

$$H \le C(S) \le H+2 \qquad (3)$$

Обираючи N достатньо великим, можна зробити величину C(S) скільки завгодно близькою до ентропії H (двійкових одиниць на знак) джерела S.

Доведення.

Нехай всі можливі блоки з N знаків джерела перелічені в алфавітному порядку, і p_i означає ймовірність появи i-го блока переліку. Нехай m_i – ціле число, для якого

$$2^{-m_i} \le p_i < 2^{1-m_i}. \qquad (4)$$

Окрім того, визначимо числа A_1, A_2, A_3, ... наступним чином:

$$A_1 = p_1/2,\ A_2 = p_1 + p_2/2,\ ...,\ A_i = (p_1 + ... + p_{i-1}) + p_i/2. \qquad (5)$$

Зауважимо, що $0 \le A_1 \le A_2 \le ... \le A_i \le ... \le 1$.

Побудуємо тепер алфавітну кодову систему.

Код i-го блока складають перші m_i+1 цифр двійкового розвинення числа A_i. В кодовій системі Шеннона той самий блок має код, утворений з перших m_i цифр розвинення деякого іншого числа.

Вартість кодової системи Шеннона позначимо $C(H_n')$. Оскільки в нашій схемі для кожного блока використовується лише на одну цифру більше, ніж в схемі Шеннона, то $C(S)=C(H_n')+1$ і з нерівності (1) випливає нерівність (3)

$$H \le C(S) \le H+2.$$

Залишається показати, що наша кодова система є такою, яка однозначно дешифрується, тобто що з кодової послідовності двійкових цифр можна однозначно відновити послідовність букв, вироблених джерелом.

Для цього достатньо довести, що наша побудова утворює перелік кодів, який має властивість префікса. Тоді кожне зашифроване повідомлення, вироблене блоком з N букв, може бути розшифровано, як тільки прийняті всі його цифри.

Щоб довести, що наш перелік має властивість префікса, розглянемо два блоки букв, наприклад, i-й та j-й, причому $i<j$. З (5) маємо:

$$A_j \ge A_i + \frac{p_j}{2} + \frac{p_i}{2}$$

$$A_j \ge A_i + 2^{-1-m_j} + 2^{-1-m_i} \qquad (6)$$

$$A_j = p_1 + ... + p_{i-1} + p_i + ... + p_{j-1} + \frac{p_j}{2}$$

Якщо $p_i \le p_j$, то $m_i \ge m_j$; але на основі (6) j-й код не може тотожно співпадати з першими $1+m_j$ позиціями i-го кода.

Подібно цьому, якщо $p_i \ge p_j$, i-й код не може бути префиксом j-го кода. Таким чином, властивість префікса, а разом з нею і теорема доведені.

Інша кодова система може бути отримана шляхом використання побудови такого самого ряду, але при $A_i = \sum_{j=1}^{i-1} 2^{-m_j} + 2^{-m_i-1}$.

В цьому випадку можна застосувати те ж саме доведення, оскільки нерівність (6) залишається в силі. Оскільки довжини кодів знову дорівнюють числам m_i+1, то ця кодова система має таку ж вартість. Але тепер числа A_j можна легко обчислити прямо в двійковій системі й тим самим уникнути багатьох арифметичних обчислень, необхідних для побудови першої системи.

В кодовій алфавітній системі всяка префіксна множина повинна складатися зі всіх букв, що знаходяться між двома даними буквами алфавіту.

Прийоми, описані нижче, дозволяють доводити, що деякі сукупності букв повинні бути префіксними множинами в будь-якій алфавітній кодовій системі. Якщо відома яка-небудь префіксна множина, то кодування можна звести до кодування для меншого алфавіту.

Теорема 2. Нехай в алфавітній кодовій системі множина букв S є префіксна множина для деякого префікса π. Побудуємо скорочений алфавіт, замінивши букви множини S на одну нову букву L^1, що займає їх місце в алфавітному порядку і що має ймовірність, яка дорівнює сумі ймовірностей колишніх букв. Тоді існує кодова система для нового алфавіту, в якій буква L^1 має код π, а всяка інша буква має свій попередній код.

У схемі кодування Хаффмена використовується припущення, аналогічне теоремі 2, але для неалфавітних кодових систем. Згідно йому дві букви з найменшою імовірністю повинні складати префіксну множину. Букви префіксної множини об'єднуються, і утворюється нова буква, інформативність якої дорівнює інформативності об'єднаних букв, а ймовірність якої дорівнює сумі ймовірностей букв префіксної множини. Ця ітерація застосовується багато разів, поки не залишаться лише дві букви, і сума їх імовірностей не буде дорівнювати 1. Якщо кодова система повинна бути алфавітною, то префіксну множину не завжди легко знайти.

1.5 Загальний алгоритм алфавітного кодування

У методі, який застосовуватиметься в загальному випадку, ми будуємо найкращу алфавітну кодову систему для всього алфавіту, складаючи спочатку найкращі алфавітні кодові системи для деяких підалфавітів. Зокрема, ми розглядатимемо лише такі підалфавіти, які можуть утворювати множини в деякій алфавітній двійковій кодовій системі для всього алфавіту.

Оскільки префіксною множиною може бути лише множина букв, що складається зі всіх тих букв, які знаходяться між деякими двома буквами алфавіту L_i та L_j, то таку множину ми називатимемо *допустимим підалфавітом.*

Позначимо допустимий підалфавіт, що складається зі всіх букв, наступних за L_i в алфавіті (включаючи саму L_i) і передуючих L_j (включаючи саму L_j), через (L_i, L_j). Так, (A, A) позначає підалфавіт, що містить лише букву A.

Якщо потрібно знайти оптимальну кодову систему, що задовольняє обмеженням, відмінним від алфавітного, то можна використовувати інші допустимі підалфавіти, причому в іншому алгоритм не змінюється.

Термін *вартість кодової системи* застосовується для середнього числа двійкових цифр на букву переданого повідомлення, тобто для $\sum_i p_i l(A_i)$.

Оскільки в алгоритмі, який описуватиметься, ми будуватимемо кодову систему для кожного допустимого підалфавіту, то ми також застосовуватимемо відповідну суму для кожного підалфавіту. Але оскільки вірогідність p_i для відповідних підалфавітів не складає в сумі 1, то $\sum_i p_i l(A_i)$ не відповідає в точності вартості передаваних повідомлень, і тому відповідна сума називатиметься *частковою вартістю* $p_ч$.

Алгоритм складається з n кроків, де n — число букв алфавіту. На k-му кроці будується найкраща алфавітна кодова система для кожного допустимого підалфавіту з k букв і обчислюється її часткова вартість. При k=1 кожен підалфавіт вигляду (L_i, L_i) кодується тривіальною кодовою системою, в якій кодом L_i є порожня послідовність; вона має вартість

$p_ч = 0$ (позначимо її $p_{Li\ ,\ Li}$),

оскільки число цифр в порожній послідовності рівне 0. При k=2 кожен підалфавіт вигляду (L_i , L_{i+1}) кодується кодовою системою, в якій кодом L_i буде символ ‘0’, а кодом L_{i+1} буде символ ‘1’. Часткова вартість $p_ч$ (позначимо її $p_{Li\ ,\ Li+1}$) цієї кодової системи дорівнює $p_i + p_{i+1}$. Взагалі на k-му кроці алгоритму, на якому відшукується найкраща алфавітна двійкова кодова система для кожного підалфавіту вигляду (L_i , L_{i+k-1}) і її часткова вартість $p_ч$ (позначимо її $p_{Li\ ,\ Li+k-1}$), використовуються коди і часткові вартості, знайдені на попередніх кроках.

Для кожного j між (i+1) та (i+k–1) можна визначити двійкову алфавітну кодову систему таким чином.

Хай існують C_i , C_{i+1}, ... C_{j-1} — коди для букв L_i , L_{i+1}, ..., L_{j-1} відповідно, вказані раніше побудованою найкращою алфавітною кодовою системою для підалфавіту (L_i , L_{j-1}), і хай C_j, C_{j+1}, ..., C_{i+k-1} — коди для букв L_j, L_{j+1}, ..., L_{i+k-1} відповідно, вказані раніше побудованою найкращою алфавітною кодовою системою для підалфавіту (L_j, L_{i+k-1}). Тоді нова кодова система для повідомлення

L_i, L_{i+1}, ..., L_{j-1}, L_j, L_{j+1}, ..., L_{i+k-1}

буде

$0C_i$, $0C_{i+1}$, ..., $0C_{j-1}$, $1C_j$, $1C_{j+1}$, ..., C_{i+k-1},

тобто одержана шляхом приписування до кодів всіх букв підалфавіту (L_i, L_{j-1}) зліва нуля, а до кодів всіх букв підалфавіту (L_j, L_{i+k-1}) — ліворуч одиниці і конкатенації (склеювання) одержаних кодових підсистем.

Таким чином, можна визначити кодову систему для кожного j, і ця кодова система є повною. З теореми 2 витікає, що найкраща кодова система для цього підалфавіту буде однією з k–1 таких кодових систем, які можна одержати при k–1 різних значеннях j. Часткова вартість $p_{Li\ ,\ Li+k-1}$ такої кодової системи, що

складається з двох кодових підсистем, дорівнює сумі часткових вартостей двох підсистем плюс сума ймовірністей всіх букв підалфавіту (L_j, L_{i+k-1}), тобто

$$p_{Li\,,\,Li+k-1} = p_{Li\,,\,Lj-1} + p_{Lj\,,\,Li+k-1} + p_i + p_{i+1} + \ldots + p_{j-1} + p_j + p_{j+1} + \ldots + p_{i+k-1}.$$

Для здійснення алгоритму не потрібно будувати всі ці кодові системи, а досить лише провести обчислення, щоб вирішити яка з k–1 різних кодових систем має найменшу часткову вартість. Для цього беруть суми часткових вартостей кожної з k–1 пар кодових систем і вибирають найкращу кодову систему.

Виконавши k-й крок цього алгоритму для k=1, 2, ..., n, одержують врешті-решт кодову систему, яка є *найкращою алфавітною кодовою* системою для всього первинного алфавіту, причому остання одержана часткова вартість буде дорівнювати вартості цієї найкращої алфавітної кодової системи.

Розглянемо приклад. Хай задане скінченне імовірнісне джерело, що є латинським алфавітом (включаючи пробіл) з визначеним на ньому розподілом вірогідності появи окремих букв в тексті (дивися таблицю 1).

Застосуємо описаний вище алгоритм побудови якнайкращої алфавітної кодової системи до цього алфавіту на прикладі перших п'яти кроків для підалфавітів довжини від 1 до 5 з букв A, B, C, D, E. Для цього складемо допустимі підалфавіти довжини від 1 до 5 і знайдемо їх часткові вартості. У таблиці 1.4 представлені дані букви і їх вірогідність.

Таблиця 1.4

Буква	Вірогідність
A	0,0642
B	0,0127
C	0,0218
D	0,0317
E	0,1031

Крок 1. k=1. Будуємо допустимі підалфавіти (Li, Li) довжини 1 (їх буде 5):
(A, A), (B, B), (C, C), (D, D), (E, E).

Такі підалфавіти кодуються порожньою послідовністю, а часткові вартості рівні нулю:

$p_{A,A} = 0$, $p_{B,B} = 0$, $p_{C,C} = 0$, $p_{D,D} = 0$, $p_{E,E} = 0$.

Крок 2. k=2.Строим допустимі підалфавіти (L_i, L_{i+1}) довжини 2 (їх буде 4):
(A, B), (B, C), (C, D), (D, E).

Їх часткові вартості знайдемо шляхом підсумовування вірогідності букв, що входять в підалфавіт:

$p_{A,B} = p_A + p_B = 0.0642 + 0.0127 = 0.0769$;
$p_{B,C} = p_B + p_C = 0.0127 + 0.0218 = 0.0345$;
$p_{C,D} = p_C + p_D = 0.0218 + 0.0317 = 0.0535$;
$p_{D,E} = p_D + p_E = 0.0317 + 0.1031 = 0.1348$.

Код першої букви в кожному підалфавіті дорівнює 0, а другої – 1, тобто коди всіх підалфавітів будуть (0, 1).

Складемо таблицю 1.5, в якій будуть представлені підалфавіти, їх часткові вартості і кодові підсистеми.

Таблиця 1.5

Підалфавіт	Часткова вартість	Кодова підсистема
A,B	0,0769	(0, 1)
B, C	0,0345	(0, 1)
C, D	0,0535	(0, 1)
D, E	0,1348	(0, 1)

Крок 3. k=3.Строим допустимі підалфавіти (L_i, L_{i+2}) довжини 3 (їх буде три):

(A, B, C), (B, C, D), (C, D, E).

Для знаходження відповідних кодових підсистем використовуватимемо коди і часткові вартості, одержані на перших двох кроках. Для цього виконаємо розбиття допустимих підалфавітів довжини 3 на підалфавіти меншої довжини (такого розбиття буде два):

(A, B, C): (A, A), (B, C);

(A, B), (C, C).

Знайдемо часткові вартості $p^1_{A,C}$ і, відповідні розбиттю (A, B, C):

$p^1_{A,C} = p_{A,A} + p_{B,C} + pA + pB + pC =$

$0+0.0345+0.0642+0.0127+0.0218=0.1332;$

$p^2_{A,C} = p_{A,B} + p_{C,C} + pA + pB + pC =$

$0.0769+0+0.0642+0.0127+0.0218=0.1756.$

Менша часткова вартість відповідає розбиттю (A, A), (B, C), по якому і побудуємо код для (A, B, C): кодом для (A, A) є порожня послідовність, до якої припишемо зліва 0 і одержимо 0; кодом для (B, C) є (0, 1), до кожної цифри якого припишемо зліва 1 і одержимо (10, 11). Таким чином, кодом для (A, B, C) буде двійкова підсистема (0, 10, 11) з вартістю $= p_{A,C} = p^1_{A,C}\, 0{,}1332.$

Аналогічно виконаємо розбиття підалфавітів (B, C, D), (C, D, E) і побудуємо коди.

(B, C, D): (B, B), (C, D);

(B, C), (D, D).

Знайдемо часткові вартості $p^1_{B,D}$ і, відповідні розбиттю (B, C, D):

$p^1_{B,D} = p_{B,B} + p_{C,D} + pB + pC + pD =$

$0+0.0535+0.0127+0.0218+0.0317=0.1197;$

$p^2_{B,D} = p_{B,C} + p_{D,D} + pB + pC + pD =$

$0.0345+0+0.0127+0.0218+0.0317=0.1007;$

По розбиттю (B, C), (D, D) побудуємо код для (B, C, D): кодом (B, C) є (0, 1), тому до кожної цифри коду припишемо зліва 0 і одержимо (00, 10); кодом (D, D) є порожня послідовність, до якої припишемо зліва 1 і одержимо 1. Таким чином, кодом для (B, C, D) буде підсистема (00, 01, 1) з вартістю

$p_{B,D} = p^2_{B,D} = 0{,}1007.$

(C, D, E): (C, C), (D, E);

(C, D), (E, E).

Знайдемо часткові вартості $p^1_{C,E}$ і, відповідні розбиттю (C, D, E):

$p^1_{C,E} = p_{C,C} + p_{D,E} + pC + pD + pE = 0+0.1348+0.0218+0.0317+0.1031=0.2914;$

$p^2_{C,E} = p_{C,D} + p_{E,E} + pC + pD + pE =0.535+0+0.0218+0.0317+0.1031=0.2001.$

Меншу вартість має друге розбиття, тому кодом для (C, D, E) буде двійкова підсистема (00, 01, 1), вартість $p_{C,E}$ якої дорівнює $p^2_{C,E}$=0,2001.

Складемо таблицю 1.6, в якій будуть представлені підалфавіти довжини 3, їх розбиття, відповідні часткові вартості і кодові підсистеми.

Таблиця 1.6

Підалфавіт	Розбиття	Часткова вартість	Кодова підсистема
A, B, C	(A, A)(B,C)	0.1332	(0, 10, 11)
	(A,B)(C, C)	0.1756	
B, C, D	(B,B)(C,D)	0,1197	(00, 01, 1)
	(B,C)(D,D)	0.1007	
C, D, E	(C, C)(D,E)	0,2914	(00, 01, 1)
	(C,D)(E,E)	0,2001	

Крок 4. k=4.Строим допустимі підалфавіти (L_i, L_{i+3}) довжини 4 (їх буде два):

(A, B, C, D), (B, C, D, E).

Для знаходження відповідних кодових підсистем використовуватимемо коди і часткові вартості, одержані на перших трьох кроках. Для цього виконаємо розбиття допустимих підалфавітів довжини 4 на підалфавіти меншої довжини (такого розбиття буде три):

(A, B, C, D): (A, A), (B, C, D);
(A, B), (C, D);
(A, B, C), (D, D).

Знайдемо часткові вартості, $p^2_{A,D}$ і, відповідні розбиттю (A, B, C, D):

$p^1_{A,D} = p_{A,A} + p_{B,D} + pA + pB + pC + pD =$
$= 0+0.1007+0.0642+0.0127+0.0218+0.0317 = 0.2311;$

$p^2_{A,D} = p_{A,B} + p_{C,D} + pA + pB + pC + pD =$
$= 0.0769+0.535+0.0642+0.0127+0.0218+0.0317=0.2608.$

$p^3_{A,D} = p_{A,C} + p_{D,D} + pA + pB + pC + pD =$
$= 0.1332+0+0.0642+0.0127+0.0218+0.0317=0.2636.$

Найменшу вартість має перше розбиття, тому кодом для підалфавіту (A, B, C, D) буде двійкова підсистема (0, 100, 101, 11), часткова вартість $p_{A,D}$ якої рівна $p_{A,D} = p^1_{A,D}$=0,2311.

(B, C, D, E): (B, B), (C, D, E);
(B,C), (D, E);

(B, C, D), (E, E).

Знайдемо часткові вартості, $p^2_{B,E}$ і, відповідні розбиттю (B, C, D, E):

$p^1_{B,E} = p_{B,B} + p_{C,E} + pB + pC + pD + pE =$

$= 0+0.2001+0.0127+0.0218+0.0317+0.1031 = 0.3694;$

$p^2_{B,E} = p_{B,C} + p_{D,E} + p_B + pC + pD + pE =$

$= 0.0345+0.1348+0.0127+0.0218+0.0317+0.1031 =0.3386;$

$p^3_{B,E} = p_{B,D} + p_{E,E} + pB + pC + pD + pE =$

$= 0.1007+0+0.0127+0.0218+0.0317+0.1031=0.27.$

Найменшу вартість має третє розбиття, тому кодом для підалфавіту (B, C, D, E) буде двійкова підсистема (000, 001, 01, 1), часткова вартість $p_{B,E}$ якої дорівнює $p_{B,E} = p^3_{B,E} = 0{,}27.$

Складемо таблицю 1.7, в якій будуть представлені підалфавіти довжини 4, їх розбиття, відповідні часткові вартості і кодові підсистеми.

Таблиця 1.7

Підалфавіт	Розбиття	Часткова вартість	Кодова підсистема
A, B, C, D	(A, A)(B,C,D)	0.2311	(0, 100, 101, 11)
	(A,B)(C,D)	0.2608	
	(A,B,C),(D,D)	0.2636	
B, C, D, E	(B,B)(C,D,E)	0,3694	(000, 001, 01, 1)
	(B,C)(D,E)	0.3386	
	(B,C,D)(E,E)	0.27	

Крок 5. k=5.Строим допустимий підалфавіт (L_i, L_{i+4}) довжини 5 (він буде один):

(A, B, C, D, E).

Для знаходження відповідної кодової підсистеми використовуватимемо коди і часткові вартості, одержані на перших чотирьох кроках. Для цього виконаємо розбиття допустимого підалфавіту довжини 5 на підалфавіти меншої довжини (такого розбиття буде чотири):

(A, B, C, D, E): (A, A), (B, C, D, E);
(A, B), (C, D, E);
(A, B, C), (D, E);
(A, B, C, D), (E, E).

Знайдемо часткові вартості,, $p^3_{A,E}$ і, відповідні розбиттю (A, B, C, D, E):

$p^1_{A,E} = p_{A,A} + p_{B,E} + pA + pB + pC + pD + pE =$

$= 0+0.27+0.0642+0.0127+0.0218+0.0317+0.1031 = 0.5035;$

$p^2_{A,E} = p_{A,B} + p_{C,E} + pA + pB + pC + pD + pE =$

$= 0.0769+0.2001+0.0642+0.0127+0.0218+0.0317+0.1031=0.5105;$

$p^3_{A,E} = p_{A,C} + p_{D,E} + pA + pB + pC + pD + pE =$

$= 0.1332+0.1348+0.0642+0.0127+0.0218+0.0317+0.1031=0.5015;$

$p^4_{A,E} = p_{A,D} + p_{E,E} + pA + pB + pC + pD + pE =$

$= 0.1332+0.1348+0.0642+0.0127+0.0218+0.0317+0.1031=0.4646$.

Найменшу вартість $p^4_{A,E} = 0.4646$ має четверте розбиття, тому кодом для підалфавіту (A, B, C, D, E) буде двійкова підсистема, одержана з кодової підсистеми (0, 100, 101, 11) підалфавіту (A, B, C, D) і порожнього коду підалфавіту (E, E) за вказаним вище правилом: (00, 0100, 0101, 011, 1). Часткова вартість $p_{A,E}$ побудованої кодової підсистеми рівна вартості всієї системи алфавіту з п'яти букв (A, B, C, D, E) і рівна $p_{A,E} = p^4_{A,E} = 0{,}4646$.

Складемо таблицю 1.8, в якій буде представлений підалфавіт довжини 5, його розбиття і відповідні їм часткові вартості, а також кодова підсистема.

Таблиця 1.8

Підалфавіт	Розбиття	Часткова вартість	Кодова підсистема
A, B, C, D, E	(A, A)(B,C,D,E)	0.5035	(0, 100, 101, 11)
	(A,B)(C,D,E)	0.5105	
	(A,B,C),(D,E)	0.5015	
		0.4646	

Таким чином, одержана двійкова підсистема (**00**, **0100**, **0101**, **011**, **1**) є найкращою алфавітною кодовою системою для алфавіту {A, B, C, D, E}.

Продовжуючи описані побудови для підалфавітів більшої довжини, можна одержати якнайкращі алфавітні кодові системи для будь-якого алфавіту довільної довжини.

1.6 Обгрунтування алгоритмів побудови кодових систем

Кодова система називається *повною*, якщо вона кодує алфавіт з двох або більшого числа букв чином, що однозначно дешифрується, і для всякої нескінченної послідовності $x=x_1x_2x_3...$ двійкових цифр існує повідомлення, яке може бути закодоване як x, або якщо вона кодує алфавіт з однієї букви за допомогою порожньої послідовності.

Теорема 3. Всяка найкраща алфавітна кодова система є повною.

Лема 1. Нехай π – деякий префікс у вторинному алфавіті D ={0,1}. Якщо в найкращій алфавітній кодовій системі є код з префіксом π0, то є код з префіксом π1. Навпаки, якщо є код з префіксом π1, то є код з префіксом π0.

Лема 2. Хай L_a – буква алфавіту з найменшою вірогідністю. Тоді в будь-якій найкращій алфавітній кодовій системі буква L_a разом з однією з букв L_{a+1} або L_{a-1} утворює префіксну множину.

Зауваження. Лема 2 і теорема 3 справедливі, коли буква L_a – єдина буква з найменшою імовірністю.

Якщо найменш імовірна буква знаходиться в кінці алфавіту, як це має місце для імовірності таблиці 1.1, то ця буква має лише одну сусідню букву і повинна складати разом з нею префіксну множину. Таким чином, як перший крок при складанні таблиці 1.1 ми можемо написати:

$$C(Y)=\pi(Y,Z)0$$
$$C(Z)=\pi(Y,Z)1$$

де $\pi(Y,Z)$ – якийсь невідомий префікс. Тоді за допомогою теореми 2 завдання приводиться до складання кодової системи для алфавіту з 26 букв, в якому Y і Z замінені однією буквою L(Y,Z) з вірогідністю 0,0169, рівній сумі вірогідності букв Y і Z відповідно. Вирішивши це нове завдання, ми знайдемо $\pi(Y,Z)$ як код для L(Y,Z). Тепер найменше вірогідними буквами є J або Q, що мають одну і ту ж вірогідність 0,0008; J, наприклад, може бути в одній префіксній множині з I або K, але лема 2 не дає вказівок для вибору між ними.

Теорема 4. Хай L_a – буква з найменшою вірогідністю. Якщо $p_{a+1} > p_a + p_{a-1}$, то букви L_a і L_{a-1} у будь-якій якнайкращій алфавітній кодовій системі повинні складати префіксну множину. Аналогічно, якщо $p_{a-1} > p_a + p_{a+1}$, то букви L_a і L_{a+1} повинні утворити префіксну множину.

Побудовані алфавітні кодові системи підтверджують справедливість сформульованих тверджень (лем і теорем), які спрощують аналіз властивостей кодових систем.

2 КРИПТОГРАФІЧНІ МЕТОДИ ЗАХИСТУ ІНФОРМАЦІЇ ВІД НЕСАНКЦІОНОВАНОГО ДОСТУПУ

Вступ

З розвитком електронних технологій і комунікацій проблема захисту інформаційних ресурсів від несанкціонованого доступу (НСД), підробки і модифікації придбаває особливу актуальність і значущість, а вивчення сучасних методів протидії вказаним деструктивним діям є необхідною умовою при підготовці кваліфікованих фахівців.

Розглянемо деякі елементи теорії, систему прийомів і способів дослідження криптографічних методів захисту інформації (електронних документів) від НСД, підробки і модифікації. Пропоновані індивідуальні завдання для самостійної роботи дозволяють оцінити рівень отриманих теоретичних знань і якість проведених досліджень, а також сприяють формуванню необхідних навичок використання методів захисту і протидії, що вивчаються, на практиці.

2.1 Принципи побудови криптографічних систем

Криптосистема – це система, реалізована програмно, апаратно або програмно-апаратно, що здійснює криптографічне перетворення інформації. Вона складається з простору ключів, відкритих текстів, шифротекстів (криптограм) та алгоритмів зашифрування і розшифрування

Одним з ефективних методів захисту інформації від несанкціонованого доступу є її спеціальне перетворення, що полягає в приведенні складових її елементів (слів, букв, цифр) з допомогою спеціальних алгоритмів до виду, що не дозволяє відтворити початкові дані без знання секрету зворотного перетворення (відновлення) або спеціального ключа. Таке перетворення інформації називається зашифруванням або криптографічним перетворенням і здійснюється з метою її приховання від сторонніх осіб, а також забезпечення її достовірності і цілісності. Інформацію, що підлягає зашифруванню, називають відкритим текстом m. Результат зашифрування відкритого тексту називають шифрованим текстом *c* або криптограмою. Застосування до шифрованого тексту *c* зворотного перетворення з метою отримати відкритий текст m називають розшифруванням. Шифратором прийнято називати спеціальний технічний пристрій, що реалізовує зашифрування і розшифрування інформації. Шифром називають сукупність алгоритмів або однозначних відображень відкритого тексту m в недоступний для сприйняття шифрований текст *c*. Ключем k називають деякий секретний параметр шифру, що дозволяє вибрати для шифрування тільки одне конкретне перетворення E_k з усієї множини перетворень, що становлять шифр. Під криптостійкістю розуміють потенційну здатність шифру протистояти розкриттю. Для цього стійкий шифр повинен задовольняти вимогам:

1. Простір ключів повинен мати достатню потужність, щоб перебір усіх можливих перетворень E_k був неможливим.
2. По криптограмі $c = E_k$ (m) дуже важко визначити k і/або m.

Для шифрування відкритого тексту m використовується спеціальний алгоритм, що реалізовується вручну або технічним пристроєм (механічним, електричним, ЕОМ). Секретність перетворення досягається за рахунок використання унікального (не відомого зловмисникові) алгоритму або ключа, що забезпечує кожного разу оригінальне шифрування інформації. Проте з розвитком криптографії базовим принципом сучасних систем шифрування стало правило Киркгофа (Kerckhoff, 1835-1903), згідно з яким популярність супротивнику алгоритму перетворення не повинна знижувати надійність системи шифрування, а її криптостійкість визначається тільки секретністю (належним збереженням в таємниці від сторонніх) і якістю використовуваних криптографічних ключів. Таким чином, без знання секретного ключа розшифрування має бути практично нездійсненним, навіть при відомому алгоритмі шифрування.

Криптографічна система складається з наступних компонент:

1. Простору відкритих текстів *M*.
2. Простору ключів *K*.
3. Простору шифрованих текстів (криптограм) *C*.
4. Двох функцій: $E_k : M \to C$ (зашифрування) і $D_k : C \to M$ (розшифрування) для $k \in K$ таких, що

$E_k(m) = c$ (де c – криптограма, $c \in C$)

$D_k(c) = m$ (де m – відкритий текст, $m \in M$)

$D_k(E_k(m)) = m$ (для будь-якого відкритого тексту $m \in M$).

Симетричними називаються криптосистеми, в яких для зашифрування і розшифрування інформації використовується один і той же ключ, званий *секретним*, що обумовлює інші найменування таких систем : одноключові або криптосистеми з секретним ключем. Узагальнена схема симетричної криптосистеми приведена на рисунку 2.1.

Симетричні криптосистеми можуть реалізовуватися на різних алгоритмах (стандартах) шифрування з секретним ключем, які можна розділити на *блокові* і *потокові*.

При *блоковому* шифруванні відкритий текст заздалегідь розбивається на рівні по довжині блоки. Блокові шифри виконують передбачені своїм алгоритмом криптографічні перетворення над одним блоком даних (блоком відкритого тексту або деякою гамованою послідовністю) фіксованої довжини, в результаті яких виходить блок шифрованого тексту такої ж довжини. Після цього аналогічному перетворенню піддається наступний блок даних.

Потокові шифри перетворять відкритий текст в шифрований текст по одному елементу за операцію (потоком – елемент за елементом). Наприклад, біти відкритого тексту складаються по модулю 2 з бітами деякої псевдовипадкової послідовності.

Сучасні симетричні криптосистеми представлені такими широко відомими стандартами як ГОСТ 28147-89 (Росія), DES, AES і Rijndael (США), які є блоковими шифрами. Ці і більшість інших шифрів з секретним ключем засновані на принципі ітерації.

Принцип ітерації (повторення) полягає в багатократному, такому, що

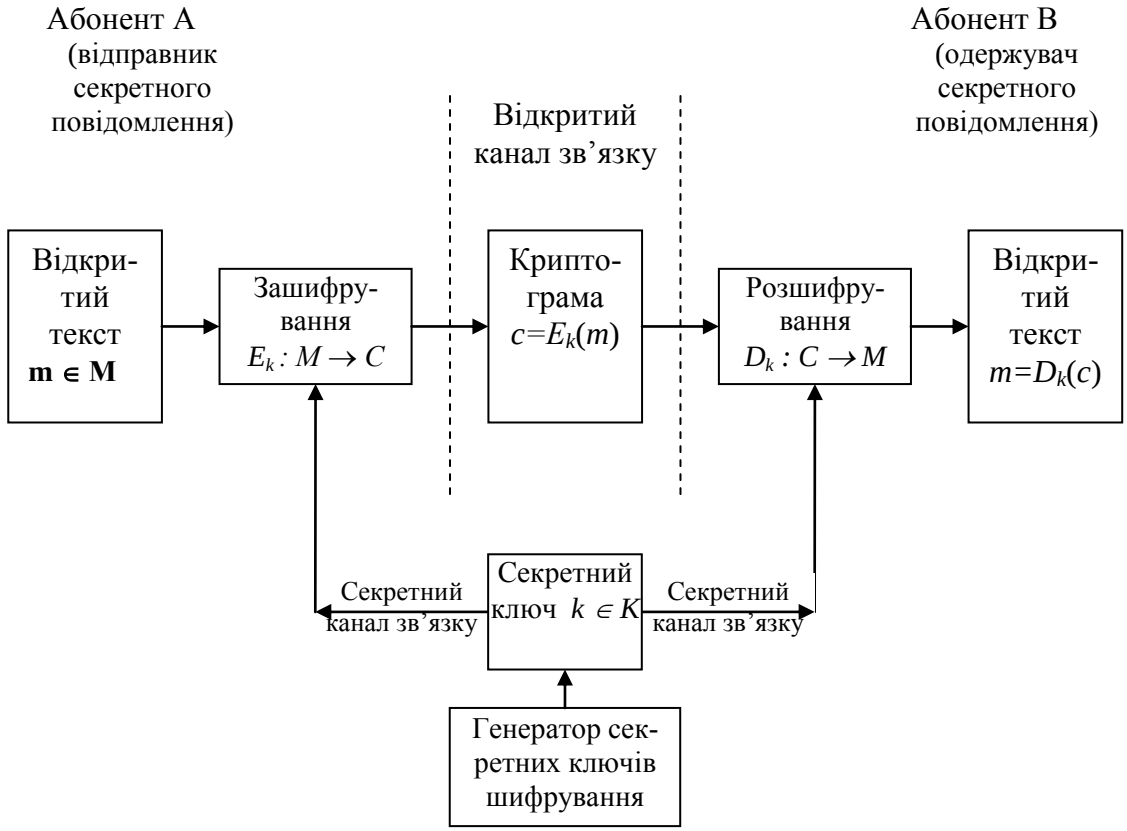


Рис. 2.1 Схема симетричної криптографічної системи

складається з однакових циклів (раундів), перетворенні одного блоку відкритого тексту. Як правило, на кожному раунді перетворення даних здійснюється за допомогою нового допоміжного ключа, отриманого з початкового секретного ключа по спеціальному алгоритму.

Стандарти ГОСТ 28147-89, DES і багато інших відомих шифрів з секретним ключем засновані на використанні конструкції (структури, мережі, петлі) Хорста Фейстеля (H. Feistel). Конструкція Фейстеля полягає в тому, що блок відкритого тексту з парним числом елементів (наприклад, біт) розбивається на дві рівні частини - ліву L і праву R. На кожному раунді одна з частин піддається перетворенню за допомогою функції шифрування f і раундового (допоміжного) ключа ki. Результат цієї операції підсумовується по модулю 2 (позначається на схемі як ⊕) з іншою частиною. Потім ліва L і права R частини міняються місцями і процес перетворення повторюється. Узагальнена схема конструкції Фейстеля представлена на рисунку 2.2.

Перевагою конструкції Фейстеля є те, що пряме і зворотне криптографічне перетворення мають однакову структуру. Тільки при розшифруванні раундові ключі використовуються в зворотному порядку. Недоліком є те, що при кожному раунді перетвориться тільки половина блоку відкритого тексту. Це призводить до необхідності збільшувати число раундів для досягнення необхідної криптостійкості шифру.

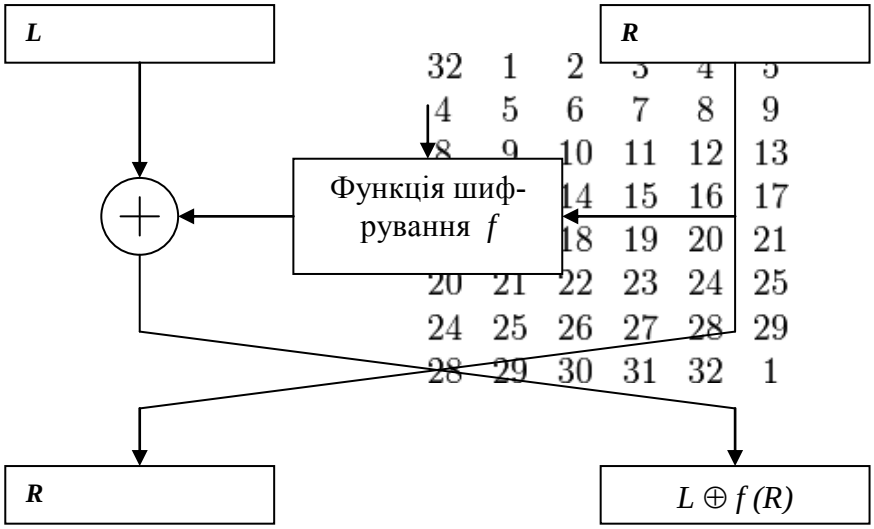


Рис. 2.2 Схема конструкції Фейстеля

Істотним *недоліком* симетричних криптосистем є складність забезпечення безпечної доставки (розподілу) і використання секретних ключів шифрування. Цей недолік виключений в *асиметричних криптосистемах* (інше найменування: двоключові або криптосистеми з відкритим ключем), в яких для зашифрування інформації використовується один ключ, званий відкритим, а для наступного розшифрування (інший ключ, званий закритим (секретним). Узагальнена схема асиметричної криптосистеми приведена на рисунку 2.3.

У асиметричних криптосистемах проблеми з доставкою відкритого ключа не існує, оскільки він ніякого секрету не представляє і може бути відомий усім, хто бажає зашифровувати інформацію. Метод шифрування з відкритим ключем, разом з відкритим розподілом ключів, був запропонований в 1976 році Діффі і Хеллманом, а його перша практична реалізація здійснена Рівестом, Шаміром і Адлеманом (алгоритм RSA).

Істотним недоліком методів шифрування з відкритим ключем є низька швидкодія: вони на 2-3 порядки повільніше за методи шифрування з секретним ключем. У свою чергу, основною (досить складною, що вимагає значних витрат) проблемою при симетричному шифруванні є забезпечення безпечного розподілу (доставки абонентам) секретних криптографічних ключів. Тому на практиці ефективно використовуються гібридні криптосистеми (від латин. hibrida – помісь), що поєднують в собі елементи симетричних і асиметричних криптосистем, і поєднують, відповідно, властиві їм достоїнства: для симетричних методів шифрування – високу швидкість і короткі криптографічні ключі, для асиметричних – можливість відкритого і безпечного розподілу ключів шифрування.

У гібридній криптосистемі методи шифрування з відкритим ключем застосовуються для зашифрування, передачі і наступного розшифрування тільки секретного ключа симетричного шифрування, який безпосередньо застосовується

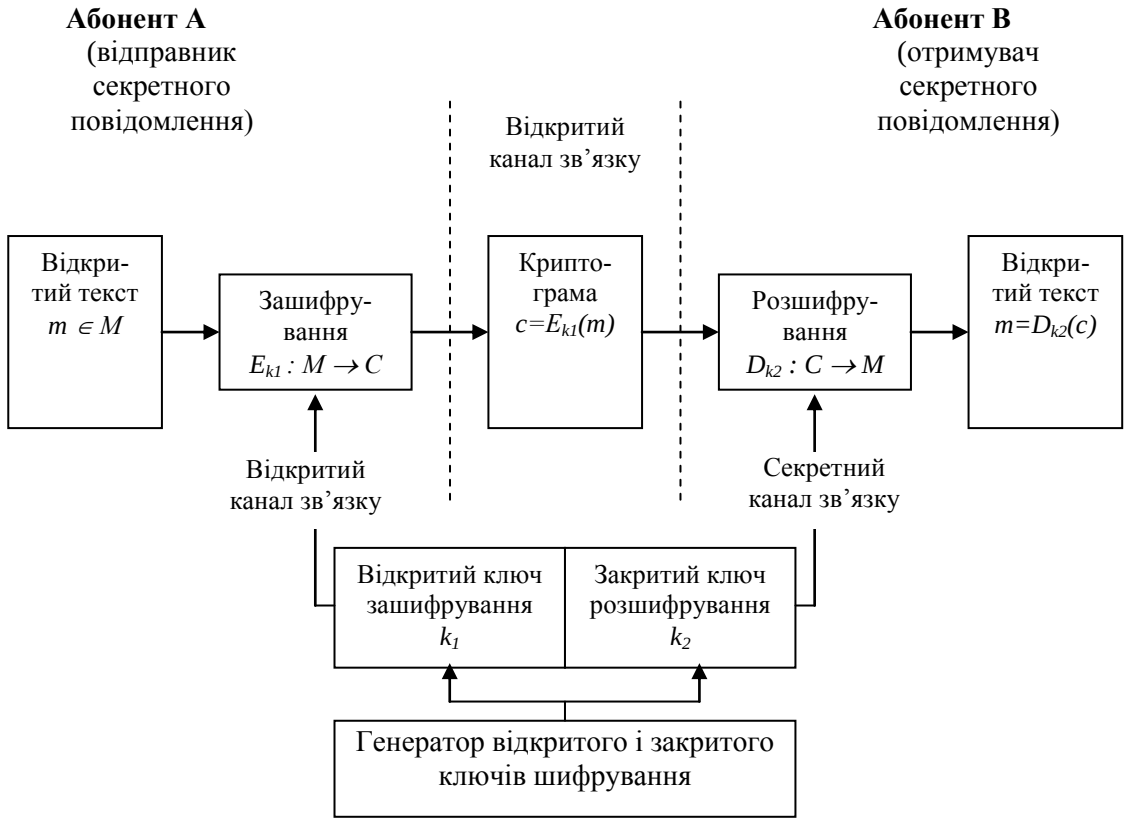


Рис.2.3 Схема асиметричної криптографічної системи

для шифрування передаваних повідомлень (відкритого тексту). Таким чином, асиметрична криптосистема гармонійно доповнює симетричну, забезпечуючи простий і безпечний розподіл (передачу) секретних ключів шифрування. Узагальнена схема гібридної криптосистеми приведена на рисунку 2.4.

Протокол сеансу секретного зв'язку (передачі секретного повідомлення) між абонентом А (відправником) і абонентом В (одержувачем) може бути наступним:

1. Абонент В генерує відкритий (k_1) і закритий (k_2) ключі для асиметричного шифрування, і передає відкритий ключ k1 по відкритому (доступному, незахищеному) каналу зв'язку абонентові А.
2. Абонент А генерує сеансовий секретний криптографічний ключ k для симетричного шифрування і зашифровує на нім підмет передачі секретне повідомлення (відкритий текст) m.
3. Абонент А зашифровує сеансовий секретний криптографічний ключ k на відкритому ключі k_1.
4. Абонент А зашифровує сеансовий секретний криптографічний ключ k на відкритому ключі k_1.
5. Абонент А зашифровує сеансовий секретний криптографічний ключ k на відкритому ключі k_1.
6. Абонент А передає по відкритому каналу зв'язки на адресу абонента В криптограму початкового відкритого тексту (зашифроване повідомлення m) ра-

зом з криптограмою сеансового секретного криптографічного ключа k, використаного для зашифрування цього повідомлення.

7. Абонент B розшифровує на закритому ключі k_2 сеансовий секретний криптографічний ключ k, за допомогою якого розшифровує криптограму повідомлення m.

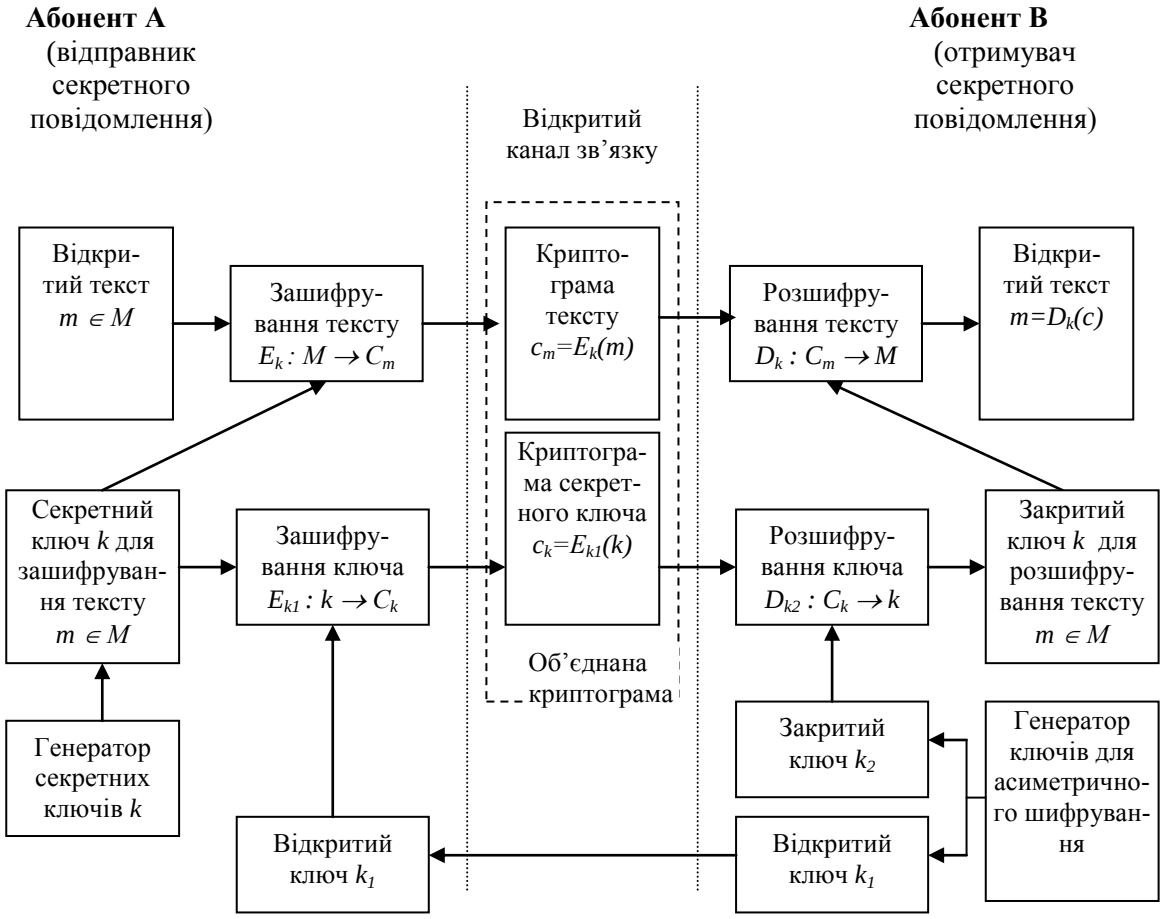


Рис. 2.4 Схема гібридної криптографічної системи

Для підвищення криптостійкості в гібридній криптографічній системі для кожного сеансу секретного зв'язку (шифрування нового повідомлення) генерується свій секретний ключ для симетричного шифрування, званий сеансовим.

Вибір розміру криптографічних ключів для симетричного і асиметричного шифрування здійснюється так, щоб їх потенційна криптостійкість до атаки по методу повного перебору можливих варіантів була порівнянною.

У разі, якщо відкритий і закритий ключі асиметричного шифрування використовуються неодноразово (довготривало), то їх криптостійкість має бути істотно вище, ніж у сеансового секретного ключа симетричного шифрування, оскільки при їх розкритті (дискредитації) супротивник дістане можливість розшифровувати передавані сеансові секретні ключі і, відповідно, зашифровані на них повідомлення.

У таблиці 2.1 приведені довжини ключів симетричних криптосистем, що мають важкість розкриття за методом повного перебору, порівнянну з трудністю факторизації відповідних модулів асиметричних криптосистем.

Таблиця 2.1

Довжина ключа симетричної криптосистеми, біт	Модуль асиметричної криптосистеми, біт
56	384
64	512
80	768
112	1792
128	2304
192	5184
256	9216

3 СИМЕТРИЧНІ КРИПТОСИСТЕМИ

3.1 Стандарт шифрування DES

Алгоритм *DES* (англ. *Data Encryption Standard*) є типовим представником класу симетричних блочних алгоритмів шифрування, в якому один ключ використовується як для зашифрування, так і для розшифрування повідомлень. Розроблений фірмою IBM і затверджений урядом США в 1977 році як офіційний стандарт DES допускає ефективну апаратну і програмну реалізацію при досягненні швидкостей шифрування до декількох мегабайт в секунду. DES призначений для шифрування блоками по 64 біта та має 16-циклову структуру мережі Фейстеля, для шифрування використовується ключ в 56 біт.

Вхідний блок данных, що складається з 64 біт, перетворюється у вихідний блок ідентичної довжини. Ключ шифрування повинен бути відомий як відправнику, так і одержувачу. В алгоритмі широко використовуються бітові перестановки. Узагальнена схема шифрування в алгоритмі DES зображена на рисунку 3.1.

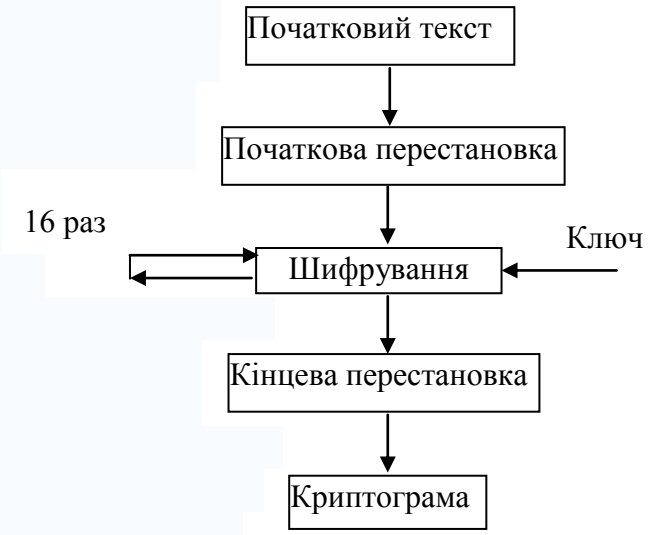


Рис. 3.1 Схема шифрування в алгоритмі DES

Кожен 64-бітний блок даних перетворюється за допомогою початкової перестановки IP. 58-й біт стає першим, 50-й – другим і т. д. Схема початкової перестановки бітів показана на рисунку 3.5. Отримана послідовність бітів розбивається на 2 частини по 32 біти: L – ліву (старші біти) та R – праву (молодші біти). Далі виконується ітераційний процес шифрування за схемою Фейстеля (рис. 3.2):

$L_i = R_{i-1}$, i = 1, 2, …, 16; $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, i=1, 2, …, 16.

На кожній ітерації повторюються наступні 4 процедури:

1. Перетворення ключа з урахуванням номера ітерації *i* (перестановки розрядів з видаленням 8 бітів, в результаті одержується 48-розрядний ключ).
2. Нехай $A = L_i$, а K — перетворений ключ. За допомогою функції $f(A, K)$ генерується 32-розрядне вихідне значення циклової (раундової) функції. Аргументами функції криптографічного перетворення f є послідовність R_{i-1},

отримана на попередньому кроці, і 48-бітовий ключ K_i, який є результатом перетворення 64-бітового ключа K.

3. Виконується операція *XOR* (R_i , *f* (*A*, *K*)), результат позначається R_{i+1}.
4. Виконується операція присвоєння $L_{i+1} = R_i$.

Після виконання 16 циклів перетворення виконується ще одна бітова перестановка, інверсна початковій. Вона передбачає наступне розміщення 64 біт зашифрованих даних (першим бітом стає 40-й, другим – 8-й і т. д.), зображене на рисунку 3.6.

Схема роботи функції криптографічного перетворення f

Спочатку 32 вхідні розряди за допомогою розширюючої перестановки перетворюються в 48 (деякі розряди повторюються). Схема цього розширення показана на рисунку 3.3 (номери відповідають номерам бітів вхідного 32- бітового блоку).

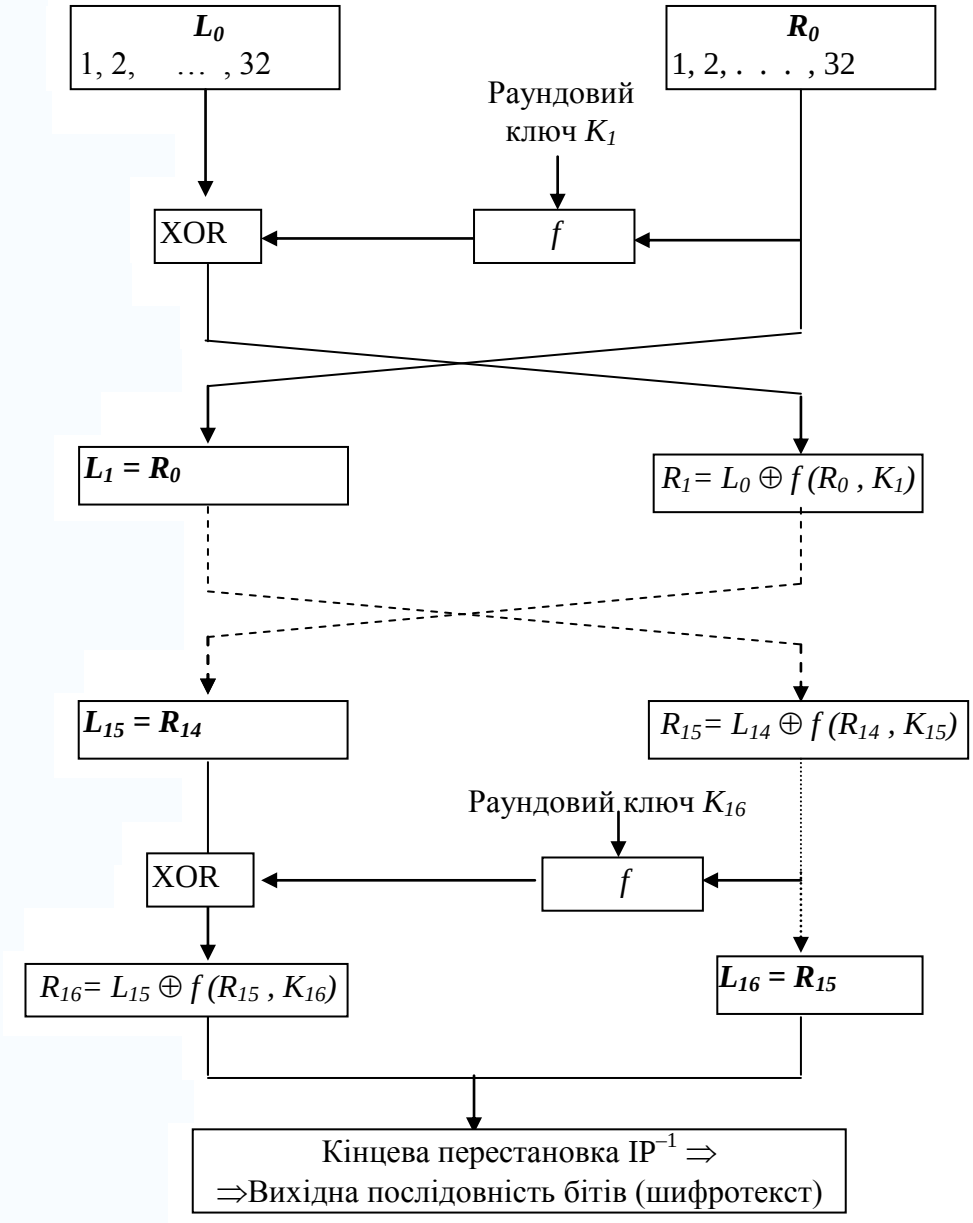


Рис. 3.2. Структурна схема алгоритма DES

Значення на виході ключового суматора одержується в результаті побітового додавання (виключаюче АБО, XOR) сформованого 48-розрядного блоку і поточного підключа. Результуючий 48-розрядний блок перетворюється в 32-розрядний за допомогою *S*-блоків. Потім вихід *S*-блоків потрапляє на P-блок, де виконується ще одна бітова перестановка згідно схеми, показаної на рисунку 3.4 (числа представляють собою порядкові номери бітів).

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

Рис. 3.3

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

Рис. 3.4

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

Рис. 3.5

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

Рис. 3.6

S-блоки являють собою таблиці, що містять 4 рядки і 16 стовпців, на перетині яких стоять 4-бітові числа від 0 до 15 (всі різні). Перший *S*-блок S_1 зображено нижче.

Вхідний 48-розрядний блок ділиться на 8 груп по 6 розрядів. Перший і останній розряд в групі використовуються в якості адреси (номера) рядка, а середні 4 розряди – в якості адреси (номера) стовпця. В результаті кожні 6 біт перетворюються в 4 біти, а весь 48-розрядний код в 32-розрядний (для цього потрібно 8 *S*-блоків).

S-блок S_1 :

No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S-блок S_2 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S-блок S_3 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*

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

S-блок S_4 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S-блок S_5 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-блок S_6 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S-блок S_7 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-блок S_8 :

№	*0*	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Схема перетворення ключів

Перетворення ключа K (де n=1, 2, .., 16, n - номер ітерації) здійснюють за алгоритмом, показаним на рисунку 3.7.

Після відкидання з 64-бітового ключа кожного 8-го біта (як біта контролю парності), маємо 56-бітний ключ.

Вибір1 і вибір2 є перестановками бітів ключа (див. таблицю 3.1). Для обчислення чергового ключа 56-бітна ключова послідовність ділиться на дві 28-бітні частини C0 і D0 (так звані регістри зрушень). У C0 увійдуть біти

57,49,41,..., 44 і 36; у D0 увійдуть біти 63,55,47,..., 12 і 4. Оскільки задано таблицю зрушень (див. таблицю 3.2), то C1, D1, C2, D2, .., C16, D16 можуть бути одержані з C0, D0 зрушенням вліво.

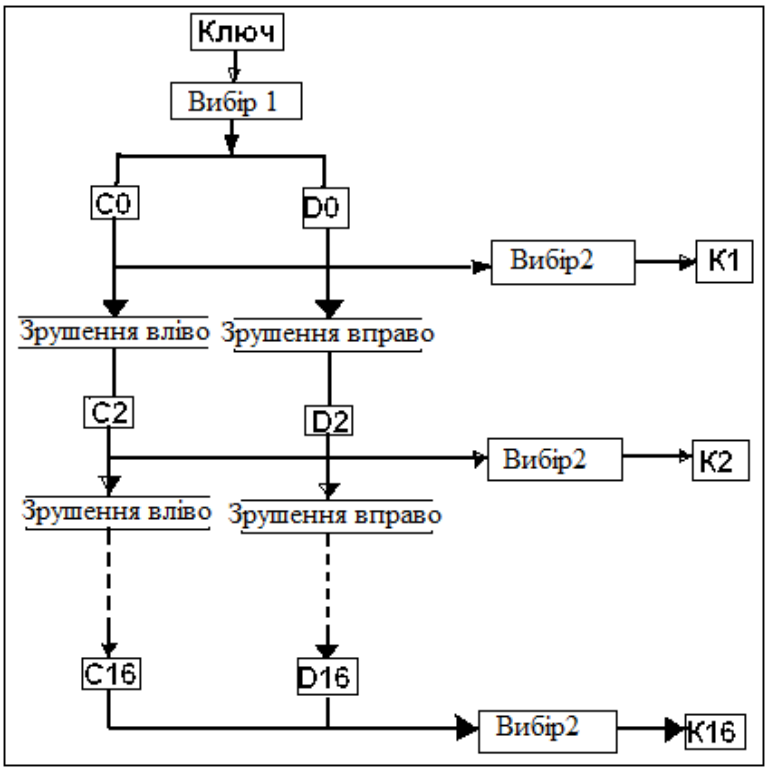


Рис. 3.7

Наприклад, C1, B1 одержуються з C0, D0 циклічним зрушенням вліво на 1 розряд; C2, D2 одержуються з C1, D1 циклічним зрушенням вліво на 1 розряд.

Далі C0 і D0 знову об'єднуються в одну 56-бітну послідовність, з якої вибираються 48 бітів згідно Вибір 2 з таблиці 3.1.

Таблиця 3.1

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

Таблиця 3.2

Номер ітерації	1	2	3	4	5	6	7	8	9	10	11	12	12	14	15	16
Число зрушень вліво	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

3.2 Стандарт шифрування ГОСТ 28147-89

Теоретичні відомості

ГОСТ 28147-89 є типовим представником сімейства блокових шифрів, що допускає ефективну апаратну і програмну реалізацію й досягає швидкостей шифрування до декількох мегабайт в секунду. Це симетричний алгоритм шифрування, в якому один ключ використовується як для зашифрування, так і для розшифрування повідомлень. ГОСТ 28147-89 призначений для шифрування да-

них 64-бітовими блоками. З одного кінця алгоритму вводиться 64-бітовий блок відкритого тексту, а з іншого кінця виходить 64-бітовий блок шифротексту.

ГОСТ, як і багато інших відомих шифрів з секретним ключем, заснований на використанні конструкції (мережі) Хорста Фейстеля (H. Feistel) (рис. 2).

В алгоритмах ГОСТу у деяких випадках елементи даних обробляються як масиви незалежних бітів, в інших випадках — як ціле число без знаку, в третіх — мають структуру як складені елементи, що складаються з декількох більш простих елементів. Тому, щоб уникнути плутанини слід домовитися про використовувані *позначення*.

Елементи даних *позначаються* великими латинськими літерами з похилим шрифтом (наприклад, *X*). Через | *X* | позначається розмір елемента даних *X* в бітах. Таким чином, якщо інтерпретувати елемент даних *X* як ціле невід'ємне число, можна записати наступну нерівність: $0 \le X < 2^{|X|}$.

Якщо елемент даних складається з декількох елементів меншого розміру, то цей факт позначається наступним чином:

$$X = (x_0, x_1, \ldots, x_{n-1}) = x_0 \,||x_1\,|| \ldots ||x_{n-1}.$$

Процедура об'єднання декількох елементів даних в один називається конкатенацією даних і позначається символом «||». Природно, для розмірів елементів даних повинно виконуватися наступне співвідношення:

$$|X| = |x_0| + |x_1| + \ldots + |x_{n-1}|.$$

При завданні складних елементів даних і операції конкатенації складові елементи даних перераховуються в порядку зростання старшинства.

В алгоритмі елемент даних може інтерпретуватися як масив окремих бітів, в цьому випадку біти позначаємо тієї ж самої буквою, що й масив, але в рядковому варіанті, як показано на наступному прикладі:

$$X = (x_0, x_1, \ldots, x_{n-1}) = x_0 + 2^1 \bullet x_1 + \ldots + 2^{n-1} \bullet x_{n-1}.$$

Якщо над елементами даних виконується деяка операція, що має логічний зміст, то передбачається, що дана операція виконується над відповідними бітами елементів. Іншими словами

$$A \bullet B = (a_0 \bullet b_0, \quad a_1 \bullet b_1, \ldots, a_{n-1} \bullet b_{n-1}), \quad \text{де } n = |A| = |B|,$$

а символом « • » позначається довільна бінарна логічна операція, як правило, мається на увазі операція виключає або, вона ж — операція підсумовування за модулем 2: $a \oplus b = (a + b) \bmod 2$.

To, Tш — масиви відповідно відкритих і зашифрованих даних;

$T_i^o, T_i^ш$ — і-ті по порядку 64-бітові блоки відповідно відкритих і зашифрованих даних:

$$To = (T_1^o, T_2^o, \ldots, T_n^o), \quad Tш = (T_1^ш, T_2^ш, \ldots, T_n^ш), \quad 1 \le i \le n,$$

останній блок може бути неповним:

$|T_i^o| = |T_i^ш| = 64$ при $1 \le i < n$, $\quad 1 \le |T_n^o| = |T_n^ш| \le 64$;

n — число 64-бітових блоків у масиві даних;

Цх — функція перетворення 64-бітового блоку даних за алгоритмом базового циклу «X».

Таким чином, треба розібрати три наступні речі:

а) що таке основний крок криптоперетворень;

б) як з основних кроків складаються базові цикли;

в) як з двох базових циклів складаються всі практичні алгоритми ГОСТу.

Слід сказати про ключову інформацію, використовувану алгоритмами ГОСТу. Відповідно до *принципу Кірхгофа*, якому задовольняють усі сучасні відомі широкому загалу шифри, саме її секретність забезпечує таємність зашифрованого повідомлення. У ГОСТі ключова інформація складається з двох структур даних. Крім власне ключа, необхідного для всіх шифрів, вона містить ще й таблицю замін. Нижче наведені основні характеристики ключових структур ГОСТу.

1. Ключ є масивом з восьми 32-бітових елементів коду, який у цій роботі позначається символом K: $K = \{K_i\}_{0<=i<=7}$. В ГОСТі елементи ключа використовуються як 32-розрядні цілі числа без знака: $0 \le Ki \le 2^{32}$. Таким чином, розмір ключа становить $32 \times 8 = 256$ біт або 32 байта.

2. Таблиця замін може бути представлена у вигляді матриці розміром 8×16, що містить 4-бітові елементи, які можна представити у вигляді цілих чисел від 0 до 15. Рядки таблиці замін називаються вузлами замін, вони повинні містити різні значення, тобто кожен вузол замін повинен містити 16 різних чисел від 0 до 15 в довільному порядку. У даній роботі таблиця замін позначається символом H:

$$H = \{H_{i,j}\}\,(0 \le i \le 7, \quad 0 \le j \le 15), \quad 0 \le H_{i,j} \le 15.$$

Таким чином, загальний обсяг таблиці замін дорівнює: 8 вузлів$\times$ 16 елементів/вузол$\times$ 4 біта/елемент = 512 біт або 64 байта.

Основний крок криптографічних перетворень

Основний крок криптоперетворень за своєю суттю є оператором, визначальним перетворення 64-бітового блоку даних. Додатковим параметром цього оператора є 32-бітовий блок, за який використовується будь-який елемент ключа. Схема алгоритму основного кроку наведена на рисунку 3.8.

Нижче дані пояснення до алгоритму основного кроку:

Крок 0. Визначає вихідні дані для основного кроку криптографічного перетворення:

N — перетворюваний 64-бітовий блок даних, в ході виконання кроку його молодша (N_1) і старша (N_2) частини обробляються як окремі 32-бітові цілі числа без знака. Таким чином, можна записати $N = (N_1, N_2)$.

X — 32-бітовий елемент ключа;

Крок 1. Додавання з ключем. Молодша половина перетвореного блоку складається по модулю 2^{32} с використовуваним на даному кроці елементом ключа, результат передається на наступний крок;

Крок 2. Поблочна заміна. 32-бітове значення, отримане на попередньому кроці, інтерпретується як масив з восьми 4-бітових блоків коду:

$$S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7).$$

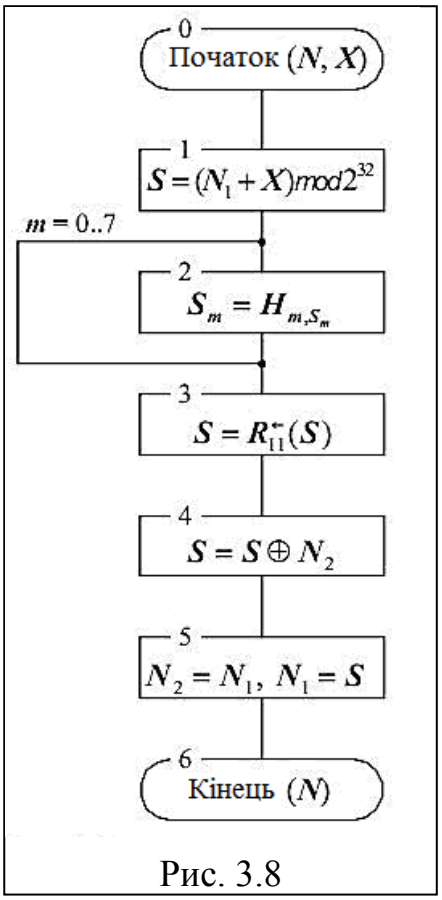


Рис. 3.8

Далі значення кожного з восьми блоків замінюється новим, яке вибирається за таблицею замін наступним чином: значення блоку S_i замінюється на S_i-тий по порядку елемент (нумерація з нуля) і-го вузла замін (тобто і-того рядка таблиці замін, нумерація також з нуля). Іншими словами, в якості заміни для значення блоку вибирається елемент з таблиці замін з номером рядка, рівним номеру замінного блоку, та номером стовпця, рівним значенню замінного блоку як 4-бітового цілого невід'ємного числа. Тепер стає зрозумілим розмір таблиці замін: число рядків у ній дорівнює числу 4-бітових елементів в 32-бітовому блоці даних, тобто восьми, а число стовпців дорівнює числу різних значень 4-бітового блоку даних, рівному як відомо 2^4, шістнадцяти.

Крок 3. Циклічний зсув на 11 біт вліво. Результат попереднього кроку зсувається циклічно на 11 біт в бік старших розрядів і передається на наступний крок. На схемі алгоритму символом R11 позначена функція циклічного зсуву аргументу на 11 біт вліво, тобто у бік старших розрядів.

Крок 4. Побітове складання: значення, отримане на кроці 3, побітно складається по модулю 2 зі старшою половиною перетвореного блоку.

Крок 5. Зрушення по ланцюжку: молодша частина перетвореного блоку зсувається на місце старшої, а на її місце поміщається результат виконання попереднього кроку.

Крок 6. Отримане значення перетвореного блоку повертається як результат виконання алгоритму основного кроку криптографічного перетворення.

Базові цикли криптографічних перетворень

ГОСТ відноситься до класу блокових шифрів, тобто одиницею обробки інформації в ньому є блок даних.

Отже, цілком логічно очікувати, що в ньому будуть визначені алгоритми для криптографічних перетворень, тобто для зашифровування і розшифрування одного блоку даних. Саме ці алгоритми і називаються базовими циклами ГОСТу, що підкреслює їх фундаментальне значення для побудови цього шифру.

Базові цикли побудовані з основних кроків криптографічного перетворення, розглянутого в попередньому розділі. У процесі виконання основного

кроку використовується тільки один елемент ключа, в той час як ключ ГОСТу містить вісім таких елементів. Отже, щоб ключ був використаний повністю, кожен з базових циклів повинен багаторазово виконувати основний крок з різними його елементами. Разом з тим здається цілком природним, що в кожному базовому циклі всі елементи ключа повинні бути використані однакове число разів, з міркувань стійкості шифру це число повинне бути більше одного.

Усі зроблені вище припущення, що спираються просто на здоровий глузд, виявилися вірними. Базові цикли полягають в багаторазовому виконанні основного кроку з використанням різних елементів ключа і відрізняються один від одного тільки числом повторення кроку і порядком використання ключових елементів. Нижче наведено порядок для різних циклів.

1. Цикл зашифровування 32-З:
   $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$
   $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$
2. Цикл розшифрування 32-Р:
   $K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$
   $K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0$

Кожен із циклів має власне буквено-цифрове позначення, що відповідає шаблоном «nX», де перший елемент позначення (n), задає число повторень основного кроку в циклі, а другий елемент позначення (X), буква, задає порядок зашифровування («З») або розшифрування («Р») у використанні ключових елементів. Цей порядок потребує додаткового пояснення:

Цикл розшифрування повинен бути зворотним циклу зашифровування, тобто послідовне застосування цих двох циклів до безпідставного блоку повинно дати в підсумку вихідний блок, що відображається таким співвідношенням: $Ц_{32-Р}(Ц_{32-З}(Т)) = Т$, де Т — довільний 64-бітовий блок даних, $Ц_x(Т)$ — результат виконання циклу X над блоком даних Т. Для виконання цієї умови для алгоритмів, подібних ГОСТу, необхідно і достатньо, щоб порядок використання ключових елементів відповідними циклами був взаємно зворотним. У справедливості записаного умови для розглянутого випадку легко переконатися, порівнявши наведені вище послідовності для циклів 32-З і 32-Р. Зі сказаного випливає один цікавий наслідок: властивість циклу бути зворотним іншому циклу є взаємним, тобто цикл 32-З є зворотним по відношенню до циклу 32-Р. Іншими словами, зашифровування блоку даних теоретично може бути виконано за допомогою циклу розшифрування, в цьому випадку розшифрування блоку даних повинне бути виконане циклом зашифровування.

З двох взаємно зворотних циклів будь-хто може бути використаний для шифрування, тоді другий повинен бути використаний для розшифрування даних, проте стандарт ГОСТ28147-89 закріплює ролі за циклами і не надає користувачеві права вибору в цьому питанні.

Схеми базових циклів наведені на рисунку 3.9 та рисунку 3.10. Кожен з них приймає як аргумент і повертає в якості результату 64-бітовий блок даних, позначений на схемах N. Символ Крок (N, X) позначає виконання основного кроку криптографічного перетворення для блоку N з використанням ключового елемента X.

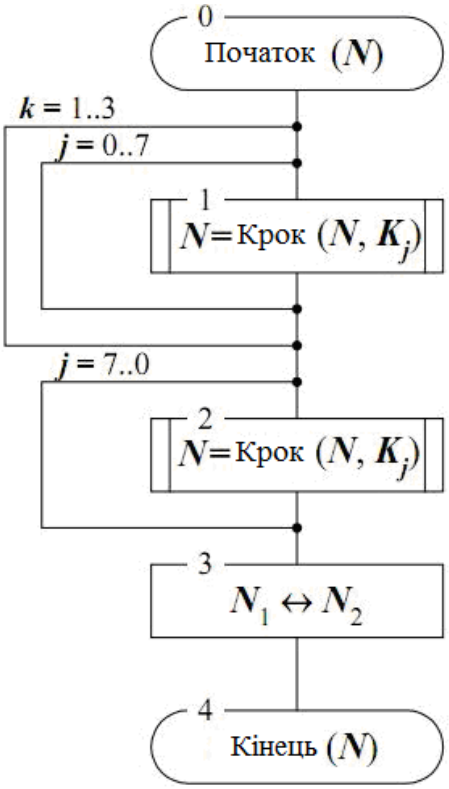


Рис. 3.9

Вимоги до якості ключової інформації і джерела ключів

У стандарті ГОСТ 28147-89 поняття ключ визначено наступним чином: "Конкретний секретний стан деяких параметрів алгоритму криптографічного перетворення, що забезпечує вибір одного перетворення із сукупності всіх можливих для даного алгоритму перетворень ".

Ключ може належати певному користувачеві або групі користувачів і бути для них унікальним. Зашифрована з використанням конкретного ключа інформація може бути розшифрована тільки за допомогою цього ж ключа.

Не всі ключі і таблиці замін забезпечують максимальну стійкість шифру. Для кожного алгоритму шифрування є свої критерії оцінки ключової інформації.

Так, для алгоритму DES відоме існування так званих «слабких ключів», при використанні яких зв'язок між відкритими і зашифрованими даними не маскується достатнім чином, і шифр порівняно просто розкривається. Вичерпну відповідь на питання про критерії якості ключів і таблиць замін ГОСТу якщо і можна взагалі де-небудь отримати, то тільки у розробників алгоритму.

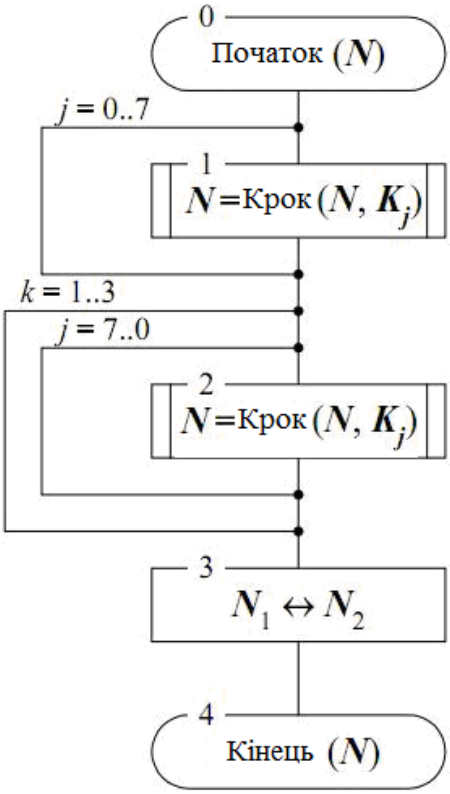


Рис. 3.10

Згідно з встановленим порядком, для шифрування інформації, що має гриф, повинні бути використані ключові дані, отримані від уповноваженої організації. Непрямим чином це може свідчити про наявність методик перевірки ключових даних на «слабкість». Сам факт існування слабких ключових даних в Російському стандарті шифрування не викликає сумніву. Очевидно, нульовий ключ і тривіальна таблиця замін, за якою будь-яке значення замінюється на нього ж самого, є слабкими, при використанні хоча б одного з них шифр досить просто

зламується, який би не був другий ключовий елемент.

Як вже було зазначено вище, критерії оцінки ключової інформації недоступні, однак на їх рахунок все ж можна висловити деякі загальні міркування:

1. Ключ повинен бути масивом статистично незалежних бітів, що приймають з однаковою ймовірністю значення 0 і 1. При цьому деякі конкретні значення ключа можуть виявитися «слабкими», тобто шифр може не забезпечувати заданий рівень стійкості в разі їх використання. Однак, імовірно, частка таких значень в загальній масі всіх можливих ключів мізерно мала. Тому ключі, вироблені за допомогою деякого датчика істинно випадкових чисел, будуть якісними з ймовірністю, що відрізняється від одиниці на дуже малу величину.

Якщо ж ключі виробляються за допомогою генератора псевдовипадкових чисел, то використовуваний генератор повинен забезпечувати зазначені вище статистичні характеристики, і, крім того, мати високу криптографічну стійкість, не меншу, ніж у самого ГОСТа. Іншими словами, задача визначення відсутніх членів вироблюваної генератором послідовності елементів не повинна бути простіше, ніж завдання розкриття шифру. Крім того, для відбракування ключів з поганими статистичними характеристиками можуть бути використані різні статистичні критерії. На практиці зазвичай вистачає двох критеріїв, для перевірки з рівною ймовірністю розподілу бітів ключа між значеннями 0 і 1 зазвичай використовується критерій Пірсона («хі квадрат»), а для перевірки незалежності бітів ключа — критерій серій. Про згадані критерії можна прочитати в підручниках чи довідниках з математичної статистики.

2. Таблиця замін є довготривалим ключовим елементом, тобто діє протягом набагато більш тривалого терміну, ніж окремий ключ. Передбачається, що вона є спільною для всіх вузлів шифрування в рамках однієї системи криптографічного захисту. Навіть при порушенні конфіденційності таблиці замін стійкість шифру залишається надзвичайно високою і не зменшується нижче допустимої межі.

До якості окремих вузлів замін можна пред'явити наведене нижче вимогу. Кожен вузол замін може бути описаний четвіркою логічних функцій, кожна з яких має чотири логічних аргументу. Необхідно, щоб ці функції були досить складними. Цю вимогу складності неможливо виразити формально, проте в якості необхідної умови можливо вимагати, щоб відповідні логічні функції, записані у мінімальній формі (тобто з мінімально можливою довжиною вираження) з використанням основних логічних операцій, не були коротші деякого необхідного мінімуму. У першому і дуже грубому наближенні ця умова може зійти і за достатнє.

Крім того, окремі функції в межах всієї таблиці замін повинні відрізнятися один від одного в достатній мірі. На практиці буває достатньо отримати вузли замін як незалежні випадкові перестановки чисел від 0 до 15, це може бути практично реалізована, наприклад, за допомогою перемішування колоди з шістнадцяти карт, за кожною з яких закріплено одне зі значень зазначеного діапазону.

Необхідно відзначити ще один цікавий факт щодо таблиці замін. Для оборотності циклів шифрування «32-З» і «32-Р» не потрібно, щоб вузли замін

були перестановками чисел від 0 до 15. Все працює навіть в тому випадку, якщо у вузлі замін є повторювані елементи, і заміна, обумовлена таким вузлом, необоротна, проте в цьому випадку знижується стійкість шифру.

Дуже широко поширене використання різних програмних датчиків випадкових чисел. При генерації невеликого за обсягом масиву ключової інформації широко застосовується метод «електронної рулетки», коли чергова порція випадкових бітів отримується в моменту часу натискання оператором деякої клавіші на клавіатурі комп'ютера.

Режим шифрування простою заміною

Зашифровування в даному режимі полягає в застосуванні циклу 32-З до блоків відкритих даних, розшифрування – циклу 32-Р до блоків зашифрованих даних. Це найбільш простий з режимів, 64-бітні блоки даних обробляються в ньому незалежно один від одного.

Розмір масиву відкритих або зашифрованих даних, що підлягає відповідно зашифровуванню або розшифруванню, повинен бути кратним 64 бітам:

$|\mathbf{T_в}| = |\mathbf{T_ш}| = 64 \cdot \boldsymbol{n}$,

після виконання операції розмір отриманого масиву даних не змінюється.

4 АСИМЕТРИЧНІ КРИПТОСИСТЕМИ

4.1 Шифрування з відкритим ключем

Розвиток основних типів криптографічних протоколів (ключовий обмін, електронний цифровий підпис, аутентифікація) був би неможливим без створення *відкритих* ключів та побудованих на їх основі *асиметричних* алгоритмів шифрування.

Ідея асиметричних алгоритмів тісно пов'язана з одного боку з теорією *односторонніх функцій*, з іншого – з *теорією складності*.

Визначення. Під *односторонньою функцією* ми будемо розуміти легко обчислюване відображення $f(x) : X \to Y$, $x \in X$, при цьому обернене відображення є складною задачею.

Визначення. Задача називається *складнообчислюваною*, якщо немає алгоритму для її розв'язання з поліноміальним часом роботи. *Легкообчислюваною* будемо називати задачу, що має алгоритм з часом роботи, представленим у вигляді полінома низького степеня відносно вхідного розміру задачі, а ще краще алгоритм з лінійним часом роботи.

Помітимо, що на сьогоднішній день теоретично *не доведено існування односторонніх функцій*. Використання їх в якості основи асиметричних алгоритмів шифрування допустимо тільки до тих пір, поки не знайдено ефективні алгоритми, що виконують обернення односторонніх функцій за поліноміальний час.

Широке розповсюдження асиметричних алгоритмів шифрування викликано необхідністю мати два ключі – відкритий для шифрування і закритий для розшифрування. Відповідно, вводячи поняття відкритого ключа, тобто ключа, потенційно відомого всім, ми позбавляємося необхідності вирішувати складну задачу обміну секретними ключами.

Розвитком ідеї односторонніх функцій стала побудова *односторонніх функцій з секретом*. Такою функцією називається функція $f(x) = y$, значення якої легко обчислити, тоді як обернене значення без знання деякого секрету складно обчислити. Знання ж секрету дозволяє достатньо просто реалізувати операцію обернення односторонніх функцій з секретом.

На практиці при застосуванні асиметричного алгоритму шифрування в ролі секретного ключа виступає саме знання секрету, а в ролі відкритого ключа – знання процедури обчислення односторонньої функції з секретом

4.2 Елементи теорії чисел

Арифметика лишків

Розглянемо приклад. 23 за модулем 12 дорівнює 11.

$(10+13) \bmod 12 = 23 \bmod 12 = 11 \bmod 12$

Іншим способом записати це є твердження про еквівалентність 23 і 11 за модулем 12:

$10 + 13 \equiv 11 \pmod{12}$

В основному, $a \equiv b \pmod{n}$, якщо $a = b + kn$ для деякого цілого k. Якщо a невід'ємне і b знаходиться між 0 і n, можна розглядати b як залишок від ділення

a на n. Число b називається *лишком* a по модулю n. Іноді a називається *конгруентним* b по модулю n (знак потрійної рівності, ≡, позначає конгруентність). Одне й те ж можна сказати різними способами.

Множина чисел від 0 до n−1 утворює те, що називається *повною множиною лишків* по модулю n. Це означає, що для будь-якого цілого a його залишок по модулю n є деяким числом від 0 до n−1. Ця операція називається *приведенням по модулю*. Наприклад, 5 mod 3 = 2.

Це визначення mod може відрізнятися від прийнятого в деяких мовах програмування. Наприклад, в мові C оператор % повертає залишок від ділення першого виразу на другий, воно може бути від'ємним числом, якщо будь-який з операндів від'ємний. Для всіх алгоритмів перевіряйте, чи додаєте ви n до результату операції одержання залишку, якщо вона повертає від'ємне число.

Арифметика залишків дуже схожа на звичайну арифметику: вона комутативна, асоціативна і дистрибутивна. Крім того, приведення кожного проміжного результату по модулю n дає той же результат, як і виконання всього обчислення з наступним приведенням кінцевого результату по модулю n.

(a + b) mod n == ((a mod n) + (b mod n)) mod n

(a–b) mod n == ((a mod n) – (b mod n)) mod n

(a*b) mod n == ((a mod n) * (b mod n)) mod n

(a *(b + c)) mod n == (((a*b) mod n) + ((a*c) mod n)) mod n

Обчислення по mod n часто використовується в криптографії, оскільки обчислення дискретних логарифмів та квадратних коренів mod n може бути нелегкою проблемою. Арифметика лишків, до того ж, легше реалізується на комп'ютерах, оскільки вона обмежує діапазон проміжних значень і результату. Для k-бітових лишків n проміжні результати будь-якого додавання, вирахування або множення будуть не довше, ніж 2k біт. Тому в арифметиці лишків ми можемо виконати зведення до степеня без величезних проміжних результатів. Обчислення степеня деякого числа по модулю другого числа, a^x mod n, представляє собою просто послідовність множень та ділень, але існують прийоми, що прискорюють цю дію. Один з таких прийомів прагне мінімізувати кількість множень по модулю, другий – оптимізувати окремі множення по модулю. Оскільки операції дистрибутивні, швидше виконати зведення до степеня як потік послідовних множень, кожного разу отримуючи лишки. Зараз ви не відчуваєте різниці, але вона буде помітна при множенні 200-бітових чисел.

Наприклад, якщо ви хочете обчислити a^8 mod n, не виконуйте наївно сім множень і одне приведення за модулем:

(a*a*a*a*a*a*a*a) mod n.

Замість цього виконайте три менших множень і три менших приведень по модулю:

$((a^2 \bmod n)^2 \bmod n)^2 \bmod n$.

Точно так,

$a^{16} \bmod n = (((a^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n$.

Обчислення a^x , где x не є степенем 2, ненабагато складніше. Двійковий запис представляє x у вигляді суми степенів 2:

25 – це бінарне 11001, тому $25 = 2^4 + 2^3 + 2^0$.

Тому

$a^{25} \bmod n = (a^* a^{24}) \bmod n = (a^* a^8 a^{16}) \bmod n =$

$= (a^*((a^2)^2)^2 (((a^2)^2)^2)^2) \bmod n = (a^*(((a^* a^2)^2)^2)^2) \bmod n.$

З продуманим збереженням проміжних результатів вам знадобиться тільки 6 множень:

$(((((((a^2 \bmod n)^* a)^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 \bmod n)^2 * a) \bmod n.$

Такий прийом називається *ланцюжком додавань*, або *методом двійкових квадратів і множень*. Він використовує простий і очевидний ланцюжок додавань, в основі якого лежить двійкове представлення числа.

Цей метод зменшує кількість операцій, в середньому, до (1.5*k), де k – довжина числа x в бітах. Знайти спосіб обчислення з найменшою кількістю операцій – важка проблема (було доведено, що послідовність повинна містити не менше k–1 операцій), але неважко знизити число операцій до (1.1*k) або навіть краще за більших k.

Операція, обернена зведенню до степеня по модулю n, обчислює *дискретний логарифм*. Розглянемо цю операцію пізніше.

Прості числа

Простим називається ціле число, більше одиниці, єдиними множниками якого є 1 і воно саме: воно не ділиться на жодне інше число. Два – це просте число. Простими є числа 73, 2521, 2365347734339 і $2^{756839}–1$. Існує нескінченно багато простих чисел. Криптографія, особливо криптографія з відкритими ключами, часто використовує великі прості числа (512 біт і навіть більше).

Найбільший загальний дільник

Два числа a і n називаються *взаємно простими*, якщо в них немає загальних множників окрім 1. Іншими словами, якщо *найбільший загальний дільник* чисел a і n дорівнює 1. Це записується як НЗД(a, n)=1.

Взаємно прості числа 25 і 28. 15 і 27 не є взаємно простими, а 13 і 500 – є. Просте число взаємно просте з усіма іншими числами, окрім чисел, кратних даному простому числу.

Одним зі способів обчислення найбільшого загального дільника двох чисел є **алгоритм Евкліда**. Евклід описав цей алгоритм в своїй книзі «Елементи», написаній в 300 році до нашої ери. Він не винайшов його. Історики вважають, що цей алгоритм років на 200 старше. Це найдавніший нетривіальний алгоритм, який дійшов до наших днів, і він все ще хороший. Кнут описав алгоритм та його сучасні модифікації в своїй книзі «Искусство программирования для ЭВМ» (D. Knuth. The Art of Computer Programming: Volume 2, Seminumerical Algorithms, 2nd edition, Addison-Wesley, 1981).

Обернені значення по модулю

Обернене значення для числа 4 – це число 1./4, тому що 4*(1/4) = 1. В світі лишків проблема ускладнюється:

$4*x = 1 \pmod 7$.

Це рівняння еквівалентне виявленню цілих чисел x і k, таких, що

$4x=7k+1$.

Загальна задача полягає в знаходженні x такого, що

$1 = (a*x) \bmod n$.

Це також можна записати як

$a^{-1} \equiv x \pmod n$.

Проблему обернених значень по модулю розв'язати нелегко. Іноді у неї є рішення, іноді ні. Наприклад, обернене значення 5 по модулю 14 дорівнює 3. З іншого боку, у числа 2 немає оберненого значення по модулю 14.

В загальному випадку у рівняння $a^{-1} \equiv x \pmod n$ існує єдиний розв'язок, якщо a та n взаємно прості числа. Якщо a і n не є взаємно простими, то $a^{-1} \equiv x \pmod n$ не має розв'язку. Якщо n є простим числом, то будь-яке число від 1 до n–1 взаємно просте з n і має в точності одне обернене значення по модулю n.

Для *знаходження оберненого значення* a по модулю n існує два шляхи. Обернене значення a по модулю n можна обчислити за допомогою *алгоритму Евкліда*. Іноді це називається *розширеним* алгоритмом Евкліда.

Алгоритм Евкліда працює наступним чином. Щоб знайти НЗД(a, b), де a>b, спочатку ділять a на b та записують частку q_1 і залишок r_1 :

$a = q_1 b + r_1$.

Потім виконують друге ділення, в якому b грає роль a, а r_1 – роль b:

$b = q_2 r_1 + r_2$.

Потім ділять r_1 на r_2 . Ділення продовжують, кожного разу ділячи передостанній залишок на останній, отримуючи нові частку і залишок. Коли в кінці кінців отримується залишок, який є дільником попереднього залишку, то цей останній ненульовий залишок і є найбільший загальний дільник чисел a і b.

Приклад. Знайти НЗД (1547, 560).

Розв'язок.

$1547 = 2*560 + 427$

$560 = 1*427 + 133$

$427 = 3*133 + 28$

$133 = 4*28 + 21$

$28 = 1*21 + 7$.

Оскільки 7|21, то НЗД (1547, 560) = 7.

Твердження. Нехай d = НЗД (a, b), где a > b. Тоді існують такі цілі числа u і v, що d = ua + bv. Іншими словами, НЗД двох чисел можна представити у вигляді лінійної комбінації цих чисел з цілими коефіцієнтами. Крім того, для знаходження чисел u і v достатньо $O(\log^3 a)$ двійкових операцій.

Схема доведення. Проходячи послідовність рівностей в алгоритмі Евкліда знизу вгору і виражаючи кожен раз d через все більш ранні залишки, в кінці кінців отримуємо вираз d через a і b. На кожному кроці необхідно виконати одне множення і одне додавання або вирахування. Таким чином, як легко впевнитися, число двійкових операцій знову є $O(\log^3 a)$.

Приклад 1 (продовження). Щоб представити 7 у вигляді лінійної комбінації 1547 и 560, запишемо

$7 = 28 – 1*21 = 28 – 1*(133 – 4*28) = 5*28 – 1*133 =$

$= 5*(427 – 3*133) – 1*133 = 5*427 – 16*133 = 5*427 – 16*(560 – 1*427) =$

$= 21*427 – 16*560 = 21*(1547 – 2*560) – 16*560 = 21*1547 – 58*560.$

Визначення. Кажуть, що два цілих числа a та b взаємно прості, якщо їх НЗД(a, b) = 1, тобто якщо a і b не мають загальних дільників, більших 1.

Наслідок. Якщо a і b – взаємно прості числа та a>b, то можна знайти представлення 1 (одиниці) у вигляді цілочислової лінійної комбінації цих чисел за поліноміальний час, точніше, за $O(\log^3 a)$ двійкових операцій.

Визначення. Для будь-якого цілого додатного числа n функція Ейлера φ(n) визначається як число невід'ємних цілих b, менших n і взаємно простих з n:

$\varphi(n) = / \text{def} / = \{b \mid 0 <= b < n \mid \text{НЗД}(b, n) = 1\}$.

Легко перевірити, що $\varphi(1) = 1$ та що $\varphi(p) = p – 1$ для будь-якого простого p. Можна впевнитися також, що для будь-якого простого p виконується

$\varphi(p^\alpha) = p^\alpha – p^{\alpha-1} = p^\alpha (1 – 1/p)$.

Для цього достатньо помітити, що числа від 0 до $p^\alpha – 1$, які не взаємно прості з p^α, – це в точності ті числа, які діляться на p, а їх кількість дорівнює числу $p^{\alpha-1}$.

Твердження (*Мала теорема Ферма*). Нехай p – просте число. Будь-яке ціле число a задовольняє порівнянню $a^p \equiv a \pmod p$, та будь-яке a, що не ділиться на p, задовольняє порівнянню $a^{p-1} \equiv 1 \pmod p$.

Наслідок. Якщо a не ділиться на p і якщо $n \equiv m \pmod{(p-1)}$, то виконується порівняння $a^n \equiv a^m \pmod p$.

Приклад 2. Знайти останню цифру в семеричному записі числа $2^{1000000}$.

Розв'язок. Нехай p = 7. Оскільки 1000000 при діленні на p–1 = 6 дає залишок 4, отримуємо $2^{1000000} \equiv 2^4 = 16 \equiv 2 \pmod 7$, і, значить, остання цифра дорівнює 2.

Наслідок (з Китайської теореми про залишки). Функція Ейлера має властивість « мультиплікативності », тобто

$\varphi(mn) = \varphi(m)\,\varphi(n)$, якщо НЗД (m, n) = 1.

4. 3 Шифрування з відкритим ключем RSA

Розвиток основних типів криптографічних протоколів (ключовий обмін, електронний цифровий підпис, аутентифікація) був би неможливим без створення *відкритих* ключів та побудованих на їх основі *асиметричних* алгоритмів шифрування.

Ідея асиметричних алгоритмів тісно пов'язана з одного боку з теорією *односторонніх функцій*, з іншого – з *теорією складності*.

Визначення. Під *односторонньою функцією* ми будемо розуміти легко обчислюване відображення $f(x) : X \to Y$, $x \in X$, при цьому обернене відображення є складною задачею.

Визначення. Задача називається *складнообчислюваною*, якщо немає алгоритму для її розв'язання з поліноміальним часом роботи. *Легкообчислюваною* будемо називати задачу, що має алгоритм з часом роботи, представленим у вигляді полінома низького степеня відносно вхідного розміру задачі, а ще краще алгоритм з лінійним часом роботи.

Помітимо, що на сьогоднішній день теоретично *не доведено існування односторонніх функцій*. Використання їх в якості основи асиметричних алгоритмів шифрування допустимо тільки до тих пір, поки не знайдено ефективні алгоритми, що виконують обернення односторонніх функцій за поліноміальний час.

Широке розповсюдження асиметричних алгоритмів шифрування викликано необхідністю мати два ключі – відкритий для шифрування і закритий для розшифрування. Відповідно, вводячи поняття відкритого ключа, тобто ключа, потенційно відомого всім, ми позбавляємося необхідності вирішувати складну задачу обміну секретними ключами.

Розвитком ідеї односторонніх функцій стала побудова *односторонніх функцій з секретом*. Такою функцією називається функція f (x) = y, значення якої легко обчислити, тоді як обернене значення без знання деякого секрету складно обчислити. Знання ж секрету дозволяє достатньо просто реалізувати операцію обернення односторонніх функцій з секретом.

На практиці при застосуванні асиметричного алгоритму шифрування в ролі секретного ключа виступає саме знання секрету, а в ролі відкритого ключа – знання процедури обчислення односторонньої функції з секретом.

Алгоритм RSA

Математичні основи

Нехай n і e – натуральні числа (n>0, e>0). Функція f, що реалізує схему RSA, побудована наступним чином:

$$f : x \to x^e \bmod n. \quad (1)$$

Для дешифрування повідомлення a = f (x) достатньо розв'язати порівняння

$$x^e \equiv a \pmod n. \quad (2)$$

Введемо *функцію Ейлера*.

Визначення. *Функцією Ейлера* називається **φ(n)** = {кількість цілих чисел на відрізку від 1 до n, взаємно простих з n}.

Оскільки $\varphi(1) = 1$, $\forall$ простого p і $r \in N$ $\varphi(p^r) = p^{r-1}(p-1)$;

$\forall$ $a, b \in N$, a, b взаємно простих: $\varphi(a, b) = \varphi(a)\varphi(b)$.

Якщо НЗД(e, φ(n)) = 1, то порівняння (2) має єдиний розв'язок. Для цього визначимо число d так, щоб виконувалось порівняння : $de \equiv 1 \pmod{\varphi(n)}$, $1 \le d < \varphi(n)$.

Класична теорема Ейлера стверджує, що $\forall x$, взаємно простого з n, тобто такого, що НЗД(x,n)=1, виконується $x^{\varphi(n)} \equiv 1 \pmod n$.

Тому $a^d = x^{de} \equiv x \pmod n$.

Таким чином, у припущенні, що НЗД(a,n) =1, єдиний розв'язок порівняння (2) може бути знайдений у вигляді

$x = a^d \ (mod\ n)$.

Оскільки $\varphi(n) = \varphi(pq) = \varphi(p)\,\varphi(q) = (p-1)(q-1)$, то єдиною умовою на вибір відкритого ключа *e* є: НЗД (e,p–1) = НЗД (e, q–1) =1.

Стандарт асиметричного шифрування *RSA* (приклад протоколу)

Кожен учасник інформаційного обміну генерує пару ключів – відкритий e і секретний d – у відповідності до правил:

1. Вибираються 2 великих цілих простих числа p і q приблизно однакового розміру. Вибір чисел p і q визначається такими міркуваннями:
   - збільшення порядку чисел призводить до уповільнення операції зашифрування/розшифрування;
   - збільшення порядку чисел призводить до збільшення стійкості алгоритму.

   Тому при виборі чисел варто керуватися практичною необхідністю. На практиці зазвичай рекомендується обирати числа, що містять порядку 150-200 десяткових знаків.
2. Обчислюється модуль системи $n = pq$ і функція Ейлера $\varphi(n) = (p-1)(q-1)$.
3. Вибирається достатньо велике число e, що задовольняє умові $1 < e < \varphi(n)$, і взаємно просте з $\varphi(n)$, тобто НЗД $(e, \varphi(n)) = 1$.
4. Обчислюється велике ціле число d, що відповідає умові: $de \equiv 1 (\mathrm{mod}\ \varphi(n))$, $1 < d < \varphi(n)$.

Таким чином, *секретним ключем* є пара чисел (n, d), а відкритим ключем – пара (n, e). Відкритий ключ розміщується в загальнодоступний довідник.

Зашифрування / розшифрування в системі RSA

Після вибору параметрів системи (n, e, d) абонент готовий до прийому *зашифрованих* повідомлень. Їх *передача* складається з наступних кроків:

1. Вхідне повідомлення розбивається на блоки m_i , їхній розмір визначається цілим k, що відповідає нерівності $10^{k-1} < n < 10^k$.
2. Обчислюється значення $c_i = m_i^e \mod n$.
3. Значення c_i , яке виявляється зашифрованим блоком повідомлення, відправляється по відкритим каналам передачі даних.

Розшифрування полягає в обчисленні значення $m_i = c_i^d \mod n$.

Доведенням того факту, що розшифрування виконується тільки абонентом, що знає секретну експоненту зашифрування d, є наступне.

$$c_i^d \mod n = m_i^{e\,d} \mod n;$$

з урахуванням того, що $de \equiv 1 (\mathrm{mod}\ \varphi(n))$. При цьому отримуємо

$$ed = 1 + k\,\varphi(n),$$

де k – ціле число, що задовольняє цій рівності. Оскільки $m_i^{\varphi(n)} \mod n$ – одиничний елемент групи лишків (залишків від ділення на n) відносно операції множення, елемент m_i також належить даній групі. В цьому випадку маємо

$$c_i^d \mod n = m_i\ m_i^{k\,\varphi(n)} \mod n = m_i .$$

Приклад. Якщо p = 47, q=71, то n = pq = 3337. Ключ e не повинен мати загальних множників з p і q: (p–1)(q–1) = 46*70 = 3220.

Виберемо (випадково) e рівним 79. В цьому випадку $d = 79^{-1} \bmod 3220 = 1019$.

При обчисленні цього числа використано розширений алгоритм Евкліда. Опублікуємо e і n, зберігши в секреті d. Відкинемо p і q. Для шифрування повідомлення m = 6882326879666683
спочатку розділимо його на маленькі блоки. Для нашого випадку підійдуть трьохсимвольні блоки. Повідомлення розбивається на 6 блоків m_i :

$m_1 = 688$; $m_2 = 232$; $m_3 = 687$; $m_4 = 966$; $m_5 = 668$; $m_6 = 003$.

1-й блок шифрується як $688^{79} \bmod 3337 = 1570 = c_1$.

Виконуючи ті ж операції для наступних блоків, створюється шифротекст повідомлення:

c = 1570 2756 2091 2276 2423 158.

Для дешифрування потрібно виконати таке же зведення до степеня, використовуючи ключ дешифрування 1019:

$1570^{1019} \bmod 3337 = 688 = m_1$.

Аналогічно відновлюється частина повідомлення, що залишилася.

5 ЕЛЕКТРОННИЙ ЦИФРОВИЙ ПІДПИС

5.1 Вступ

Найбільш важливою областю застосування криптографії з відкритим ключем є електронні цифрові підписи (ЕЦП). Вони забезпечують цілий комплекс захисту, який було б доволі складно здійснити для звичайного паперового документообігу.

Однак при цьому виникають проблеми автентифікації автора документа й самого документа (тобто встановлення достовірності підпису та відсутність змін в отриманому документі).

Проблема автентифікації є актуальною в обчислювальних системах управління и взагалі там, де необхідно упевнитися в достовірності отриманого по каналам зв'язку або на машинних носіях повідомлення (документа).

В ситуаціях, коли немає повної довіри між відправником та одержувачем, вимагається щось більше, ніж проста автентифікація.

Найбільш привабливим рішенням цієї проблеми виявляється ЕЦП. Він є аналогом підпису, зробленого від руки, і повинен забезпечувати наступні можливості:

- можливість встановити автора, а також дату і час підпису;
- можливість встановити достовірність вмісту повідомлення на час підпису;
- можливість перевірки підпису третьою стороною на випадок виникнення суперечки.

На основі цих можливостей можуть бути сформульовані вимоги, що висуваються до цифрового підпису:

- підпис повинен бути двійковим кодом, який залежить від повідомлення, що підписується;
- підпис повинен використовувати деяку інформацію, унікальну для відправника, щоб запобігти можливість як фальсифікації, так і заперечення авторства;
- ЕЦП можна відносно просто виробити;
- ЕЦП можна відносно просто розпізнати і перевірити;
- з точки зору обчислень нереально фальсифікувати ЕЦП ані за допомогою створення нового повідомлення для наявного ЕЦП, ані за допомогою створення фальшивого ЕЦП для наявного повідомлення.

Безпосередній ЦП має на увазі участь тільки сторін, які обмінюються даними (відправник, одержувач). Припускається, що одержувач знає відкритий ключ відправника. ЕЦП може бути сформований за допомогою зашифрування всього повідомлення особистим ключем відправника або за допомогою шифрування хеш-коду повідомлення особистим ключем відправника.

При формуванні ЕЦП відправник насамперед обчислює хеш-функцію $h(M)$ тексту M, що підписується. Обчислене значення хеш-функції $h(M)$ являє собою короткий блок інформації m, який характеризує весь текст M в цілому.

Потім число m шифрується секретним ключем відправника. Отримувана при цьому пара чисел являє собою ЕЦП для даного повідомлення M.

При перевірці ЦП одержувач повідомлення знову обчислює хеш-функцію $m = h(M)$ прийнятого по каналу тексту M, потім за допомогою відкритого ключа відправника перевіряє, чи відповідає отриманий підпис обчисленому значенню m хеш-функції.

Без знання секретного ключа підписування відправника неможна підробити ЕЦП. В якості документа, що підписується, можна використовувати будь-який файл.

Підписаний файл створюється з непідписаного шляхом додавання в нього одного або більше електронних підписів, які повинні містити наступну інформацію:

- дату підпису;
- строк закінчення дії ключа даного підпису;
- інформацію про особу, що підписала файл (документ) (П.І.Б., посаду, коротке найменування фірми);
- ідентифікатор особи, яка підписала документ (ім'я відкритого ключа);
- власне цифровий підпис.

5.2 Концепція формування ЦП

Абоненти посилають один одному підписані електронні документи. Для кожного абонента генерується пара ключів: секретний K_c і відкритий $K_в$. Причому секретний ключ зберігається абонентом в таємниці і використовується ним тільки для формування ЦП. Відкритий ключ відомий всім іншим користувачам і призначений для перевірки ЦП одержувачем підписаного електронного документа. Відкритий ключ не дозволяє обчислити секретний, а є необхідним інструментом, що дозволяє перевірити достовірність електронного документа і автора підпису.

Для генерації пари ключів (відкритого і секретного) в алгоритмах цифрового підпису використовуються різні математичні схеми, засновані на застосуванні однонаправлених функцій. За відомими складними обчислювальними задачами ці схеми розділяють на дві групи:

- задача факторизації (розкладання на прості множники);
- задача дискретного логарифмування.

Для формування ЦП використовують наступні алгоритми:

- RSA;
- Ель Гамаля;
- DSA;
- ГОСТ 34.10-94;
- на базі еліптичних кривих;
- цифрових підписів з функціональними додатковими властивостями.

Закон про електронний цифровий підпис

З січня 2004 року вступив у дію прийнятий Верховної Радою України Закон "Про електронний цифровий підпис", в якому чітко визначене поняття, сфера використання та юридична сила електронного цифрового підпису.

Електронний підпис – дані в електронній формі, які додаються до інших електронних даних або логічно з ними пов'язані та призначені для ідентифікації підписувача цих даних.

Електронний цифровий підпис (ЕЦП) – вид електронного підпису, отриманого за результатом криптографічного перетворення набору електронних даних, який додається до цього набору або логічно з ним поєднується і дає змогу підтвердити його цілісність та ідентифікувати підписувача. Електронний цифровий підпис накладається за допомогою особистого ключа та перевіряється за допомогою відкритого ключа.

Електронний цифровий підпис може використовуватись юридичними та фізичними особами як аналог власноручного підпису для надання електронному документу юридичної сили. Юридична сила електронного документа, підписаного ЕЦП, еквівалентна юридичній силі документа на паперовому носії з власноручним підписом правоздатної особи та скріпленого печаткою.

ЕЦП володіє всіма основними властивостями власноручного підпису:

- засвідчує те, що отриманий документ надійшов від особи, яка його підписала;
- гарантує цілісність та захист від спотворення/виправлення підписаного документа;
- не дає можливості особі, яка підписала документ, відмовитись від зобов'язань, що виникли в результаті підписання цього електронного документа.

Безпека використання ЕЦП забезпечується тим, що засоби, які використовуються для роботи з ЕЦП, проходять експертизу в Державній службі спеціального зв'язку та захисту інформації України.

Переваги ЕЦП

Юридична сила електронних документів

Закони України прирівнюють за юридичною силою електронні документи, що підписані ЕЦП, і документи з власноручним підписом або печаткою, а також створюють правову основу для застосування ЕЦП і здійснення юридично значимих дій шляхом електронного документообігу.

Конфіденційність і безпека інформації

Використовуючи ЕЦП, Ви отримуєте додаткові можливості шифрування документів. Завдяки надійним криптографічним алгоритмам забезпечується конфіденційність інформації, яка передбачає неможливість доступу до неї будь-якої особи, що не володіє секретним ключем.

Безпека використання ЕЦП гарантується тим, що ПЗ "БЕСТ ЗВІТ ПЛЮС", яке використовується для роботи з ЕЦП, пройшло експертизу в Державній службі спеціального зв'язку та захисту інформації України.

Можливість ведення електронного документообігу з державними структурами

Можливість використовувати одні й ті ж засоби ЕЦП при обміні даними з усіма міністерствами, відомствами, при подачі звітності в будь-які контролюючі органи на території України.

Удосконалення бізнес-процесів на підприємствах

Ведення ЕДО на підприємствах значно скорочує об'єми паперової бухгалтерської документації, економить час співробітників і витрати підприємств, пов'язані з укладенням договорів, оформленням платіжних документів та їх пересиланням контрагентові.

Ведення ділових відносин на сучасному рівні

Використання ЕЦП значно прискорює проведення численних комерційних операцій, виключає необхідність додаткових зустрічей і багатогодинних переговорів.

Електронний цифровий підпис – ефективне рішення для всіх, хто хоче використовувати новітні технології та бути успішним бізнесменом. Документи, підписані електронним цифровим підписом, можуть бути передані до місця призначення за лічені секунди. Всі учасники електронного документообігу отримують рівні можливості незалежно від їх фізичного розташування.

5.3 Алгоритм цифрового підпису DSA

Національний інститут стандартів і технології (NIST) США опублікував федеральний стандарт обробки інформації *FIPS PUB* 186, відомий також як DSS (Digital Signature Standard – стандарт цифрового підпису). Стандарт *DSS* заснований на алгоритмі хешування *SHA* (Secure Hash Algorithm – захищений алгоритм хешування) і являє собою нову технологію використання цифрового підпису – алгоритм *DSA* (Digital Signature Algorithm – алгоритм цифрового підпису).

DSA являє собою варіант алгоритму Ель Гамаля в модифікації Шнора. Алгоритм використовує наступні параметри:

p – просте число довжиною l бітів, де l приймає значення, кратне 64, в діапазоні від 512 до 1024;

q – 160-бітове просте число – дільник (p–1).

Генератор $g = h^{(p-1)/q} \bmod p$, де h – довільне число, менше за p–1, таке, що $h^{(p-1)/q} \bmod p > 1$.

Секретний ключ x, $x < q$.

Відкритий ключ $y = g^x \bmod p$.

В алгоритмі також використовується одно направлена хеш-функція $H(m)$ стандарта *SHA*.

Перші три параметра p, q і g відкриті і можуть бути загальними для абонентів криптомережі. Число y є відкритим ключем і передається всім одержувачам документів. Секретним ключем є число x.

Розглянемо алгоритм формування ЦП за схемою *DSA*.

1. Відправник генерує випадкове число $k < q$ і обчислює
   $r = (g^k \bmod p) \bmod q$,

$s = (k^{-1} (H(m) + x\, r))\ mod\ q$.

Підписом відправника служать числа r та s.

2. Одержувач перевіряє підпис, обчислюючи

$w = s^{-1}\ mod\ q$,

$u_1 = (H(m)\ w)\ mod\ q$,

$u_2 = (r\ w)\ mod\ q$,

$v = (g^{u_1} g^{u_2}\ mod\ p)\ mod\ q$.

Якщо $v = r$, то підпис під документом визнається одержувачем правильною.

У порівнянні з алгоритмом цифрового підпису Ель Гамаля алгоритм *DSA* має ряд переваг:

- при будь-якій парі чисел q і p (від 512 до 1024) числа q, x, r, s мають довжину по 160 біт, скорочуючи довжину підпису до 320 біт, при будь-якому допустимому рівні стійкості;
- більшість операцій з числами k, r, s, x при обчисленні підпису виконуються по модулю числа q довжиною 160 біт, що скорочує час обчислення підпису;
- при перевірці підпису більшість операцій з числами u_1, u_2, v, w також виконуються по модулю числа q довжиною 160 біт, що скорочує об'єм пам'яті і час обчислення.

Алгоритм *DSA* не дозволяє отримувати максимальну швидкодію, оскільки при підписанні та при перевірці підпису доводиться виконувати складні операції ділення по модулю q:

$s = (k^{-1} (H(m) + x\, r))\ mod\ q$, $w = (1/s)\ (mod\ q)$, що є недоліком алгоритму *DSA*.

5.4 Алгоритм цифрового підпису RSA

Процедура вироблення цифрового підпису складається з наступних етапів.

Дії *відправника*:

1. Обчислення *секретного* і *відкритого* ключів, для чого відправник електронних документів обирає два великих простих числа p і q, знаходить їх добуток (модуль) n = p * q.

Потім обчислює значення функції Ейлера φ(n): $\varphi(n) = (p-1)(q-1)$.

2. Обчислення відкритого ключа $К_в$ з умов:

$К_в \le \varphi(n)$, НЗД ($К_в$, $\varphi(n)$) =1,

і секретного ключа $К_с$ з умов: $К_с < n$, $К_в * К_с \equiv 1\ (\mathrm{mod}\ \varphi(n))$.

3. Передача користувачем мережі пари чисел ($К_в$, n), яка є відкритим ключем повідомлення, партнерам по листуванню для перевірки його цифрового підпису. Число $К_с$ зберігається відправником як секретний ключ для підписування.

Для формування ЦП за схемою RSA необхідно:

1. Повідомлення M (блок інформації, файл и т.д.) стиснути за допомогою хеш-функції $h(M)$ в ціле число m.

2. Обчислити цифровий підпис s під електронним документом *M*, використовуючи хеш-значення m і секретний ключ $К_с$:

$s = m^{K}{}_{c} \pmod n$.

3. Передати одержувачу електронний документ M, підписаний ЦП s.

Дії одержувача.

1. Після прийому пари (*M, s*) одержувач обчислює хеш-значення повідомлення M як m = h(M) і відновлює хеш-значення m' цифрового підпису s за формулою

$m' = s^{K}{}_{в} \pmod n$.

2. Якщо $s^{K}{}_{в} \pmod n = h(M)$, то одержувач визнає пару (M, s) справжньою.

Отже, процедура перевірки ЕЦП складається з двох етапів: обчислення хеш-функції документа і власне математичних обчислень, передбачених даним алгоритмом підпису. Останні полягають у перевірці того або іншого співвідношення, що зв'язує хеш-функцію документа, підпис під ним і секретний ключ $К_с$ абонента, який підписав. Якщо розглядуване співвідношення виявляється виконаним, то підпис визнається правильним, а сам документ – справжнім, в противному випадку документ вважається зміненим, а підпис під ним – підробленим.

Доведено, що тільки володар секретного ключа $К_с$ може сформувати ЕЦП s по документу M, а визначити секретне число $К_с$ по відкритому ключу $К_в$ не легше, ніж розкласти число n на прості множники.

Результат перевірки цифрового підпису s буде позитивним тоді і тільки тоді, коли при обчисленні s був застосований секретний ключ $К_с$, який відповідає відкритому ключу $К_в$. У зв'язку з цим відкритий ключ $К_в$ іноді називають «ідентифікатором» того, хто підписав.

Розглянемо числовий *приклад* використання хеш-функції, що має наступний вигляд:

$$H_i = [(H_{i-1} \oplus M_i)^2] \pmod n, \qquad (*)$$

де H_0 – вектор ініціалізації; $M_i = M_1, M_2, \ldots, M_n$.

Приклад 1. Повідомлення, що хешується, «531». Обираємо числа p = 7, q = 3, H_0 =6. Визначаємо n = pq = 7*3 = 21. Обчислюємо хеш-код повідомлення 531 поблочно за формулою (*) $H_i = [(H_{i-1} \oplus M_i)^2] \pmod n$.

1. $M_1 + H_0 = 5+6 = 11$;

$[M_1 + H_0]^2 \pmod n = 11^2 \pmod{21} = 16 = H_1$;

2. $M_2 + H_2 = 3+16 = 19$;

$[M_2 + H_1]^2 \pmod n = 19^2 \pmod{21} = 4 = H_2$;

3. $M_3 + H_2 = 1+4 = 5$;

$[M_3 + H_2]^2 \pmod n = 5^2 \pmod{21} = 4 = H_3$.

У підсумку отримуємо хеш-значення повідомлення «531», що дорівнює 4.

Приклад 2. Отримати хеш-код для повідомлення «HESHING» за допомогою хеш-функції, обчислюваної за формулою (*).

Обираємо числа p = 17, q = 19.

Порядок обчислення хеш-коду:

1) обчислюємо значення модуля n = p*q = 323;
2) представляємо повідомлення «HESHING» у вигляді символів ASCII:

H	A	S	H	I	N	G
72	65	83	72	73	78	71

3) представляємо коди ASCII бітовим рядком:

H	A	S	H	I	N	G
72	65	83	72	73	78	71
01001000	01000001	01010011	01001000	01001001	01001110	01000111

4) розбиваємо байти навпіл, потім додаємо на початок півбайта одиниці і отримуємо блоки M_i , що хешуються.

M_1	M_2	M_3	M_4	M_5	M_6	M_7
11110100	11111000	11110100	11110001	11110101	11110011	11110100

M_8	M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}
11111000	11110100	11111001	11110100	11111110	11110100	11110111

Обчислюємо хеш-код повідомлення поблочно за формулою (*)

$H_i = [(H_{i-1} \oplus M_i)^2] \pmod n$.

1. Обчислюємо H_1

M_1	= 11110100
$\oplus$	
H_0	= <u>00000000</u>
$H_0 \oplus M_1$	= $11110100_2 = 244_{10}$
$[(H_0 \oplus M_1)^2] \pmod{323}$	= $244^2 \pmod{323} = 104$
H_1	= $104_{10} = 01101000_2$

2. Обчислюємо H_2

M_2	= 11111000
$\oplus$	
H_1	= <u>01101000</u>
$H_1 \oplus M_2$	= $10010000_2 = 144_{10}$
$[(H_1 \oplus M_2)^2] \pmod{323}$	= $144^2 \pmod{323} = 64$
H_2	= $64_{10} = 01000000_2$

Обчислюємо $H_i \; \forall$ i = 3, 4, … і так далі.

<u>Зауваження</u>. Тут і далі символ 64_{10} означає число 64 в 10-ковій системі числення, а 01000000_2 – число 64 в 2-ковій системі числення.

Для алгоритму цифрового підпису RSA властиві наступні недоліки:

- При обчисленні модуля n і ключів $К_в$, $К_с$ для системи ЦП RSA необхідно перевірити велику кількість додаткових умов, що зробити достатньо складно. Невиконання будь-якої з цих умов робить можливою фальсифікацію ЦП з боку того, хто виявив таке невиконання. При підписанні важливих документів неможна допустити таку можливість навіть теоретично;
- Для забезпечення криптостійкості ЦП RSA по відношенню до спроб фальсифікації необхідно використовувати при обчисленнях n, $К_в$, $К_с$ цілі числа не менше 2^{512} кожне, що вимагає великих обчислювальних витрат, які перевищують на 20÷30% обчислювальні витрати інших алгоритмів ЕЦП при збереженні того ж рівня криптостійкості;
- ЦП RSA чутливий до атак, тобто алгоритм дозволяє зловмиснику без знання $К_с$ сформувати підписи під тими документами, у яких результат хешування можна обчислити як добуток результатів хешування вже підписаних документів.

Колізією називається пара аргументів M і $M^{'}$, яким відповідає одне й те саме значення хеш-функції: h(M) = h($M^{'}$). Якщо число таких аргументів дорівнює r, то кажуть про колізію кратності r. Якщо хеш-функція не має колізій при заданих обмеженнях на довжину аргументу, то вона називається вільною від колізій.

Припустимо, що є два повідомлення M_1 і M_2. Перше повідомлення достовірне, а для другого крипто аналітик намагається отримати теж саме значення хеш-коду, видавши повідомлення M_1 за повідомлення M_2 (тобто намагається отримати **колізію**). Для цього крипто аналітик готує приблизно $\sqrt{n}$ різних версій M_1 і M_2, що трохи відрізняються, і для кожної з них обчислює хеш-код. З високою ймовірністю вдається виявити пару версій $M^{'}_1$ і $M^{'}_2$, які мають один і той самий хеш-код. В подальшому при використанні даного хеш-коду в схемі ЦП можна видати повідомлення $M^{'}_2$ замість $M^{'}_1$, зміст якого близький до змісту повідомлення M_1.

До основних методів запобігання даної атаки можна віднести збільшення довжини отримуваних хеш-кодів.

6 КРИПТОСИСТЕМИ НА ОСНОВІ МЕТОДУ ЕЛІПТИЧНИХ КРИВИХ

Хоча RSA є безпечною асиметрично-ключовою криптографічною системою, її безпека забезпечується ціною її великих ключів. Дослідники шукали альтернативний метод, який дає той самий рівень безпеки з меншими розмірами ключів. Один з таких перспективних варіантів – криптосистема на основі методу еліптичних кривих (Elliptic Curve Cryptosystem – ECC). Система базується на теорії еліптичних кривих.

6.1 Еліптичні криві у дійсних числах

Еліптичні криві, які безпосередньо не зв'язані з еліпсами, являють собою кубічні рівняння двох змінних та зазвичай застосовуються для обчислення довжини кривої в колі еліпса. Загальне рівняння для еліптичної кривої:

$y^2 + b_1 x y + b_2 y = x^3 + a_1 x^2 + a_2 x + a_3$.

Еліптичні криві в полі дійсних чисел використовують спеціальний клас форми еліптичних кривих:

$y^2 = x^3 + a x + b$.

В цьому випадку, якщо $4a^3 + 27b^2 \neq 0$, рівняння являє собою *несингулярну еліптичну криву*, в противному випадку воно описує *сингулярну еліптичну криву*. Для несингулярної еліптичної кривої рівняння $x^3 + a x + b = 0$ має три відмінних кореня (дійсних або комплексних); для сингулярної рівняння $x^3 + a x + b = 0$ не має трьох відмінних коренів.

У рівнянні, як можна бачити, ліва частина (y^2) має степінь 2, в той час як права частина має степінь 3 (x^3). Це означає, що горизонтальна лінія може перетинати криву в трьох точках, якщо всі корені дійсні. Однак вертикальна лінія може перетнути криву найбільше в двох точках.

Приклад 1. Рисунок 6.1 а,б показує дві еліптичні криві з рівняннями $y^2 = x^3 - 4 x$ та $y^2 = x^3 - 1$. Обидва рівняння несингулярні. Однак перше має три дійсних кореня ($x = -2$, $x = 0$ і $x = 2$), але друге – тільки один дійсний корінь ($x = 1$) і два уявних.

Абелева група

Визначимо абелеву (комутативну) групу, що використовує точки на еліптичній кривій. Кортеж $P = (x_1, y_1)$ являє точку на кривій, якщо x_1 і y_1 – координати точки на кривій, які задовольняють рівнянню цієї кривої. Наприклад, точки P = (2,0; 0,0), Q = (0,0; 0,0), R = (–2,0; 0,0), S = (10,0; 30,98), T= (10,0;–30,98) є точками на кривій $y^2 = x^3 - 4 x$. Зверніть увагу, що кожна точка представлена двома дійсними числами.

Визначимо *множину* як точки на кривій, де кожна точка – пара дійсних чисел. Наприклад, множина E для еліптичної кривої $y^2 = -x^3 - 4 x$ буде

E = {(2,0; 0,0), (0,0; 0,0), (-2,0; 0,0), (10,0; 30,98), (10,0; -30,98), …}

Задані властивості несингулярної еліптичної кривої дозволяють нам визначити операцію додавання точок на кривій. Однак ми повинні пам'ятати, що операція додавання тут відрізняється від операції, яка була визначена для цілих

чисел. Операція «додавання двох точок на кривій» проводиться, щоб отримати іншу точку на кривій.

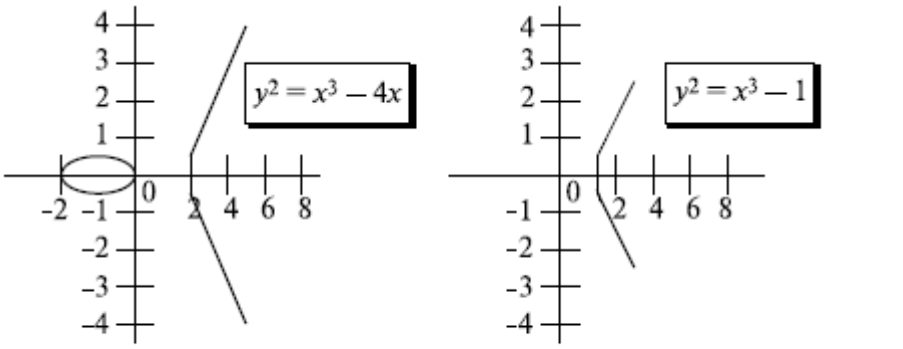


Рис.6.1 a. Три дійсних кореня. b. Один дійсний корінь і два уявних.

$R = P + Q$, $P = (x_1, y_1)$, $Q = (x_2, y_2)$, $R = (x_3, y_3)$.

Для того, щоб знайти R на кривій, розглянемо три випадки, як це показано на рисунку 6.2.

1. В першому випадку дві точки $P = (x_1, y_1)$, $Q = (x_2, y_2)$ мають різні x-координати і y-координати ($x_1 \neq y_1$ і $x_2 \neq y_2$), як показано на рисунку 2a. Лінія, що з'єднує P і Q, перетинає криву в точці, позначеній –R. R є відображення (–R) відносно y-осі. Координати точки R, x_3, y_3 , можуть бути знайдені по нахилу лінії, λ, і потім можна обчислити значення x_3, y_3 , як показано нижче:

$\lambda = (y_2 - y_1)/(x_2 - x_1)$,

$x_3 = \lambda^2 - x_1 - x_2$,

$y_3 = \lambda(x_1 - x_3) - y_1$.

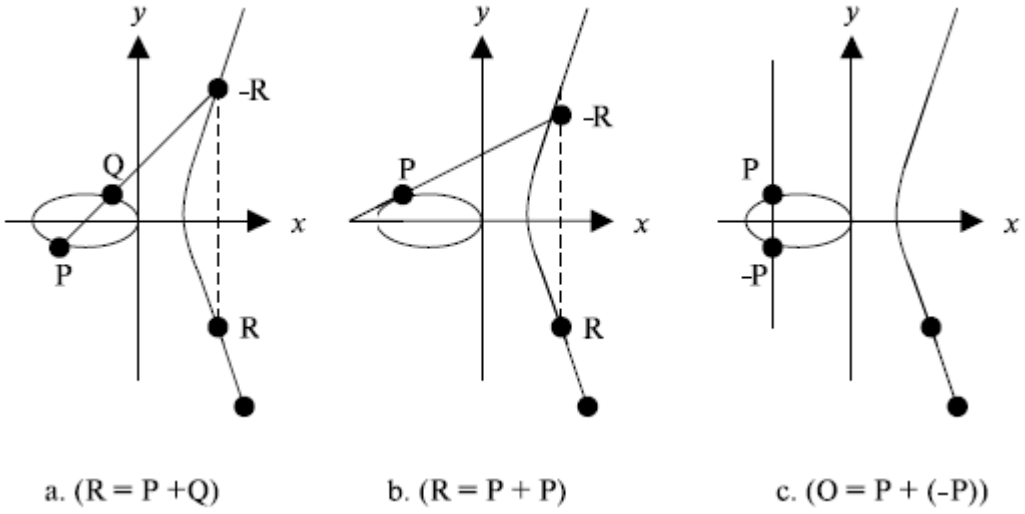


Рис. 6.2 Три випадки додавання на еліптичній кривій

2. В другому випадку дві точки збігаються (R = P + P), як показано на рисунку 2b. Нахил лінії і координати точки R можуть бути знайдені, як показано нижче:

$\lambda = (3x_1^2 - a)/(2y_1)$,

$x_3 = \lambda^2 - x_1 - x_2$,

$y_3 = \lambda(x_1 - x_3) - y_1$.

3. В третьому випадку дві точки – адитивні інверсії одна одної, як це показано на рисунку 2с. Якщо перша точка P = (x_1, y_1), а друга точка дорівнює Q = (x_1, -y_1), лінія, що з'єднує ці дві точки, не перетинає криву в третій точці. Математики кажуть в цьому випадку, що точка перетину знаходиться в нескінченності. Вони визначають точку O (рис. 2с) як точку в нескінченності або нульову точку, яка є адитивним нейтральним елементом групи.

6.2 Еліптичні криві в GF(p)

Криптографія вимагає модульної арифметики. Ми визначили групу еліптичної кривої з операцією додавання, але операція на координатах з точками в даному випадку є операція в GF(p) з p >3. В модульній арифметиці точки на кривій не представляють графи, як це можна було бачити на рисунках, але зберігаються ті ж самі основні концепції. Ми використовуємо ту ж саму операцію додавання, але з обчисленням по модулю p. В результаті ми отримуємо еліптичну криву $E_p(a, b)$, де p визначає модуль, a і b – коефіцієнти рівняння

$y^2 = x^3 + a\,x + b$.

Зверніть увагу, що хоча значення x в цьому випадку від 0 до p, зазвичай не всі точки знаходяться на кривій.

Знаходження інверсії

Інверсія точки (x, y) дорівнює (x, -y), де (-y) – адитивна інверсія y. Наприклад, якщо p = 13, інверсія (4, 2) дорівнює (4, 11).

Знаходження точок на кривій

Алгоритм 1 показує програму на псевдокоді для знаходження точок на кривій E_p (a, b).

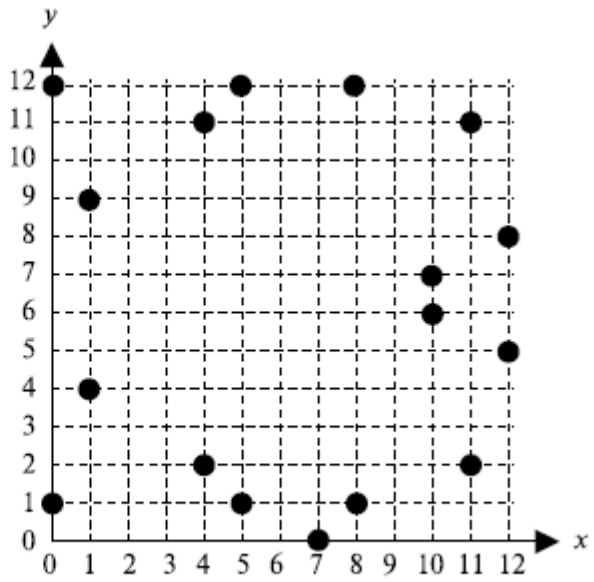
Алгоритм 1

```
Elliptic_points (p, a, b) // p – модуль
{
x ← 0
while (x < p)
{
w ← (x^3 + a x + b) mod p        // w – це y^2
if (w – ціле значення квадратного кореня в Z_p )
вихід ((x, √w )( x, −√w ))
x ← x + 1
}
}
```

Приклад 2. Визначте еліптичну криву E_{13} (1, 1) за рівнянням $y^2 = x^3 + x + 1$ і обчисліть по модулю 13. Точки на кривій можуть бути знайдені, як показано на рисунку 3.

(0,1)	(0,12)
(1,4)	(1,9)
(4,2)	(4,11)
(5,1)	(5,12)
(7,0)	(7,0)
(8,1)	(8,12)
(10,6)	(10,7)
(11,2)	(11,11)

Точки



Граф

Рис. 6.3 Точки на еліптичній кривій в полі GF (p)

Зверніть увагу на наступне:

а. Деякі значення y^2 не мають квадратного кореня по модулю 13. Вони не є точками на еліптичній кривій. Наприклад, точки x = 2, x=3, x = 6 і x = 9 не знаходяться на кривій.

б. Кожна точка, визначена на кривій, має інверсію. Інверсії перераховані як пари. Помітимо, що (7, 0) – інверсія самої себе.

в. Зверніть увагу, що для пари зворотних точок значення y – адитивні інверсії друг друга в Z_p. Наприклад, 4 і 9 – адитивні інверсії Z_{13} .Так що ми можемо сказати, що якщо 4 – це значення y, то 9 – це значення (–y).

г. Інверсії знаходяться на тих же самих вертикальних лініях.

Додавання двох точок

Ми використовуємо групу еліптичної кривої, визначену раніше, але обчислення зроблені у GF (p) (з операцією додавання за модулем p). Замість вирахування та ділення ми застосовуємо адитивні і мультиплікативні інверсії (протилежні відносно додавання та зворотні відносно множення).

Приклад 3.

Додамо дві точки з прикладу 2, R = P + Q, де P = (4, 2) і Q = (10, 6).

a. $X = (6 – 2)*(10 – 4)^{-1} \bmod 13 = 4 * 6^{-1} \bmod 13 = 5 \bmod 13$.
b. $x = (5^2 – 4 – 10) \bmod 13 = 11 \bmod 13$.
c. $y = [5 (4 – 11) – 2] \bmod 13 = 2 \bmod 13$.
d. R = (11, 2) є точкою на кривій в прикладі 2.

Множення точки на константу

В арифметиці множення числа на константу k означає додавання числа самого до себе k раз. Тут ситуація та ж сама. Множення точки P на еліптичній кривій на константу k означає додавання точки P самої до себе k раз. Наприклад, в E_{13} (1, 1), якщо точка (1, 4) множиться на 4, результат є точка (5,4). Якщо точка (8, 1) множиться на 3, результат – точка (10, 7).

6.3 Алгоритм цифрового підпису на базі еліптичних кривих ECDSA

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 субекспонеціального 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.

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 точок 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."

Вибір параметрів

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 підписуване 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 – порядок однієї з циклічних підгруп групи точок еліптичної кривої.

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 – скінченне просте поле. Причому, якщо необхідно, конструкцію можна легко адаптувати для еліптичних кривих над іншим полем.

Нехай E – еліптична крива, визначена над F_p , і P – точка простого порядку q кривої $E(F_p)$. Крива E і точка P є системними параметрами. Число p – просте. Кожен користувач A конструює свій ключ за допомогою наступних дій:

1. Вибирає випадкове або псевдовипадкове ціле число x з інтервалу [1, q–1].
2. Обчислює добуток (кратне) $Q = xP$.

Відкритим ключем користувача A є точка Q, а закритим – x.

Замість використання E і P в якості глобальних системних параметрів, можна фіксувати лише поле F_p для всіх користувачів і дозволити кожному користувачеві вибирати свою власну еліптичну криву E і точку P$\in$ E(F_p). У цьому випадку певне рівняння кривої E, координати точки P, а також порядок q цієї точки P повинні бути включені в відкритий ключ користувача. Якщо поле F_p фіксовано, то апаратна і програмна складові можуть бути побудовані так, щоб оптимізувати обчислення в тому полі. У той же час є величезна кількість варіантів вибору еліптичної кривої над полем F_p .

Обчислення цифрового підпису

Для того, щоб підписати яке-небудь повідомлення, для якого підраховано значення h хеш-функції H, користувач A повинен зробити наступне:

1. Вибрати випадкове ціле число $k\in$[1, q–1].
2. Обчислити $k \cdot P = (x_1 , y_1)$ і покласти в $r = x_1 \bmod q$,, де r виходить з цілого числа x_1 між 0 і (p – 1) приведенням по модулю q.

Зауваження: якщо r = 0, то рівняння підпису $s = k^{-1} (h + x \cdot r) \bmod q$ не залежить від секретного ключа x, і отже, (r, s) не підходить в якості цифрового підпису. Значить, у випадку r = 0 необхідно повернутися до кроку 1.

3. Обчислити $k^{-1} \pmod q$ і покласти $s = k^{-1} (h + x \cdot r) \bmod q$, де h – значення хеш-функції підписуваного повідомлення.

Зауваження: якщо s = 0, то значення $s^{-1} \pmod q$, потрібне для перевірки, не існує. Значить, у випадку s = 0 необхідно повернутися до кроку 1.

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

Перевірка цифрового підпису

Для того, щоб перевірити підпис користувача A (r, s) на повідомлення, користувач B повинен зробити наступне:

1. Отримати підтверджену копію відкритого ключа Q користувача A.
2. Перевірити, що числа r і s є цілими числами з інтервалу [1, q–1], і обчислити значення хеш-функції h від сполучення;
3. Обчислити $u_1 = s^{-1} h \pmod q$ і $u_2 = s^{-1} r \pmod q$;
4. Обчислити $u_1 P + u_2 Q = (x_0 , y_0)$, і щодо x_0, як цілого числа між 0 і (p–1), покласти $v = x_0 \bmod q$.
5. Прийняти підпис, якщо і тільки якщо v = r.

Зауважимо, що, якщо користувач A вирахував свій підпис правильно, то

$u_1 P + u_2 Q = (u_1 + x u_2) P = (s^{-1} h \bmod q + x s^{-1} r \bmod q) P = k P$,

оскільки $k = s^{-1} (h + x r) \pmod q$, і тому v = r.

Для підтвердження публічного ключа Q потрібно виконати наступне (O тут позначає нескінченно віддалену точку):

1. Перевірити, що Q не дорівнює O і координати вірні.
2. Перевірити, що Q лежить на кривій.
3. Перевірити, що $qQ = O$.;

7 ІДЕНТИФІКАЦІЯ І ПЕРЕВІРКА ДОСТОВІРНОСТІ

Будь-який об’єкт комп’ютерної системи (КС) може бути *ідентифікований* числом, словом або рядком символів. Якщо об’єкт має деякий ідентифікатор, зареєстрований в мережі, його називають *законним* (легальним), інші об’єкти відносяться до *незаконних* (нелегальних).

Ідентифікація об’єкта виконується в тому випадку, коли об’єкт робить спробу увійти в мережу. Якщо ідентифікація пройшла успішно, то даний об’єкт є законним для даної мережі.

Аутентифікація об’єкта (перевірка достовірності) встановлює, чи є (являється) цей об’єкт саме тим, ким себе оголошує.

Надання повноважень (**авторизація**) встановлює сферу діяльності об’єкта та доступні йому ресурси КС після ідентифікації і перевірки його достовірності.

Аутентифікація об’єктів при захисті каналів передачі даних означає взаємне встановлення достовірності об’єктів, зв’язаних між собою по лінії зв’язку. Ця процедура виконується зазвичай на початку сеансу в процесі встановлення логічного зв’язку між двома об’єктами мережі (з’єднання), так званих абонентів. Зрештою забезпечується упевненість в тому, що з’єднання встановлено із законним об’єктом, і вся інформація дійде до місця призначення.

7.1 Ідентифікація і аутентифікація користувача

Щоб отримати допуск до ресурсів КС, користувач повинен пройти ідентифікацію (тобто повідомити системі за її запитом своє ім’я), потім аутентифікацію (тобто вводити в КС унікальну, не відому іншим користувачам інформацію про себе). Для цього необхідна наявність відповідного суб’єкта (модуля) аутентифікації і аутентифікуючого об’єкта, який зберігає унікальну інформацію для аутентифікації користувача.

Розрізняють декілька форм зображення (представлення) об’єктів, що аутентифікують користувача, а саме на основі знання якихось даних. Прикладом можуть служити стандартні паролі, персональні ідентифікаційні номери (*PIN*), а також секретні і відкриті ключі, знання яких демонструється в протоколах типу «запит-відповідь»:

- *на основі засобів зберігання* («знання чогось»); зазвичай це магнітні карти, смарт-карти, touch memory та персональні генератори, які використовуються для створення одноразових паролів;
- *на основі біометричних характеристик* («володіння чимось»); тут включаються методи, засновані на перевірці користувацьких біометричних характеристик (голос, сітківка ока, відбитки пальців і т.д.). В даному випадку не використовуються криптографічні методи та засоби. Аутентифікація на основі біометричних характеристик застосовується для контролю доступу в приміщення або до будь-якої техніки.

Протоколи аутентифікації можна класифікувати за рівнем забезпечуваної безпеки або за можливістю протистояти певному класу атак: *проста аутентифікація* (на основі використання паролів), строга аутентифікація (на основі ви-

користання криптографічних методів та засобів) та протоколи, які мають властивість доведення (доказу) з нульовим знанням.

Кожен з перелічених типів аутентифікації сприяє рішенню своїх специфічних задач, тому всі вони використовуються на практиці, за виключенням протоколів аутентифікації, що мають властивість доведення (доказу) з нульовим знанням, цікавість до яких носить поки що чисто теоретичний характер.

Найбільш небезпечною загрозою протоколам аутентифікації є випадок, коли зловмисник видає себе за яку-небудь іншу сторону, що має суттєві привілеї в системі. Крім того, існує атака такого виду, коли після успішного проходження аутентифікації між двома користувачами і встановлення з'єднання порушник «прибирає» якого-небудь користувача із з'єднання та продовжує роботу від його імені.

Типові схеми ідентифікації і аутентифікації користувача

Нехай в КС зареєстровано n користувачів та і-й об'єкт, що аутентифікує і-го користувача містить два інформаційних поля:

Id_i – незмінний ідентифікатор і-го користувача, який є аналогом імені і використовується для ідентифікації користувача;

K_i – інформація, що аутентифікує користувача, яка може змінюватися и служити для аутентифікації (наприклад, пароль $P_i = K_i$).

Подібна структура відповідає практично будь-якому ключовому носію інформації, який використовується для пізнання користувача. Так, для носіїв типу пластикових карт виділяється незмінна інформація Id_i первинної персоналізації користувача та об'єкт у файловій структурі карти, що містить K_i .

Первинною інформацією, що аутентифікує і-го користувача, можна назвати сукупну інформацію в ключовому носії. Внутрішній об'єкт, що аутентифікує, не повинен існувати більше часу роботи конкретного користувача. Для тривалого зберігання варто використовувати дані у захищеній формі.

Розглянемо дві типові схеми ідентифікації і аутентифікації.

Згідно з алгоритмом за схемою 1 в КС виділяється об'єкт-еталон для ідентифікації і аутентифікації користувачів. Структуру об'єкта-еталона для схеми 1 показано в табл.1. Тут $E_i = F(Id_i , K_i)$, де F – функція, яка має властивість «невідновлюваності» значення K_i за E_i та Id_i . «Невідновлюваність» значення K_i оцінюється деякою трудомісткістю T_0 розв'язання задачі відновлення інформації K_i, що аутентифікує за E_i та Id_i . Для пари K_i та K_j можливий збіг відповідних значень E. Тому ймовірність хибної аутентифікації користувача не повинна бути більшою деякого порогового значення P_0. На практиці задають $T_0 = 10^{20}$, …, 10^{30} , $P_0 = 10^{-7}$, …, 10^{-9} .

Таблиця 7.1

Номер користувача	Інформація для ідентифікації	Інформація для аутентифікації
1	Id_1	E_1
2	Id_2	E_2
…	…	…
N	Id_N	E_N

7.2 Протокол ідентифікації і аутентифікації

Протокол ідентифікації і аутентифікації за схемою 1

1. Спочатку користувач пред'являє свій ідентифікатор Id .
2. При неспівпадінні Id з жодним з Id_i , які зареєстровані в КС, ідентифікатор відкидається, і користувач не допускається до роботи. У випадку $Id = Id_i$ користувач пройшов ідентифікацію.
3. Суб'єкт аутентифікації робить запит у користувача на його аутентифікатор K і обчислює значення $Y = F(Id_i , K)$.
4. Суб'єкт аутентифікації виконує порівняння значень Y та E_i . За умови рівності цих значень встановлюється, що користувач успішно аутентифікований в системі. Інформація про цього користувача передається у програмні модулі, що використовують ключі користувачів. В противному випадку користувач не допускається до роботи.

Дану схему ідентифікації і аутентифікації користувача можна модифікувати за схемою 2.

Згідно зі схемою 2 в КС виділяється модифікований об'єкт-еталон, структуру якого показано в таблиці 2.

Таблиця 7.2

Номер користувача	Інформація для ідентифікації	Інформація для аутентифікації
1	Id_1 , S_1	E_1
2	Id_2 , S_2	E_2
…	…	…
N	Id_N , S_N	E_N

В схемі 2 значення $E_i = F(S_i , K_i)$, де – S_i – випадковий вектор, який задається при створенні ідентифікатора користувача; F – функція, що має властивість «невідновлюваності» значення K_i по відомим E_i та S_i .

Протокол ідентифікації і аутентифікації за схемою 2

1. Спочатку користувач пред'являє свій ідентифікатор Id .
2. При неспівпадінні Id з жодним із Id_i , які зареєстровані в КС, ідентифікатор відкидається, і користувач не допускається до роботи. У випадку $Id = Id_i$ користувач пройшов ідентифікацію.
3. За ідентифікатором Id_i виділяється (обирається) вектор S_i .
4. Суб'єкт аутентифікації робить запит у користувача на його аутентифікатор K_i і обчислює значення $Y = F(S_i , K_i)$.
5. Суб'єкт аутентифікації виконує порівняння значень Y та E_i . За умови рівності цих значень встановлюється, що користувач успішно аутентифікований в системі. В противному випадку користувач не допускається до роботи.

Необхідною умовою стійкості схем аутентифікації до відновлення інформації K_i є випадковий рівноймовірний вибір K_i з множини можливих значень.

Особливості застосування паролю для аутентифікації користувача

Кожен користувач КС отримує ідентифікатор і пароль, який на початку сеансу він пред'являє системі. На рисунку 7.1 проілюстровано метод підтвердження дійсності з використанням пароля, заснований на порівнянні пароля P_A,

що пред'являє користувач, з початковим значенням P'_A, що зберігається в комп'ютерному центрі. Оскільки пароль зберігається в таємниці, то перед пересиланням по незахищеному каналу він повинен шифруватися. Якщо P_A і P'_A збігаються, то пароль P_A вважається дійсним, а користувач – законним.

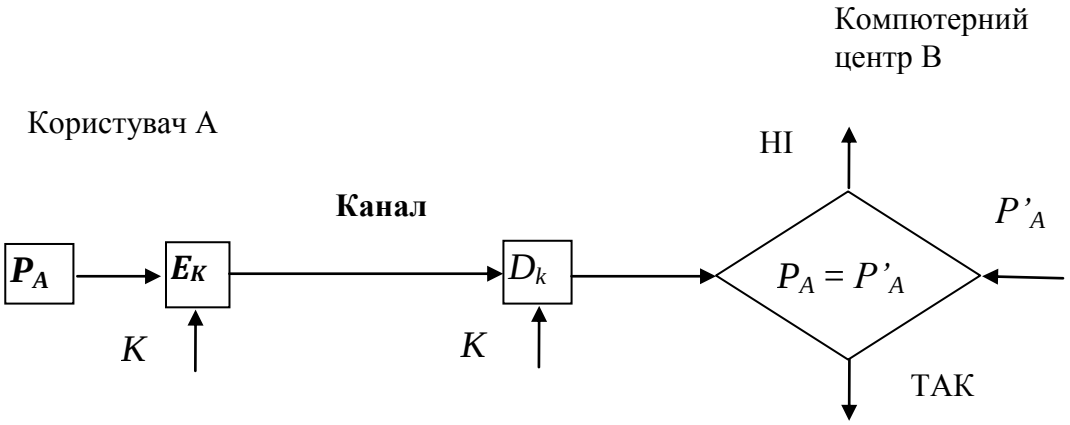


Рис.7.1 Схема простої аутентифікції за допомогою пароля

Доступ в таку систему може отримати порушник, якщо дізнається якимось чином пароль та ідентифікатор законного користувача.

Якщо одержувач не повинен розкрити початкову відкриту форму пароля, то відправник повинен переслати замість відкритої форми пароля відображення пароля, отримуване з використанням односторонньої функції $\alpha(\cdot)$ пароля, яке повинне гарантувати неможливість розкриття супротивником пароля за його відображенням, і порушник наштовхується на нерозв'язну задачу. Функція $\alpha(\cdot)$ може бути обчислена за формулою

$$\alpha(P) = E_p(Id),$$

де P – пароль відправника, Id – ідентифікатор відправника, E_p – функція шифрування, виконувана з використанням пароля P в якості ключа.

Якщо довжина ключа і пароля однакові, то такі функції особливо зручні, оскільки у випадку підтвердження дійсності за допомогою процес ідентифікації складається з пересилання одержувачу відображення $\alpha(P)$ і порівняння його попередньо обчисленим еквівалентом $\alpha'(P)$.

Щоб користувач запам'ятав пароль, він повинен бути коротким, однак це робить пароль вразливим до атаки повного перебору всіх варіантів. Для запобігання такої атаки функцію $\alpha(P)$ визначають так

$$\alpha(P) = E_{P \oplus K}(Id),$$

де K – ключ відправника, Id – ідентифікатор відправника.

Підтвердження дійсності пароля полягає у порівнянні двох відображень $\alpha(P_A)$ і $\alpha'(P_A)$ та визнання пароля P_A, якщо ці відображення рівні.

8 КРИПТОГРАФІЧНІ ПРОТОКОЛИ

8.1 Вступ

Криптографія вирішує проблеми секретності, перевірки істинності, цілісності та людської нечесності. Криптографічні алгоритми і методи являють собою лише академічний інтерес, якщо не використовуються для вирішення якої-небудь проблеми. Саме тому спочатку розглянемо протоколи.

Протокол – це порядок дій, які здійснюють дві або більше сторін, призначений для розв'язання визначеної задачі. «Порядок дій» означає, що протокол виконується у визначеній послідовності, з початку до кінця. Кожна дія повинна виконуватися в свою чергу і тільки після закінчення попередньої. «Здійснюваних двома або більше сторонами» означає, що для реалізації протоколу потрібні щонайменше дві людини, одна людина не зможе реалізувати протокол. Людина сама може виконати деякі дії, розв'язуючи задачу (наприклад, купуючи торт),але це не протокол (для того, щоб вийшов справжній протокол, хтось повинен з'їсти торт). Наостанок, «призначений для розв'язання визначеної задачі» означає, що протокол повинен призводити до результату. Щось схоже на протокол, але не розв'язуюче ніякої задачі – це не протокол, а втрата часу. У протоколів є й інші характеристики:

- Кожен учасник протоколу повинен знати протокол і послідовність дій, що його складають.
- Кожен учасник протоколу повинен погодитися дотримуватися протоколу.
- Протокол повинен бути непротирічним, кожна дія повинна бути визначена так, щоб не було можливості нерозуміння.
- Протокол повинен бути повним, кожній можливій ситуації повинна відповідати визначена дія.

Виконання протоколу відбувається по діям, лінійно, поки не буде команди перейти до наступної дії. Кожна дія включає принаймні одне з двох: обчислення, що виконуються однією або декількома сторонами, або повідомлення, якими обмінюються сторони.

Криптографічний протокол – це протокол, що використовує криптографію. Сторони можуть бути друзями і довіряти один одному або ворогами та не вірити один одному навіть при повідомленні часу доби. Криптографічний протокол містить деякий криптографічний алгоритм,але взагалі призначення протоколу виходить за рамки простої безпеки. Учасники протоколу можуть захотіти поділитися секретом один з одним, сумісно генерувати випадкову послідовність, підтвердити один одному свою істинність або підписати контракт в один і той самий момент часу. Сенс використання криптографії в протоколі – у запобіганні або виявленні шкідництва та шахрайства. Загальне правило можна сформулювати наступним чином: «Неможливо зробити або дізнатися більше, ніж визначено в протоколі». Як і для алгоритмів, значно легше довести можливу небезпечність протоколу, ніж його повну безпечність.

Сенс протоколів. У повсякденному житті майже для всього існують неформальні протоколи: замовлення товарів по телефону, гра в карти, голосування на виборах і так далі. Сьогодні все більше і більше людей спілкуються не особисто, а використовуючи комп'ютерну мережу. Наївно вважати, що корис-

тувачі комп'ютерних мереж завжди чесні. Також наївно вважати, що завжди чесні розробники комп'ютерних мереж. Формалізуючи протоколи, можна перевірити способи, використовувані шахраями для злому протоколів. Так можна розробити протоколи, стійкі до злому.

Гравці (учасники протоколу). Для демонстрації роботи протоколів використовуються декілька гравців-учасників. Перші двоє – A і B. Вони приймають участь у всіх двосторонніх протоколах. Як правило, A починає всі протоколи, а B відповідає. Якщо для протоколу потрібна третя або четверта сторона, у гру вступають C і D. Інші гравці грають спеціальні допоміжні ролі, вони будуть представлені пізніше (ролі зазначені в таблиці).

Ім'я гравця	Роль гравця
A	Перший учасник всіх протоколів
B	Другий учасник всіх протоколів
C	Третій учасник у протоколах за участю трьох та чотирьох сторін
D	Четвертий учасник у протоколах за участю трьох та чотирьох сторін
E	Зловмисник
M	Зломщик
T	Посередник, що заслуговує на довіру
W	Контролер, захищає A і B в багатьох протоколах
P	Свідок
V	Перевіряє істинність

Протоколи з посередником. Посередник – це незацікавлена третя сторона, якій довірено завершення протоколу. Незацікавленість означає, що у посередника немає зацікавленості в результаті роботи протоколу та схильності до однієї зі сторін. «Довірено» означає, що всі учасники протоколу приймають все, що скаже посередник, за істину, всі його дії – як правильні, і впевнені в тому, що посередник виконає свою частину протоколу. Посередники допомагають реалізувати роботу протоколів взаємодії сторін, що не довіряють одна одній.

В реальному світі в якості посередників часто виступають *юристи*. Наприклад, A продає незнайомому їй B автомобіль. *B* хоче заплатити чеком, але в A немає способу перевірити, чи дійсний чек. A хоче, щоб розрахунок за чеком був здійснений перш, ніж право власності перейде до *B*. *B*, який довіряє A не більше, ніж вона йому, не хоче передавати чек, не отримавши права власності.

Посередництво юриста влаштує обох. З його допомогою A і B можуть виконати наступний протокол, щоб захистити себе від обману:

1. *A* передає право власності юристу.
2. *B* передає чек юристу.
3. *A* депонує чек.
4. Дочекавшись сплати чека, юрист передає право власності *B*. Якщо чек не сплачений протягом визначеного часу, *A* доводить цей факт юристу, і той повертає право власності *A*.

В цьому протоколі А вірить, що юрист не передасть В право власності до тих пір, поки чек не буде сплачено, і поверне право власності А, якщо чек сплачено не буде. В вірить, що юрист буде володіти правом власності до тих пір, поки чек не буде сплачено, і передасть право власності В одразу ж після сплати чека. Юрист не турбується про сплату чека. Він у будь-якому випадку виконає свою частину протоколу, адже йому заплатять в будь-якому разі.

Іншим загальноприйнятим посередником є *нотаріус*. Коли *B* отримує від *A* завірений нотаріусом документ, він переконаний, що *A* підписав документ за своїм бажанням й власноручно. При необхідності нотаріус може виступити в суді і засвідчити цей факт.

Але з комп'ютерними посередниками існує ряд проблем:

1. Легко знайти нейтральну третю сторону, якій можна довіряти, якщо ви знаєте посередник та можете особисто побачити його. Дві сторони, що відносяться одна до одної з підозрою, з тією ж підозрою віднесуться і до посередника без обличчя, загубленого десь у мережі.
2. Комп'ютерна мережа повинна забезпечити підтримку посередника. Зайнятість юристів загальновідома, на кого в мережі впадуть додаткові витрати ?
3. Існує затримка, притаманна всім протоколам з посередником.
4. Посередник повинен приймати участь в кожній транзакції, що є вузьким місцем у великомасштабних реалізаціях будь-якого протоколу. Зростання числа посередників пом'якшить цю проблему, але зросте й ціна цієї послуги.
5. Оскільки кожен в мережі повинен довіряти посереднику, то посередник являє собою слабке місце мережі при спробі її злому.

Не дивлячись на це посередництво все ще активно використовується. У протоколах з використанням посередника цю роль буде грати *T*.

Арбітражні протоколи. Використовуваний завдяки високій вартості найму посередників арбітражний протокол може бути розбитий на два **підпротокола** нижнього рівня. Перший являє собою протокол без посередника, використовуваний при бажанні сторін виконати протокол. Другий являє собою протокол з посередником, що запрошується у виключних обставинах – при наявності розбіжностей між сторонами. Відповідний спеціальний посередник називається **арбітром.**

Арбітр, як і посередник, являє собою незацікавленого учасника протоколу, якому довіряють обидві сторони. На відміну від посередника він безпосередньо не приймає участь в кожній окремій реалізації протоколу і запрошується тільки для перевірки чесності виконання протоколу сторонами.

Професійними арбітрами є судді. На відміну від нотаріусів до суддів звертаються лише при появі розбіжностей. *A* і *B* можуть укласти контракт без участі судді. Суддя ніколи не дізнається про контракт, якщо одна зі сторін не подасть на іншу до суду. Протокол підписання контракту можна формалізувати наступним чином:

Протокол без посередника (виконується завжди):

1. *A* і *B* домовляються про умови контракту.
2. *A* підписує контракт.

3. *B* підписує контракт.

Протокол з використанням арбітра (виконується при наявності розбіжностей):

4. *A* і *B* постають перед суддею.
5. *A* надає свої докази.
6. *B* надає свої докази.
7. Суддя приймає рішення на підставі доказів.

Різниця використовуваних понять посередника та арбітра полягає в тому, що участь арбітра відбувається не завжди. Сторони звертаються до судді тільки при розбіжностях. Якщо розбіжностей немає, суддя не потрібен.

Існують арбітражні комп'ютерні протоколи. Вони припускають, що сторони , що приймають участь, чесні, але при підозрі про можливе шахрайство за існуючим набором даних третя сторона, якій довіряють учасники, зможе виявити факт шахрайства. Хороший арбітражний протокол дозволяє арбітру встановити і особистість шахрая. Арбітражні протоколи виявляють, а не попереджають шахрайство. Невідворотність виявлення виступає в якості попереджувальної міри, запобігаючи шахрайство.

Самодостатні протоколи. Самодостатній протокол є кращим типом протоколу. Він повністю забезпечує чесність сторін. Для виконання протоколу не потрібен посередник, не розв'язуючий суперечку арбітр. Самопобудова протоколу забезпечує відсутність суперечок. Якщо дна зі сторін спробує зшахраювати, шахрайство буде негайно виявлено іншою стороною, і протокол припинить виконуватися. Чого б на намагалась добитися сторона, що займається шахрайством, цьому не призначено статися.

У кращому світі будь-який протокол повинен бути самодостатнім, але, на лихо, *не існує* самодостатніх протоколів для кожної ситуації.

Спроби розкриття протоколів. Криптографічні спроби злому можуть бути направлені проти криптографічних алгоритмів, використовуваних у протоколах, проти криптографічних методів, використовуваних для реалізації алгоритмів та протоколів або безпосередньо проти протоколів. Будемо припускати, що криптографічні алгоритми й методи безпечні і розглядати тільки спроби розкриття протоколів.

Існує багато способів зламати протокол. Якщо хтось, хто не є учасником протоколу, «підслуховує» якусь частину протоколу або весь протокол, то це називається **пасивним** розкриттям, оскільки зломщик не може подіяти на протокол. Все, що він може зробити – це простежити за протоколом і спробувати добути інформацію. Цей тип розкриття відповідає розкриттю з використанням тільки шифротексту, тобто у крипто аналітика є шифротексти декількох повідомлень, зашифрованих одним і тим же алгоритмом шифрування, задача криптоаналітика полягає в розкритті відкритого тексту якомога більшої кількості повідомлень або, що краще, отриманні ключа (ключів), використовуваного для шифрування повідомлень, для дешифрування інших повідомлень, зашифрованих тими ж ключами. Оскільки пасивні розкриття важко виявити, протоколи намагаються запобігати, а не виявляти їх. В цих протоколах роль зловмисника буде грати *E*.

Зломщик може намагатися змінити протокол для власної вигоди. Він може видати себе за іншого, ввести нові повідомлення в протокол, замінити одне повідомлення іншим, повторно передати старі повідомлення, розірвати канал зв'язку або змінити інформацію, що зберігається на комп'ютері. Такі дії називаються **активним** розкриттям, оскільки вони потребують активного втручання.

Пасивні зломщики намагаються отримати інформацію про учасників протоколу. Вони збирають повідомлення, передані різними сторонами, і намагаються криптоаналізувати їх. Спроби *активного* розкриття, з іншого боку, переслідують більш широкий набір цілей. Зломщик може бути зацікавлений в отриманні інформації, яка погіршує роботу системи, або отриманні несанкціонованого доступу до ресурсів. Роль активного зломщика буде грати *M*.

Зломщиком може бути й один з учасників протоколу. Він може обманювати, виконуючи протокол, але зовсім не слідувати правилам протоколу. Такий зломщик називається **шахраєм. Пасивні шахраї** виконують правила протоколу, але намагаються отримати більше інформації, ніж передбачено протоколом. Активні шахраї порушують роботу протоколів, намагаючись зшахраювати.

Звичайно, протоколи повинні бути захищені від активного та пасивного шахрайства.

8.2 Протокол передачі інформації з використанням симетричної криптографії

Для того щоб двом сторонам безпечно обмінюватися інформацією,треба шифрувати свої повідомлення. Опишемо спрощений протокол (повний набагато складніший).

1. *A* і *B* обирають систему шифрування.
2. *A* і *B* обирають ключ.
3. *A* шифрує відкритий текст свого повідомлення з використанням алгоритму шифрування і ключа, отримуючи шифроване повідомлення.
4. *A* посилає шифроване повідомлення *B*.
5. *B* дешифрує шифротекст з використанням алгоритму шифрування і ключа, отримуючи текст повідомлення.

Що може *E*, знаходячись між *A* і *B*, узнати, підслуховуючи цей протокол? Якщо *E* може підслухати тільки передачу на етапі 4, йому доведеться піддавати шифротекст крипто аналізу. Це пасивне розкриття являє собою розкриття з використанням тільки шифротексту, застосовувані алгоритми стійкі (наскільки нам відомо) по відношенню до будь-яких обчислювальних потужностей, які може отримати *E* для вирішення проблеми.

Однак, Е може також підслухати й етапи 1 і 2. Тоді йому стануть відомі алгоритм та ключ – так само, як і *B*. Коли *E* перехопить повідомлення на етапі 4, то йому залишиться тільки дешифрувати його самостійно.

В хорошій криптосистемі *безпека* повністю залежить від *знань* ключа й абсолютно не залежить від знань алгоритму. Саме тому керування ключами так важливо в криптографії. Використовуючи симетричний алгоритм, *A* і *B* можуть відкрито виконувати етап 1, але етап 2 вони повинні зберігати в таємниці. Ключ

повинен залишатися в секреті перед, після та протягом роботи протоколу – до тих пір, поки повинно залишатися в таємниці повідомлення, що передається, – у противному випадку повідомлення одразу ж буде розкрито.

Активний зломщик *M* може зробити дещо інше. Він може спробувати порушити лінію зв'язку на етапі 4, зробивши так, що *A* взагалі не зможе передавати інформацію *B*. *M* також може перехопити повідомлення А та замінити його своїм власним. Якщо йому вдалося дізнатися ключ (перехопивши обмін інформацією на етапі 2 або зламавши криптосистему), він зможе зашифрувати своє власне повідомлення і відправити його *B* замість перехопленого, і *B* не зможе дізнатися, що повідомлення відправлено не учасником *A*. Якщо *M* не знає ключа, він може тільки створити повідомлення, що перетворюється при дешифровці у нісенітницю. *B*, вважаючи, що повідомлення відправив *A*, може вирішити, що або в *A*, або в мережі виникли серйозні проблеми.

Зі свого боку учасник *A* також може зіпсувати протокол: він може передати копію ключа учаснику *E*, і тоді *E* зможе читати все, що каже *B*, та надрукувати його слова в засобах масової інформації. Проблема тут не в протоколі. Учасник *A* і так може передавати *E* будь-які відкриті тексти, що передаються з використанням протоколу. Звичайно, те ж саме може зробити і учасник *B*. Протокол передбачає, що *A* і *B* довіряють один одному. Отже, симетричним криптосистемам притаманні наступні проблеми:

1. Розподіл ключів повинен проводитися в секреті. Ключі так само важливі, як і повідомлення, зашифровані цими ключами, оскільки знання ключа дозволяє розкрити всі повідомлення. Для розповсюджених систем шифрування задача розподілу ключів – найсерйозніша задача. Часто кур'єри особисто доставляють ключі за призначенням.
2. Якщо ключ скомпрометовано (вкрадено, розгадано, отримано за хабар і так далі), то учасник *E* зможе розшифрувати всі повідомлення, зашифровані цим ключем. Він зможе також виступити в якості однієї зі сторін і створювати хибні повідомлення, обдурюючи іншу сторону.
3. У припущенні, що кожна пара користувачів мережі використовує окремий ключ, загальна кількість ключів швидко зростає з ростом числа користувачів. Мережа з *n* користувачів вимагає $n(n–1)/2$ ключів. Наприклад, для спілкування 10 користувачів між собою потрібно 45 різних ключів, для 100 користувачів потрібно4950 ключів. Розв'язання проблеми – у зменшенні числа користувачів, але це не завжди можливо.

Однонаправлені функції. Поняття однонаправленої функції є центральним у криптографії з відкритими ключами. Вони не є протоколами безпосередньо, але являють собою основу («наріжний камінь») більшості протоколів.

Однонаправлені функції відносно легко обчислюються, але інвертуються з великим трудом. Тобто, знаючи *x*, просто розрахувати $f(x)$, але по відомому $f(x)$ нелегко обчислити *x*. Тут «нелегко» означає, що для обчислення *x* по $f(x)$ можуть знадобитися мільйони років, навіть якщо над цією проблемою будуть битися всі комп'ютери світу.

Хорошим *прикладом* однонаправленої функції служить розбита тарілка: легко розбити тарілку на тисячу крихітних шматочків. Однак, нелегко знову скласти тарілку з цих шматочків.

Математично строгого доведення існування однонаправлених функцій немає, немає й реальних свідоцтв можливості їх побудови. Не дивлячись на це, багато функцій виглядають в точності як однонаправлені: ми можемо розраховувати їх і, досі не знаємо простого способу інвертувати їх. Наприклад, в обмеженому околі легко обчислити x^2 , але набагато складніше $x^{1/2}$.

Однонаправлені функції безпосередньо не можна використовувати для шифрування. Повідомлення, зашифроване однонаправленою функцією, непотрібне – його не можна дешифрувати. Для криптографії з відкритими ключами потрібно щось інше, хоча існують і безпосередні криптографічні застосування одно направлених функцій (посвідчення дійсності).

Однонаправлені функції з секретом – це особливий тип однонаправленої функції, з секретною лазівкою. Її легко обчислити в одному напрямку і важко – в іншому. Але якщо відомий секрет, можна легко підрахувати і зворотну функцію. Тобто легко обчислити *f(x)* по заданому x, але важко по відомому *f(x)* обчислити *x*. Проте, існує невелика секретна інформація, *y*, що дозволяє, при знанні *f(x)* та *y*, легко обчислити *x*.

Однонаправлені хеш-функції – це інша частина фундаменту багатьох протоколів. *Хеш-функції* являють собою математичні або інші функції, які отримують на вхід рядок змінної довжини – *прообраз* – і перетворюють його в рядок фіксованої, зазвичай меншої, довжини – *значення* хеш-функції. В якості простої хеш-функції можна розглядати функцію, яка отримує прообраз і повертає байт, що являє собою XOR усіх вхідних бітів. Сенс хеш-функції полягає в отриманні характерної ознаки прообразу – значення, по якому аналізуються різні прообрази при розв'язанні зворотної задачі.

Однонаправлена хеш-функція – це хеш-функція, яка працює тільки в одному напрямку: легко обчислити значення хеш-функції по прообразу, але важко створити прообраз, значення хеш-функції якого дорівнює заданій величині. Хорошою однонаправленою хеш-функцією є хеш-функція **без зіткнень** – важко створити два прообрази з однаковим значенням хеш-функції.

8.3 Протокол передачі інформації з використанням криптографії з відкритими ключами

Криптографія з відкритими ключами використовує два різних ключа – один відкрити і один закритий. Визначення закритого ключа по відкритому вимагає величезних обчислювальних витрат. Хто завгодно, використовуючи відкритий ключ, може зашифрувати повідомлення, але не розшифрувати його. Розшифрувати повідомлення може тільки власник закритого ключа.

Математичною основою процесу є обговорювані раніше однонаправлені хеш-функції з секретом. Секретом служить закритий ключ, він робить дешифрування таким же простим, як і шифрування.

Опишемо *протокол*.

1. *A* і *B* узгоджують криптосистему з відкритими ключами.

2. *B* посилає *A* свій відкритий ключ.
3. *A* шифрує своє повідомлення і відправляє *B*.
4. *B* розшифровує повідомлення *A* за допомогою свого закритого ключа.

Зверніть увагу, що криптографія з відкритими ключами усуває проблему розподілення ключів, притаманну симетричним криптосистемам. Раніше *A* і *B* повинні були тайно домовитися про ключ. *A* міг обрати будь-який ключ, але їй треба було передати його *B*. *A* міг зробити це заздалегідь, але це потребує від нього певної передбачливості. *A* міг би послати ключ із секретним кур'єром, але для цього потрібен час. У учасника *E*, що підслуховує абсолютно все, є відкритий ключ *B* і повідомлення, зашифроване цим ключем, але він не зможе отримати ані закритий ключ *B*, ані текст повідомлення.

Якщо користувачі заздалегідь узгоджують використовувану криптосистему, у кожного з них є відкритий та закритий ключ, а відкриті ключі розміщують у загальнодоступній базі даних. В цьому випадку протокол виглядає ще простіше:

1. *A* витягує відкритий ключ *B* з бази даних.
2. *A* шифрує своє повідомлення за допомогою відкритого ключа *B* і посилає його *B*.
3. *B* розшифровує повідомлення *A* за допомогою свого закритого ключа.

В першому протоколі *B* повинен послати *A* свій відкритий ключ перш, ніж він міг відправити *B* повідомлення. Другий протокол більше схожий на звичайну пошту. *B* не приймає участь в протоколі до тих пір, поки він не почне читати повідомлення.

Змішані криптосистеми

В реальному світі алгоритми з відкритими ключами не замінюють симетричні алгоритми і використовуються не для шифрування повідомлень, а для шифрування ключів з наступних двох причин:

1. Алгоритми з відкритими ключами працюють повільно. Симетричні алгоритми принаймні у 1000 разів швидше, ніж алгоритми з відкритими ключами. Так, комп'ютери стають все швидше, але вимоги до обсягу інформації, що передається, також зростають, і завжди буде вимагатися шифрувати дані швидше, ніж це зможе зробити криптографія з відкритими ключами.
2. Криптосистеми з відкритими ключами уразливі по відношенню до розкриття з обраним відкритим текстом. Якщо $C=F(P)$, де P – відкритий текст з n можливих відкритих текстів, то криптоаналітику потрібно тільки зашифрувати всі n можливих текстів та порівняти результати з C. Він не зможе розкрити ключ дешифрування, але він зможе визначити P.

В більшості реалізацій криптографія з відкритими ключами використовується для засекречення та розповсюдження сеансових ключів, які використовуються симетричними алгоритмами для закриття потоку повідомлень. Такі реалізації називають змішаними (гібридними) криптосистемами.

Змішаний протокол має наступний зміст.

1. *B* посилає *A* свій відкритий ключ.

2. *A* створює випадковий сеансів ключ, шифрує його за допомогою відкритого ключа *B* та передає його *B*: $F_B(K)$.
3. *B* розшифровує повідомлення *A*, використовуючи свій закритий ключ, для отримання сеансового ключа: $D_k(F_B(K))=K$.
4. Обидва учасника шифрують свої повідомлення за допомогою одного сеансового ключа.

Використання криптографії з відкритими ключами для розподілення ключів вирішує дуже важливу проблему розподілення ключів. Якщо учасник Е отримає ключ шифрування симетричної криптографії, він зможе розшифрувати всі закриті цим ключем повідомлення. За допомогою наведеного протоколу при необхідності зашифрувати повідомлення створюється сеансів ключ, який знищується по закінченні сеансу зв'язку. Це значно зменшує ризик компрометації сеансового ключа.

8.4 Електронні цифрові підписи (ЕЦП)

Цифровий підпис має наступні властивості.

1. Підпис достовірний. Він переконує отримувача документу в тім, що той, хто підписав, свідомо це зробив.
2. Підпис непідробний. Він доводить, що саме той, хто підписав, і ніхто інший, свідомо підписав документ.
3. Підпис не може бути використаний повторно. Він є частиною документа, злодій не зможе перенести підпис на інший документ.
4. Підписаний документ не можна змінити. Після того, як документ підписано, його неможливо змінити.
5. Від підпису неможливо відмовитися. Підпис і документ матеріальні. Той, хто підписав, не зможе згодом стверджувати, що він не підписував документ.

В дійсності, жодне з цих тверджень не є повністю справедливим. Однак, шахрайство *ускладнено* і може бути *виявлено*.

Протокол підпису документу за допомогою симетричної криптосистеми та посередника. Учасник *A* хоче підписати цифрове повідомлення і відправити його *B*. Вона може це зробити за допомогою *T* і симетричної криптосистеми. Т – це посередник, що має владу, якому довіряють. Він може зв'язатися і з *A* і з *B*. Він видає секретний ключ K_A учаснику *A* та інший секретний ключ K_B – учаснику *B*. Ці ключі визначаються задовго до початку дії протоколу і можуть бути використані багаторазово для багатьох підписів.

Зміст проколу.

1. *A* шифрує своє повідомлення *B* ключем K_A та посилає його *T*.
2. *T*, знаючи ключ K_A , розшифровує повідомлення.
3. *T* додає до розшифрованого повідомлення твердження, що він отримав це повідомлення від *A*, шифрує це нове повідомлення ключем K_B .
4. *T* посилає нове повідомлення *B*.

5. *B* розшифровує повідомлення ключем K_B . Він може прочитати і повідомлення *A*, і підтвердження, що повідомлення відправлено саме учасником *A*.

Якщо *B* захоче показати *C* документ, підписаний *A*, він не зможе розкрити свій секретний ключ. Йому доведеться знову звернутися до *T*.

Тристоронній протокол підпису документа має наступний зміст.

1. Учасник *B* бере повідомлення *A* і твердження *T*, що повідомлення отримано від *A*, шифрує їх ключем K_B та посилає назад учаснику *T*.
2. *T* розшифровує отриманий пакет за допомогою ключа K_B .
3. *T* перевіряє свою базу даних і підтверджує, що відправником оригінального повідомлення був *A*.
4. *T* шифрує отриманий пакет за допомогою ключа K_C , який він виділив для *C*, і посилає *C* шифрований пакет.
5. *C* розшифровує отриманий пакет за допомогою ключа K_C , тепер *C* може прочитати і повідомлення, і підтвердження *T*, що повідомлення відправлено *A*.

Ці протоколи працюють, але вони вимагають від *T* немалих витрат часу. Він буде вузьким місцем будь-якої системи, навіть якщо він – комп'ютерна програма. Такого посередника, як *T*, якому будуть довіряти усі кореспонденти, важко знайти і важко зберегти.

Протокол підпису документу за допомогою асиметричної криптосистеми. Існують алгоритми з відкритими ключами. Які можна використовувати для цифрових підписів. *Основний протокол* простий:

1. *A* шифрує документ своїм закритим ключем, таким чином підписуючи його.
2. *A* посилає підписаний документ *B*.
3. *B* розшифровує документ, використовуючи відкритий ключ *A*, таким чином перевіряючи підпис.

Цей протокол краще попереднього, бо учасник *T* не потрібен ані для підпису документів, ані для його перевірки.

Підпис документа та мітка часу. При певних умовах учасник *B* може зшахраювати. Він може повторно використати документ і підпис сумісно. Це не має значення, якщо *A* підписав контракт (на одну копію підписаного контракту більше), але якщо *A* поставив цифровий підпис під чеком, *B*, який виступає в ролі шахрая, зможе зробити це ще раз (або більше разів).

Тому в цифрові підписи часто включають мітки часу. Дата і час підпису документа додаються до документа та підписуються разом із вмістом повідомлення.

Протокол підпису документу за допомогою асиметричної криптосистеми та однонаправлених хеш-функцій. На практиці алгоритми з відкритими ключами недостатньо ефективні для підпису великих документів. Для економії часу протоколи цифрового підпису нерідко використовують разом з однонап-

равленими функціями. Учасник *A* підписує не документ, а значення хеш-функції для даного документа. В цьому протоколі однонаправлена хеш-функція і алгоритм цифрового підпису узгоджуються заздалегідь.

Зміст протоколу:

1. *A* отримує значення однонаправленої хеш-функції для документа.
2. *A* шифрує це значення своїм закритим ключем, таким чином підписуючи документ.
3. *A* посилає *B* документ та підписане значення хеш-функції.
4. *B* отримує значення однонаправленої хеш-функції для документа, що надійшов від *A*. Потім, використовуючи алгоритм цифрового підпису, він розшифровує підписане значення хеш-функції за допомогою відкритого ключа *A*. Якщо підписане значення хеш-функції співпадає з розрахованим, підпис правильний.

Цифрові підписи і шифрування

Об'єднавши цифрові підписи та криптографію з відкритими ключами, ми розробляємо протокол, що комбінує безпеку шифрування і достовірність цифрових підписів.

Зміст протоколу:

1. *A* підписує повідомлення за допомогою свого закритого ключа: $S_A(M)$.
2. *A* шифрує підписане повідомлення відкритим ключем *B* і посилає його *B*: $F_B(S_A(M))$.
3. *B* розшифровує повідомлення за допомогою свого закритого ключа: $D_B(F_B(S_A(M))) = S_A(M)$.
4. *B* перевіряє підпис за допомогою відкритого ключа *A* та відновлює повідомлення: $V(S_A(M)) = M$.

8.5 Протоколи обміну ключами

Передача загального сеансового ключа в руки осіб, що обмінюються інформацією, являє собою складну проблему.

Протокол обміну ключами за допомогою симетричної криптографії. Цей протокол припускає, що користувачі мережі, *A* і *B*, отримують секретний ключ від Центру розподілення ключів, який в протоколах грає роль *T*. Перед початком протоколу ці ключі вже повинні бути у користувачів.

Зміст протоколу:

1. *A* звертається до *T* і запитує сеансів ключ для зв'язку з *B*.
2. *T* генерує випадковий сеансів ключ. Він зашифровує дві копії ключа: одну для *A*, а другу – для *B*. Потім *T* посилає обидві копії *A*.
3. *A* розшифровує свою копію сеансового ключа.
4. *A* посилає *B* його копію сеансового ключа.
5. *B* розшифровує свою копію сеансового ключа.
6. *A* і *B* використовують цей сеансів ключ для безпечного обміну інформацією.

Протокол обміну ключами за допомогою асиметричної криптографії. Зміст протоколу:

1. *A* отримує відкритий ключ *B* із Центру розподілення ключів.
2. *A* генерує випадковий сеансів ключ, зашифровує його відкритим ключем *B* і посилає його *B*.
3. *B* розшифровує повідомлення *A* за допомогою свого закритого ключа.
4. *A* і *B* шифрують свій обмін інформацією цим сеансовим ключем.

8.6 Ідентифікація

Коли учасник *A* підключається до головного комп'ютера, то їй треба пройти ідентифікацію. Зазвичай ця проблема вирішується за допомогою паролів: *A* вводить свій пароль, а сервер перевіряє його правильність.

Протокол ідентифікації за допомогою однонаправлених функцій. Зміст протоколу:

1. *A* посилає серверу свій пароль.
2. Сервер обчислює однонаправлену функцію пароля.
3. Сервер порівнює отримане значення з тим, що зберігається.

Оскільки сервер не зберігає більше таблицю правильних паролів всіх користувачів, знижується загроза того, що хтось пройде в сервер і викраде таблицю паролів.

Протокол ідентифікації за допомогою криптографії з відкритими ключами. Сервер зберігає файл відкритих ключів всіх користувачів, а всі користувачі зберігають свої закриті ключі. Зміст протоколу:

1. Сервер посилає *A* випадкову послідовність (рядок).
2. *A* шифрує цей рядок свої закритим ключем і посилає її назад серверу разом зі своїм ім'ям.
3. Сервер знаходить в базі даних відкритий ключ *A* та дешифрує повідомлення, використовуючи відкритий ключ.
4. Якщо відправлений спочатку та дешифрований рядки співпадають, сервер надає *A* доступ до системи.

Ніхто інший не зможе скористатися закритим ключем *A*, отже ніхто не зможе видати себе за неї. Що більш важливо, *A* ніколи не посилає на сервер свій закритий ключ, який повинен бути достатньо довгим і не бути мнемонічним. Але ані сервер, ані лінії зв'язку не зобов'язані бути безпечними.

Безпечні ідентифікаційні протоколи мають наступну, більш складну форму:

1. *A* виконує обчислення, засноване на деяких випадкових числах та своєму закритому ключі, і посилає результат на сервер.
2. Сервер посилає інше випадкове число.
3. *A* виконує деяке обчислення, засноване на випадкових числах (як створеному нею, так і отриманому від сервера) та своєму закритому ключі, і посилає результат на сервер.
4. Сервер виконує деяке обчислення для різних чисел, отриманих від *A*, та її відкритого ключа, перевіряючи, що йому (*A*) відомий його закритий ключ.
5. Якщо перевірка завершується успішно, особистість *A* підтверджується.

Якщо *A* довіряє серверу не в більшому степені, ніж той довіряє *A*, то А повинен зажадати підтвердження істинності сервера аналогічним чином.

9 ЛАБОРАТОРНИЙ ПРАКТИКУМ

9.1 Лабораторна робота 1

Тема. **Побудова та аналіз двійкової кодової системи з мінімальною надмірністю.**

Завдання.

1. Побудувати кодову систему Хаффмена, використовуючи наведений вище алгоритм.
2. Множину, що кодується, обрати довільно, при цьому число повідомлень у множині повинно бути не менше 10 (можна обрати приведений в таблиці 9.1 (стовпці 1 та 2) англійський алфавіт із заданим розподіленням ймовірностей для букв або ввести довільний рядок і підрахувати частоти букв).
3. Знайти числові характеристики отриманої кодової системи: ентропію H, вартість C та надмірність R=C–R.

Таблиця *9.1*

Буква			
пробіл	0,1859	0,20	10
A	0,0642	0,18	000
B	0,0127	0,10	011
C	0,0218	0,10	110
D	0,0317	0,10	111
E	0,1031	0,06	0101
F	0,0208	0,06	00100
G	0,0152	0,04	00101
H	0,0467	0,04	01000
I	0,0575	0,04	01001
J	0,0008	0,04	00110
K	0,0049	0,03	001110
L	0,0321	0,01	001111
M	0,0198		
N	0,0574		
O	0,0632		
P	0,0152		
Q	0,0008		
R	0,0484		
S	0,0514		
T	0,0796		
U	0,0228		
V	0,0083		
W	0,0175		
X	0,0013		

Y	0,0164		
Z	0,0005		

9.2 Лабораторна робота 2

Тема. Дослідження стандартів шифрування із секретним ключем Дослідження стандарту шифрування ГОСТ 28147-89

Мета. Вивчення принципів побудови сучасних симетричних криптографічних систем. Дослідження і апробація сучасних методів блокового криптографічного перетворення з секретним ключем: стандарту на шифрування даних ГОСТ 28147(89), стандарту США на шифрування даних DES.

Завдання.

1. Вивчити:
   стандарт криптографічного перетворення з секретним ключем ГОСТ 28147–89 (функції шифрування і таблиці вибору 32 раундових 32-бітових криптографічних ключів). Виділити в алгоритмі конструкцію Фейстеля.
2. Реалізувати програмно режим шифрування простою заміною.
3. Апробувати процес зашифрування і розшифрування різних варіантів тексту на різних криптографічних ключах за алгоритмом ГОСТ 28147–89 в режимі простої заміни:
   а) ввести в режимі ручного введення текст для шифрування довжиною не більше 240 символів або вибрати його з довільного файлу;
   б) ввести в режимі ручного введення 256-бітовий криптографічний ключ для шифрування або вибрати його з підготовлених файлів;
   в) виконати вибірку з початкового 256-бітового ключа 32-х раундових 32-бітових криптографічних ключів, вивести їх на екран монітора;
   г) вибрати і вивести на екран таблицю замін (підстановки) H, використовувану у функції шифрування;
   д) виконати процес зашифрування і розшифрування введеного відкритого тексту, отримати результати на всіх етапах шифрування на екрані монітора;
   е) впевнитися в коректності роботи програми і правильності проведених перетворень, порівняти початковий відкритий текст з результатом розшифрування його криптограми;
   ж) повторити пункти а–е для відкритих текстів різної довжини та нових криптографічних ключів;
   з) виконати пункти а–е для оригінальних відкритого тексту довжиною не більше 8 символів (64 біт) і 256-бітового криптографічного ключа.
4. Дослідити результати розшифрування при спотворенні елементів (бітів) криптограми (в умовах дії перешкод). Зробити висновки.

9.3 Лабораторная работа 3

Тема. Дослідження алгоритму шифрування з відкритим ключем RSA

Мета. Вивчення принципів побудови сучасних асиметричних криптографічних систем. Дослідження і апробація методу криптографічного перетворення з відкритим ключем за алгоритмом RSA.

Завдання.

1. Вивчити:
   стандарт криптографічного перетворення з відкритим ключем RSA, правила обчислення відкритого і закритого криптографічних ключів, прямого і оберненого криптографічних перетворень.
2. Розробити структури даних і реалізувати алгоритм RSA.
3. Апробувати процес зашифрування і розшифрування різних варіантів тексту на різних криптографічних ключах за алгоритмом RSA:
   а) ввести в режимі ручного введення текст для шифрування довжиною не більше 240 символів;
   б) виконати процедуру обчислення і вивести на екран відкритий та закритий (секретний) криптографічні ключі для шифрування або ввести їх в режимі ручного введення;
   в) представити введений відкритий текст як послідовність чисел у відповідності з обраним модулем криптографічного перетворення;
   г) виконати процес зашифрування і розшифрування введеного відкритого тексту, отримати результати за всіма етапами шифрування на екрані дисплея;
   д) впевнитися у коректності роботи програми, правильності виконання обчислень, порівняти первинний відкритий текст з результатом розшифрування його криптограми;
   е) повторити пункти а–д для нових відкритих текстів та криптографічних ключів різної довжини;
   ж) виконати пункти а–д для оригінальних відкритого тексту і криптографічних ключів.

Текст для шифрування та початкові значення простих чисел p і q для обчислення відкритого (E) і закритого (D) криптографічних ключів приведено в таблицях 9.2 і 9.3. Варіант тексту для шифрування визначається за передостанньою цифрою (i) номера залікової книжки (таблиця 3). Варіант значень p і q визначається за останньою (j) цифрою номера залікової книжки (таблиця 4). Отримані ключі e і d повинні задовольняти умові: $e > 7$ и $e \neq d$.

Таблиця 9.2

i	Текст для шифрування
0	ВЗЛОМ
1	ВИРУС
2	ДАТЧИК
3	ДОСТУП

Таблиця 9.3

j	Значення p	Значення q
0	3	17
1	3	19
2	3	23
3	3	29

4	МЕТОД
5	ОБЪЕКТ
6	ПАРОЛЬ
7	ПЭМИН
8	РЕЖИМ
9	СЕКРЕТ

4	5	11
5	5	17
6	7	17
7	7	19
8	13	17
9	19	29

Числові зображення (еквіваленти) літер алфавиту представлені в таблиці 9.4.

Таблиця 9.4

Буква	Число	Буква	Число	Буква	Число
А	1	К	12	Х	23
Б	2	Л	13	Ц	24
В	3	М	14	Ч	25
Г	4	Н	15	Ш	26
Д	5	О	16	Щ	27
Е	6	П	17	Ъ	28
Ё	7	Р	18	Ы	29
Ж	8	С	19	Ь	30
З	9	Т	20	Э	31
И	10	У	21	Ю	32
Й	11	Ф	22	Я	33

Прості числа від 2 до 2239 представлені в таблиці 9.5.

Таблиця 9.5

2	163	379	613	859	1109	1409	1657	1951
3	167	383	617	863	1117	1423	1663	1973
5	173	389	619	877	1123	1427	1667	1979
7	179	397	631	881	1129	1429	1669	1987
11	181	401	641	883	1151	1433	1693	1993
13	191	409	643	887	1153	1439	1697	1997
17	193	419	647	907	1163	1447	1699	1999
19	197	421	653	911	1171	1451	1709	2003
23	199	431	659	919	1181	1453	1721	2011
29	211	433	661	929	1187	1459	1723	2017
31	223	439	673	937	1193	1471	1733	2027
37	227	443	677	941	1201	1481	1741	2029
41	229	449	683	947	1213	1483	1747	2039
43	239	457	691	953	1217	1487	1753	2053
47	233	461	701	967	1223	1489	1759	2063
53	241	463	709	971	1229	1493	1777	2069
59	251	467	719	977	1231	1499	1783	2081
61	257	479	727	983	1237	1511	1787	2083
67	263	487	733	991	1249	1523	1789	2087
71	269	491	739	997	1259	1531	1801	2089
73	271	499	743	1009	1277	1543	1811	2099
79	277	503	751	1013	1279	1549	1823	2111
83	281	509	757	1019	1283	1553	1831	2113

89	283	521	761	1021	1289	1559	1847	2129
97	293	523	769	1031	1291	1567	1861	2131
101	307	541	773	1033	1297	1571	1867	2137
103	311	547	787	1039	1301	1579	1871	2141
107	313	557	797	1049	1303	1583	1873	2143
109	317	563	809	1051	1307	1597	1877	2153
113	331	569	811	1061	1319	1601	1879	2161
127	337	571	821	1063	1321	1607	1889	2179
131	347	577	823	1069	1327	1609	1901	2203
137	349	587	827	1087	1361	1613	1907	2207
139	353	593	829	1091	1367	1619	1913	2213
149	359	599	839	1093	1373	1621	1931	2221
151	367	601	853	1097	1381	1627	1933	2237
157	373	607	857	1103	1399	1637	1949	2239

9.4 Лабораторна робота 4

Тема. **Криптографічні протоколи**

Завдання.

1. Скласти заданий криптографічний протокол.
2. Кожен крок протоколу зобразити в описовій формі й формально.

Варіанти завдань

1. Передачі даних (ключової інформації або зашифрованої).
2. Шифрування даних на основі симетричного алгоритму.
3. Шифрування даних на основі асиметричного алгоритму.
4. Аутентифікації пароля на основі асиметричного алгоритму.
5. Аутентифікації (перевірки) ключа на основі асиметричного алгоритму.
6. Ідентифікації користувача системи на основі асиметричного алгоритму.
7. Аутентифікації пароля на основі симетричного алгоритму.
8. Аутентифікації (перевірки) ключа на основі симетричного алгоритму.
9. Аутентифікації на основі хеш-функції.
10. Ідентифікації користувача системи на основі хеш-функції.
11. Аутентифікації на основі розділення секрету (знання с нульовим розголошенням).
12. Передачі даних (ЕЦП) на основі симетричного шифрування.
13. Передачі даних (ЕЦП) на основі асиметричного шифрування.
14. Ідентифікації користувача системи на основі ЕЦП.
15. Ідентифікації користувача системи на основі хеш-функції.
16. Передачі даних (хеш-коду).
17. Шифрування даних на основі хеш-функції.
18. Шифрування даних на основі ЕЦП.
19. Аутентифікації пароля на основі хеш-функції.
20. Ідентифікації користувача системи на основі розділення секрету (знання с нульовим розголошенням).

Контрольна робота

Тема 1. *Побудова кодової системи із застосуванням загального алгоритму алфавітного кодування.*

Завдання 1. Заданий алфавіт, що згенерований імовірнісним джерелом, з деяким розподілом вірогідності появи букв. Написати програму, що реалізовує побудову якнайкращої алфавітної кодової системи для заданого алфавіту за допомогою загального алгоритму алфавітного кодування (див. підрозділ 1.5), тобто для кожної букви алфавіту побудувати якнайкращий алфавітний код. Для побудованої кодової системи знайти числові характеристики: ентропію, вартість і надмірність. Показати роботу загального кроку алгоритму, тобто вивести на екран по кроках розбиття підалфавітів на підсистеми, кодові підсистеми і їх часткові вартості.

Тема 2. *Кодування за методом зонного стиску.*

Метод зонного стиску інформації

При записі інформації в пам'яті ЕОМ кожній букві або цифрі відводиться зазвичай 8 двійкових розрядів (байт), бо останній є найменшою структурною одиницею пам'яті, що адресується.

Необхідність стискування буквеної інформації напрошується вже виходячи з того, що 8 двійкових розрядів, що становлять байт, дозволяють закодувати двійковим кодом алфавіт з 256 букв, тоді як реальні алфавіти, разом з цифрами і допоміжними символами зазвичай не перевищують 50-60 знаків. Проте і 50-60 знаків вимагають двійкових комбінацій або 5-6-бітову структуру осередку. Така структура не вирішує проблем, оскільки в 2-3-х бітах, що залишилися, можна записати лише 4-8 букв (не кажучи вже про проблему прочитування).

Спробуємо тепер використовувати ½ байта для представлення букв деякого абстрактного алфавіту, потім кодувати інформацію в цьому 16-ти-буквенном (m=16) алфавіті по n=2 букви в кодовому слові. Тоді, використовуючи один байт, можна буде передати ті ж символів.

буква	Код
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Рис. 1

Якщо цей 16-ти-буквенный алфавіт побудувати так, щоб 13 якісних ознак використовувати як основні символи, а 3 як допоміжні, то можна побудувати алфавіт, зображений на рисунку 1.

Перші 13 символів умовно називатимемо ЦИФРА, а інші 3 - ЗОНА (співвідношення "цифр" і "зон" обумовлюється кількістю кодових слів у вторинному алфавіті і може мінятися від 8:8 до 15:1).

Кодові слова у вторинному алфавіті надалі будуватимемо таким чином, що перші 4 розряди завжди представлятимуть зону, а другі 4 - цифри. Число можливих комбінацій вторинного алфавіту в цьому випадку зменшиться і буде рівне N=3(13=39 (4 зони дали б 12(4=48; п'ять - 11(5=55 і так далі), та зате такий вторинний алфавіт дає принципову можливість побудови 4-бітової структури пам'яті і дозволяє застосувати зонний метод стискування інформації.

Ідея зонного стискування базується на тому, що букви вторинного алфавіту розбиваються по зонах, і якщо в тексті поруч зустрічаються букви, що належать одній зоні, то номер її вказують тільки перед першою буквою, а запис наступних букв обмежується записом їх цифрової частини.

Якщо окрім частоти появи окремих букв в тексті врахувати ще і вірогідність різних буквених поєднань, то кодові слова у вторинному алфавіті матимуть вигляд, представлений в таблиці 1 (вірогідність вказує тільки частоту букви в російському тексті).

Таблиця 1

зона	D			E			F		
	код	буква	вірогідність	код	буква	вірогідність	код	буква	вірогідність
1	D0	пробіл	0,175	E0	З	0,16	F0	Ц	0,004
2	D1	О	0,09	E1	У	0,021	F1	Ж	0,007
3	D2	Е	0,072	E2	Д	0,025	F2	Х	0,009
4	D3	А	0,062	E3	Я	0,018	F3	Ч	0,012
5	D4	Р	0,04	E4	Ь, Ъ	0,014	F4	Э	0,003
6	D5	П	0,023	E5	Ф	0,002	F5	Ю	0,006
7	D6	Т	0,053	E6	Ы	0,016	F6	,	
8	D7	Н	0,053	E7	Щ	0,003	F7	.	
9	D8	В	0,038	E8	Ш	0,006	F8	;	
10	D9	И	0,062	E9	Б	0,014	F9	:	
11	DA	С	0,045	EA	Г	0,013	FA	!	
12	DB	М	0,026	EB	К	0,028	FB	?	
13	DC	Л	0,035	EC	Й	0,01	FC	–	

Коефіцієнт стиску визначається як відношення кількості байтів в початковому тексті n1 до байтів в стислому тексті n2 : Ксж = n1 / n2 .

***Завдання 2*.** Перекласти задане повідомлення на російську мову і застосувати до одержаного повідомлення метод зонного стиску (крапки не враховувати). Обчислити коефіцієнт стиску.

Варіанти.

1. Continue reading this page for highlights of new content in the Library.
2. It contains technical programming information, including sample code.
3. The kit includes build documentation, a hardware reference specification.
4. This article is an introduction to creating your own Microsoft Visio shapes.
5. Drop a Process shape on your drawing and make a modification.
6. Drag the shape onto the blank stencil to create a master.
7. The icon can also be edited to create a more meaningful symbol on the stencil.
8. You can add your shape to the existing stencil by clicking Yes on the above error message.
9. Select the shape and choose Special from the Format menu to enter the copyright information.
10. On the Insert menu, click Picture, and then select a graphic image to insert in your drawing.
11. This turns it into a Visio shape so you can then change its components or add text.

12. Drag the object's selection handles to crop, or drag inside the border to pan the object.
13. Click the arrow next to the Rectangle tool to select the Rectangle or Ellipse tool.
14. The shape will take on an opaque fill if it is successfully closed.
15. The Pencil tool can draw lines and arcs, in addition to the separate Line and Arc tools.
16. The following steps show how to create a repeated series of shapes with equal spacing.
17. To create a repeated series of lines or shapes with equal spacing.
18. When you glue something to a shape's vertex, Visio creates a connection point.
19. The following steps show how to add, move, and delete a new connection.
20. Visio adds a connection point to that location to show the new connection point.
21. A Connection object represents a unique session with a data source.
22. This abstract class defines parameters for asymmetric cryptographic algorithms.
23. Creates a cryptographic object to perform the asymmetric algorithm.
24. Many public-key systems can also be used to form digital signatures.
25. Returns the name of the current Visual FoxPro resource file.

Контрольні питання до заліку з дисципліни

1. Джерела повідомлень ті їх класифікація.
2. Поняття коду, кодування. Представлення кодів.
3. Рівномірні і нерівномірні коди.
4. Скінченні комбінаторне і ймовірнісне джерела повідомлень.
5. Ентропія, вартість, надмірність джерела.
6. Класифікація кодів. Двійкові кодові системи.
7. Кодова система Хаффмена, алфавітна кодова система.
8. Ефективні коди. Оптимальні нерівномірні коди.
9. Теорема про кодування. Теорема Шеннона.
10. Алгоритм побудови алфавітної кодової системи.
11. Префіксна множина. Коди з мінімальною надмірністю.
12. Побудова кодів Шеннона-Фано.
13. Метод оптимального кодування. Повна кодова система.
14. Загальний алгоритм алфавітного кодування.
15. Порядкове число. Кодова площина.
16. Інверсні коди.
17. Геометрична модель цифрових кодів. Кодове дерево.
18. Геометрична інтерпретація властивостей префіксних кодів.
19. Необхідні і достатні умови однозначного декодування.
20. Сегментні класи й алгоритми їх побудови.
21. Типи алгоритмів шифрування.
22. Стійкість алгоритмів шифрування.
23. Алгоритми блокового шифрування.
24. Симетричні алгоритми: ГОСТ 28147-89.
25. Симетричні алгоритми: DES,AES.
26. Асиметричні алгоритми шифрування.

27. Алгоритм RSA.
28. Криптосистема Ель-Гамаля.
29. Криптосистема Діффі-Хелмана.
30. Криптосистема Рівеста-Шаміра-Адлемана.
31. Аутентифікація і підпис.
32. Алгоритми генерації та перевірки електронно-цифрового підпису (ЕЦП).
33. ДСТУ 4045-2002.
34. ЕЦП на основі симетричних і асиметричних алгоритмів.
35. Криптосистеми на основі еліптичних кривих.
36. Типи хеш-функцій.
37. Вимоги до хеш-функцій.
38. Ключова інформація.
39. Вимоги до криптографічних протоколів. Типи протоколів.
40. Криптопротоколи на основі симетричних і асиметричних алгоритмів.
41. Класифікації криптографічних протоколи.
42. Атаки на протоколи. Засоби захисту протоколів.
43. Формальні методи аналізу протоколів.

СПИСОК ДЖЕРЕЛ

1. Шнайер Б. Прикладная криптография. 2-е изд. Электронная версия.
2. Хамидуллин Р.Р., Бригаднов И.А., Морозов А.В. Методы и средства защиты компьютерной информации: Учеб. Пособие.– СПб.: СЗТУ, 2005. –178 с. Электронная версия.
3. http://www.intuit.ru/studies/courses/
4. Кнут Д. Искусство программирования для ЭВМ. Т. 1. Электронная версия.
5. Домарев, В. В. Защита информации и безопасность компьютерных систем [Текст] / В. В. Домарев. К. : ДиаСофт, 1999. - 480 с. – 3 прим.
6. Вертузаєв М. С., Юрченко О. М. Захист інформації в комп`ютерних системах від несанкціонованого доступу: Навч. посібник / Ред. Лаптєв С. Г.; Європ. ун-т. К.: Вид-во Європ. ун-ту, 2001, укр. – 1 прим.
7. Коваленко М. М. Комп`ютерні віруси і захист інформації: Навч. посібник / Національна академія внутрішніх справ України. – 268 с. К.: Наукова думка, 1999. , укр. – 6 прим.
8. Галатенко В. А. Основы информационной безопасности: Курс лекций / Ред. Бетелин В. Б.; ИНТУИТ. - 278 с.: ил. - (Основы информ. технологий). М.: , 2003, рос. – 1 прим.
9. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. М., изд-во «ДМК», 2000.
10. Домашев А.В. и др. Программирование алгоритмов защиты информации. Учебное пособие. М., Изд-во «Нолидж», 2000 г. – 279 с.
11. Левин М. Криптография: Руководство пользователя. М., Изд-во «Познавательная книга плюс», 2001. – 319 с.
12. Введение в криптографию. Учебник. Под ред. В.В.Ященко. М., Изд-во «Познавательная книга плюс», 2001. – 287 с.