Лекція 10.

Тема. Канали передачі та прийому інформації.

Розділ. Захист інформації в незахищених каналах обміну інформацією.

План лекції:

10.1. Види кодування інформації в незахищених каналах обміну інформацією.

10.2. Симетричні системі кодування.

10.3. Асиметричні системи кодування.

10.3.1. Генерація таємних ключів кодування.

10.4. Реалізація асиметричної системи кодування у протоколі Kerneros.

10.1. Види кодування інформації в незахищених каналах обміну інформацією.

В зв'язку з широким використанням загальнодоступних каналів обміну інформацією (інтернет, телефонні лінії, телекомунікаційні канали) виникає задача захисту інформації, що передається. Для захисту інформації, що передається через незахищені канали зв'язку, використовується кодування інформації з використанням програмних кодових ключів. Для можливості розкодування інформації кодові ключі повинні бути відомі обом сторонам, що виконують обмін. Тоді виникає задача генерації, зберігання ключів та обміну ними. Залежно від способу генерації та обміну ключами розрізняють симетричні і несиметричні системи кодування інформації (далі криптосистеми).

Симетричні криптосистеми (симетричне кодування, симетричні шифри) - спосіб кодування, в якому для кодування і розкодування застосовується один і той же кодовий (криптографічний) ключ. Ключ алгоритму кодування повинен зберігатися в таємі обома сторонами. Тому симетричну систему ще називають криптосистемою з закритими ключами. Алгоритм кодування обирається сторонами до початку обміну повідомленнями і не передаються через незахищені канали. Симетричні криптосистеми виникли раніше асиметричних. Найбільш відомі симетричні криптосистеми - DES (Data Encryption Standard, стандарт кодування даних), 3DES (Triple-DES, потрійний DES), AES (Advanced Encryption Standard, покращений стандарт кодування).

Асиметричні криптоситеми – спосіб кодування, в якому відкритий ключ передається по незахищеному каналу і використовується для генерації таємного ключа обома сторонами обміну. Таємний ключ не передається через канали обміну, а тому незахищений канал не може стати причиною розсекречення. Таємний ключ використовується для кодування і розкодування повідомлень. Асимертичні криптосистеми дістали назву систем з відкритим ключем. Найбільш відомі асиметричні ситсеми - RSA (Rivest-

Shamir-Adleman), DSA (Digital Signature Algorithm). Асимертичні криптосистеми широко застосовуються в відомих мережевих протоколах, зокрема, в протоколах TLS і його попереднику SSL (що лежать в основі HTTPS), в SSH.

10.2. Симетричні системі кодування.

Головним принципом у них є умова, що передавач і приймач заздалегідь знають алгоритм кодування і таємний ключ, без яких інформація являє собою всього лише набір символів, що не мають сенсу.
Головні симетричні криптографічні алгоритми перераховані нижче:

- Проста перестановка.
- Одиночна перестановка по ключу.
- Подвійна перестановка.
- Перестановка "Магічний квадрат"

Проста перестановка

Проста перестановка без ключа - один з найпростіших методів кодування. Повідомлення записується в таблицю в колонки. Після того, як відкритий текст записаний колонками, для утворення коду він зчитується по рядках. Для використання цього коду відправнику і отримувачу потрібно домовитися про спільний ключ у вигляді розміру таблиці. Об'єднання букв в групи не входить у ключ шифру і використовується лише для зручності запису несмислового тексту.

Одиночна перестановка по ключу.

Більш практичний метод кодування, названий одиночною перестановкою по ключу дуже схожий на попередній. Він відрізняється лише тим, що колонки таблиці переставляються за ключовим словом, фразі або набору чисел довжиною в рядок таблиці.

Подвійна перестановка.

Для відвищення таємності закодоване повідомлення додатково кодується ще раз. Цей спосіб отримало назву - подвійна перестановка. Для цього розмір другої таблиці підбирають так, щоб довжини її рядків і колонок були іншими, ніж в першій таблиці. Найкраще, якщо вони будуть взаємно простими. Крім того, в першій таблиці можна переставляти стовпці, а в другій рядки. Нарешті, можна заповнювати таблицю згідно якоїсь функції або якимось іншим способом.

Перестановка «Магічний квадрат».

Магічними квадратами називаються квадратні таблиці з вписаними в їх клітини послідовними натуральними числами починаючи від 1, які дають в сумі по кожному стовпцю, кожному рядку і кожній діагоналі одне і те ж число. Подібні квадрати широко застосовувалися для вписування тексту що шифрується за наведеною в них нумерацією. Якщо потім виписати вміст таблиці по рядках, то виходила шифровка перестановкою літер. На перший погляд здається, ніби магічних квадратів дуже мало. Проте їх число дуже швидко зростає із збільшенням розміру квадрата. Так, існує лише один магічний квадрат розміром 3 х 3, якщо не брати до уваги його повороти.

Магічних квадратів 4 х 4 налічується вже 880, а число магічних квадратів розміром 5 х 5 близько 250000. Тому магічні квадрати великих розмірів застосовувалися для надійної системи кодування, коли перебори робилися вручну, тому що ручний перебір всіх варіантів ключа для цього коду був надзвичайно великим. Якщо у квадрат розміром 4 на 4 вписати числа від 1 до 16, сума чисел по рядках, стовпцях і повним діагоналям дорівнює одному й тому числу - 34. Вперше ці квадрати з'явилися в Китаї, де їм і була приписана якась «магічна сила».

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

Кодування по магічному квадрату вироблялося наступним чином. Наприклад, потрібно закодувати фразу: «ВідлітаюЗавтра.». Букви цієї фрази вписуються послідовно в квадрат згідно записаним в них числах: позиція літери в реченні відповідає порядковому числу. У порожні клітини ставиться крапка.

16.	3д	2і	13р
5і	10а	11в	8ю
9З	6т	7а	12т
4л	15.	14а	1В

Після цього закодований текст записується в рядок (зчитування проводиться зліва направо, порядково):

.діріавюЗтатл.аВ

При розкодуванні текст вписується в квадрат, і розкодований текст читається згідно послідовності чисел «магічного квадрата». Програма розкодування повинна генерувати «магічні квадрати» і по таємному ключу вибирати необхідний.

Повна втрата всіх статистичних закономірностей закодованого повідомлення є важливою вимогою до симетричного коду. Для цього код повинен мати «ефект лавини» - при зміні хоча б одного біту даних повинен змінюватись вихідний код дуже сильно. Також важливою вимогою є відсутність лінійності в алгоритмі кодування.

Розрізняють такі симетричні коди:

- блокові коди. Обробляють інформацію блоками певної довжини (зазвичай 64, 128 біт), застосовуючи до блоку ключ в установленому порядку, як правило, кількома циклами перемішування і підстановки, що мають назву раундів. Результатом повторення раундів є лавинний ефект - наростаюча втрата відповідності бітів між блоками відкритих і закодованих даних.

- потокові коди, в яких кодування проводиться над кожним бітом або байтом вихідного (відкритого) тексту з використанням операції XOR (Виключне АБО).

Більшість симетричних кодів використовують складну комбінацію великої кількості підстановок і перестановок у великій кількості проходів, використовуючи на кожному проході «ключ проходу». Безліч «ключів проходу» для всіх проходів називається «розкладом ключів» (key schedule). Як правило, воно створюється з єдиного таємного ключа виконанням над ним якихось операцій, в тому числі перестановок і підстановок. Операція перестановки перемішує біти повідомлення по якомусь закону. В апаратних реалізаціях вона тривіально реалізується як плутанина провідників. Саме операції перестановки дають можливість досягнення «ефекту лавини». Операція перестановки линейна - f (a) xor f (b) = = f (a xor b). Тому додатково використовується операція підстановки як заміна значення якоїсь частини повідомлення (часто в 4, 6 або 8 біт) на стандартне, жорстко вбудоване в алгоритм інше число шляхом звернення до масиву констант. Операція підстановки привносить в алгоритм нелінійність.
Найчастіше стійкість алгоритму, особливо до диференціального криптоаналізу, залежить від вибору значень в таблицях підстановки (S-блоках). Як мінімум вважається небажаним наявність нерухомих елементів S (x) = x, а також відсутність впливу якогось біта вхідного байта на якийсь біт результату - тобто випадки, коли біт результату однаковий для всіх пар вхідних слів, що відрізняються тільки в даному біті .

Існує безліч не менше двох десятків алгоритмів симетричних шифрів, істотними параметрами яких є стійкість, довжина ключа, число раундів, довжина оброблюваного блоку, складність апаратної / програмної реалізації, складність перетворення.

10.3. Асиметричні системи кодування.

Опублікована в листопаді 1976 року стаття Уітфілд Діффі і Мартіна Хеллмана «Нові напрямки в криптографії» (англ. New Directions in Cryptography) перевернула уявлення про криптографічні системи, заклавши основи криптографії з відкритим ключем. Розроблений згодом алгоритм Діффі - Хеллмана дозволив двом сторонам отримати спільний таємний ключ, використовуючи нетаємний ключ, який передається через незахищений канал зв'язку. Однак цей алгоритм не вирішував проблему аутентифікації. Без додаткових затрат користувачі не могли бути впевнені, з ким саме вони згенерували спільний таємний ключ. Вивчивши цю статтю, троє вчених Рональд Ривест, Аді Шамір і Леонард Адлеман з Массачусетського технологічного інституту (MIT) знайшли алгоритм, заснований на відмінності в тому, наскільки легко знаходити великі прості числа і наскільки складно розкладати на множники добуток двох великих простих чисел, який отримав згодом назву RSA. Система була названа за першими літерами прізвищ її творців. Повний опис нової криптосистеми булв опублікований в журналі «Communications of the ACM» в лютому 1978 року. Заявка на патент була подана 14 грудня 1977 року, в якості власника був вказаний MIT. Патент 4405829 був виданий 20 вересня 1983 року, а 21 вересня 2000 року

строк його дії закінчився. Проте за межами США у винахідників патенту на алгоритм не було через патентне законодавство.
У 1982 році Ривест, Шамір і Адлеман організували компанію RSA Data Security (англ.) (в даний момент - підрозділ EMC). У 1989 році RSA, разом з симетричним шифром DES, згадується в RFC 1115, тим самим починаючи використання алгоритму в мережі Internet, а в 1990 році використовувати алгоритм починає міністерство оборони США. У листопаді 1993 року відкрито публікується версія 1.5 стандарту PKCS1 (англ.), що описує застосування RSA для шифрування і створення електронного підпису. Остання версія стандарту доступна у вигляді RFC 3447 - 2.1, 2002 рік.

Ідея криптографії з відкритим ключем дуже тісно пов'язана з ідеєю односторонніх функцій, тобто таких функцій f(x), що за відомим x досить просто знайти значення f(x), тоді як за значенням f(x) неможливо знайти x за розумний термін. Але сама одностороння функція марна в застосуванні: нею можна зашифрувати повідомлення, але розшифрувати неможливо. Тому криптографія з відкритим ключем використовує односторонні функції з лазівкою. Лазівка - це такий таєм, який допомагає розшифрувати повідомлення. Тобто існує такий y, що знаючи y і f(x), можна вирахувати x. Для прикладу, якщо розібрати дерев'яний будинок на колоди і інші частини, то дуже складно зібрати знову такий самий будинок, але якщо пронумерувати всі частини перед розбиранням і записати послідовність розбирання згідно нумерації, то можна легко зібрати будинок, використовуючи записану послідовність і зворотній порядок нумерації.
На прикладі зберігання паролів в комп'ютері можна зрозуміти дію цієї системи. Кожен користувач в мережі має свій пароль. При вході він вказує ім'я і вводить таємний пароль. Якщо зберігати пароль на диску комп'ютера, то хтось його може зчитати і отримати доступ до таємної інформації. Для вирішення завдання використовується одностороння функція. При створенні таємного пароля в комп'ютері зберігається не сам пароль, а результат обчислення функції від цього пароля та імені користувача. Наприклад, користувач sam придумала пароль jurav1. При збереженні цих даних обчислюється результат функції f(jurav1), нехай результатом буде рядок абрівіатура, яка і буде збережена в системі. В результаті файл паролів прийме наступний вигляд:
Ім'я (імя_пароль)
sam абрівіатура
Вхід в систему тепер виглядає так:
Ім'я: sam
Пароль: jurav1
Коли sam вводить пароль jurav1, комп'ютер перевіряє, дає чи ні функція, f(jurav1), правильний результат абрівіатура, що зберігається на диску комп'ютера. «Таємний» пароль jurav1 не зберігається в комп'ютері ні в якому вигляді. Файл паролів може бути тепер переглянутим іншими користувачами без втрати таємності, так як функція f(jurav1) практично незворотня. У цьому прикладі використовується одностороння функція без лазівки, оскільки не

потрібно по зашифрованому повідомленню отримати вхідне. У наступному прикладі розглядається схема з можливістю відновити вихідне повідомлення за допомогою «лазівки», тобто важкодоступній інформації. Для шифрування тексту можна взяти великий абонентський довідник, що складається з декількох товстих томів (по ньому дуже легко знайти номер будь-якого жителя міста, але майже неможливо за відомим номером знайти абонента). Для кожної букви з інформаційного повідомлення вибирається ім'я, що починається на ту ж літеру. Таким чином букві ставиться у відповідність номер телефону абонента. Інформаційне повідомлення, наприклад «Мастак», буде закодоване наступним чином:

М	Михайлов	345-84-34
а	Абакумов	548-47-23
с	Соломенко	5467-34-85
т	Туз	674-56-95
а	Андрійченко	536-64-76
к	Костильов	967-02-57

Закодоване повідомлення буде списком чисел, які відповідають відповідним літерам інформаційного повідомлення. Лазівкою до розкодування цього закодованого повідомлення буде довідник з сортуванням по зрастанню номерів телефонів. Якщо не мати такого відсортованого довідника, розкодувати повідомлення можливо тільки за дуже довгий час.

Асимертична система кодування використовує алгоритм Ди́ффи — Хе́ллмана **(англ. *Diffie-Hellman*, *DH*).**

Нехай обом абонентам відомі деякі два числа g і p (наприклад, вони можуть бути «зашиті» в програмне забезпечення), які не є таємними і можуть бути відомі також іншим зацікавленим особам. Для того, щоб створити невідомий більш нікому таємний ключ, обидва абонента генерують великі випадкові числа: перший абонент - число a, другий абонент - число b. Потім перший абонент обчислює певне значення і пересилає його другому, а другий, використовуючи отримане значення і свій таємний ключ, обчислює своє певне значення і передає першому. Передбачається, що зловмисник може отримати обидва ці значення, але не може змінити їх (тобто в нього немає можливості втрутитися в процес передачі). На другому етапі, перший абонент на основі наявної в нього і отриманого з мережі значень обчислює таємний ключ, а другий абонент на основі наявної в нього і отриманого по мережі значень обчислює свій таємний ключ. Якщо обидва абоненти отримали одне і те ж значення спільного таємного ключа, його вони і можуть використовувати в якості таємного ключа для кодування і розкодування, оскільки зловмисник зустрінеться з практично нерозв'язною (за розумний час) проблемою обчислення таємного ключа по перехопленим даним і, якщо числа вибрані досить великими.

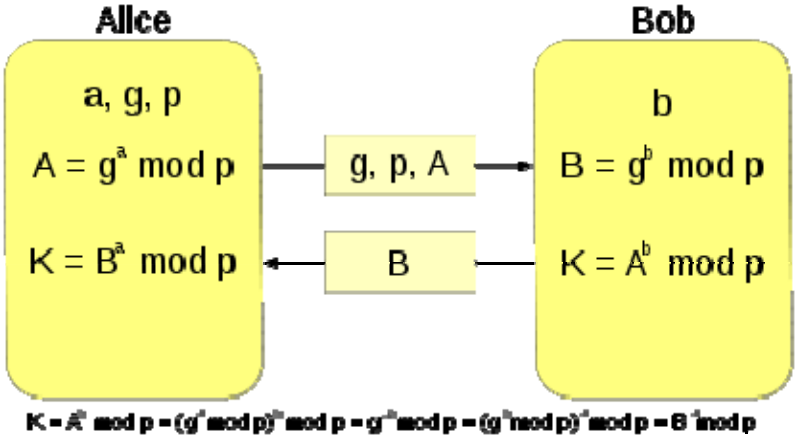

Рисунок 10.1. Алгоритм Діффі - Хеллмана, де K - підсумковий спільний таємний ключ

При роботі алгоритму, кожна сторона:

1. Генерує випадкові натуральні числа **a** і **b** - закриті ключі.

2. Спільно з віддаленою стороною встановлює відкриті параметри **p** і **g** (зазвичай значення **p** і **g** генеруються на одній стороні і передаються іншій), де:

p є випадковим простим числом,

g є первообразним коренем за модулем **p.**

3. Передавач підраховує відкритий ключ A, використовуючи перетворення над закритим ключем **a**:

$A = g^a \bmod p$.

4. Передавач обмінюється відкритими ключами **A, g, p** з віддаленою стороною.

5. Приймач підраховує спільний таємний ключ $K = A^b \bmod p$, а також свій відкритий ключ $B = g^b \bmod p$ і повертає свій відкритий ключ **B** передавачу.

K виходить рівним з обох сторін, тому що:

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

У практичних реалізаціях, для **a** і **b** використовуються числа порядку 10100 і для **p** близько 10300. Число **g** не зобов'язане бути великим і зазвичай має значення в межах першого десятка.

Приклад:

Єва - Розкодувальник. Вона читає пересилку Боба і Аліси, але не змінює вмісту їх повідомлень.

- s = таємний ключ. s = 2
- g = відкрите просте число. g = 5
- p = відкрите просте число. p = 23
- a = таємний ключ Аліси. a = 6
- A = відкритий ключ Аліси. $A = g^a \bmod p = 8$
- b = таємний ключ Боба. b = 15
- B = відкритий ключ Боба. $B = g^b \bmod p = 19$

Аліса	
знає	не знає
p = 23	b = ?
g = 5	
a = 6	
$A = 5^6 \bmod 23 = 8$	
$B = 5^b \bmod 23 = 19$	
$s = 19^6 \bmod 23 = 2$	
$s = 8^b \bmod 23 = 2$	
$s = 19^6 \bmod 23 = 8^b \bmod 23$	
s = 2	

Боб	
знає	не знає
p = 23	a = ?
g = 5	
b = 15	
$B = 5^{15} \bmod 23 = 19$	
$A = 5^a \bmod 23 = 8$	
$s = 8^{15} \bmod 23 = 2$	
$s = 19^a \bmod 23 = 2$	
$s = 8^{15} \bmod 23 = 19^a \bmod 23$	
s = 2	

Єва	
знає	не знає
p = 23	a = ?
g = 5	b = ?
	s = ?
$A = 5^a \bmod 23 = 8$	
$B = 5^b \bmod 23 = 19$	
$s = 19^a \bmod 23$	
$s = 8^b \bmod 23$	
$s = 19^a \bmod 23 = 8^b \bmod 23$	

Цифровий підпис (RSA).

Система RSA може використовуватися не тільки для шифрування, але і для цифрового підпису.

Припустимо, що Алісі (відправнику) потрібно відправити Бобу (приймачу) повідомлення, підтверджене електронним цифровим підписом.

Алгоритм відправки підпису:

- Взяти відкритий текст m
- Створити цифровий підпис s за допомогою свого таємного ключа

$$s = S_A(m) = m^d \mod n$$

- Передати пару $\{m, s\}$, що складається з повідомлення і підпису.

Алгоритм отримання підпису:

- Прийняти пару $\{m, s\}$
- Взяти відкритий ключ Аліси

$$\{e, n\}$$

Підрахувати прообраз повідомлення з підпису

$$m' = P_A(s) = s^e \mod n$$

- Перевірити справжність підпису (і незмінність повідомлення), порівнявши m **та** m'

Оскільки цифровий підпис забезпечує як аутентифікацію автора повідомлення, так і підтвердження цілісності вмісту підписаного повідомлення, вона служить аналогом підпису, зробленого від руки в кінці рукописного документа.

Важлива властивість цифрового підпису полягає в тому, що її може перевірити кожен, хто має доступ до відкритого ключа її автора. Один з учасників обміну повідомленнями після перевірки справжності цифрового підпису може передати підписане повідомлення ще комусь, хто теж в змозі перевірити цей підпис. Наприклад, одна сторона може переслати іншій стороні електронний чек. Після того як сторона, що отримала повідомлення, перевірить підпис на чеку, вона може передати його у свій банк, службовці якого також мають можливість перевірити підпис, і здійснити відповідну грошову операцію.

Підписане повідомлення не зашифровано. Воно пересилається в початковому вигляді і його вміст не захищений від порушення конфіденційності. Шляхом спільного застосування представлених вище схем кодування і цифрового підпису в системі RSA можна створювати повідомлення, які будуть і закодовані і містити цифровий підпис. Для цього автор спочатку повинен додати до повідомлення свій цифровий підпис, а потім закодувати пару (що складається з самого повідомлення і підпису до нього) за допомогою відкритого ключа, що належить отримувачу. Отримувач розкодовує отримане повідомлення за допомогою свого таємного ключа.

Якщо проводити аналогію з пересилкою звичайних паперових документів, то цей процес схожий на те, якби автор документа поставив під ним свій підпис, а потім поклав його в паперовий конверт і заклеїв, з тим щоб конверт був розклеєний лише тією людиною, кому адресоване повідомлення .