МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ ЦИФРОВИХ ТЕХНОЛОГІЙ, ДИЗАЙНУ ТА ТРАНСПОРТУ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ТА ДИЗАЙНУ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Технології захисту інформації»
для студентів усіх форм навчання
(частина 1)

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 122 КОМП'ЮТЕРНІ НАУКИ
Освітня програма – КОМП'ЮТЕРНИЙ ДИЗАЙН

Одеса – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ПОЛІТЕХНІКА»
ІНСТИТУТ ЦИФРОВИХ ТЕХНОЛОГІЙ, ДИЗАЙНУ ТА ТРАНСПОРТУ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЄКТУВАННЯ ТА ДИЗАЙНУ

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни «Технології захисту інформації»
для студентів усіх форм навчання
(частина 1)

Перший (бакалаврський) рівень вищої освіти
Спеціальність – 122 КОМП'ЮТЕРНІ НАУКИ
Освітня програма – КОМП'ЮТЕРНИЙ ДИЗАЙН

Затверджено на засіданні
кафедри інформаційних технологій
проєктування та дизайну
Протокол № 3 від 29.10.2024

Одеса – 2024

Конспект лекцій з дисципліни «Технології захисту інформації» для студентів першого (бакалаврського) рівня за спеціальністю 122 Комп’ютерні науки, за освітньо-професійною програмою «Комп’ютерний дизайн» (частина 1) / Укл.: А. С. Коляда, О. С. Лопаков, В. В. Космачевський – Одеса : Національний університет «Одеська політехніка», 2024. – 72 с.

Укладачі:

Коляда А. С., к.т.н., доц. кафедри
Лопаков О. С., ст. викладач
Космачевський В. В., ст. викладач

Даний конспект лекцій розглядає криптографічний захист інформаційних систем від викривлення та несанкціонованого використання даних. Інформація є цінним ресурсом, тому існують ризики зловмисних дій щодо систем, які її обробляють і передають. Проблема захисту інформації має давню історію, починаючи з військової та дипломатичної сфери, і сьогодні актуальна для бізнесу та приватних осіб. Розвиток комп’ютерних технологій суттєво змінив методи захисту. Криптографія стала важливою складовою мережевих технологій, забезпечуючи конфіденційність даних та контроль їхньої справжності. Зростання безпаперового документообігу, електронних платежів та цифрових архівів вимагає надійних механізмів захисту, адже відсутність фізичних підписів і печаток створює нові загрози. Водночас розвиток комп’ютерних технологій сприяє появі нових методів атак. Таким чином, для ефективного використання інформаційних систем необхідно розробляти та застосовувати сучасні засоби забезпечення конфіденційності та цілісності даних.

ЗМІСТ

ВСТУП

Даний курс лекцій присвячений проблемам криптографічного захисту інформаційних систем (ІС) від навмисних дій по викривленню та несанкціонованому використанню інформації, яка зберігається в них. Оскільки інформація може являти собою певну цінність, можливі різноманітні зловмисні дії по відношенню до систем, що зберігають, обробляють або передають таку інформацію.

Проблема захисту інформації має давню історію. Вона виникла з потреб таємної передачі, спочатку військових і дипломатичних повідомлень. В даний час вона актуальна в багатьох областях діяльності, в тому числі і для комерційних організацій і приватних осіб. Особливу актуальність проблема захисту інформації набула в інформаційних системах. Широке застосування комп'ютерів і комп'ютерних комунікацій радикально змінило характер і діапазон проблем захисту інформації.

Криптографія є в даний час невід'ємною частиною мережевих технологій. При використанні комп'ютерних мереж, по яким передаються великі обсяги інформації державного, комерційного, військового і приватного характеру, необхідно не допустити можливість доступу до цієї інформації сторонніх осіб.

Крім того, необхідно забезпечити контроль справжності інформації, що зберігається в електронному вигляді. Широке впровадження безпаперового документообігу також вимагає додаткових засобів захисту в силу відсутності на електронних документах підписів і печаток. Постає також питання про захист права на приватне життя, якщо в цьому житті використовуються електронна пошта, електронне зберігання особистих архівів. Багато хто користується такими криптографічними засобами, як шифрування електронної пошти, банківські картки та інше. Дедалі більшого поширення набувають безготівкові розрахунки з використанням телекомунікаційних мереж, електронні гроші. У той же час поява нових потужних комп'ютерів, технологій мережевих і нейронних обчислень зробило можливим дискредитацію серйозних систем захисту.

Таким чином, як при розробці ІС, так і при роботі з ними досить важливо вміти створювати і застосовувати ефективні засоби для реалізації всіх необхідних функцій, пов'язаних із забезпеченням конфіденційності та цілісності інформації.

Лекція №1 ОСНОВИ БЕЗПЕКИ ІНФОРМАЦІЙНИХ СИСТЕМ

1.1 Основні поняття і визначення інформаційної безпеки

Інформація, яка потребує захисту, оголошується *захищуваної, приватної, конфіденційної, таємної*. Для найбільш типових, часто зустрічаються ситуації такого типу введені навіть спеціальні поняття: державна таємниця; військова таємниця; комерційна таємниця; юридична таємниця; лікарська таємниця і т. д.

Далі ми будемо говорити про інформацію, яка захищається, маючи на увазі наступні ознаки такої інформації:

- є якесь певне коло *законних користувачів*, які мають право володіти цією інформацією. Вони здійснюють *санкціонований доступ* до інформації;
- є *незаконні користувачі*, які прагнуть оволодіти цією інформацією з тим, щоб звернути її собі на благо, а законним користувачам на шкоду. Вони здійснюють *несанкціонований доступ* до інформації;

Конфіденційність даних - це статус, наданий даним і визначає ступінь їх захисту.

Розглянемо основні визначення інформаційної безпеки комп'ютерних систем.

Під ***безпекою інформаційних систем*** розуміють її захищеність від випадкового або навмисного втручання в нормальний процес її функціонування, а також від спроб крадіжки, зміни або руйнування її компонентів. Природа впливів на ІС може бути найрізноманітнішою. Це і стихійні лиха (землетрус, ураган, пожежа), і вихід з ладу складових елементів ІС, і помилки персоналу, і спроба проникнення зловмисника.

Безпека ІС досягається прийняттям заходів щодо забезпечення конфіденційності і цілісності оброблюваної нею інформації, а також доступності та цілісності компонентів і ресурсів системи.

Під ***доступом до інформації*** розуміється ознайомлення з інформацією, її обробка, зокрема копіювання, модифікація або знищення інформації.

Розрізняють санкціонований і несанкціонований доступ до інформації.

Санкціонований доступ до інформації - це доступ до інформації, що не порушує встановлені правила розмежування доступу.

Правила розмежування доступу служать для регламентації права доступу суб'єктів доступу до об'єктів доступу.

Несанкціонований доступ до інформації характеризується порушенням встановлених правил розмежування доступу. Особа або процес, які здійснюють несанкціонований доступ до інформації, є порушниками правил розмежування доступу. Несанкціонований доступ є найбільш поширеним видом комп'ютерних порушень.

Конфіденційність даних - це статус, наданий даним і визначає необхідний ступінь їх захисту. По суті конфіденційність інформації - це властивість інформації бути відомою лише допущеним і які пройшли перевірку (авторизованим) суб'єктам системи (користувачам, процесам, програмам). Для інших суб'єктів системи ця інформація повинна бути невідомою.

Суб'єкт - це активний компонент системи (користувач, програма), який може стати причиною потоку інформації від об'єкта до суб'єкту або зміни стану системи.

Об'єкт - пасивний компонент системи (диск, канал зв'язку), який зберігає, приймає або передає інформацію. Доступ до об'єкту означає доступ, який міститься в цій інформації.

Цілісність інформації забезпечується в тому випадку, якщо дані в системі не відрізняються в семантичному відношенні від даних у вихідних документах, тобто якщо не відбулося їх випадкового або навмисного викривлення або руйнування.

Цілісність компонента або ресурсу системи - це властивість компонента або ресурсу бути незмінними в семантичному сенсі при функціонуванні системи в умовах випадкових або навмисних спотворень або руйнівних впливів.

Доступність компонента або ***ресурсу*** системи - це властивість компонента або ресурсу бути доступним для авторизованих законних суб'єктів системи.

Під ***загрозою безпеки*** ІС розуміються можливі дії на ІС, які прямо або побічно можуть завдати шкоди її безпеці. *Збиток безпеки* має на увазі порушення стану захищеності інформації, яка міститься і оброблюється в ІС. З поняттям загрози безпеки тісно пов'язане поняття уразливості ІС.

Уразливість ІС - це деяка невдала властивість системи, яка робить можливим виникнення і реалізації загрози.

Атака на комп'ютерну систему - це дії, що робляться зловмисником, яке полягає в пошуку і користуванні тієї чи іншої уразливості системи. Таким чином, атака - це реалізація загрози безпеки.

Протидія загрозам безпеки є метою захисту систем обробки інформації.

Безпечна чи ***захищена система*** - це система із засобами захисту, які успішно і ефективно протистоять загрозам безпеки.

Надійна система визначається як «система, яка використовує достатні апаратні і програмні засоби для забезпечення одночасної обробки інформації різного ступеня секретності групою користувачів без порушення прав доступу». Надійність системи оцінюється за двома основними критеріями: політика безпеки і гарантованість.

Комплекс засобів захисту представляє собою сукупність програмних і технічних засобів, які створюються і підтримуються для забезпечення інформаційної безпеки ІС. Він створюється і підтримується відповідно з певною політикою безпеки.

1.1.1 Політика безпеки

Політика безпеки - набір законів, правил і норм, що визначають дисципліну обробки, захисту та поширення інформації. Визначає вибір конкретних механізмів, що забезпечують безпеку системи, і є *активним* компонентом захисту, що включає в себе аналіз можливих загроз і вибір заходів протидії. Вона повинна включати в себе, принаймні, такі елементи:

Довільне керування доступом. Полягає в обмеженні доступу до об'єктів на основі врахування персональних характеристик суб'єкта або групи, в яку суб'єкт входить. Довільність полягає в тому, що власник об'єкта на свій розсуд може дозволяти, забороняти або обмежувати доступ інших суб'єктів до даного об'єкта.

Поточний стан прав доступу при довільному управлінні описується матрицею, в рядках якої перераховані суб'єкти, а в стовпчиках - об'єкти. На перетині рядків і стовпців знаходяться ідентифікатори способів доступу, допустимі для суб'єкта по відношенню до об'єкта, наприклад читання, запис, виконання, можливість передачі прав іншим суб'єктам і т.п. Крім матриці доступу використовується більш компактне представлення, засноване на структуруванні сукупності суб'єктів. Застосовуються також списки управління доступом - уявлення матриці по стовпцях, коли для кожного об'єкта перераховуються суб'єкти разом з їх правами доступу.

Безпека повторного використання. Дана міра дозволяє захиститися від випадкового або навмисного вилучення секретної інформації з «сміття». Безпека повторного використання повинна гарантуватися для областей оперативної пам'яті (зокрема, для буферів з образами екрана, паролями, ключами і т.п.) і для різних носіїв інформації.

Сучасні периферійні пристрої ускладнюють забезпечення безпеки повторного використання. Наприклад, принтер може буферизувати кілька сторінок документа, які залишаться в пам'яті навіть після закінчення друку. Необхідно вжити спеціальних заходів, щоб «очистити» пам'ять пристрою.

Мітки безпеки. Примусове управління доступом реалізується за допомогою міток безпеки, асоційованих з суб'єктами і об'єктами. Мітка суб'єкта описує його благонадійність, мітка об'єкта - ступінь закритості, яка міститься в ньому інформації.

Мітки складаються з двох частин - рівня секретності і списку категорій. Найбільш часто використовуваний набір рівнів секретності складається з наступних елементів: «цілком таємно», «таємно», «конфіденційно», «нетаємно». Для систем різного призначення набір рівнів секретності може бути різним. Призначення категорій - опис предметної області, до якої відносяться дані. Механізм категорій дозволяє розділити інформацію, що сприяє підвищенню безпеки системи. Так, суб'єкт не зможе отримати доступ до «чужим» категоріям, навіть якщо він абсолютно благонадійний.

Примусове управління доступом. Примусове управління доступом засновано на зіставленні міток безпеки суб'єкта та об'єкта.

Суб'єкт може читати інформацію об'єкта, якщо його рівень секретності не нижче, ніж у об'єкту, а всі категорії, перераховані в мітці безпеки об'єкта, вказані в мітці суб'єкта.

Суб'єкт може записувати інформацію в об'єкт, якщо рівень безпеки об'єкта не нижче, ніж у суб'єкта, а всі категорії, перераховані в мітці безпеки суб'єкта, вказані в мітці об'єкта. Так, зокрема, суб'єкт, що відноситься до розряду «конфіденційно», може писати в секретні файли, але не може - в несекретні (при дотриманні обмежень на набір категорій). Рівень секретності інформації не повинен знижуватися, хоча зворотній процес можливий.

Описаний спосіб управління називається примусовим, тому що не залежить від волевиявлення суб'єктів. Після того як мітки безпеки суб'єктів і об'єктів зафіксовані, виявляються зафіксованими і права доступу.

1.1.2 Гарантованість

Гарантованість - рівень довіри, яке може бути надано конкретної реалізації системи. Гарантованість відображає ступінь коректності механізмів безпеки. Гарантованість можна вважати *пасивним* компонентом захисту, що наглядає за механізмами забезпечення безпеки.

Гарантованість - це міра впевненості в тому, що обраний набір засобів дозволяє реалізувати сформульовану політику безпеки. Розрізняють два види гарантованості - операційна і технологічна.

Операційна гарантованість. Даний вид гарантованості включає аналіз:

- архітектури і цілісності системи;
- прихованих каналів витоку інформації;
- методів адміністрування;
- технології відновлення після збоїв.

Архітектура системи повинна розроблятися з урахуванням сформульованих заходів безпеки або допускати принципову можливість їх вбудовування. Необхідно передбачити кошти для контролю цілісності програмного і апаратного забезпечення.

Аналіз прихованих каналів витоку інформації особливо важливий в тих випадках, коли необхідно забезпечити конфіденційність інформації. Прихованим називається канал, що не призначений для передачі інформації при традиційному використанні. Наприклад, для передачі інформації по таємному каналу зловмисник може змінити ім'я або розмір відкритого (доступного для читання) файлу. Інший спосіб полягає в зміні тимчасових характеристик різних процесів.

Надійне відновлення включає в себе підготовку до збою (відмови) і власне відновлення. Підготовка до збою - це, перш за все регулярне виконання резервного копіювання, а також вироблення планів дій в екстрених випадках. Відновлення, як правило, пов'язане з перезавантаженням системи і виконанням різних технологічних і адміністративних процедур.

У період відновлення система не повинна залишатися незахищеною; не можна допускати станів, коли захисні механізми повністю або частково відключені.

Технологічна гарантованість. Цей вид гарантованості поширюється на весь життєвий цикл системи - проектування, реалізацію, тестування, передачу замовнику і супровід. Методи контролю спрямовані на недопущення витоку інформації і впровадження нелегальних «закладок».

Основний метод полягає в тестуванні реалізованих механізмів безпеки і їх інтерфейсу з метою докази адекватності захисних механізмів, неможливості їх обходу або руйнування, демонстрації дієвості засобів управління доступом, захищеності реєстраційної та автентифікаційної інформації.

Інший метод - верифікація опису архітектури. Мета верифікації - доказ того, що архітектура системи відповідає сформульованій політиці безпеки.

Здійснюється також комплекс заходів щодо захисту і перевірці версії, яка поставляється, системи, починаючи від перевірок серійних номерів апаратних компонентів і закінчуючи верифікацією контрольних сум програм і даних.

Крім того: Концепція надійної обчислювальної бази є центральною при оцінці ступеня гарантованості надійної системи. Надійна обчислювальна база являє собою сукупність захисних механізмів (включаючи програмне і апаратне забезпечення), що гарантують безпеку системи. Компоненти поза обчислювальної бази можуть бути ненадійними (наприклад, канали зв'язку), однак це не повинно впливати на безпеку системи в цілому.

Механізм протоколювання також є важливим засобом забезпечення безпеки. Ведення протоколів повинно доповнюватися аудитом, тобто аналізом реєстраційної інформації.

1.1.3 Загрози Безпеки ІС

Як загрози можна розглядати конкретну фізичну особу або подію, що представляє небезпеку для ресурсів і призводить до порушення їх конфіденційності, цілісності, доступності та законного використання. Атака є реалізацією тієї чи іншої загрози або їх комбінації. Контрзаходи зводяться до набору превентивних дій по захисту ресурсів від можливих загроз.

За ***метою впливу*** розрізняють **три основних типи загроз** безпеки ІС:

1) загрози порушення ***конфіденційності*** інформації, спрямовані на розголошення конфіденційної або секретної інформації. У термінах комп'ютерної безпеки загроза порушення конфіденційності має місце щоразу, коли отримано несанкціонований доступ до деякої закритої інформації, що зберігається в комп'ютерній системі чи переданої від однієї системи до іншої.

2) загрози порушення ***цілісності*** інформації, що зберігається в комп'ютерній системі чи переданої по каналу зв'язку, спрямовані на її зміну або викривлення, що приводить до порушення її якості або повного знищення. Ця загроза особливо актуальна для систем передачі інформації - комп'ютерних мереж і систем телекомунікацій.

3) загрози порушення ***працездатності*** системи (відмови в обслуговуванні) спрямовані на створення таких ситуацій, коли певні навмисні дії або знижують працездатність ІС, або блокують доступ до деяких її ресурсів.

Загрози підрозділяються на ***навмисні*** (наприклад, атака з боку хакера) і ***випадкові*** (наприклад, посилка за неправильною адресою в результаті збоїв під час передачі повідомлення). Навмисні загрози поділяються на пасивні і активні.

Пасивні загрози випливають з прослуховування (несанкціонованого зчитування інформації) і не пов'язані з будь-якою зміною інформації.

Активні загрози є наслідком спроб перехоплення і зміни інформації. Як правило, реалізація пасивних загроз вимагає менших витрат, ніж реалізація погроз активних. Розглянемо типові загрози, характерні для сучасних розподілених систем. При класифікації загроз виділяють фундаментальні загрози, первинні ініціюючі загрози і базові загрози.

До ***фундаментальних загроз*** відносяться наступні.

- *Витік інформації. Розкриття інформації неавторизованому користувачеві або процесу.*
- *Порушення цілісності.* Компрометація узгодженості (несуперечності) даних шляхом цілеспрямованого створення, підміни і руйнування даних.
- *Відмова в послузі.* Навмисне блокування легального доступу до інформації або іншим ресурсам (наприклад, за допомогою перевантаження сервера потоком запитів).
- *Незаконне використання.* Використання ресурсів незаконним чином. Використання ресурсів неавторизованим об'єктом або суб'єктом. Наприклад, використання віддаленого комп'ютера з метою «злому» інших комп'ютерів мережі.

Реалізація фундаментальних загроз багато в чому залежить від реалізації первинних загроз. Первинні загрози ініціюють фундаментальні загрози. Первинні загрози поділяються на загрози проникнення і загрози впровадження.

До ***загроз проникнення*** відносяться наступні:

- *Маскарад*. Користувач (або інша сутність - процес, підсистема і т.п.) маскується і намагається видати себе за іншого користувача. Дана загроза, як правило, пов'язана зі спробами проникнення всередину периметра безпеки і часто реалізується хакерами.

- *Обхід захисту*. Використання слабких місць системи безпеки для обходу захисних механізмів з метою отримання законних прав та привілеїв.

- *Порушення повноважень*. Використання ресурсів не за призначенням. Дана загроза пов'язана з діями внутрішнього порушника.

До ***загроз впровадження*** відносяться наступні:

- *Троянські програми*. Програми, які містять прихований або явний програмний код, при виконанні якого порушується функціонування системи безпеки. Приклад троянської програми - текстовий редактор, який крім звичайних функцій редагування виконує таємне копіювання редагованого документа в файл зловмисника.

- *Потаємні ходи*. Деякі додаткові можливості, таємно вбудовуються в систему або її компоненти, що порушують функціонування системи безпеки при введенні специфічних даних. Наприклад, підсистема login може опускати запит і перевірку пароля при введенні певного імені користувача.

Подібні загрози, як правило, реалізуються за допомогою спеціальних агентів впровадження, що активізуються після деякого періоду латентності.

Розглядаючи фундаментальні загрози, слід враховувати також загрози базові. Наприклад, витік інформації пов'язаний з такими базовими погрозами, як:

- підслуховування;
- аналіз трафіку;
- персональна необережність;
- «копання в смітті».

Рис. 1.1. Взаємозв'язок різних видів загроз

Взаємозв'язок різних загроз представлений на рис. 1.1. Відзначимо, що цей взаємозв'язок може бути досить складним. Так, маскарад є загрозою, який ініціює фундаментальні загрози, в тому числі витік інформації.

Однак маскарад сам по собі також може залежати від витоку інформації. Наприклад, розкриття пароля може ініціювати загрозу маскараду.

Аналіз понад три тисячі комп'ютерних злочинів показав, що найчастіше виникають такі загрози (в порядку убування):

1) порушення повноважень;

2) маскарад;

3) обхід захисту;

4) троянські програми або потаємні ходи;

5) «копан явсміт і».

Існують і інші загрози для інформації, що захищаються з боку незаконних користувачів: підміна, імітація, відмова від повідомлення, зміна статусу доступу, руйнування системи і ін.

1.1.4 Послуги безпеки

Розглянемо послуги безпеки, характерні для розподілених систем. Питання реалізації цих послуг розглядаються в наступному параграфі.

Аутентифікація. Розрізняють аутентифікацію партнерів по взаємодії і аутентифікацію джерела даних (повідомлень).

- Аутентифікація партнерів по взаємодії використовується при встановленні з'єднання або виконується періодично під час сеансу зв'язку і служить для запобігання таких загроз, як маскарад і несанкціоноване відтворення даних (повідомлень) попереднього сеансу зв'язку.

- Аутентифікація джерела полягає в підтвердженні справжності джерела окремих повідомлень. Відзначимо, що даний вид аутентифікації не забезпечує захисту від несанкціонованого відтворення повідомлень попереднього сеансу.

Управління доступом. Управління доступом забезпечує захист від несанкціонованого використання ресурсів мережі.

Конфіденційність даних. Конфіденційність забезпечує захист від несанкціонованого отримання інформації. Розрізняють такі види конфіденційності:

- конфіденційність при взаємодії з встановленням з'єднання (в цьому і наступному випадку захищається вся інформація користувача);
- конфіденційність при взаємодії без встановлення з'єднання;
- конфіденційність окремих полів повідомлення (виборча конфіденційність);
- конфіденційність трафіку (протидія різним методам розкриття, заснованим на аналізі потоків повідомлень).

Цілісність даних. Дана послуга підрозділяється на підвиди залежно від того, який тип взаємодії використовується - з встановленням з'єднання або без, захищається чи повідомлення цілком або тільки окремі поля, чи забезпечується відновлення в разі порушення цілісності.

Належність. Дана послуга (доказ приналежності в разі відмови від раніше переданого / прийнятого повідомлення) забезпечує:

- доказ приналежності з підтвердженням справжності джерела повідомлень;
- доказ приналежності з підтвердженням доставки.

1.1.5 Механізми реалізації послуг безпеки

Для реалізації послуг безпеки можуть використовуватися такі механізми і їх комбінації.

Шифрування. Шифрування підрозділяється на симетричне (один і той же секретний ключ для шифрування і дешифрування) і асиметричне (різні ключі для шифрування і дешифрування).

Електронний цифровий підпис. Механізм електронного підпису включає в себе дві процедури:

- вироблення (обчислення) підпису;
- перевірку підписаних повідомлень.

Процедура вироблення підпису використовує інформацію, відому тільки тому, хто підписує. Процедура перевірки підпису є загальнодоступною, при цьому, однак, вона не дозволяє розкривати секретну інформацію, того хто підписує.

Механізми управління доступом. В ході прийняття рішення про надання запитуваного доступу можуть використовуватися такі види і джерела інформації:

- бази даних управління доступом. У такій базі, підтримуваної централізовано або на кінцевих системах, можуть зберігатися списки управління доступом або структури аналогічного призначення;

- паролі або інша аутентифікаційна інформація;
- різні посвідчення, пред'явлення яких свідчить про наявність прав доступу;
- мітки безпеки, асоційовані з суб'єктами і об'єктами доступу;
- час запитуваного доступу;
- маршрут запитуваного доступу;
- тривалість запитуваного доступу.

Механізми управління доступом можуть перебувати у будь-яких з взаємодіючих сторін або в проміжній точці. У проміжних точках доцільно перевіряти права доступу до

комунікаційних ресурсів. Вимоги механізму, розташованого на приймальному кінці, повинні бути відомі заздалегідь, до початку взаємодії.

Механізми контролю цілісності. Розрізняють два аспекти цілісності: цілісність повідомлення або окремих його полів і цілісність потоку повідомлень.

Процедура контролю *цілісності окремого повідомлення* (або поля) включає в себе два процеси - один на передавальній стороні, інший - на приймальній. На передавальній стороні до повідомлення додається надмірність (той або інший різновид контрольної суми), яка є функцією повідомлення. Отримане приймальною стороною повідомлення також використовується для обчислення контрольної суми. Рішення приймається за результатами порівняння прийнятих і обчислених контрольних сум. Відзначимо, що даний механізм не захищає від несанкціонованого відтворення (наприклад, дублювання) повідомлень.

Для перевірки *цілісності потоку повідомлень* (тобто для захисту від вилучення, переупорядкування і вставки повідомлень) використовуються порядкові 30номера, тимчасові мітки, криптографічні методи (у вигляді різних режимів шифрування) або інші аналогічні прийоми.

При взаємодії без встановлення з'єднання використання тимчасових команд можуть забезпечити частковий захист від несанкціонованого відтворення повідомлень.

Механізми аутентифікації. Аутентифікація може досягатися за рахунок використання паролів, персональних карток або інших пристроїв аналогічного призначення, криптографічних методів, пристроїв вимірювання і аналізу біометричної інформації.

Аутентифікація буває односторонньою (наприклад, клієнт доводить свою справжність серверу) і двосторонньою (взаємною). Приклад односторонньої аутентифікації - вхід користувача в систему.

Для захисту від несанкціонованого відтворення автентифікаційної інформації можуть використовуватися тимчасові мітки і система єдиного часу, а також різні методи на основі хеш-функцій.

Механізми доповнення («набивання») трафіку. Механізми «набивання» трафіку ефективні тільки в поєднанні з заходами щодо забезпечення конфіденційності; в іншому випадку зловмисник зможе виділити корисні повідомлення із загального потоку, що містить шумове «набивання».

Механізми нотарізації. Механізм нотарізації служить для засвідчення справжності. Засвідчення забезпечується надійною третьою стороною, яка володіє достатньою інформацією, для того щоб її запевненням можна було довіряти. Як правило, нотарізація спирається на механізм електронного цифрового підпису.

1.1.6 Адміністрування

Адміністрування включає в себе управління інформацією, необхідною для реалізації послуг безпеки та їх механізмів, а також збір і аналіз інформації про їх функціонування. Прикладами можуть служити поширення криптографічних ключів, установка значень параметрів захисту, ведення реєстраційного журналу і т.п.

Як правило, всі об'єкти адміністрування зводяться в єдину інформаційну базу управління безпекою. База може не існувати як єдине розподілене сховище, проте кожна з кінцевих систем повинна мати у своєму розпорядженні інформацію, необхідну для реалізації функцій адміністрування.

Основні завдання адміністратора коштів безпеки зводяться до адміністрування:

- системи в цілому;
- послуг безпеки;
- механізмів реалізації послуг безпеки.

Адміністрування системи в цілому полягає в проведенні адекватної політики

безпеки, у взаємодії з іншими службами, в реагуванні на події, що відбуваються, аудит і безпечному і надійному відновленні.

Адміністрування послуг безпеки включає в себе визначення об'єктів, що захищаються, вибір і комбінування механізмів реалізації послуг безпеки, взаємодія з іншими адміністраторами для забезпечення узгодженої роботи.

Адміністрування механізмів реалізації послуг безпеки полягає у виборі і своєчасної зміни параметрів в разі виникнення загрози.

Обов'язки адміністратора визначаються переліком задіяних механізмів реалізації послуг безпеки. Як правило, адміністрування включає наступні напрямки:

1) управління ключами (генерація і розподіл). До даного виду управління можна віднести і адміністрування механізмів електронного цифрового підпису, а також управління цілісністю, якщо цілісність забезпечується криптографічними засобами;

2) адміністрування управління доступом (розподіл інформації, необхідної для управління, - паролів, списків доступу);

3) адміністрування аутентифікації (розподіл інформації, необхідної для аутентифікації, - паролів, ключів і т.п.);

4) управління «набиванням» трафіку - вироблення правил, які задають характеристики «шумових» повідомлень. Характеристики можуть варіюватися в залежності від дати і часу доби;

5) управління нотарізацією (поширення інформації про нотаріальні служби і їх адміністрування).

1.1.7 Протоколювання і аудит

Протоколювання і аудит забезпечують реєстрацію наслідків різних порушень і виявлення зловмисників шляхом аналізу накопиченої реєстраційної інформації.

Розумний підхід полягає в спільному аналізі реєстраційних журналів окремих складових системи з метою перевірки повноти і несуперечності подій, які в ній відбулися.

Протоколювання допомагає відслідковувати дії користувачів і реконструювати минулі події. Реконструкція подій дозволяє проаналізувати випадки порушень, оцінити розміри збитку і вжити відповідних заходів. При протоколюванні подій повинна реєструватися, принаймні, наступна інформація:

- дата і час події;
- ідентифікатор користувача або процесу - ініціатора події;
- тип події;
- результат події;
- джерело запиту події;
- імена порушених об'єктів (наприклад, файлів, що відкриваються і видаляються);
- опис змін, що стосуються механізмів захисту (наприклад, поява нової мітки безпеки);
- мітки безпеки суб'єктів і об'єктів події.

Аудит має справу з подіями, які зачіпають безпеку системи, і являє собою аналіз накопиченої інформації з метою виявлення спроб порушення інформаційної безпеки. До числа таких подій відносяться:

- вхід в систему і вихід з неї;
- звернення до віддаленої системи;
- операції з файлами (відкрити, закрити, перейменувати, видалити);
- зміна привілеїв чи інших атрибутів безпеки (режиму доступу, рівня благонадійності і т.п.).

Повний перелік подій, що підлягають аналізу, залежить від обраної політики безпеки.

Активний аудит - відстеження підозрілих дій в реальному масштабі часу. Активний аудит включає:

- виявлення нетипової активності суб'єктів (користувачів, програм і т.п.);
- виявлення початку злочинних дій.

Для виявлення нетипової активності і початку злочинних дій використовується метод зіставлення з попередньо отриманими зразками тих чи інших системних подій, а також сигнатурами відомих атак.

1.2 Основні поняття і історія криптографії

Криптографічні методи в даний час є базовими для забезпечення безпеки сучасних розподілених інформаційних систем (ІС). Криптографія є складовою частиною науки *криптології* (*kryptos* - таємний, *logos* - наука), що займається проблемами захисту інформації з найдавніших часів, і що отримала в останні десятиліття значного розвитку. Криптологія розділяється на два напрямки - *криптографію* і *криптоаналіз*:

Криптологія = криптографія + криптоаналіз

Співвідношення криптографії та криптоаналізу очевидно: криптографія - захист, а криптоаналіз - напад. Завдання *криптографа* - забезпечити конфіденційність (секретність) і автентичність (справжність) переданих повідомлень. Завдання *криптоаналітика* - "зламати" систему захисту, розроблену криптографами. Він намагається розкрити зашифрований текст або видати підроблене повідомлення за сьогодення. Однак ці дві дисципліни пов'язані один з одним, і не буває хороших криптографів, які не володіють методами криптоаналізу.

1.2.1 Історія криптографії

Криптографія забезпечує захист інформації шляхом її перетворення, що виключає її прочитання сторонньою особою. Проблема ця хвилювала людство з давніх часів. Історія криптографії - ровесниця історії людської мови. Більш того, спочатку писемність сама по собі була криптографічною системою, так як в древніх суспільствах нею володіли лише обрані. Священні книги Стародавнього Єгипту, Стародавньої Індії тому приклади. З широким поширенням писемності криптографія стала формуватися як самостійна наука. Перші криптосистеми зустрічаються вже на початку нашої ери.

Протягом більш ніж тисячолітньої історії криптографії вона представляла собою набір, який постійно оновлюється і удосконалюється, технічних прийомів шифрування і розшифрування, які зберігалися в суворій таємниці, оскільки застосовувалися в основному для захисту державних і військових секретів.

Перші канали зв'язку були дуже простими. Їх організовували, використовуючи надійних кур'єрів. Безпека таких систем зв'язку залежала як від надійності кур'єра, так і від його здатності не потрапляти в ситуації, при яких могло мати місце розкриття повідомлення.

Історія криптографії пов'язана з великою кількістю дипломатичних і військових таємниць і тому огорнута туманом легенд. Свій слід в історії криптографії залишили багато відомих історичних особистостей. Наведемо кілька найбільш яскравих прикладів.

Перші відомості про використання шифрів в військовій справі пов'язані з ім'ям спартанського полководця Лісандра (шифр «Сцітала»). Цезар використовував в листуванні шифр, який увійшов в історію як «шифр Цезаря». У стародавній Греції був винайдений вид шифру, який в подальшому став називатися «квадрат Полібія». Одну з перших книг з криптографії написав абат І. Трітелій (1462-1516), що жив в Німеччині. У 1566 році відомий математик Д. Кардано опублікував роботу з описом винайденої їм

системи шифрування («решітка Кардано»). Франція XVI століття залишила в історії криптографії шифри короля Генріха IV і Рішельє. Деякі відомості про властивості шифрів і їх застосуванні можна знайти і в художній літературі, особливо в пригодницькій, детективній і військовій. Гарне докладне пояснення особливостей одного з найпростіших шифрів - шифру заміни і методів його розтину, міститься в двох відомих оповіданнях: «Золотий жук» Е. По і «Танцюючі чоловічки» А. Конан Дойла.

Розглянемо більш докладно два приклади.

Шифр «Сцітала». Цей шифр відомий з часів війни Спарти проти Афін в V столітті до н.е. Для його реалізації використовувалася сцітала - жезл, що має форму циліндра. На сціталу виток до витка намотувалася вузька папірусна стрічка (без просвітів і нахлестів), а потім на цій стрічці вздовж осі сцітали записувався відкритий текст. Стрічка розмотувалася і виходило (для непосвячених), що поперек стрічки в безладді написані якісь літери. Потім стрічка відправлялася адресату. Адресат брав таку ж сціталу, таким же чином намотував на неї отриману стрічку і читав повідомлення уздовж осі сцітали.

Клас шифрів, до яких відноситься і шифр «Сцітала», називається *шифрами перестановки*, оскільки процес шифрування полягає в певній перестановці букв відкритого тексту.

Шифр Цезаря. Більше 2000 років тому Юлій Цезар писав Цицерону в Рим, використовуючи шифр, тепер названий його ім'ям. Цей шифр реалізує наступне перетворення відкритого тексту: кожна буква відкритого тексту замінюється третьою після неї буквою в алфавіті, який вважається написаним по колу, тобто після букви «я» слідує буква «а». Клас шифрів, до яких відноситься і шифр Цезаря, називається *шифрами заміни*.

Донаукова криптологія

Довгий час заняття криптографією було долею диваків-одинаків. Серед них були обдаровані вчені, дипломати, священнослужителі. Відомі випадки, коли криптографія вважалася навіть чорною магією. Криптологією займалися тоді майже виключно як мистецтвом, а не як наукою. Цей період розвитку криптографії як мистецтва тривав з незапам'ятних часів до початку XX століття, коли з'явилися перші шифрувальні машини. Бурхливий розвиток криптографічні системи отримали в роки першої і другої світових війн. Лише з початком другої світової війни криптологічні служби воюючих держав усвідомили, що математики можуть внести вагомий внесок у розвиток криптології. Зокрема, в Англії в цей час був покликаний на службу в якості спеціаліста з криптології Алан Тюрінг.

Період розвитку криптології з давніх часів до 1949 р прийнято називати ерою *донаукової криптології*, оскільки досягнення тих часів були засновані на інтуїції і не підкріплювалися доказами.

Ера наукової криптології

Розуміння математичного характеру розв'язуваних криптографією завдань прийшло тільки в середині XX століття - після робіт видатного американського вченого Клода Шеннона. Публікація в 1949 р статті К. Шеннона "Теорія зв'язку в секретних системах" стала початком нової ери наукової криптології з секретними ключами. У цій блискучій роботі Шеннон пов'язав криптографію з теорією інформації. Поява обчислювальних засобів прискорило розробку та вдосконалення криптографічних методів.

1.2.2 Завдання сучасної криптографії

З середини сімдесятих років в зв'язку з появою таких нових фундаментальних ідей, як асиметрична (з відкритим ключем) криптографія, доказово стійкі протоколи, надійність яких заснована на гарантованої складності рішення математичних задач, і т.п., криптографія не тільки перестала бути секретним зведенням прийомів шифрування-розшифрування, і почала оформлятися в нову математичну теорію.

Сучасна криптографія включає в себе чотири великих розділи:

1. симетричні криптосистеми (класична криптографія);

2. асиметричні алгоритми шифрування;

3. системи електронного підпису;

4. криптографічні протоколи;

Основні напрямки використання криптографічних методів:

- передача конфіденційної інформації з каналів зв'язку (наприклад, електронна пошта),
- зберігання інформації (документів, баз даних) на носіях в зашифрованому вигляді,
- встановлення автентичності (аутентифікація) переданих повідомлень і абонентів системи,
- захист інформації в електронних платіжних системах, де в якості універсального платіжного засобу використовуються банківські пластикові картки.

Багато розробок в області криптографії секретні, і, зокрема, «відкритим» фахівцям невідомо, які є досягнення в «закритій» сфері.

Слід звернути увагу також на те, яке відношення мають криптографічні алгоритми до державних стандартів. Справа в тому, що в зв'язку з поширенням криптографії виникають деякі політичні проблеми. Наприклад, уряд не любить, коли громадяни користуються криптографією в особистих цілях. Ілюстрацією служить прагнення американського уряду стандартизувати алгоритми шифрування, що дозволяє владі отримувати доступ до зашифрованої інформації за рішенням суду.

З політичними проблемами також пов'язані обмеження експорту криптографічних засобів. У США їх прирівнюють до військового спорядження. Для експорту дозволені криптографічні засоби, в яких довжина ключа не перевищує 40 бітів. Як показує практика, ключ такої довжини можна підібрати за кілька годин за допомогою декількох персональних комп'ютерів.

Відзначимо, що історично в криптографії закріпилися деякі військові слова: противник, атака на шифр і інші. Вони найбільш точно відображають зміст відповідних криптографічних понять. Разом з тим широко відома військова термінологія, заснована на понятті коду, вже не застосовується в теоретичній криптографії. Це пов'язано з тим, що поняття коду в математичної теорії інформації закріплено за *теорією кодування*, що займається методами захисту інформації від випадкових викривлень в каналах зв'язку. І якщо раніше терміни «кодування» і «шифрування» вживалися як синоніми, то тепер це неприпустимо.

1.2.3 Роль ключа в криптографії

Процес перетворення інформації з метою зробити її недоступною для несанкціонованого доступу називають *шифруванням*. Щоб не дозволити противнику витягти інформацію з перехоплюваних повідомлень, по каналу зв’язку передається вже не сама інформація, яка захищається, а результат її перетворення.

Під *шифром* в криптографії зазвичай розуміють *алгоритм шифрування*. Якщо секретним є весь алгоритм захисту інформації, то його розголошення негайно призводить до краху всієї системи. Щоб збільшити «час життя» хорошого шифру (алгоритму) і

використовувати його для шифрування як можна більшої кількості повідомлень, в шифрі вводять *ключ* - змінний елемент шифру, який застосовується для шифрування конкретного повідомлення. На його основі шифрований текст перетвориться у вихідний і навпаки. Наприклад, в шифрі «Сцітала» ключем є діаметр сцітали, а в шифрі Цезаря ключем є величина зсуву літер шифртекста щодо букв відкритого тексту.

Якщо противник вже розгадав (розкрив) шифр і читає інформацію, яка захищається, то замінивши ключ, можна зробити так, що розроблені противником методи вже не дають ефекту.

Описані міркування привели до того, що безпека, яка захищається, стала визначатися в першу чергу ключем. Сам шифр, тобто алгоритм шифрування або шифрмашина стали вважати відомими противнику і доступними для попереднього вивчення, але в них з'явився невідомий для супротивника ключ, від якого значно залежать використовувані перетворення інформації.

Тепер законні користувачі можуть обмінюватися шифрованими повідомленнями по загальнодоступному каналу зв'язку. А секретний канал використовується тільки для обміну ключами. Це набагато простіше, оскільки навантаження на нього стає невелика. А для криптоаналітика з'явилося нове завдання - визначити ключ, після чого можна легко прочитати зашифровані на цьому ключі повідомлення.

1.2.4 Узагальнена схема криптосистеми з секретним ключем

Щоб не дозволити противнику витягти інформацію з перехоплюваних повідомлень або через з'єднання передається вже не сама інформація, яка захищається, а результат її перетворення за допомогою шифру, і для супротивника виникає завдання розтину шифру. Розглянемо загальну схему (рис.2.1) захищеної передачі інформації.

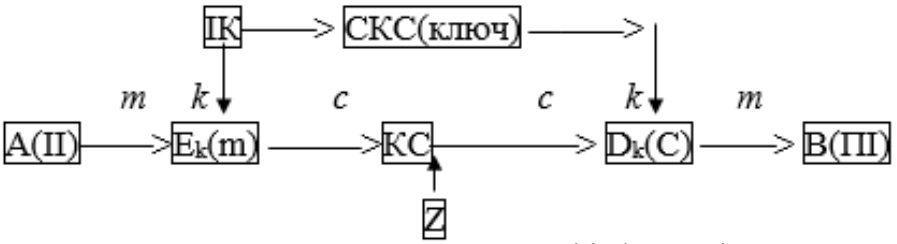


Рис. 2.1. Загальна схема захищеної передачі інформації.

Тут:

- A (*відправник*, джерело інформації (II), в англомовній літературі зазвичай позначається ім'ям Alice) і B (*одержувач*, приймач інформації (III), звичайно позначається ім'ям Bob) - віддалені *законні користувачі* інформації, яка захищається,
- *m* - вихідне повідомлення (*відкритий текст*), який генерує джерело інформації або відправник,
- ІК - джерело секретних ключів,
- *k* - *ключ* (секретний параметр алгоритму шифрування/дешифрування),
- *СКС* - *канал зв'язку для передачі секретного ключа*,
- E_k - блок шифрування, який здійснює перетворення інформації з метою зробити її недоступною для несанкціонованого доступу.
- *c* - шифрограма (шифрований текст, шифртекст),
- КС - загальнодоступний фізичний канал зв'язку,
- D_k - блок дешифрування, процесу, зворотного шифрування,
- Z - зловмисник (противник, *криптоаналітик*), який має доступ до каналу передачі інформації, який може перехоплювати надіслані через з'єднання повідомлення і намагатися витягти з них потрібну йому інформацію.

Місце захищеного каналу зв'язку в загальній схемі передачі інформації. У загальній схемі передачі інформації, розглянутої в теорії інформації, захищений канал зв'язку займає місце фізичного каналу зв'язку. Для забезпечення оптимальності роботи інформаційної системи за всіма критеріями (швидкість передачі, захищеність від випадкових і навмисних спотворень, конфіденційність) необхідно, щоб спочатку виконувалося оптимальне мало надмірне кодування, потім криптографічне шифрування, а потім шифртекст кодується перешкодозахищеним кодом. Це не тільки збільшує якість і надійність передачі інформації, але і ускладнює розкриття шифрів.

Процес криптографічного закриття даних може здійснюватися як програмно, так і апаратно. Апаратна реалізація відрізняється більшою вартістю, однак їй властиві і переваги: висока продуктивність, простота, захищеність і т.п. Програмна реалізація більш практична, допускає відому гнучкість у використанні.

1.2.5 Проблема стійкості криптосистеми

Розтин (злом) шифру - процес отримання інформації, яка захищається, з шифрованого повідомлення без знання застосованого шифру. Спроба розкриття шифру називають *атакою на шифр*.

Здатність шифру протистояти всіляким атакам на нього називають *стійкістю шифру*. Поняття стійкості шифру є центральним для криптографії.

Найважливішим для розвитку криптографії був результат К. Шеннона про існування та єдність абсолютно стійкого шифру. Єдиним таким шифром є якась форма так званої *стрічки одноразового використання*, в якій відкритий текст «об'єднується» з повністю випадковим ключем такої ж довжини. Але абсолютно стійкий шифр виявився дуже дорогим і непрактичним.

Зрозуміло, що найчастіше для захисту своєї інформації законні користувачі змушені застосовувати не абсолютно стійкі шифри. Такі шифри, принаймні теоретично, можуть бути розкриті. Питання тільки в тому, чи вистачить у противника сил, засобів і часу для розробки і реалізації відповідних алгоритмів. Зазвичай цю думку висловлюють так: противник з необмеженими ресурсами може розкрити будь-який не абсолютно стійкий шифр. Протягом багатьох століть серед фахівців не затихали суперечки про стійкість шифрів і про можливість побудови абсолютно стійкого шифру. Батько кібернетики Норберт Вінер говорив: «Будь-який шифр може бути розкритий, якщо тільки в цьому є наполеглива необхідність і інформація, яка підлягає отриманню, стоїть витрачених коштів, зусиль і часу ...»

Як же повинен діяти в цій ситуації законний користувач, вибираючи для себе шифр? Найкраще, звичайно, було б довести, що ніякий противник не може розкрити обраний шифр, скажімо, за 10 років і тим самим отримати теоретичну оцінку стійкості. На жаль, математична теорія ще не дає потрібних теорем - вони відносяться до невирішеної *проблеми нижніх оцінок обчислювальної складності завдань*.

Тому у користувача залишається єдиний шлях-отримання практичних оцінок стійкості. Цей шлях складається з наступних етапів:

- Зрозуміти і чітко сформулювати, від якого супротивника ми збираємося захищати інформацію; необхідно усвідомити, що саме противник знає або зможе дізнатися про систему шифру, а також які сили і засоби він зможе застосувати для її розкриття;

- В думках стати в положення противника і намагатися з його позицій атакувати шифр, тобто розробляти різні алгоритми розтину шифру; при цьому необхідно в максимальній мірі забезпечити моделювання сил, засобів і можливостей противника;

- Найкращий з розроблених алгоритмів використовувати для практичної оцінки стійкості шифру.

Тут корисно для ілюстрації згадати про двох простіших методах розкриття шифру: випадкове вгадування ключа (він спрацьовує з маленькою ймовірністю, зате має маленьку складність) і перебір всіх підряд ключів аж до знаходження істинного (він спрацьовує завжди, зате має дуже велику складність). Відзначимо також, що не завжди потрібна атака на ключ: для деяких шифрів можна відразу, навіть не знаючи ключа, відновлювати відкритий текст по шифрованому.

Отже, стійкість конкретного шифру оцінюється тільки шляхом всіляких спроб його розтину і залежить від кваліфікації *криптоаналітиків*, атакуючих шифр. Таку процедуру іноді називають *перевіркою стійкості*. Є кілька показників криптостійкості, серед яких: кількість всіх можливих ключів; середній час, необхідний для криптоаналіза.

Важливим підготовчим етапом для перевірки стійкості шифру є продумування різних передбачуваних можливостей, за допомогою яких противник може атакувати шифр. Поява таких можливостей у супротивника звичайно не залежить від криптографії, це є деякою зовнішньою підказкою і значно впливає на стійкість шифру. Тому оцінки стійкості шифру завжди містять ті припущення про цілі і можливості противника, в умовах яких ці оцінки отримані.

У загальному випадку всі оцінки стійкості системи повинні вестися в припущенні, що зловмисникові відомий застосовуваний шифр. Тобто зазвичай вважається, що противник знає сам алгоритм шифрування і має можливості для його попереднього вивчення. Противник також знає деякі характеристики відкритих текстів, наприклад, загальну тематику повідомлень, їх стиль, деякі стандарти, формати і т. п.

У зв'язку з вищесказаним розглянемо основні поняття і завдання криптоаналізу.

1.2.6 Основні поняття і завдання криптоаналізу

Криптоаналітична атака - це загроза безпеці інформаційної системи з боку зловмисника. Розрізняють активні і пасивні атаки. Розглянемо, в чому може полягати *мета криптоаналітичної атаки*:

Мета *пасивної* атаки - визначити повідомлення m або, краще, ключ k.

Мета *активної* атаки:

- створити або підмінити повідомлення, переконавши одержувача в тому, що воно відправлено відправником.
- видалити повідомлення таким чином, щоб одержувач цього не помітив.

Види пасивних атак. Розглянемо, які вихідні дані можуть виявитися в руках у зловмисника, і атаки якого виду можна очікувати (в порядку збільшення складності для атакуючого і небезпеки для атакованого):

- атака з *відомою шифрограмою*. Противник може перехоплювати всі шифровані повідомлення c, але не має відповідних їм відкритих текстів. Робота криптоаналітика полягає в тому, щоб розкрити вихідні тексти m, по можливості, більшості повідомлень або обчислити ключ k;
- атака з *відомим повідомленням* (plaintext only). Противник може перехоплювати всі шифровані повідомлення c_i і добувати відповідні їм відкриті тексти m_i. Матеріал для такої атаки можуть дати випадкове перехоплення, старі (розсекречені) повідомлення, відомі фрагменти текстів (заголовки, бібліотеки в виконуваних файлах і т.п.). Робота криптоаналітика полягає в тому, щоб обчислити ключ k;
- атака з *вибором повідомлення* (chosen plaintext), коли противник має доступ до шифру (але не до ключів!) і тому може зашифровувати і розшифровувати будь-яку інформацію щодо вибору.

При цьому він має можливість вибирати для шифрування ті блоки, які дадуть більше інформації про ключі. Така атака можлива, наприклад, за допомогою спільника, який працює всередині атакованої організації, або в деяких спеціальних випадках, таких, як система впізнання «свій-чужий»;

- атака з *вибором шифрограми* (chosen ciphertext), коли зловмисник може отримувати результати розшифровки для побудованих їм шифрограм.
- *адаптивна* атака з вибором повідомлення або шифрограми, що використовує при виборі результати попередніх експериментів.
- *силова атака* або атака методом *повного перебору* всіх можливих ключів. Вона передбачає використання відомої шифрограми і здійснюється шляхом повного перебору всіх можливих ключів з перевіркою, чи є осмисленим вихідний текст. Такий підхід вимагає залучення граничних обчислювальних ресурсів.

Однак крім перехоплення і розтину шифру противник може намагатися отримати інформацію, яка захищається, багатьма іншими способами. Найбільш відомим з таких способів є агентурний, коли противник будь-яким шляхом схиляє до співпраці одного із законних користувачів і за допомогою цього агента отримує доступ до інформації, що захищається. У такій ситуації криптографія безсила.

Для захисту від активних атак (не одержати, а знищити або модифікувати інформацію, яка захищається, в процесі її передачі) розробляються свої специфічні методи. На шляху від одного законного користувача до іншого інформація, як правило, захищається різними способами, що протистоять різним загрозам. Виникає ситуація ланцюга з різнотипних ланок захисту інформації. Звісно, противник буде прагнути знайти найслабшу ланку, щоб з найменшими витратами дістатися до інформації. А значить, і законні користувачі повинні враховувати цю обставину в своїй стратегії захисту: безглуздо робити якусь ланку дуже міцним, якщо є свідомо більш слабкі ланки («принцип рівноміцності захисту»).

Відзначимо тепер, що не існує єдиного шифру, придатного для всіх випадків. Вибір способу шифрування залежить від особливостей інформації, її цінності і можливостей власників по захисту своєї інформації. Перш за все, підкреслимо велике розмаїття видів інформації, що захищається: документальна, телефонна, телевізійна, комп'ютерна і т. п. Кожен вид інформації має свої специфічні особливості. Велике значення мають обсяги і необхідна швидкість передачі шифрованої інформації.

Вибір виду шифру і його параметрів значно залежить від характеру секретів, які захищаються, або таємниці. Деякі таємниці (наприклад, державні, військові та ін.) повинні зберігатися десятиліттями, а деякі (наприклад, біржові) - вже через кілька годин можна розголосити. Необхідно враховувати також і можливості того противника, від якого захищається дана інформація. Одна справа - протистояти одиночці або навіть банді кримінальників, а інша справа - потужній державній структурі.

Не слід забувати і ще про одну важливу проблему: проблему співвідношення *ціни інформації*, витрат на її захист і витрат на її добування. Перш ніж захищати інформацію, задайте собі два питання:

1) чи є вона для противника ціннішою, ніж вартість атаки;

2) чи є вона для вас ціннішою, ніж вартість захисту.

1.2.7 Вимоги до криптосистем

Для сучасних криптографічних систем захисту інформації сформульовані наступні загальноприйняті вимоги:

1. зашифроване повідомлення повинно піддаватися читанню тільки при

наявності ключа;

2. число операцій, необхідних для визначення використаного ключа шифрування по фрагменту шифрованого повідомлення і відповідного йому відкритого тексту, має бути не менше загального числа можливих ключів;

3. число операцій, необхідних для розшифрування інформації шляхом перебору всіляких ключів, повинно мати строгу нижню оцінку і виходити за межі можливостей сучасних комп'ютерів (з урахуванням можливості використання мережевих обчислень);

4. знання алгоритму шифрування не повинно впливати на надійність захисту;

5. незначна зміна ключа повинно приводити до значної зміни виду зашифрованого повідомлення навіть при використанні одного і того ж ключа;

6. структурні елементи алгоритму шифрування повинні бути незмінними;

7. додаткові біти, що вводяться в повідомлення в процесі шифрування, повинні бути повністю та надійно сховані в зашифрованому тексті;

8. не повинно бути простих і легко встановлюваних залежностей між ключами, послідовно використовуваними в процесі шифрування;

9. будь який ключ з множини можливих повинен забезпечувати надійний захист інформації;

10. алгоритм повинен допускати як програмну, так і апаратну реалізацію, при цьому зміна довжини ключа не повинна вести до якісного погіршення алгоритму шифрування.

Лекція №2 КЛАСИЧНІ СИМЕТРИЧНІ КРИПТОСИСТЕМИ

Розглянемо *традиційні* (класичні) *методи* шифрування, що відрізняються симетричною функцією шифрування. Сучасні симетричні одноключові криптоалгоритми базуються на принципах, викладених в роботі Шеннона «Теорія зв'язку в секретних системах» (1949), де він узагальнив накопичений до нього досвід розробки шифрів. Виявилося, що навіть в складних шифрах в якості складових компонентів можна виділити *шифри* простої та складної *заміни*, *шифри перестановки*, а також деякі їх модифікації і комбінації. Слід зазначити, що *комбінації шифрів перестановок* і *шифрів заміни* утворюють все різноманіття застосовуваних на практиці симетричних шифрів.

Розглянемо приклади шифрів кожної групи спочатку в хронологічному порядку, що дозволить зрозуміти суть цих методів на простих і наочних прикладах з історії розвитку цієї науки.

1.1 Шифри перестановки. Шифрувальні таблиці

При шифруванні перестановкою символи шифрованого тексту переставляються за певним правилом в межах блоку цього тексту. Шифри перестановки є найпростішими і, ймовірно, найдавнішими шифрами.

Першим найпростішим криптографічним пристроєм, що реалізує шифр перестановки, є древній шифр "сцітала". Для дешифрування такого шифру потрібно не тільки знати правило шифрування, але і володіти ключем у вигляді стрижня певного діаметру. Ідея цього шифру в наступні часи отримала розвиток в методах шифрування за допомогою спеціальних таблиць.

З початку епохи Відродження (кінець XIV століття) почала відроджуватися і криптографія. У розроблених шифрах перестановки того часу застосовуються шифрувальні таблиці, які, по суті, задають правила перестановки літер в повідомленні. Як ключ в шифруючих таблицях використовуються:

- розмір таблиці;
- слово або фраза, що задають перестановку;
- особливості структури таблиці.

Одним з найбільш примітивних табличних шифрів перестановки є ***проста перестановка***, для якої ключем служить розмір таблиці. Оригінальний текст повідомлення записується в таблицю по стовпцях. Для отримання шифртекста вміст таблиці зчитують по рядкам. При дешифруванні дії виконують у зворотному порядку. Ключем шифру є розмір таблиці - кількість рядків і стовпців. Якщо текст не поміщається в таблицю цілком, його розбивають на блоки довжиною, рівній кількості елементів таблиці, додаючи при необхідності потрібну кількість символів. Оскільки кількість варіантів різних ключів відносно невелике, стійкість такого алгоритму шифрування невисока.

Дещо більшою стійкістю до розкриття володіє метод шифрування, званий ***одиночною перестановкою по ключу***. Цей метод відрізняється від попереднього тим, що стовпчики таблиці переставляються за ключовим словом, фразою або набором чисел довжиною в рядок таблиці.

У верхньому рядку таблиці до перестановки записується ключ, а номери під буквами ключа визначені відповідно з очевидним порядком відповідних букв ключа в алфавіті. У правій таблиці стовпці переставлені відповідно з впорядкованими номерами букв ключа.

Приклад. Зашифруємо повідомлення:

ТЕРМІНАТОР ПРИБУВАЄ ДВАЦЯТОГО ОПІВНОЧІ

за допомогою таблиці 5х7 і ключового слова «Пелікан»:

П	Е	Л	І	К	А	Н
7	2	5	3	4	1	6
Т	Н	П	В	А	Г	В
Е	А	Р	А	Ц	О	Н
Р	Т	И	Є	Я	О	О
М	О	Б	Д	Т	П	Ч
І	Р	У	В	О	І	І

А	Е	І	К	Л	Н	П
1	2	3	4	5	6	7
Г	Н	В	А	П	В	Т
О	А	А	Ц	Р	Н	Е
О	Т	Є	Я	И	О	Р
П	О	Д	Т	Б	Ч	М
І	Р	В	О	У	І	І

При зчитуванні вмісту правої таблиці по рядкам і запису шифртекста групами по п'ять букв отримаємо зашифроване повідомлення:

ГНВАПВТ ОААЦРНЕ ОТЄЯИОР ПОДТБЧМ ІРВОУІІ

Особливості структури таблиці використовуються в якості ключової інформації в магічних квадратах, які також застосовувалися в середні століття. ***Магічними квадратами*** називають квадратні таблиці з вписаними в їх клітини послідовними натуральними числами, починаючи від 1, які дають в сумі по кожному стовпцю, кожному рядку і кожній діагоналі одне і те ж число.

Шифрований текст вписували в магічні квадрати відповідно з нумерацією їх клітин. Якщо потім виписати вміст такої таблиці по рядкам, то вийде шифртекст,

сформований завдяки перестановці букв вихідного повідомлення. У ті часи вважалося, що створені за допомогою магічних квадратів шифртекст охороняє не тільки ключ, але і магічна сила.

Приклад. Нижче наведено один з варіантів магічного квадрата розміром 4x4. Зашифруємо з його допомогою повідомлення:

ПРИЛІТАЮ ВОСЬМОГО

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

О	И	Р	М
І	О	С	Ю
В	Т	А	Ь
Л	Г	О	П

Шифртекст, одержуваний при зчитуванні містимо правої таблиці по рядкам, має цілком загадковий вигляд:

ОИРМ ІОСЮ ВТАЬ ЛГОП

Число магічних квадратів швидко зростає зі збільшенням розміру квадрата. Існує тільки один магічний квадрат розміром 3x3 (якщо не враховувати його повороти). Кількість магічних квадратів 4x4 становить вже 880, а кількість магічних квадратів 5x5 - близько 250000. Магічні квадрати середніх і великих розмірів могли служити хорошою базою для забезпечення потреб шифрування того часу, оскільки практично нереально виконати перебір вручну всіх варіантів для такого шифру.

Для забезпечення додаткової стійкості можна повторно зашифрувати повідомлення, яке вже пройшло шифрування. Такий метод шифрування називається ***подвійною перестановкою***. У випадку подвійної перестановки стовпців і рядків таблиці перестановки визначаються окремо для стовпців і окремо для рядків. Спочатку в таблицю записується текст повідомлення, а потім по черзі переставляються стовпці, а потім рядки. При дешифруванні порядок перестановок повинен бути зворотним.

Число варіантів подвійної перестановки швидко зростає при збільшенні розміру таблиці. Однак і подвійна перестановка не відрізняється високою стійкістю і порівняно просто "зламується" при будь-якому розмірі таблиці шифрування.

Висновок: Шифрування перестановкою полягає в тому, що символи шифрованого тексту переставляються за певним правилом в межах деякого блоку цього тексту. При достатній довжині блоку, в межах якого здійснюється перестановка, і складним неповторним порядком перестановки можна досягти прийнятної для простих практичних застосувань стійкості шифру.

До шифрів перестановки відноситься і ***шифр Кардано***. У середині XVI століття італійським математиком Кардано була висунута ідея використання частини самого переданого відкритого тексту в якості ключа і новий спосіб шифрування, який отримав назву «решітка Кардано». Для її виготовлення береться квадратний шматок картону, в якому за спеціальними правилами прорізаються «вікна». При шифруванні решітка накладалася на аркуш паперу, і відкритий текст вписувався в вікна. Потім решітка поверталася на 90 градусів, і знову вписувався текст, всього 4 рази.

Головна вимога до «решітки Кардано» - при всіх поворотах вікна не повинні потрапляти на одне і те ж місце паперу. Порожні місця заповнювалися довільними літерами, потім шифрограма виписувалася порядково. Решітка Кардано дозволяє здійснювати перестановку букв досить швидко.

Як математик, Кардано зумів обчислити кількість квадратів-решіток (ключів) заданого розміру NxN. Якщо N - парне число, 4^{N-1}. При $N = 10$ то ця кількість дорівнює воно має порядок десять у п'ятнадцятій ступені.

1.1 Стеганографічні шифри

Ідея шифру-решітки, запропонованого Кардано, лежить і в основі знаменитого ***шифру Рішельє***, в якому зашифрований текст зовні мав вигляд осмисленого повідомлення, що не має ніякого стосунку до переданої інформації. З щільного матеріалу вирізали прямокутник розміром, наприклад, 7x10; в ньому прорублялися вікна (на малюнку вони заштриховані).

I	.	L	O	V	E	.	Y	O	U
I	.	H	A	V	E	.	Y	O	U
D	E	E	P	.	U	N	D	E	R
M	Y	.	S	K	I	N	.	M	Y
L	O	V	E	.	L	A	S	T	S

F	O	R	E	V	E	R	.	I	N
H	Y	P	E	R	S	P	A	C	E

Секретний текст вписувався в ці вікна, потім решітка знімалася і залишилися клітини заповнювалися так, щоб виходило повідомлення, що має зовсім інший сенс. Сувору команду англійською мовою: «YOU KILL AT ONCE» за допомогою такої решітки можна заховати в «невинний» текст любовного змісту: «I LOVE YOU. I HAVE YOU. DEEP UNDER MY SKIN. MY LOVE. LASTS LEVER IN HYPERSPACE».

Такі шифри відносяться вже до *стеганографічним методам* захисту інформації, які припускають приховування самого факту передачі інформації. До них відносяться також, які не мають відношення до криптографії, тайнопис (невидиме чорнило, яке проявляється при спеціальній обробці, мікроточки в тексті книги і т.п.).

Стеганографічні методи у вигляді замаскованих закладок використовуються для захисту програмних продуктів від несанкціонованого копіювання.

1.1 Шифри простої заміни

У шифрі простої заміни кожен символ вихідного тексту замінюється символами того ж алфавіту однаково на всьому протязі тексту. Шифри простої заміни називають також шифрами *одноалфавітної підстановки*.

Одним з перших шифрів простої заміни вважається так званий ***полібіанський квадрат***. За два століття до нашої ери грецький полководець і історик Полібій винайшов для цілей шифрування квадратну таблицю розміром 5x5, заповнену літерами алфавіту у випадковому порядку.

При шифруванні в цьому полібіанському квадраті знаходили чергову букву відкритого тексту і записували в шифртекст букву, розташовану нижче її в тому ж стовпці. Якщо буква тексту виявлялася в нижньому рядку таблиці, то для шифртекста брали саму верхню букву з того ж стовпця. Концепція полібіанського квадрата виявилася плідною і знайшла застосування в криптосистемах наступних часів.

Шифр Цезаря є окремим випадком шифру простої заміни (одноалфавітної підстановки). При шифруванні вихідного тексту кожна буква замінялася на іншу букву того ж алфавіту шляхом зміщення за алфавітом від вихідної букви на К букв. При досягненні кінця алфавіту виконувався циклічний перехід до його початку. Цезар використовував шифр заміни при зміщенні К = 3. Наприклад, послання Цезаря VENI VIDI VICI (в перекладі на українську означає "Прийшов, Побачив, Переміг"), спрямоване його другу Амінтію після перемоги над понтійським царем Фарнаком, сином Мітрідата, виглядало б в зашифрованому вигляді так:

YHQL YLGL YLFL

У той же час, такий шифр заміни можна задати таблицею підстановок, що містить відповідні пари букв відкритого тексту і шифртекста.

Перевагою системи Цезаря є простота шифрування і дешифрування, що обумовлює її застосування і в складних сучасних ширах в якості складового елементу.

До недоліків слід віднести наступне:

- число можливих ключів *k* мало (не більш букв алфавіту);
- зберігається алфавітний порядок в послідовності букв, які замінюються; при зміні значення *k* змінюються тільки початкові позиції такої послідовності і досить розшифрувати заміну однієї літери, щоб визначити всі інші заміни;
- шифр Цезаря легко розкривається на основі аналізу частот появи літер в шифртексті, так як підстановки, що виконуються відповідно з шифром Цезаря, не маскують частот появи різних букв вихідного відкритого тексту.

Модифікацією цього шифру є ***система шифрування Цезаря з ключовим словом***. Ця система також є одноалфавітною. Особливістю її є використання ключового слова для зміщення і зміни порядку символів в алфавіті підстановки.

Ключове слово записується під літерами алфавіту, починаючи з літери, числовий код якої збігається з обраним числом *k*. Необхідно, щоб всі букви ключового слова були різні (інакше можна повторювані букви виключити). Букви алфавіту підстановки, які не ввійшли в ключове слово, записуються після ключового слова в алфавітному порядку. Виходить підстановка для кожної літери довільного повідомлення.

Приклад: виберемо ключове слово «інформація» і $k = 3$. Тоді правило підстановки буде наступним:

літери початкового тексту: абвгґдеєжзиіїйклмнопрстуфхцчшщьюя

літери шифртекста: ьюінформацябвгґдеєжзиїйклпстухчшщ

Перевагою системи Цезаря з ключовим словом є те, що кількість можливих ключових слів практично невичерпний.

1.1 Криптоаналітична статистична атака і система омофонів

Недоліком всіх шифрів простої заміни є можливість злому шифртекста на основі аналізу частот появи літер. Криптоаналітична атака проти системи одноалфавітної заміни починається з підрахунку частот появи символів:

- визначається число появ кожної букви в шифртексті.
- отриманий розподіл частот букв в шифртексті порівнюється з розподілом частот букв в алфавіті вихідних повідомлень, наприклад в англійському.
- буква з найвищою частотою появи в шифртексті замінюється на букву з найвищою частотою появи в англійській мові і т.п.

Згідно із законом великих чисел ймовірність успішного розкриття системи шифрування підвищується зі збільшенням довжини шифртекста.

Найпростіший захист проти атак, заснованих на підрахунку частот, забезпечується в криптосистемі омофонів (HOMOPHONES), яка також є одноалфавітною: тільки при цьому літери вихідного повідомлення мають кілька замін. І число замін для кожної літери пропорційне частоті її появи. Таким чином, розмір алфавіту шифртексту більше розміру вихідного алфавіту.

При шифруванні кожна буква алфавіту замінюється на трьохрозрядне число. Заміни (їх ще називають омофонами) представлені числами від 000 до 999. Найбільш рідко букви, які зустрічаються, отримують по одному числу для заміни, інші - по кілька чисел, одне з яких вибирається випадковим рівноймовірним чином при шифруванні даної літери. Число чисел для заміни кожної букви береться пропорційно частоті її появи. При шифруванні букви вихідного повідомлення, вибирається одна з її замін, тобто кожна літера вихідного алфавіту шифрується трьома цифрами і шифртекст в три рази довше вихідного тексту.

Наприклад, англійська літера E зустрічається в 3 рази частіше букви L і в 123 рази частіше букв J і Z, а українська буква Е зустрічається в 36 разів частіше, ніж буква Ф. Тоді в англійському алфавіті літери J і Z для заміни отримують по одному числу, а буква Е замінюється на вибране випадковим чином одне з 123 тризначних чисел з призначеної їй в таблиці замін.

Таким чином, кожен омофон з'являється в шифртексті рівноймовірно, і простий підрахунок частот нічого не дасть криптоаналітику. Однак доступна також інформація про розподіл пар і трійок букв в різних звичайних мовах. Криптоаналіз, заснований на такій інформації, буде більш успішним, але і більш трудомістким.

Подібні шифри, де одній букві відкритого тексту ставиться у відповідність декілька букв з алфавіту шифртекста називають багатозначними шифрами заміни. Хоча

криптоаналіз таких шифрів і більш трудомісткий, принциповий підхід до дешифрування таких систем той же, що і для шифрів простої заміни.

1.1 Шифри складної заміни

Шифри складної заміни називають *багатоалфавітними*, так як для шифрування кожного символу вихідного повідомлення застосовують свій шифр простої заміни. Багатоалфавітня підстановка послідовно і циклічно змінює використовувані алфавіти.

При *r*-алфавітній підстановці символ x_0 вихідного повідомлення замінюється символом з алфавіту B_0, символ x_1 - символом з алфавіту B_1 і так далі, символ x_{r-1} замінюється символом з алфавіту B_{r-1}, символ x_r замінюється символом знову з алфавіту B_0 і т.п.

Приклад багатоалфавітної підстановки ($r = 4$):

Вхідний символ $x_0\ x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8\ x_9$

Алфавіт підстановки $B_0\ B_1\ B_2\ B_3\ B_0\ B_1\ B_2\ B_3\ B_0\ B_1$

Ефект використання багатоалфавітної підстановки полягає в тому, що забезпечується маскування звичайної статистики вихідної мови, так як конкретний символ з вихідного алфавіту X може бути перетворений в кілька різних символів шифрувальних алфавітів B. Ступінь забезпечуваного захисту теоретично пропорційне довжині періоду *r* в послідовності використовуваних алфавітів B.

Багатоалфавітні шифри заміни запропонував і ввів в практику криптографії Леон Батист Альберті, який також був відомим архітектором і теоретиком мистецтва. Його книга "Трактат про шифр", написана в 1566 р, представляла собою першою в Європі наукову працю по криптології. Криптологи усього світу шанують Альберті основоположником криптології. Він же вперше висунув ідею повторного шифрування, яка у вигляді ідеї багаторазового шифрування лежить в основі всіх сучасних шифрів з секретним ключем. Крім шифру багатоалфавітної заміни Альберті також докладно описав пристрої для його реалізації.

Диск Альберті являє собою систему із зовнішніх нерухомих та внутрішніх рухомих дисків, на які нанесені символи алфавіту і цифри. На зовнішньому диску символи розташовані в алфавітному порядку, на внутрішньому - в довільному. Ключем шифрування є порядок букв на внутрішньому диску і початкове положення внутрішнього диска щодо зовнішнього. Після шифрування слова внутрішній диск зміщувався на один крок. Кількість алфавітів *r* в ньому дорівнює числу символів на диску.

Шифр Гронсфельда. Шифр складної заміни, званий шифром Гронсфельда, є модифікацією шифру Цезаря числовим ключем. Для цього під літерами вихідного повідомлення записують цифри числового ключа. Якщо ключ коротше повідомлення, то його запис циклічно повторюють.

Шифртекст отримують аналогічно, як в шифрі Цезаря, але відраховують за алфавітом не третю букву (як це робиться в шифрі Цезаря), а вибирають ту букву, яка зміщена за алфавітом на відповідну цифру ключа. Наприклад, застосовуючи в якості ключа групу з чотирьох початкових цифр числа *e* (основи натуральних логарифмів), а саме 2718, отримуємо для вихідного повідомлення ПІВДЕННИЙ ЕКСПРЕС наступний шифртекст:

Повідомлення	П	І	В	Д	Е	Н	Н	И	Й		Е	К	С	П	Р	Е	С
Ключ	2	7	1	8	2	7	1	8	2		7	1	8	2	7	1	8
Шифртекст	С	О	Г	Й	Ж	Ф	О	О	Л		Й	Л	Щ	С	Ч	Є	Щ

Щоб зашифрувати першу букву повідомлення П, використовуючи першу цифру ключа 2, потрібно відрахувати другу по порядку букву від П, виходить перша буква шифртекста С.

Слід зазначити, що шифр Гронсфельда розкривається відносно легко, якщо врахувати, що в числовому ключі кожна цифра має тільки десять значень, а значить, є лише десять варіантів прочитання кожної букви шифртекста. З іншого боку, шифр Гронсфельда допускає подальші модифікації, що поліпшують його стійкість, зокрема подвійне шифрування різними числовими ключами. Цей шифр є по суті окремим випадком системи шифрування Віженера, який проаналізував і об'єднав запропоновані Трітеміем, Белазом і Портом підходи, по суті не вносячи в них нічого оригінального.

Система шифрування Віженера. Система Віженера вперше була опублікована в 1586 році і є однією з найстаріших і найбільш відомих багатоалфавітних систем. Свою назву вона отримала по імені французького дипломата XVI століття Блеза Віженера, який розвивав і удосконалював криптографічні системи. Вона подібна до такої системи шифрування Цезаря, у якій ключ підстановки змінюється від букви до букви. Автор шифру запропонував для шифрування таблицю, яку називають таблицею (або квадратом) Віженера. Її розмір дорівнює довжині алфавіту. Верхній (нульовий) рядок заповнюється символами алфавіту, лівий стовпець - цифрами.

Сама таблиця заповнюється за таким правилом. Перший рядок має цифровий ключ «0» і заповнюється усіма символами за алфавітом, другий має цифровий ключ «1» і заповнюється тими ж символами, зсунутими вправо на один символ по колу, і далі, *k*-а має цифровий ключ «*k*-1» і заповнюється тими ж символами, зсунутими вправо на (*k*-1) символ по колу.

Наведемо фрагмент таблиці Віженера для українського алфавіту.

	а	***б***	***в***	***г***	***ґ***	***д***	***е***	***є***	***ж***	***з***	***и***	***і***	***ї***	***й***	***к***	***л***	***м***	***н***	***о***	***п***	***р***	***с***	***т***	***у***	***ф***	***х***	***ц***	***ч***			***ь***	***ю***	***я***
0	а	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	ш	ь	ю	я
1	б	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	ш	ь	ю	я	а
2	в	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	ш	ь	ю	я	а	б
3	г	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в
4	ґ	д	е	є	ж	з	и	і	ї	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ю	я	а	б	в	г
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	ґ
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	ґ	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	ґ	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	ґ	д	е	є

При шифруванні верхній рядок підкреслених символів, використовується для пошуку чергової букви відкритого тексту. Крайній лівий стовпчик використовується для пошуку відповідного цій букві елемента ключа. Чергова буква шифртекста знаходиться на перетині стовбця, що визначається шифрованою буквою, і рядки, які визначаються числовим значенням ключа.

Краще, якщо ключова послідовність вибирається як набір випадкових чисел. Іноді, щоб ключ легше було запам'ятати, використовують слово або фразу, а потім замінюють літери в ній їх номерами в алфавіті. Якщо ключ виявився коротше повідомлення, то його циклічно повторюють.

Розглянемо приклад отримання шифртекста за допомогою таблиці Віженера. Нехай вибрано ключове слово АМБРОЗІЯ. Необхідно зашифрувати повідомлення ПРИЛІТАЮ ВОСЬМОГО. Випишемо вихідне повідомлення в рядок і запишемо під ним ключове слово з повторенням. У третій рядок випишемо шифртекст.

Повідомлення	П Р И Л І Т А Ю В О С Ь М О Г О
Ключ	А М Б Р О З І Я А М Б Р О З І Я
Шифртекст	П Г І В Щ Ю І Ь В Б Т Н Б Ч К Н

Шифри Віженера з коротким періодичним ключем використовуються і в наші дні в системах шифрування, від яких не потрібна висока криптостійкість.

З розвитком математики необхідність в таблицях шифрування відпала. Якщо замінити букви на числа, то операції шифрування і дешифрування легко виражаються простими математичними формулами. Так, в шифрі Віженера використовуються операції циклічного, або модульного складання.

2.6. Одноразова система шифрування

Найважливішим для розвитку криптографії був результат К. Шеннона про існування та єдність абсолютно стійкого шифру. Єдиним таким шифром є якась форма так званої *стрічки одноразового використання*, в якій відкритий текст «об'єднується» з повністю випадковим ключем такої ж довжини. Цей результат був доведений К. Шенноном за допомогою розробленого ним теоретико-інформаційного методу дослідження шифрів. Обговоримо особливості будови абсолютно стійкого шифру і можливості його практичного використання.

Нехай $\{K_i;\ 0 < i < n\}$ - незалежні випадкові змінні, що приймають рівноімовірні значення на множині Z_m:

$$P\ \{(K_0, K_1, ..., K_{n-1}) = (k_0, k_1, ..., k_{n-1})\} = (1/m)^n$$

Одноразова система шифрування перетворює вихідний текст $(x_0, x_1, ..., x_{n-1})$ в шифрований текст $(y_0, y_1, ..., y_{n-1})$ за допомогою підстановки Цезаря:

$$y_i = E_k(x_i) = (k_i + x_i) \bmod m,\ i = 0, ..., n-1$$

Для такої системи підстановки використовують також термін "одноразова стрічка" і "одноразовий блокнот". Простір ключів K системи одноразовою підстановки складається з векторів $(k_0, k_1, ..., k_{n-1})$ і містить m^n точок.

Процес шифрування фактично являє собою накладення білого шуму в вигляді нескінченного ключа на вихідний текст, який змінює статистичні характеристики мови джерела. Системи одноразового використання *теоретично не розшифровані*, так як не містять достатньої інформації для відновлення тексту.

Підкреслимо, що для абсолютної стійкості значним є кожна з таких вимог до стрічки одноразового використання:

1) повна випадковість і рівноймовірність ключа (це, зокрема, означає, що ключ не можна виробляти за допомогою будь-якого детермінованого пристрою);

2) рівність довжини ключа і довжини відкритого тексту;

3) однократність використання ключа.

У разі порушення хоча б однієї з цих умов шифр перестає бути абсолютно стійким, і з'являються принципові можливості для його розкриття (хоча вони можуть бути важко реалізованими). Але саме ці умови і роблять абсолютно стійкий шифр дуже дорогим і непрактичним. Перш ніж користуватися таким шифром, ми повинні забезпечити всіх абонентів достатнім запасом випадкових ключів і виключити можливість їх повторного застосування.

У разі передачі інформації через комп'ютерні мережі, зокрема для інформаційних систем, де доводиться шифрувати не один мільйон знаків, завдання, хоча, і вирішене теоретично, з практичної точки зору не посунулася не так на крок, так як передача

секретного тексту замінюється передачею секретного ключа точно такий ж довжини.

2.7. Шифр Вернама

Простим прикладом реалізації абсолютно стійкого шифру (при використанні ключа, рівного по довжині вихідного тексту) є шифр, запропонований в 1926 році співробітником фірми AT&T Вернамом. Розроблене Вернамом на основі цього апарату, що зчитує і обладнання для шифрування використовувалося в той час корпусом зв'язку армії США.

Шифр використовує двійкове подання символів вихідного тексту з використанням телеграфного коду Бодо, де кожна літера вихідного тексту переводилася в п'ятибітовий символ з використанням старовинного телетайпа фірми AT&T і записувалася на паперовій перфострічці. Таким же чином записувався на перфострічці і ключ. При шифруванні, яке могло бути виконано простим накладенням двох перфострічок однакової довжини один на одного, ключ додавався до початкового тексту шляхом побітового додавання $\oplus$ по модулю два символів відкритого тексту і ключа:

$y_i = x_i \oplus k_i$, $i = 1, ..., n$

Тут $x_1 x_2 ... x_n$ - відкритий текст, $k_1 k_2 ... k_n$ - ключ, $y_1 y_2 ... y_n$ - шифрований текст.

Дешифрування виконується аналогічно (накладенням перфострічок) і складається в додаванні по модулю два символів шифртекста з тією ж послідовністю ключів:

$y_i \oplus k_i = x_i \oplus k_i \oplus k_i = x_i$, $i=1,...,n$

При цьому послідовності ключів, використані при шифрування і дешифрування, при додаванні по модулю два компенсують один одного і в результаті відновлюються символи x вихідного тексту.

Однак при реалізації методу Вернама виникають серйозні проблеми, пов'язані з необхідністю доставки одержувачу в такій же послідовності ключів, як у відправника, або з необхідністю, безпечного зберігання ідентичних послідовностей ключів у відправника і одержувача. Ці недоліки системи шифрування Вернама подолані при шифруванні методом гамування.

2.8. Шифрування методом гамування

Хоча одноразова система шифрування і виявилася непридатна на практиці в силу неможливості практичної реалізації всіх вимог, що забезпечують її теоретичну стійкість, ідея, що лежить в її основі знайшла практичне застосування в широко використовуваній і в даний час системі шифрування, що отримала назву методу гамування.

Гамма шифру - це псевдовипадкова послідовність, вироблена за певним алгоритмом для шифрування відкритих даних і дешифрування зашифрованих даних. Вона грає роль ключа в одноразовій системі шифрування. Строго кажучи, вона не задовольняє ні вимогу випадковості, так як використовується детермінований алгоритм для її вироблення, ні вимогу нескінченної довжини, так як всі псевдовипадкові послідовності мають кінцевий період. Проте, при правильно обраному алгоритмі генерації гами шифру можна отримати метод шифрування з хорошою практичною стійкістю, достатньою для вирішення реальних завдань захисту інформації.

Один і той же алгоритм генерації гами шифру виконується і під час пересилання і на стороні одержувача інформації, обидва мають однакову псевдовипадкову послідовність, яка використовується для шифрування і дешифрування. При цьому фактичний (підлягає передачі) обсяг ключової інформації дуже малий і складається з декількох чисел, які задають значення параметрів алгоритму та нульового елемента послідовності.

Процес шифрування цим методом називають *гамуванням*. Він шифрування полягає в генерації гами шифру і її накладення на вихідний відкритий текст за певним законом оборотним чином, наприклад з використанням операції додавання по модулю два.

Шифрування ведеться або посимвольно, або шляхом шифрування даних,

об'єднаних в блоки. При шифруванні блоками відкритий текст m розбивають на блоки m_i, i = 1, ..., M однакової довжини, зазвичай по 64 біта. Гамма шифру виробляється у вигляді послідовності блоків γ_i, i = 1, ..., M аналогічної довжини. Рівняння *зашифрування* можна записати у вигляді:

$c_i = \gamma_i \oplus m_i$, i = 1, ..., M

де c_i - i-й символ (блок) шифртекста; γ_i - i-й символ (блок) гами шифру; m_i - i-й символ (блок) відкритого тексту; M - кількість символів (блоків) відкритого тексту.

Процес *дешифрування* зводиться до повторної генерації гами шифру на стороні одержувача інформації за тим же алгоритмом і накладенню цієї гами на зашифровані дані. Рівняння дешифрування має вигляд:

$m_i = \gamma_i \oplus c_i$, i = 1, ..., M

Отриманий зашифрований текст є досить важким для розкриття в тому випадку, якщо гамма шифру не містить повторюваних послідовностей. В ідеалі гамма шифру повинна змінюватися випадковим непередбачуваним чином для кожного шифрованого слова. Якщо період гами перевищує довжину всього тексту і невідома ніяка частина вихідного тексту, то шифр можна розкрити тільки прямим перебором ключа. Тобто криптостійкість методу визначається періодом гами.

Слабке місце методу гамування в тому, що він стає безсилим, якщо зловмиснику стає відомий фрагмент вихідного тексту довжиною більш ніж період гами, і відповідна йому шифрограма. Простим вирахуванням по модулю виходить ключ і по ньому відновлюється вся послідовність. Криптоаналітик також може частково або повністю відновити ключ на основі припущень про зміст вихідного тексту. Так, якщо більшість повідомлень, які посилаються, починається зі слів "СОВ. СЕКРЕТНО", то криптоаналіз всього тексту значно полегшується. Також багато текстових редакторів, наприклад Word, вставляють в початок файлу стандартну службову інформацію, що знижує криптостійкість при шифруванні цих файлів методом гамування.

2.9. Методи генерації псевдовипадкових чисел

Один з методів генерації послідовності елементів гами, довжина яких перевищує розмір шифрованих даних - це *датчики псевдовипадкових чисел* (ПСЧ). На основі теорії груп розроблено декілька типів таких датчиків.

Найбільш доступними і ефективними є *конгруентні* генератори ПСЧ. Для цього класу генераторів можна зробити математично строгий висновок про те, якими властивостями володіють вихідні сигнали цих генераторів з точки зору періодичності та випадковості.

Одним з хороших конгруентних генераторів є лінійний конгруентний датчик ПСЧ. Він виробляє послідовності псевдовипадкових чисел γ_i, описувані співвідношенням:

$\gamma_{i+1} = (a\gamma_i + b) \bmod m$, i = 1,2, ..., M

де a і b - константи, γ_0 - вихідна величина, обрана в якості породжуваного числа. Ці три величини і утворюють ключ. Значення m зазвичай встановлюється рівним 2^n-1, де n - довжина машинного слова в бітах (m - максимально можливе число різних чисел, записаних за допомогою n біт).

З наведеної формули випливає, що як тільки ми отримаємо в якості чергового елемента $\gamma_M = \gamma_0$, елементи послідовності почнуть повторюватися. Значення M називають періодом датчика ПСЧ. Цей період залежить від значень a і b, які бажано вибрати так, щоб період був максимальним ($M = 2^n$-1). Як показано Кнутом, лінійний конгруентний датчик ПСЧ має максимальну довжину M тоді і тільки тоді, коли b - непарне і взаємно просте з m і величина, $a \bmod 4 = 1$. За іншими рекомендаціями b - взаємно просте з m і a непарне .

Також популярні так звані M-*послідовності* завдяки відносній легкості їх реалізації. M-послідовності є лінійними рекурентні послідовності максимального періоду, що формуються k-розрядними генераторами на основі регістрів зрушення. На кожному такті біт, який надійшов, зрушує k попередніх і до нього додається сума цих біт

по модулю 2. Витісняється біт додається до гами. Більш детально цей метод буде розглянуто далі.

Також перспективними представляються *нелінійні датчики* ПСЧ (наприклад, зрушенні регістри з елементом «І» в колі зворотного зв'язку), однак їх властивості ще недостатньо вивчені. Можливі й інші, більш складні варіанти вибору породжуваних чисел для гами шифру.

Лекція №3 СУЧАСНІ СИМЕТРИЧНІ КРИПТОСИСТЕМИ (DES, Магма)

3.1 Загальні принципи

На думку К. Шеннона, в практичних шифрах необхідно використовувати два загальних принципа: розсіювання та перемішування.

Розсіювання є поширення впливу одного знака відкритого тексту на багато знаків шифртекста, що дозволяє приховати статистичні властивості відкритого тексту.

Перемішування передбачає використання таких шифруючих перетворень, які ускладнюють відновлення взаємозв'язку статистичних властивостей відкритих і шифрованих текстів.

Поширеним способом досягнення ефектів розсіювання та перемішування є використання складного шифру, тобто такого шифру, який може бути реалізований у вигляді деякої послідовності простих шифрів, кожен з яких вносить свій внесок в значне сумарне розсіювання та перемішування.

У *складених шифрах* в якості простих шифрів найчастіше використовуються прості перестановки і підстановки. При перестановці просто перемішують символи відкритого тексту, причому конкретний вид перемішування визначається секретним ключем. При підстановці кожен символ відкритого тексту замінюють іншим символом з того ж алфавіту, а конкретний вид підстановки також визначається секретним ключем.

При багаторазовому чергуванні простих перестановок і підстановок, керованих досить довгим ключем, можна отримати дуже стійкий шифр з хорошим розсіюванням і перемішуванням. Однак шифр повинен не тільки ускладнювати розкриття, а й забезпечувати легкість шифрування і розшифрування при відомому користувачеві секретному ключі. Розглянуті нижче криптоалгоритми DES, IDEA і стандарт шифрування Магма даних побудовані в повній відповідності з вказаною методологією.

Важливим завданням у забезпеченні гарантованої безпеки інформації в ІС є розробка і використання стандартних алгоритмів шифрування даних.

Першим серед подібних стандартів став американський DES, що представляє собою послідовне використання замін і перестановок. В даний час все частіше говорять про невиправдану складність і невисоку крипостійкість. На практиці доводиться використовувати його модифікації.

3.2 Стандарт шифрування даних DES

Офіційна історія стандарту шифрування США почалася в 1972 році з того, що національне бюро стандартів (НБС, National Bureau of Standards, NBS) США висунуло програму розробки системи стандартів щодо захисту від несанкціонованого доступу даних, що зберігаються і оброблюються на електронно-обчислювальних машинах. Цей напрямок був визнаний одним з найбільш пріоритетних областей, яка ДСТУ 28147:2009ро потребується в регламентуючих документах. Одним з найважливіших напрямків була визнана розробка стандартного алгоритму шифрування. Діяльність щодо розробки та затвердження стандарту почалася в травні 1973 року, коли НБС опублікувало в Федеральному Реєстрі США звернення з закликом до закладів і фірмам США пропонувати

свої алгоритми шифрування як кандидати на місце стандарту. Однак відгуків з конкретними пропозиціями не було. Але вже до початку сімдесятих років двадцятого століття комп'ютери набули масового поширення і проблема захисту даних, що зберігаються і обробляються на них, стала усвідомлюватися дедалі більшим числом фахівців. Багато великих корпорацій, не кажучи вже про державні служби, проводили власні дослідження в даній області. Одним з найвідоміших дослідних центрів такого роду в ті часи була наукова лабораторія фірми IBM, очолювана доктором Фейстелем. Проведена в ній дослідницька робота привела до створення практичної системи шифрування, що отримала назву "Люцифер", і до розробки архітектурних принципів, що дозволяють створювати ефективні та надійні шифри, добре підходять для реалізації на комп'ютерних пристроях. Архітектура шифрів, що базується на цих принципах, пізніше отримала назву "мережа Фейстеля" по імені свого творця, який у своїй відомій роботі "Криптографія і комп'ютерна безпека" виклав новий підхід до створення стійких шифрів для комп'ютерного застосування.

Шифр DES (Data Encryption Standard) був розроблений фірмою IBM і сертифікований NSA (Національної Безпеки США). У 1977 р. прийнятий в якості федерального стандарту США і протягом двох десятиліть вважався досить надійним і універсальним.

Стандарт DES призначений для захисту від несанкціонованого доступу до важливої, але несекретної інформації в державних і комерційних організаціях США. Алгоритм, покладений в основу стандарту, поширювався досить швидко, і вже в 1980 р. був схвалений Національним інститутом стандартів і технологій США. З цього моменту DES перетворюється в стандарт не тільки за назвою, а й фактично. З'являється програмне забезпечення та спеціалізовані мікроЕВМ, призначені для шифрування і розшифрування інформації в мережах передачі даних.

До теперішнього часу DES є найбільш поширеним алгоритмом, використовуваним в системах захисту комерційної інформації. Більш того, реалізація алгоритму DES в таких системах стає ознакою хорошого тону.

Основні переваги алгоритму DES:

- використовується тільки один ключ довжиною 56 біт;
- зашифрувавши повідомлення за допомогою одного пакету програм, для розшифровки можна використовувати будь-який інший пакет програм, який відповідає стандарту DES;
- відносна простота алгоритму забезпечує високу швидкість обробки.

3.2.1 Мережа Фейстеля

Багато сучасних стандартних криптосистем відносяться до класу блокових шифрів, які виконують параметризоване секретним ключем перетворення блоків вихідного в блоки шифрограми.

У сучасному блоковому шифрі блоки відкритого тексту і шифр-тексту є двійкові послідовності зазвичай довжиною 64 біта. В принципі кожен блок може приймати 2^{64} значень. Тому підстановки виконуються в дуже великому алфавіті, що містить до $2^{64} \cong 10^{19}$ "символів".

Для побудови блочного шифру часто використовують конструкцію, звану *мережею Фейстеля* (Feistel Network). Блок повідомлення T розбивається на дві частини L і R по 32 біта кожен: T = L R.

Одна ітерація або крок мережі Фейстеля є наступне перетворення:

$L_i = R_{i-1}$

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

Кроки повторюються багаторазово, причому (L_i, R_i) використовується в якості (L_{i-1}, R_{i-1}) на наступній ітерації. Функція f_i, може залежати або не залежати від номера кроку i. На кожному кроці використовується з'єднання K_i, який вираховується по ключу K. Вже після двох кроків кожна з частин результату залежить від обох частин вихідного

повідомлення.

Крім того, навіть якщо функція f не є взаємно однозначною, перетворення мережі Фейстеля можна зупинити:

$R_{i-1} = L_i$

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

3.2.2 Структура алгоритму DES

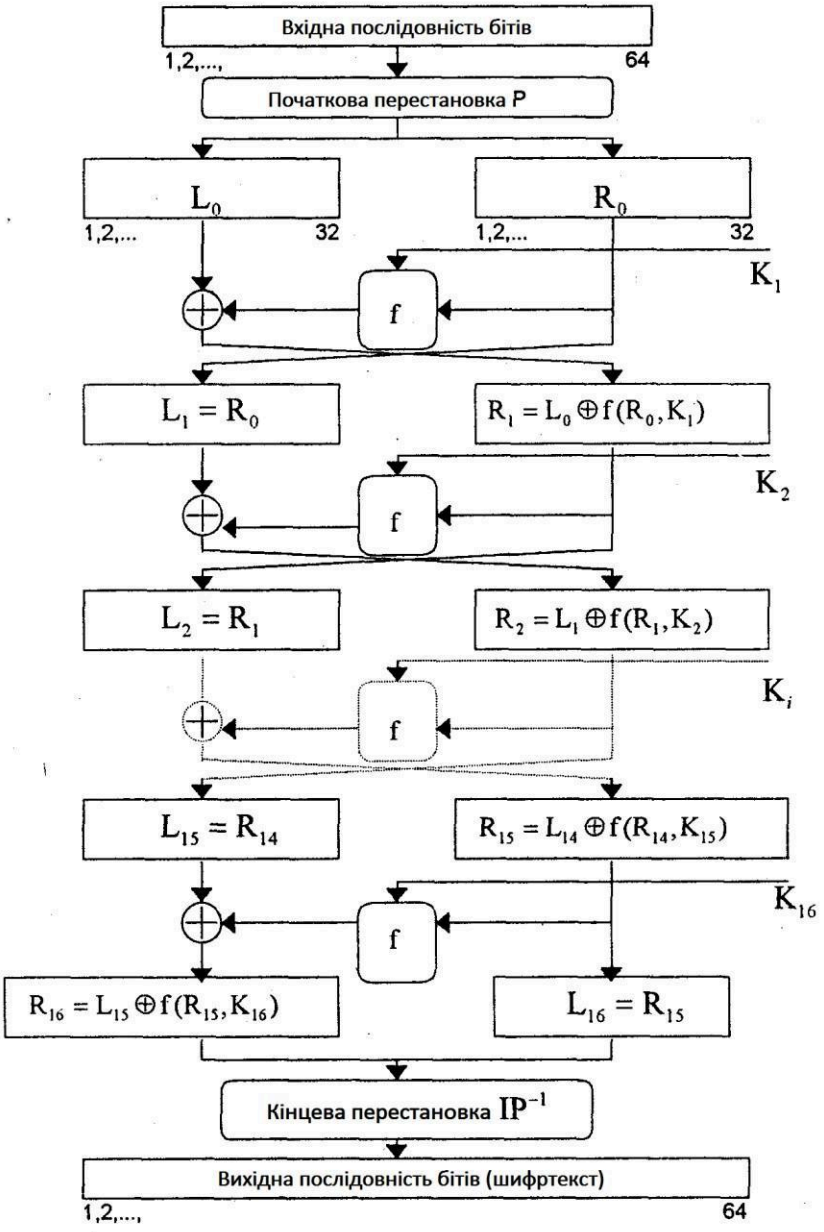


Рис. 3.1. Структура алгоритму DES

Алгоритм DES також використовує комбінацію підстановок і перестановок. DES здійснює шифрування 64-бітових блоків даних за допомогою 64-бітового ключа, в якому значущими є 56 біт (інші 8 біт - перевірочні біти для контролю на парність).

Процес шифрування полягає в початковій перестановці бітів 64-бітового блоку, шістнадцяти циклах шифрування в мережі Фейстеля і в кінцевій перестановці бітів. Дешифрування в DES є операцією, зворотною шифруванню, і виконується шляхом повторення операцій шифрування в зворотній послідовності.

Таблиця 3.1. Матриця початкової перестановки I

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

P

Усі наведені таблиці є стандартними і повинні включатися в реалізацію алгоритму DES в незмінному вигляді.

Всі перестановки і коди в таблицях підібрані розробниками таким чином, щоб максимально ускладнити процес розшифровки шляхом підбору ключа. При описі алгоритму DES застосовані наступні позначення:

L і R - послідовності бітів (ліва (Left) і права (Right));

LR - конкатенація послідовностей L і R.

$\oplus$ - операція побітового додавання за модулем 2 (операція XOR).

Нехай з файлу вихідного тексту лічений черговий 64-бітовий (8-байтовий) блок T. Цей блок T перетворюється за допомогою матриці початкової перестановки IP (див. табл. 3.1), яка має розмірність 8x8 і являє собою записані по рядкам нові номери бітів.

Перестановка переміщує кожен вхідний біт в іншу позицію, жоден біт не використовується двічі і жоден біт не ігнорується. Наприклад, біт 58 вхідного блоку T стає бітом 1, біт 50 - бітом 2 і т.п. Цю перестановку можна описати виразом T = IP (T).

Отримана послідовність бітів T_0 розділяється на дві по 32 біта: L_0 - ліві (старші) і R_0-праві (молодші) біти. Потім виконується ітеративний процес шифрування, що складається з 16 циклів.

Нехай T_i - результат *i*-й ітерації:

$T_i = L_i R_i$

де $L_i = t_1\ t_2\ ...\ t_{32}$ (перші 32 біта); $R_i = t_{33}\ t_{34}\ ...\ t_{64}$ (останні 32 біта). Тоді результат *i*-й ітерації описується наступними формулами:

$L_i = R_i$

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

-1

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

Функція *f* називається функцією шифрування. Її аргументами є послідовність R_{i-1}, що отримується на попередньому кроці ітерації, і 48-бітову ключ K_i, який є результатом перетворення 64-бітового ключа шифру K. (Детальніше функція шифрування *f* і алгоритм отримання ключа K описані нижче.)

На останньому кроці ітерації отримують послідовності R_{16} і L_{16} (без перестановки місцями), які конкатенуються в 64-бітову послідовність $R_{16} L_{16}$.

Таблиця 3.2. Матриця зворотної перестановки

Після закінчення шифрування здійснюється відновлення позицій бітів за

допомогою матриці зворотної перестановки IP^{-1} (таблиця 3.2).

Початкова перестановка і відповідна заключна перестановка не впливають на криптостійкість DES. (Перестановка в першу чергу служить для полегшення побайтного завантаження даних відкритого тексту і шифртекста в мікросхему DES.) Перестановки називають також P-блоками.

Процес розшифрування даних є інверсним по відношенню до процеса шифрування. Він може бути описаний наступними формулами:

$R_{i-1} = L_i$

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

Таким чином, для процесу розшифрування з переставленим вхідним блоком $R_{16}L_{16}$ на першій ітерації використовується ключ K_{16}, на другий ітерації - K_{15} і т.п. На 16-й ітерації використовується ключ K_1. На останньому кроці ітерації будуть отримані послідовності L_0 і R_0, які конкатенуються в 64-бітову послідовність L_0R_0. Потім в цій послідовності 64 біта переставляються відповідно з матрицею IP^{-1}. Результат такого перетворення - вихідна послідовність бітів (розшифроване 64-бітове значення).

3.2.3 Функція шифрування

Тепер розглянемо, що ховається під перетворенням, позначеною буквою *f* (рис.3.2).

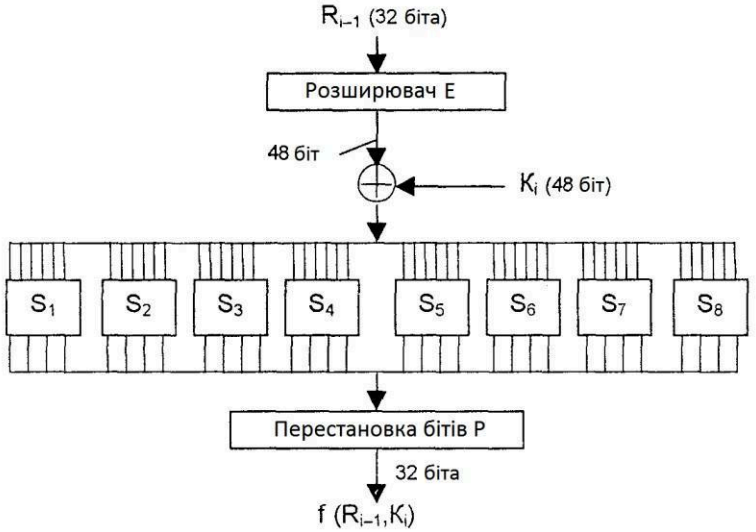


Рис. 3.2. Схема обчислення функції шифрування *f*

Аргументами функції шифрування *f* є R_{i-1} (32 біта) і K_i (48 біт). Результат функції розширення E (R_{i-1}) є 48-бітове число Для обчислення значення функції *f*

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

використовуються:

- функція E (розширення 32 біт до 48);
- функції S_1, S_2, ..., S_8 (перетворення 6-бітового числа в 4-бітове);
- функція P (перестановка бітів в 32-бітовій послідовності).

Функція розширення E, що виконує операцію перестановки з розширенням правої половини даних R_i від 32 до 48 бітів і зміною їх порядку, називається перестановкою з розширенням (приймає блок з 32 біт і породжує блок з 48 біт), визначається таблицею 3.3.

Таблиця 3.3. Функція E.

У цій операції два завдання: привести розмір правої половини у відповідність з ключем для операції ⊕ (XOR) і отримати більш довгий результат, який можна буде стиснути в ході операції підстановки. Однак криптографічний сенс даної операції полягає в тому, що за рахунок впливу одного біта на дві підстановки швидше зростає залежність бітів результату від бітів вихідних даних. Дана залежність називається *лавинним ефектом*. DES спроектований так, щоб якомога швидше домогтися залежності кожного блоку шифртекста від кожного біта відкритого тексту і кожного біта ключа.

Отриманий після перестановки з розширенням результат (позначимо його E (R_{i-1})) складається по модулю 2 з поточним значенням ключа K_i і потім розбивається на вісім 6-бітових блоків B_1, ..., B_8:

$$E(R_{i-1}) \oplus K_i = B_1 B_2 \ldots B_8$$

Номер стовпця																	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
номер рядка	0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
	2	4	1	4	8	13	6	2	11	15	12	9	7	3	10	5	0	
	3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
	0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
	2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
	3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
	0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
	2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
	3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
	0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
	2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
	3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
	0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
	2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
	3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
	0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
	2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	6	
	3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
	0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
	2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
	3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
	0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
	2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
	3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

Таблиця 3.4. Функції перетворення (вузли заміни) S_1, ..., S_8

Кожен 6-бітовий блок B_j перетворюється в 4-бітовий блок за допомогою відображення, заданого таблицями S_1, ..., S_8. Ці таблиці називаються вузлами заміни (S-boxes, S-блоками) і опубліковані разом з алгоритмом, в той час як критерії їх вибору опубліковані не були (табл 3.4).

Елементами матриць S_i розміром 4 рядки і 16 стовпців є числа від 0 до 15, які в двійковому запису являють собою 4-бітові значення.

Сукупність 6-бітових блоків B_1, ..., B_8 забезпечує вибір 4-бітового елемента (числа 0-15) в кожній з матриць S_i. Кожен блок B_i використовується для вибору номера елемента в матриці S_i наступним чином.

Нехай на вхід матриці S_i надходить 6-бітовий блок $B_j = b_1 b_2 \ldots b_6$, тоді:

- двохбітове число b_1b_6 вказує номер рядка матриці (число від 0 до 3),
- чотирьохбітове число $b_2 ... b_5$ - номер стовпця (число від 0 до 15).

Результати вузлів заміни об'єднуються в один 32-бітовий блок:
$S_1(B_1) ... S_8(B_8)$

Підстановка за допомогою вузлів заміни є ключовим етапом DES. Інші дії алгоритму лінійні і легко піддаються аналізу. S-блоки нелінійні і саме вони в більшій мірі, ніж всі інші, забезпечують криптостійкість DES. Отриманий блок перетворюється за допомогою функції перестановки бітів P. Таким чином, функція шифрування має вигляд:
f (Ri-1, Ki), = P (S1 (B1), ..., S8 (B8))

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

Таблиця 3.5. Функція перестановки бітів P

3.2.4 Генерація ключів

Процедура вироблення фактично використовуваних в алгоритмі підключів, сумарна довжина яких велика, по вихідному короткому ключу називається Key scheduling. Цю операцію немає сенсу прискорювати, оскільки при виконанні «законного» шифрування або дешифрування з відомим ключем, процедуру потрібно виконати тільки один раз, а при підборі ключа зловмисник змушений повторювати її багаторазово.

На кожній ітерації використовується нове значення ключа K_i, (довжиною 48 біт). Нове значення ключа K_i обчислюється з початкового ключа K. Процедура перетворення ключа K зводиться до наступних дій (див. Рис.3.3).

Спочатку 64-бітовий ключ DES зменшується до 56-бітового ключа відкиданням кожного восьмого біта. Це біти, розташовані в позиціях 8, 16, 24, 32, 40, 48, 56, 64. Вони використовуються для контролю по парності і дозволяють перевіряти правильність ключа. Таким чином, в дійсності ключ шифру є 56-бітовим.

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

Таблиця 3.6. Функція G

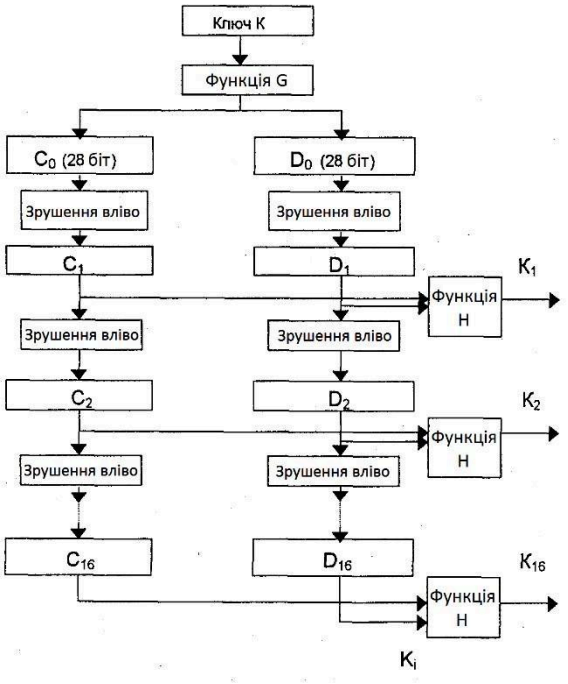


Рис. 3.3. Схема алгоритму обчислення підключів

Для видалення контрольних бітів і підготовки ключа до роботи використовується функція G (K) початкової підготовки ключа (табл. 3.6). Таблиця 3.6 розділена на дві частини. Результат перетворення G (K) розбивається на дві половини C_0 і D_0 по 28 біт кожен. Перші чотири рядки матриці G визначають, як вибираються біти послідовності C_0.(першим бітом C_0 буде біт 57 ключа шифру, потім біт 49 і т.п., а останніми бітами - біти 44 і 36 ключа).

Наступні чотири рядки матриці G визначають, як вибираються біти послідовності D_0 (тобто послідовність D_0 буде складатися з бітів 63, 55, 47, ..., 12, 4 ключа шифру).

Після визначення C_0 і D_0 рекурсивно визначаються C_i, і D_i, i = 1, 2, ..., 16. Для цього застосовуються операції циклічного зрушення вліво на один або два біта в залежності від номера кроку ітерації, як показано в таблиці 3.7. Операції зрушення виконуються для послідовностей C_i, і D_i незалежно.

Таблиця 3.7.

Номер ітерації	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Кількість зрушень вліво (біт)	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Після зрушення вибирається 48 з 56 бітів. Отже, ключ K_i, який визначається на кожному кроці ітерації, є результат вибору конкретних бітів з 56-бітової послідовності C_i D_i

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

і їх перестановки. Так як при цьому не тільки вибирається підмножина бітів, але і змінюється їх порядок, ця операція отримала назву *перестановка із стисненням*. Її також називають функцією H завершальної обробки ключа. Функція H визначається матрицею, наведеної в таблиці 3.8.

Через зрушення для кожного підключа використовуються різні підмножини біт ключа. Кожен біт використовується приблизно в чотирнадцяти з шістнадцяти підключів при цьому не всі біти використовують однакове число разів. Отже, ключ K_i визначається перетворенням:

Таблиця 3.8. Функція H

$$K_i = H (C_i D_i)$$

3.2.5 Недоліки алгоритму DES

Алгоритм DES має ряд суттєвих недоліків. Основні з них:

- Бітові операції в вузлах заміни неефективно реалізуються програмним шляхом.
- Коротка довжина ключа (56 бітів), що, власне, і допомогло організувати повний перебір.
- Виявлена теоретична можливість зменшити простір перебору за допомогою диференціального криптоаналізу (з вибором шифрограми) і лінійного криптоаналізу (з

відомим повідомленням), якщо відомо досить багато (близько 2^{47}) пар повідомлення — шифрограма.

- Відомо, що незалежний вибір підключів практично не збільшує стійкості алгоритму.

3.2.6 Крипостійкість DES

У січні 1997р. компанія RSA Data Security опублікувала зашифрований за допомогою DES текст і призначила приз в 10 тис. доларів тому, хто його розшифрує. Результат був отриманий за 4 місяці «грубою силою», шляхом повного перебору ключів, в якому брало участь близько 78 000 комп'ютерів в США і Канаді, в основному звичайних персон альних (в середньому 14 000 комп'ютерів щодня). Користувачі цих машин отримували по інтернету програму для перебору і інтервали простору ключів. Всього було випробувано близько $18x10^{15}$ варіантів з приблизно $72x10^{15}$ можливих. Пік продуктивності склав $7x10^{9}$ варіантів в секунду. Результат був отриманий на комп'ютері з процесором Pentium-90 і 16 мегабайт пам'яті. Таким чином, DES можна вважати надійним

шифром, тим більше при багаторазовому збільшенню з тих пір потужності комп'ютерів. Однак, DES справив великий вплив на розвиток криптосистем з секретним ключем і до сих пір використовується для захисту інформації невисокого рівня секретності.

3.2.7 Способи поліпшення DES. Потрійний DES

Перетворення DES не утворюють групи, тобто композиція двох операцій шифрування з різними ключами не є в загальному випадку DES-шифруванням з деякими третім ключем. Отже, можна намагатися збільшити простір ключів за рахунок багаторазового застосування DES.

Ключ DES має довжину 56 біт, тому існує 2^{56} можливих варіантів такого ключа.

Подвійний DES:

$c = DES_{k1}(DES_{k2}(m))$

Не забезпечує збільшення в 2^{56} разів обсягу перебору, необхідного для визначення ключа, оскільки при атаці з відомим повідомленням можна підбирати паралельно ключі k_1 і k_2, накопичувати в хеш-таблиці значення $DES_{k2}(m)$ і $DES_{k1}^{-1}(c)$ і шукати збіги між ними .

Фахівцями рекомендується потрійний DES:

- в режимі ECB: $c = DES_{k1}(DES_{k2}(DES_{k3}(m)))$
  або $c = DES_{k1}(DES_{k2}^{-1}(DES_{k3}(m)))$,
- в інших режимах: $c = DES_{k1}(DES_{k}^{-1}(DES_{k1}(m)))$.

Застосування функції дешифрування на другому кроці пояснюється бажанням досягти сумісності з одноразовим алгоритмом DES в разі, якщо всі ключі рівні.

Потрійне шифрування з двома ключами все одно зводиться до однократного при використанні атаки з вибором повідомлення.

3.3 Основні режими роботи алгоритму DES

Алгоритм DES цілком підходить як для шифрування, так і для аутентифікації даних. Він дозволяє безпосередньо перетворювати 64-бітовий вхідний відкритий текст в 64-бітовий вихідний шифрований текст, проте дані рідко обмежуються 64 розрядами.

Щоб скористатися алгоритмом DES для вирішення різноманітних криптографічних завдань, розроблені чотири робочих режиму:

- електронна кодова книга ECB (Electronic Code Book);
- зчеплення блоків шифру CBC (Cipher Block Chaining);
- зворотний зв'язок по шифртексту CFB (Cipher Feed Back);
- зворотний зв'язок по виходу OFB (Output Feed Back).

Ці режими можуть бути використані для шифрування за допомогою довільного блокового шифру (не тільки DES) потоку даних, який в загальному випадку довший блоку шифрування, хоча вперше вони введені в стандарті DES.

3.3.1 Режим ECB - Електронна кодова книга

У режимі Electronic Code Book (ECB) або "Електронна кодова книга" шифруємий файл розбивають на 64-бітові відрізки (блоки) по 8 байтів. Кожен з цих блоків шифрують незалежно з використанням одного і того ж ключа шифрування.

Режим придатний для шифрування при прямому доступі до фрагментів шифруємої інформації. У разі помилки при передачі відбувається відновлення, тобто помилка впливає тільки на один блок і не поширюється далі.

Основна перевага - простота реалізації. Крипостійкість режиму ECB не нижче криптостійкості використовуваного блочного алгоритму. Основний недолік режиму в

тому, що однакові блоки шифруються однаково, що дає матеріал для статистичних атак. Крім того, відкритий текст може бути легко змінений шляхом видалення, реплікації і перестановки блоків шифртекста.

Застосування режиму рекомендується тільки для шифрування короткої і випадкової інформації, наприклад ключів.

3.3.2 Режим CBC - "Зчеплення блоків шифру"

У режимі Cipher Block Chaining (CBC) або "Зчеплення блоків шифру" вихідний файл також розбивається на 64-бітові блоки, але обробляються вони інакше. Перший блок відкритого тексту m_1 складається по модулю два з 64-бітовим секретним початковим вектором c_0, званим вектором ініціалізації. Отримана сума шифрується за схемою DES. Отриманий 64-бітовий блок шифртекста c_1 складається по модулю два з другим блоком відкритого тексту, результат шифрується і виходить другий 64-бітовий блок шифртекста c_2 і так до кінця файлу. Таким чином, результат шифрування c_i визначається наступним чином:

$$c_i = DES\,(m_i \oplus c_{i-1}),\ \forall\, i = 1, \ldots, n$$

де n - число блоків вихідного файлу, а c_0 - початкове значення шифру, рівне вектору ініціалізації. Очевидно, що останній 64-бітовий блок шифртекста є функцією секретного ключа, початкового вектора і кожного біта відкритого тексту незалежно від його довжини. Цей блок шифртекста називають кодом аутентифікації повідомлення (КАП).

Код КАП може бути легко перевірено одержувачем, що володіє секретним ключем і початковим вектором, шляхом повторення процедури, виконаної відправником. Сторонній, однак, не може здійснити генерацію КАП, який сприймався б одержувачем як справжній, щоб додати його до неправдивого повідомлення або відокремити КАП від істинного повідомлення для використання його зі зміненим або неправдивим повідомленням.

Розшифрування здійснюється шляхом побітового складання попереднього блоку шифртекста з розшифрованим блоком поточного тексту:

$$m_i = c_{i-1} \oplus \mathrm{DES}^{-1}(c_i)$$

Застосування режиму CBC дозволяє усунути недолік режиму ECB: кожен блок відкритого тексту «маскується» блоком шифртекста, отриманим на попередньому етапі. Таким чином, можливість зміни відкритого тексту досить обмежена майже всі маніпуляції з блоками шифртекста будуть виявлені.

Гідність даного режиму в тому, що він не дозволяє накопичуватися помилкам при передачі. Блок m_i є функцією тільки c_i і c_{i-1} Якщо при передачі буде викривлений один з блоків шифрограми c_i, то при розшифровці блок m_i виявиться повністю викривленим, а в блоці m_{i+1} будуть спотворені тільки біти, що змінилися в блоці c_i. Режим забезпечує відновлення після помилок при передачі, але тільки після таких, які призводять до зміни окремих бітів, а не до випадання бітів і додавання нових.

Швидкість обробки в даному режимі дорівнює продуктивності використовуваного блочного алгоритму, затримка при виконанні операції $\oplus$ зневажливо мала.

В цьому режимі останній блок може виявитися коротшим 64 бітів. У такому випадку застосовується один з наступних способів (тут j - розмір останнього блоку m_n, операції >> і << означають, як і в мові програмування С, арифметичні зрушення вправо і вліво на задане число розрядів):

- неповний блок m_n відкритого тексту доповнюється фіксованим доповненням p_{64-j} до 64 бітів, наприклад пропусками:
  $c_n = \mathrm{DES}\,(((m_n << (64-j)) + p_{64-j}) \oplus c_{n-1})$
- неповний блок m_n відкритого тексту складається побітно зі старшими j розрядами DES перетворення попереднього блоку шифртекста:

$c_n = m_n \oplus (\text{DES}(c_{n-1}) >> (64-j))$.

В останньому випадку зберігається довжина повідомлення.

3.3.3 Режим CFB - "Зворотній зв'язок по шифру"

У режимі Cipher Feed Back (CFB) або "Зворотній зв'язок по шифру" розмір блоку може відрізнятися від 64 біт. Розмір блоку вихідного тексту може бути будь-яким, що не перевершує розміру блоку шифрування 64. Це дозволяє зашифрувати і відправити будь-яку кількість бітів, не чекаючи кінця блоку.

Файл, що підлягає шифруванню (розшифруванню), зчитується послідовними блоками довжиною k біт ($k = 1, ..., 64$). Нехай в результаті розбиття на блоки ми отримали n блоків по k бітів кожен (залишок дописується нулями або пропусками). Тоді блоки шифртекста виходять у вигляді:

$c_i = m_i \oplus p_{i-1}, \ \forall i = 1, ..., n$

де p_{i-1} позначає k старших бітів попереднього зашифрованого блоку.

Вхідний блок DES шифрування (64-бітовий регістр зрушення) спочатку містить вектор ініціалізації, вирівняний по правому краю. Оновлення зрушеного регістру здійснюється шляхом видалення його старших k бітів і записів c_i в регістр (рис.3.4).

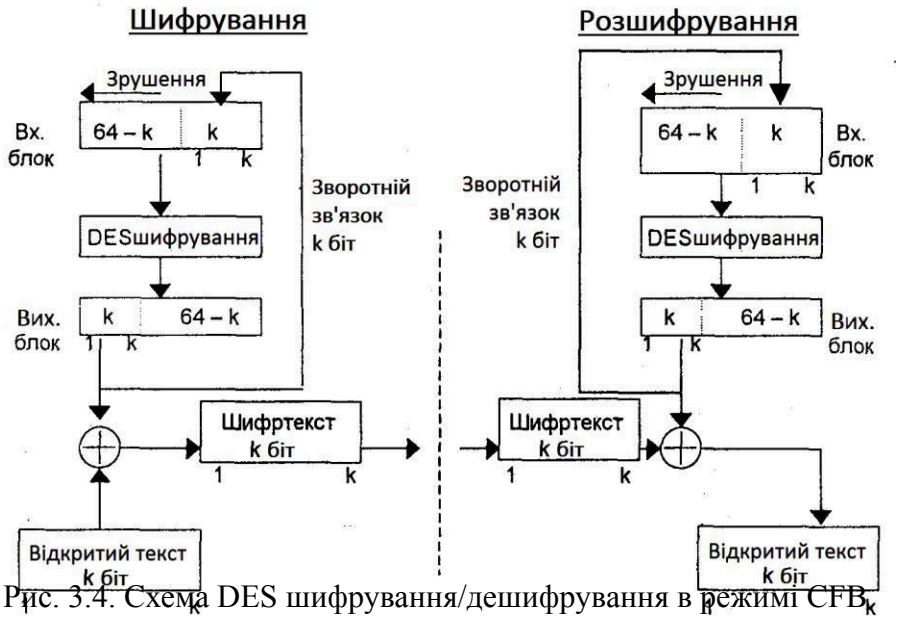


Рис. 3.4. Схема DES шифрування/дешифрування в режимі CFB_k

Відновлення зашифрованих даних також виконується відносно просто: p_{i-1} та c_i, обчислюється аналогічним чином:

$m_i = c_i \oplus p_{i-1}$

Режим також має властивість відновлення після зміни окремих бітів в шифрограмі. При розшифровці використовується функція шифрування, а не функція дешифрування блочного шифру.

Можливості зміни відкритого тексту ті ж, що в режимі CBC. Швидкість шифрування з повноблоковим відкритим зв'язком те ж, що і блочного шифру.

3.3.4 Режим OFB - "Зворотній зв'язок по виходу"

Режим Output Feed Back (OFB) або "Зворотній зв'язок по виходу" схожий на режим CFB. Відмінність полягає в методі поновлення зрушеного регістру. Воно здійснюється шляхом відкидання старших k бітів вхідного блоку і дописування справа k старших бітів вихідного блоку DES-шифрування на попередньому кроці.

Вхідний блок спочатку також містить вектор ініціалізації c_0, вирівняний по правому краю. При цьому для кожного сеансу шифрування даних необхідно використовувати новий початковий стан регістра, який може пересилатися по каналу відкритим текстом. Або породжується псевдовипадкова послідовність і побітно складається з повідомленням.

Розіб'ємо вхідну послідовність m на блоки довжиною k біт: $m = m_1 m_2 \dots m_n$. Тоді блоки шифртекста:

$c_i = m_i \oplus p_{i-1} \forall i = 1 \dots n$

де p_{i-1}- старші k бітів результату операції DES (c_{i-1}).

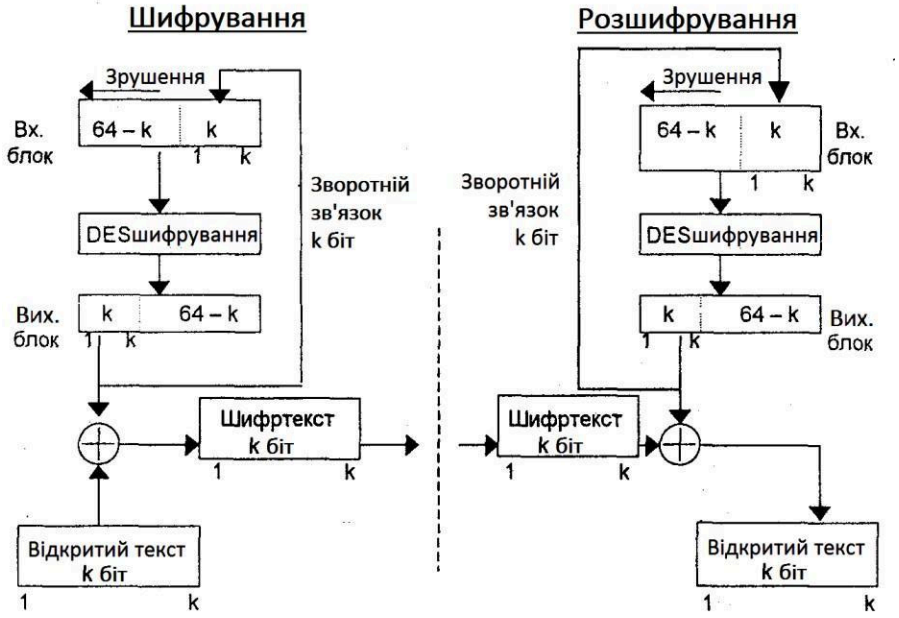


Рис. 3.5. Схема DES шифрування/дешифрування в режимі OFB

Режим OFB має наступну перевагу в порівнянні з CFB: помилки, що виникають при передачі по каналу з шумом, при дешифрування не «розмазуються» по всьому шифртексту, а локалізуються в межах одного блоку. Однак відкритий текст може бути змінений шляхом певних маніпуляцій з шифртекстом.

Для усунення відомих недоліків розроблені деякі удосконалення різних режимів, але вони не є стандартними.

3.3.5 Області застосування режимів алгоритму DES

Кожному з розглянутих режимів властиві свої переваги і недоліки, що обумовлює області їх застосування.

Режим ECB (Електронна кодова книга) добре підходить для шифрування ключів. Проте, не дивлячись на те, що цей режим найбільш вразливий, в більшості існуючих комерційних програм використовується саме він, а режим CBC використовується рідко, хоча він лише незначно більш складний і забезпечує більшу криптостійкість.

Режим CFB (Зворотній зв'язок по шифру), як правило, призначається для шифрування окремих символів.

Режим OFB (Зворотній зв'язок по виходу) нерідко застосовується для шифрування в супутникових системах зв'язку.

Режими CBC (Зчеплення блоків шифру) і CFB (Зворотній зв'язок по шифру) придатні для аутентифікації даних. Ці режими дозволяють використовувати алгоритм DES для:

- інтерактивного шифрування при обміні даними між терміналом і головною ЕВМ;
- шифрування криптографічного ключа в практиці автоматизованого поширення ключів;

- шифрування файлів, поштових відправлень, даних супутників та інших практичних завдань.

Хоча спочатку стандарт DES призначався для шифрування і розшифрування даних, його застосування було узагальнено і на аутентифікацію. Дані захищаються від ознайомлення шифруванням, а зміни виявляються за допомогою аутентифікації.

У системах автоматичної обробки даних людина не в змозі переглянути дані, щоб встановити, чи не внесені в них будь-які зміни. При величезних обсягах даних, що проходять в сучасних системах обробки, перегляд зайняв би надто багато часу. До того ж надмірність даних може виявитися недостатньою для виявлення помилок. Навіть в тих випадках, коли перегляд людиною можливий, дані можуть бути змінені таким чином, що виявити ці зміни людині дуже важко. Наприклад, "do" може бути замінено на "do not", "$1900" - на "$9100". Без додаткової інформації людина при перегляді може легко прийняти змінені дані за справжні. Такі ризики можуть існувати навіть при використанні шифрування даних. Тому бажано мати автоматичний засіб виявлення навмисних і ненавмисних змін даних.

Звичайні коди, що виявляють помилки, непридатні, тому що, якщо алгоритм утворення коду відомий, противник може виробити правильний код після внесення змін до даних. Однак за допомогою алгоритму DES можна утворити криптографічну контрольну суму, яка може захистити як від випадкових, так і навмисних, але несанкціонованих змін даних.

Цей процес описується в *стандарті для аутентифікації даних ЕВМ* (FIPS 113). Суть стандарту полягає в тому, що дані зашифровуються в режимі зворотного зв'язку по шифртексту (режим CFB) або в режимі зчеплення блоків шифру (режим CBC), в результаті чого виходить остаточний блок шифру, який представляє собою функцію всіх розрядів відкритого тексту. Після цього повідомлення може бути передано з використанням обчисленого остаточного блоку шифру, що служить як криптографічна контрольна сума.

Шифрування і аутентифікацію використовують для *захисту даних*, що *зберігаються в ЕВМ*. Наприклад, *паролі* зашифровують незворотнім чином і зберігають в пам'яті машини. Нерідко зашифрований пароль виробляють за допомогою алгоритму DES, причому ключ вважається рівним паролю, а відкритий текст - коду ідентифікації користувача.

Алгоритм аутентифікації можна застосувати як до відкритого, так і до зашифрованого тексту. При *фінансових операціях*, коли в більшості випадків реалізуються і шифрування, і аутентифікація, остання застосовується і до відкритого тексту.

Захист повідомлень *електронної системи платежів* (ЕСП) є одним з найбільш важливих застосувань алгоритму DES при операціях з широкою клієнтурою і між банками. Алгоритм DES реалізується в банківських автоматах, терміналах в торгових точках, автоматизованих робочих місцях і головних ЕВМ. Діапазон захищуваних їм даних досить широкий – сплата від $50 до перекладів на багато мільйонів доларів. Гнучкість основного алгоритму DES дозволяє використовувати його в найрізноманітніших областях застосування електронної системи платежів.

3.4 Алгоритм Магма (ДСТУ 28147:2009)

Стандарт країн СНД рекомендований до використання для захисту будь-яких даних, представлених у вигляді двійкового коду, хоча не виключаються й інші методи шифрування. Даний стандарт формувався з урахуванням світового досвіду, і зокрема, були прийняті до уваги недоліки і нереалізовані можливості алгоритму DES.

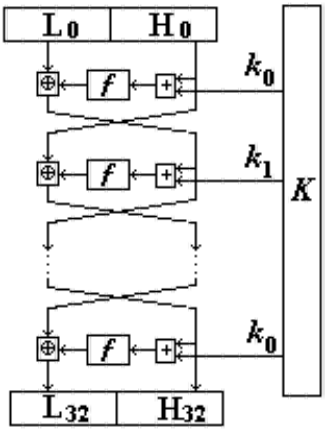


Рис. 3.6. Схема алгоритму ДСТУ 28147:2009

Алгоритм ДСТУ 28147:2009 - це ітеративний 32-цикловий оборотний блоковий шифр Фейстеля. Розмір вхідного блоку - 64 біта, розмір ключа шифру - 256 біт. Алгоритм досить складний і нижче буде описана в основному його концепція. Таблиця замін алгоритму ДСТУ не опублікована. Загальна схема алгоритму наведена Рис. 3.6.

У цій схемі: $\boxplus$ - складання по модулю 2^{32},

$\oplus$ - побітове додавання по модулю 2.

Для додавання ключа і складання підблоків в мережі Фейстеля використовуються операції + і $\oplus$. Ця пара операцій не має властивості асоціативності і дистрибутивності, що підвищує стійкість алгоритму до аналітичних атак, заснованим на апроксимації таблиці замін лінійними співвідношеннями.

Ключ k довжиною 256 біт представляється у вигляді восьми 32-розрядних чисел:

$k = k_{(0)} \| k_{(1)} \| \ldots \| k_{(7)}$

Алгоритм криптографічного перетворення передбачає 4 режими роботи:

- Шифрування даних в режимі простої заміни;
- Шифрування даних в режимі гамування;
- Шифрування даних в режимі гамування зі зворотним зв'язком;
- Вироблення імітовставки.

3.4.1 Режим простої заміни

Кожен 64-бітовий блок M шифруємого повідомлення розглядається як пара 32-бітових блоків M = H L (H - праві чи молодші і L - ліві чи старші біти). Перший біт блоку вважається старшим.

Далі виконується ітеративний (i - номер ітерації) процес шифрування (f - функція шифрування):

$H_i = f(H_{i-1} [+] k_{((i-1) \bmod 8)}) \oplus L_{i-1}$ для i=1, 2, ..., 24 $Li = H_{i-1}$

$H_i = f(H_{i-1} [+] k_{(32-i)}) \oplus L_{i-1}$ для i=25, 26, ..., 31 $Li = H_{i-1}$

$H_{32} = H_{31}$ $L_{32} = f(H_{31} [+] k_0) \oplus L_{31}$

64-розрядний блок зашифрованих даних C представляється у вигляді:

$C = H_{32} L_{32}$

Решта блоків відкритих даних в режимі простої заміни зашифровуються аналогічно.

Підключи k_i, використовувані в ітераціях алгоритму, є 8 блоків $k_{(0)}$, $k_{(1)}$, ..., $k_{(7)}$ по 32 біта, на які розбивається 256-бітовий секретний ключ K. Ключі повторюються циклічно 3 рази, а в четвертий раз вибираються в зворотному порядку.

3.4.2 Функція шифрування

Функція шифрування включає дві операції над 32-розрядним аргументом. Схема функції шифрування в алгоритмі ДСТУ 28147:2009 показана на Рис. 4.7.

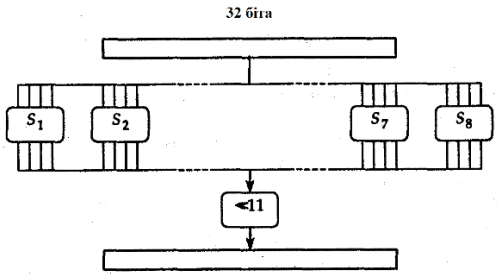


Рис. 3.7. Функція шифрування в алгоритмі ДСТУ 28147:2009

Перша операція є підстановкою. Блок підстановки складається з 8 вузлів заміни $S_{(1)}$ ... $S_{(8)}$ з пам'яттю 64 біта кожен. Вступник на блок підстановки 32-розрядний вектор розбивається на 8 послідовно йдуть 4-розрядних вектора, кожен з який перетворюється в 4-розрядний вектор відповідним вузлом заміни.

Кожен вузол заміни можна представити у вигляді таблиці-перестановки 16-и чотирьохрозрядних двійкових чисел в діапазоні 0 ... 15. Вхідний вектор визначає адресу рядка в таблиці, число з якої є вихідним вектором. Потім 4-розрядні вектори послідовно об'єднуються в 32-розрядний вихідний вектор.

Друга операція - циклічне зрушення вліво (на 11 розрядів) 32-розрядного вектора, отриманого в результаті підстановки.

Слід враховувати, що даний режим шифрування має обмежену криптостійкість. Режим простої заміни допустимо використовувати для шифрування даних тільки в обмежених випадках - при шифруванні ключа із забезпеченням імітозахисту для передачі по каналам зв'язку або зберігання його в пам'яті ЕВМ.

3.4.3 Режим гамування

Режим гамування нагадує режим OFB, також будучи варіантом потокового шифру.

Відкриті дані, розбиті на 64-розрядні блоки m_i (i = 1,2, ..., n) (n визначається обсягом шифрованих даних), зашифровуються в режимі гамування шляхом порозрядного додавання за модулем 2 з гамою шифру $\blacksquare_ш=(\blacksquare_1, \blacksquare_2,...., \blacksquare_m)$, яка виробляється блоками по 64 біта. c_i (64-розрядний блок зашифрованого тексту) виходить з рівняння

$c_i = m_i \oplus \blacksquare_i$

де гамма шифру: $\blacksquare_i = F(y_{i-1}, z_{i-1})$.

Тут F позначає функцію шифрування в режимі простої заміни (аргументами цієї функції є два 32-розрядних числа).

Величини y_i і z_i визначаються ітераційно по мірі формування гами наступним чином:

$(y_0, z_0)=F(s)$

$(y_i , z_i)=(y_{i-1}+ a_2, z_{i-1}+ a_1)$, i=1, 2, ..., n

де s - початкове значення (64-розрядна двійкова послідовність, звана синхропосилкою), не є секретним елементом шифру. Вона відома як на передавальній стороні, так і на приймальні або передається по каналу зв'язку разом з шифртекстом.

$a_1 = 01010101_{16}$ і $a_2 = 01010104_{16}$ - 32-розрядні двійкові константи, задані в ДСТУ 28147:2009.

3.4.4 Режим гамування зі зворотним зв'язком

Цей режим дуже схожий на режим гамування. Він в точності збігається з найпростішим варіантом режиму CFB.

Як і в режимі гамування, відкриті дані, розбиті на 64-розрядні блоки m_i, зашифровуються шляхом порозрядного додавання за модулем 2 з гамою шифру ■ш, яка виробляється блоками по 64 біт:

■ш=(■(1), ■(2), ..., ■(*m*))

Рівняння шифрування даних в режимі гамування зі зворотним зв'язком виглядають наступним чином:

c_1=F(s) ⊕ m_1=■1 ⊕ m_1, s - синхропосилка,

c_i = F(m_{i-1}) ⊕ m_i = ■$_i$ ⊕ m_i, i=2, 3, ..., n.

У цьому режимі для вироблення гами шифру використовується попередній блок відкритого тексту. Він розбивається на дві частини, які служать аргументами функції F.

Якщо довжина останнього відкритого блоку даних менше 64 розрядів, то інші розряди гами шифру просто відкидаються. Таким чином, можна шифрувати блоки будь-якої довжини в тому числі і в потоковому режимі.

3.4.5 Вироблення імітовставки

У ДСТУ 28147:2009 визначається процес вироблення імітовставки, який одноманітний для всіх режимів шифрування.

Імітовставка - це блок з p біт (імітовставка I_p), який виробляється за певним правилом з відкритих даних з використанням ключа і потім додається до зашифрованих даних для забезпечення їх імітозахисту, тобто захисту від їх *нав'язування помилкових даних*.

Імітовставка виробляється або перед шифруванням усього повідомлення, або паралельно з шифруванням по блокам. Параметр р вибирається відповідно з необхідним рівнем імітозахищеності з урахуванням того, що ймовірність нав'язування помилкових перешкод дорівнює $1/2^p$.

Алгоритм. Для отримання імітовставки, відкриті дані представляються також у вигляді блоків по 64 біт. Перший блок відкритих даних m_1 піддається перетворенню, що відповідає першим 16 циклам алгоритму режиму простої заміни. Причому в якості ключа використовується той же ключ, що і для шифрування даних.

Отримане 64-розрядне число підсумовується по модулю 2 з другим блоком відкритих даних m_2, і сума знову піддається 16 циклам шифрування для режиму простої заміни.

Дана процедура повторяться для всіх m блоків повідомлення. З отриманого 64-розрядного числа вибирається відрізок I_p довжиною p біт з номерами бітів з (32-p + 1) -го по 32-й в такий спосіб:

32-p біт	p біт	32 біта

Імітовставка I_p передається по каналу зв'язку після зашифрованих даних:

c_1, ..., c_m, I_p

На приймальній стороні після розшифрування отриманого повідомлення повторюється процедура отримання імітовставки на основі розшифрованого тексту.

Отримана імітовставка порівнюється з прийнятою в кінці повідомлення. Що стосується розбіжності імітовставок прийняте повідомлення вважається помилковим.

3.4.6 Крипостійкість ДСТУ 28147:2009. Порівняння алгоритмів DES і ДСТУ 28147:2009

Розробники ДСТУ 28147:2009у намагалися досягти рівноваги між криптостійкістю і ефективністю. Взявши за основу конструкцію Фейстеля, вони розробили криптоалгоритм, який краще, ніж DES, підходить для програмної реалізації. Для підвищення криптостійкості, введений наддовгий ключ і подвоєно кількість циклів.

Головні відмінності між DES і ДСТУ 28147:2009 полягають в наступному:

- В DES 56-бітний ключ, а в ДСТУ 28147:2009 - 256-бітний. Якщо додати секретні перестановки S- блоків, то повний обсяг секретної інформації ДСТУ 28147:2009у складе приблизно 610 біт;
- В DES 16 циклів, а в ДСТУ 28147:2009 - 32.
- DES використовує складну процедуру для генерації підключів з вихідного ключа, в ДСТУ 28147:2009 ця процедура дуже проста;
- У S-блоків DES 6-бітові входи і 4-бітові виходи (таблиці 4x16 = 64), а у S-блоків ДСТУ 28147:2009у 4- бітові входи і виходи (таблиці 4x4 = 16). В обох алгоритмах використовується по вісім S- блоків, але розмір S-блоку ДСТУ 28147:2009у дорівнює чверті розміру S-блоку DES;
- В DES використовуються нерегулярні початкові і кінцеві перестановки, а в ДСТУ 28147:2009 використовується 11-бітне циклічне зрушення вліво.

Силова атака на ДСТУ 28147:2009 абсолютно безперспективна. ДСТУ 28147:2009 використовує 256- бітовий ключ, а якщо враховувати секретні S-блоки, то довжина ключа буде ще більшою.

Крім того, ДСТУ 28147:2009, мабуть, більш стійкий до диференціального і лінійного криптоаналізу, ніж DES. Хоча випадкові S-блоки ДСТУ 28147:2009 при деякому виборі не гарантують високої криптостійкості в порівнянні з фіксованими S-блоками DES, їх секретність збільшує стійкість ДСТУ 28147:2009 до диференціального і лінійного криптоаналізу. До того ж ефективність цих криптоаналітичних методів залежить від кількості циклів перетворення - чим більше циклів, тим важче криптоаналіз. ДСТУ 28147:2009 використовує в два рази більше циклів, ніж DES, що, можливо, призводить до неспроможності диференціального і лінійного криптоаналізу.

ДСТУ не використовує існуючу в DES перестановку з розширенням. Видалення цієї перестановки з DES послаблює його через зменшення лавинного ефекту; розумно припустити, що відсутність такої операції в ДСТУ 28147:2009 негативно позначається на його криптостійкості. З точки зору криптостійкості операція арифметичного додавання, яка використовується в ДСТУ 28147:2009, не гірше, ніж операція XO*R* в DES.

Основною відмінністю є використання в ДСТУ 28147:2009 циклічного зрушення замість перестановки. Перестановка DES збільшує лавинний ефект, що складається в поширенні впливу одного біта відкритого тексту на багато бітів шифртекста. У ДСТУ 28147:2009 зміна одного вхідного біта впливає на один S-блок одного циклу перетворення, який потім впливає на два S-блоків наступного циклу, потім на три блоки наступного циклу і т.п. Буде потрібно вісім циклів, перш ніж зміна одного вхідного біта вплине на кожен біт результату; в DES для цього потрібно тільки п'ять циклів. Однак ДСТУ 28147:2009 складається з 32 циклів, а DES з 16.

3.5 Алгоритм IDEA

Алгоритм шифрування IDEA (International Data Encryption Algorithm) розроблений в 1989 р. в Швейцарії, в інституті ETH Zurich. Алгоритм запатентований в Європі (Швейцарія) і США. Тримач патенту - фірма Ascom-Tech AG. Він використовується, зокрема, у відомій програмі pgp.

Блок-схема алгоритму IDEA зображена на рис. 3.8. Він являє собою варіант 64-бітного ітеративного блочного шифру з 128-бітовим ключем і вісьмома циклами криптографічного перетворення. IDEA не відповідає схемі Фейстеля, однак процедура дешифрування виконується аналогічно процедурі шифрування.

Структура криптоалгоритма допускає просту програмну і апаратну реалізацію. Секретність перетворення, що забезпечує ефекти розсіювання і перемішування, досягається за рахунок трьох різних типів арифметичних операцій: над 16-бітними словами.

- Додавання по модулю 2, на схемі позначено $\oplus$;
- Додавання по модулю 2^{16}, на схемі позначено +, далі +;
- Множення по модулю (2^{16}-1), на схемі позначено - $\otimes$.

Таким чином, циклове відображення $f(x, k)$ побудовано з використанням «змішування» операцій над різними алгебраїчними структурами, що ускладнює криптоаналіз алгоритму.

Всі операції виконуються над 16-бітовими підблоками, тобто кожен 64-бітовий блок розбивається на 4 підблока. Вхідні підблоки першого циклу позначимо через x_1, x_2, x_3, x_4, підблоки вихідного відображення (шифртекста) позначимо через y_1, y_2, y_3, y_4, ключові підблоки i-го циклу позначимо через $k_1^{(i)}$, $k_2^{(i)}$, $k_3^{(i)}$, …,$k_6^{(i)}$, i = 1,2, ..., 8, ключові підблоки вихідного відображення $k_1^{(9)}$, $k_2^{(9)}$, $k_3^{(9)}$, $k_4^{(9)}$.

Опишемо перший цикл роботи алгоритму (інші цикли реалізуються аналогічно). Протягом першого циклу шифрування реалізуються наступні кроки обчислень (для простоти замість позначення $k_1^{(i)}$, будемо використовувати k_i, i = 1,2, ..., 6).

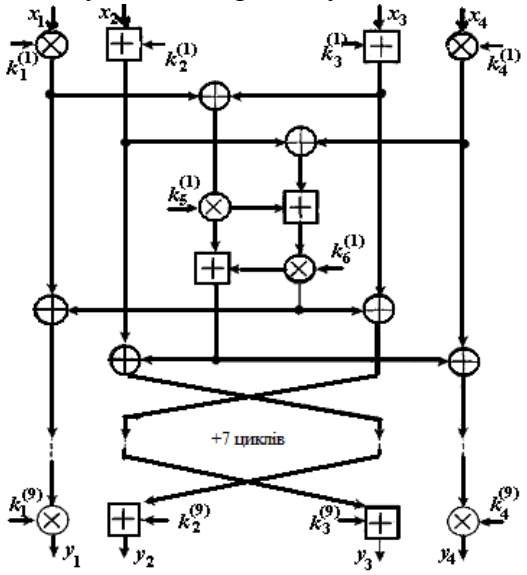


Рис. 3.8. Блок-схема алгоритму шифрування IDEA

Крок 1. $c_1 = x_1 \otimes k_1$ Крок 2. $c_2 = x_2 + k_2$ Крок 3. $c_3 = x_3 + k_3$

Крок 4. $c_4 = x_4 \otimes k_4$ Крок 5. $c_5 = c_1 \oplus c_3$ Крок 6. $c_6 = c_2 \oplus c_4$

Крок 7. $c_7 = c_5 \otimes k_5$ Крок 8. $c_8 = c_6 + c_7$ Крок 9. $c_9 = c_8 \otimes k_6$

Крок 10. $c_{10} = c_9 + c_7$ Крок 11. $c_{11} = c_1 \oplus c_9$ Крок 12. $c_{12} = c_3 \oplus c_9$

Крок 13. $c_{13} = c_2 \oplus c_{10}$ Крок 14. $c_{14} = c_4 \oplus c_{10}$

Вхідні підблоки наступного циклу є c_{11}, c_{13}, c_{12}, c_{14}. Після 8-го циклу реалізується вихідне відображення:

Крок 1. $y_1 = x_1^{(9)} \otimes k_1^{(9)}$ Крок 2. $y_2 = x_2^{(9)} + k_2^{(9)}$

Крок 3. $y_3 = x_3^{(9)} + k_3^{(9)}$ Крок 4. $y_4 = x_4^{(9)} \otimes k_4^{(9)}$

Шифрованим текстом є блок (y_1, y_2, y_3, y_4).

Ключовий розклад реалізується за допомогою обчислення 52 ключових підблоків з 128-бітового ключа K, на кожному циклі використовується по 6 підблоків ключа і при реалізації вихідного відображення - 4 підблока ключа. Спочатку 128-бітовий ключ K представляється як 8 підблоків, які в наших позначеннях рівні:

$k_1^{(1)}$, $k_2^{(1)}$, $k_3^{(1)}$, $k_4^{(1)}$, $k_5^{(1)}$, $k_6^{(1)}$, $k_1^{(2)}$, $k_2^{(2)}$

де старшими бітами підблоків є ліві біти. Далі 128-бітовий ключ циклічно зрушується вліво на 25 біт і знову розбивається на 8 підблоків:

$k3^{(2)}$, $k4^{(2)}$, $k5^{(2)}$, $k6^{(2)}$, $k1^{(3)}$, $k2^{(3)}$, $k3^{(3)}$, $k4^{(3)}$.

Потім 128-бітовий ключ знову циклічно зрушується вліво на 25 біт і знову розбивається на 8 підблоків і т. п., поки не згенерує всі 52 підблока.Підблоки ключа розшифрування (позначимо їх через $q_i^{(r)}$) визначаються наступним чином: перші чотири

- При $r = 2,3, ..., 8$: $(q_1^{(r)}, q_2^{(r)}, q_3^{(r)}, q_4^{(r)}) = ([k_1^{(10-r)}]^{-1}, -k_2^{(10-r)}, -k_3^{(10-r)}, [k_4^{(10-r)}]^{-1})$,
- При $r = 1,9$: $(q_1^{(r)}, q_2^{(r)}, q_3^{(r)}, q_4^{(r)}) = ([k_1^{(10-r)}]^{-1}, -k_2^{(10-r)}, -k_3^{(10-r)}, [k_4^{(10-r)}]^{-1})$,
  і два останніх $(q_5^{(r)}, q_6^{(r)}) = (k_5^{(r)}, k_6^{(r)})$, $r = 1,2, ..., 8$;

де z^{-1} є зворотній до z елемент щодо множення по модулю $2^{16} + 1$ (зворотним до 0 вважаємо 0); і $-z$ є зворотній до z елемент щодо складання по модулю 2^{16}.

Алгоритм IDEA може працювати в будь-якому режимі блокового шифру, передбаченому для алгоритму DES. Ефективність програмної реалізації IDEA не поступається DES. Продуктивність алгоритму IDEA перевищує в 1,5-4 рази (в залежності від обчислювача) продуктивність DES-алгоритму.

Алгоритм IDEA безпечніше алгоритму DES, оскільки, по-перше, має ключ вдвічі довші. По-друге, внутрішня структура алгоритму IDEA забезпечує кращу, ніж у DES стійкість до криптоаналізу.

Специфічна особливість IDEA - порівняльна простота аналізу криптостійкості алгоритму по відношенню до диференціального криптоаналізу. Результати успішного лінійного криптоаналізу також невідомі. Методи аналізу алгоритму IDEA, менш трудомісткі, ніж повний перебір ключів, не відомі. В ході дослідження криптостійкості IDEA було встановлено, що існує підмножина з 2^{51} «слабких» ключів, які можуть бути розкриті шляхом аналізу процедури шифрування на конкретному ключі з даної підмножини. Результат, однак, не робить значного впливу на практичну криптостійкість, так як потужність підмножини «слабких» ключів мала в порівнянні з потужністю всього ключового простору (2^{128} різних ключів).

3.6 Атаки на блокові шифри

3.6.1 Диференціальний криптоаналіз

Метод диференціального криптоаналізу вперше був застосований для атаки на блоковий шифр FEAL-4. Згодом метод був удосконалений і успішно застосований для криптоаналізу DES. Метод диференціального криптоаналізу дозволив отримати відповідь на питання про кількість циклів криптографічного перетворення стандарту DES. Ця відповідь важлива з точки зору розробки ефективних методів криптоаналізу довільних ітерованих блокових шифрів конструкції Фейстеля.

Питання формулювалося наступним чином: чому число циклів перетворення рівно шістнадцяти, не більше і не менше? Відомо, що після п'яти циклів кожен біт шифртекста є функцією всіх бітів відкритого тексту і ключа. Після восьми циклів спостерігається пік лавинного ефекту - шифртекст є випадковою функцію всіх бітів відкритого тексту і ключа.

Успішні атаки на DES з трьома, чотирма і шістьма циклами підтвердили результати досліджень лавинного ефекту. На перший погляд, криптографічне перетворення з великим числом циклів (>8) видається необґрунтованим з точки зору ефективності реалізації. Однак успішний диференційний криптоаналіз DES довів: трудомісткість (обсяг перебору) атаки на відкритому тексті для DES з будь-якою кількістю циклів, меншим шістнадцяти, нижче, ніж трудомісткість силової атаки. При цьому необхідно зазначити, що ймовірність

успіху при силовій атаці вище, ніж при атаці методом диференціального криптоаналізу. Той факт, що метод диференціального криптоаналізу був відомий в момент розробки DES, але засекречений з очевидних міркувань, підтверджений публічними заявами самих розробників.

Ядро методу складає атака на основі вибіркового відкритого тексту. Ідея полягає в аналізі процесу зміни відмінності для пари відкритих текстів, що мають певні вихідні відмінності, в процесі проходження через цикли шифрування з одним і тим же ключем. Не існує будь-яких обмежень на вибір відкритих текстів. Досить відмінностей і в деяких позиціях. В якості міри відмінності, як правило, використовують відстань Хеммінга, яке дорівнює кількості розрізнюючих бітів.

Виходячи з відмінностей в одержуваних шифртекстах, ключам присвоюються різні ймовірності. Істинний ключ визначається в процесі подальшого аналізу пар шифртекстів - це найбільш ймовірний ключ серед множини претендентів.

Найкращий відомий результат успішного диференціального криптоаналізу DES з шістнадцятьма циклами вимагає 2^{47} вибірковий відкритий текст. Можна скористатися атакою на відомому відкритому тексті, але для цього буде потрібно вже 2^{55} відомих зразків.

Трудомісткість атаки - 2^{37} DES-шифрування. Диференціальний криптоаналіз ефективний проти DES і аналогічних алгоритмів з постійними S-блоками. Трудомісткість атаки залежить від структури S-блоків. Для всіх режимів шифрування - ECB, CBC, CPB і OFB - атака має однакову трудомісткість.

Крипостійкість DES може бути підвищена шляхом збільшення числа циклів. Трудомісткість диференціального криптоаналізу DES с сімнадцятьма або вісімнадцятьма циклами еквівалентна трудомісткості силової атаки. При дев'ятнадцяти і більш циклах, диференційний криптоаналіз неможливий в принципі - для його реалізації буде потрібно більше 2^{64} вибіркових відкритих текстів (DES оперує блоками по 64 біта, отже, існує 2^{64} різних відкритих текстів).

У загальному випадку доказ криптостійкості по відношенню до атаки методом диференціального криптоаналізу полягає в оцінці числа відкритих текстів, необхідних для виконання атаки. Атака неможлива, якщо отримана оцінка перевищує 2^b, де b - розрядність блоку в бітах.

3.6.2 Лінійний криптоаналіз

Метод лінійного криптоаналізу вперше був застосований для атаки блокового шифра FEAL і потім DES. Даний метод використовує лінійні наближення. Це означає, що, якщо виконати операцію $\oplus$ над деякими бітами відкритого тексту, потім над деякими бітами шифртекста, а потім виконати $\oplus$ результатів попереднього підсумовування, можна отримати біт, що дорівнює сумі деяких біт ключа. Це називається лінійним наближенням, яке може бути вірним з певною ймовірністю p. Якщо $p \neq 1/2$, то цей факт можна використовувати для розкриття біт ключа.

3.6.3 Силова атака на основі розподілених обчислень

Для вирішення ряду завдань практичної криптографії необхідні значні обчислювальні ресурси. При цьому деякі з цих завдань піддаються розпаралелюванню. Таким чином, інтернет, який об'єднує багато тисяч робочих станцій, дозволяє ефективно вирішувати трудомісткі завдання шляхом координованої та одночасної роботи великої кількості комп'ютерів. Такий підхід, який реалізує можливості комп'ютерних мереж, відомий під назвою *метод розподілених обчислень*.

Існує два підходи до організації силової атаки на основі методу розподілених обчислень. Перший - централізоване управління процесом пошуку за допомогою сервера. Другий - випадковий і незалежний вибір стартової точки в ключовому просторі, кожним з тими, хто бере участь в проекті комп'ютером і оповіщення в разі розкриття ключа.

Перший підхід більш кращий, але пов'язаний з певними труднощами. Так, не виключена можливість перевантаження сервера потоком запитів з боку клієнтів. До того ж деякі ненавмисні дії клієнтів можуть привести до збоїв в роботі центрального сервера. Не виключений також ризик зловмисного деструктивного впливу. Для зведення до

мінімуму зазначених ризиків необхідні додаткові заходи.

Відомий метод забезпечення відмовостійкості передбачає введення додаткової надмірності. Реплікація серверів, а також принцип ієрархії в проектуванні структури зв'язків дозволяють усунути деякі ризики і обмежити наслідки збоїв і відмов. Застосування коду аутентифікації гарантує справжність всіх повідомлень при передачі як від клієнта до сервера, так і в зворотному напрямку. Ведення реєстраційного журналу спрощує аналіз і відстеження наслідків відмов.

Розвитку подібної технології сприяла низка конкурсів, оголошених відомими фірмами - розробниками криптосистем, зокрема компанією RSA Data Security, з призовою сумою до 10 тис. Доларів. Розроблено спеціальне програмне забезпечення на основі технології «клієнт-сервер». Центральний сервер виконує реєстрацію клієнта і видає інтервал ключів для перебору. Секретний ключ вважається розкритим, якщо результат дешифрування на ньому призводить до змістовного тексту англійською мовою.

Лекція №4 СУЧАСНІ СИМЕТРИЧНІ КРИПТОСИСТЕМИ (AES, RIJNDAEL)

4.1 Шифри зі змінною довжиною ключа

Зі збільшенням потужності обчислювальної техніки алгоритми шифрування застарівають в міру того, як стає можливим повний перебір простору секретних ключів. При цьому доводиться шукати нові алгоритми і вибирати нові стандарти.

Звичайним рішенням в цій ситуації є вибір алгоритму шифрування зі змінною довжиною ключа, щоб при збільшенні можливостей перебору можна було збільшувати її, не змінюючи самого алгоритму. При цьому бажано, щоб:

- складність (кількість виконуваних операцій) алгоритмів шифрування $E_k(m)$ і дешифрування $D_k(m)$ була *поліномною*:
  $O(|k|^m)$, де $|k|$ - довжина ключа k в бітах, а $m > 0$ - константа,
- в той час як складність алгоритму, що реалізує будь-яку атаку, була б не менше складності повного перебору ключів, тобто *експоненційною* - $O(a^{|k|})$, де $a > 1$ - константа.

Нехай час, який можна витратити на шифрування або дешифрування, так само t_1, а час, який зловмисник може дозволити собі витратити на атаку, так само t_2. При швидкості обчислень, що дорівнює v операцій в одиницю часу, отримуємо такі обмеження на довжину ключа:

$$\log_a c_2^{-1} v t_2 < |k| < (c_1^{-1} v t_1)^{1/m}$$

Тут c_1 і c_2 - константи, що визначають складність даного алгоритму.

Так як з ростом v верхня межа зростає швидше нижньої, при досить великих значеннях швидкості, обов'язково знайдуться відповідні довжини ключів.

4.1.1 Алгоритм RC2

Криптоалгоритм RC2 є блоковий шифр з ключем змінної довжини. Розроблений Рівестом на замовлення компанії RSA Data Security, Inc. Абревіатура «RC» означає «код Рональда» (Ron's Code), або «шифр Ривеста» (Rivest's Cipher).

Криптоалгоритм розроблявся як альтернатива криптостандарта DES. RC2 працює з блоками по 64 біта, програмна реалізація криптоалгоритма приблизно в два-три рази швидше, ніж DES.

Мінлива довжина ключа дозволяє домагатися адекватної криптостійкості з урахуванням можливостей силової атаки. Криптоалгоритм RC2 дозволяє шифрувати дані в різних режимах - ЄСВ, CBC, CFB.

Уряд США не забороняє експортувати апаратні і програмні реалізації криптоалгоритмів RC2 і RC4 (потоковий шифр, він буде розглянутий далі) - при дотриманні обмеження на довжину ключа (40 біт). Американські компанії за межами США і Канади можуть використовувати криптоалгоритми з довжиною ключа 65 біт. При

шифруванні по криптоалгоритму RC2 до секретного ключа методом конкатенації додається деякий допоміжний ключ від 40 до 88 біт. Для виконання дешифрування допоміжний ключ передається одержувачу зашифрованого повідомлення у відкритому вигляді.

4.1.2 Алгоритм RC5

Криптоалгоритм RC5 також розроблений Рівестом на замовлення компанії RSA Data Security, Inc. Це блоковий шифр зі змінними довжинами блоку, ключа і числом циклів криптографічного перетворення. Параметрами алгоритму є такі величини:

- Розмір слова в бітах $w \in \{32,64,128\}$. RC5 є блоковим шифром з розміром блоку $2w$ бітів.
- Кількість ітерацій $R \in (0, ..., 255\}$.
- Довжина ключа в байтах $b \in (0, ..., 2048\}$.

Обраний варіант алгоритму позначається RC5-*w*/*r*/*b*. Можливість параметризації дозволяє гнучко налаштовувати криптоалгоритм з урахуванням конкретних вимог щодо криптостійкості і ефективності реалізації. Не всі варіанти алгоритму стійкі, і для практичного використання рекомендуються RC5-32/12/16 або RC5-64/16/16.

Конструкція алгоритму RC5 є модифіковану мережу Фейстеля, в якій для додавання ключа і складання підблоків використовуються різні операції (як в алгоритмі ДСТУ 28147:2009), а замість таблиці замін використане параметризоване циклічне зрушення.

Криптоалгоритм RC5 складається з трьох основних процедур: розширення ключа, шифрування і дешифрування.

У процедурі розширення ключа заданий секретний ключ піддається спеціальному перетворенню з метою заповнення ключової таблиці, причому розмір таблиці залежить від числа циклів криптографічного перетворення. Ключова таблиця використовується потім для шифрування і дешифрування.

Процедура шифрування складається з трьох основних операцій: цілочисельного підсумовування, підсумовування по модулю 2 і циклічного зрушення. Безумовна перевага криптоалгоритма полягає в простоті реалізації. Непередбачуваність результату операції циклічного зрушення, що залежить від конкретних вхідних даних при шифруванні, забезпечує необхідний рівень криптостійкості.

Дослідження криптостійкості RC5 показали, що варіант криптоалгоритма з розрядністю блоку 64 біта і дванадцятьма (і більше) циклами перетворення, гарантує адекватну криптостійкість по відношенню до диференціального і лінійного криптоаналізу.

4.2 Потокові шифри

4.2.1 Шифрування великих повідомлень і потоків даних

Ця проблема з'явилася порівняно недавно з появою засобів мультимедіа та мереж з високою пропускною здатністю, що забезпечують передачу мультимедійних даних.

До сих пір йшлося про захист повідомлень. При цьому під ними малася на увазі скоріш деяка текстова або символічна інформація. Однак в сучасних ІС та інформаційних системах починають застосовуватися технології, які вимагають передачі значно великих обсягів даних. Серед таких технологій:

- факсимільний, відео і мовленнєвий зв'язок;
- голосова пошта;
- системи відеоконференцій.

Так як передача оцифрованої звукової, графічної і відеоінформації в багатьох випадках вимагає конфіденційності, то виникає проблема шифрування величезних інформаційних масивів. Для інтерактивних систем типу телеконференцій, ведення аудіо-або відеозв'язку таке шифрування має здійснюватися в реальному масштабі часу і, по можливості, бути "прозорим" для користувачів.

Найбільш поширеним є потокове шифрування даних. Якщо в описаних раніше криптосистемах передбачалося, що на вході є деяке кінцеве повідомлення, до якого і застосовується криптографічний алгоритм, то в системах з потоковим шифруванням принцип інший. Система захисту не чекає, коли закінчиться передане повідомлення, а відразу ж здійснює його шифрування і передачу.

Найбільш очевидним є побітове додавання вхідної послідовності (повідомлення) з деяким нескінченним або періодичним ключем, одержуваним, наприклад, від генератора ПСЧ. Частково цей метод схожий на гамування, але важливою відмінністю потокового шифрування є те, що шифруванню піддаються не символи повідомлення, а окремі біти. Потокові шифри перетворюють відкритий текст в шифртекст по одному біту за операцію.

Генератор потоку ключів (званий генератором з біжучим ключем) видає потік бітів: $k_1, k_2, k_3, \ldots, k_i$. Цей потік бітів (званий біжучим ключем) і потік бітів відкритого тексту $x_1, x_2, x_3, \ldots, x_i$ підсумовуються по модулю 2 (операція $\oplus$ або XOR- «виключне АБО»), і в результаті виходить потік бітів шифртекста:

$c_i = x_i \oplus k_i$.

При дешифруванні, для відновлення бітів відкритого тексту, операція $\oplus$ виконується над бітами шифртекста і тим же самим потоком ключів:

$x_i = c_i \oplus k_i$

Відзначимо, що для отримання ключа за формулою:

$k_i = c_i \oplus x_i$

досить, щоб довжина шифртекста і відкритого тексту була більше ключа. Отже, безпека системи повністю залежить від властивостей генератора потоку ключів. Якщо генератор потоку ключів видає нескінченний рядок нулів, шифртекст буде збігатися з відкритим текстом і перетворення буде безглуздим. Якщо генератор потоку ключів видає повторюваний 16-бітовий шаблон, криптостійкість буде зневажливо малою. У разі нескінченного потоку випадкових бітів криптостійкість поточного шифру буде еквівалентна криптостійкості одноразового блокнота. Режим OFB блочного шифру також є окремим випадком потокового шифру.

Генератор потоку ключів створює бітовий потік, який схожий на випадковий, але в дійсності детермінований і може бути безпомилково відтворений при дешифруванні. Ключем для псевдовипадкового генератора є його початковий стан.

Чим ближче вихід генератора потоку ключів до випадкового, тим вище трудомісткість криптоаналітичної атаки. Псевдовипадковий генератор називається криптографічно стійким, якщо стійким є потоковий шифр, отриманий з використанням даного генератора.

Блокові і потокові шифри реалізуються по-різному. Потокові шифри, шифрувальні та дешифрувальні дані по одному біту не дуже підходять для програмних реалізацій. Блокові шифри легше реалізовувати програмно, так як вони дозволяють уникнути трудомістких операцій з бітами і оперують зручними для комп'ютера блоками даних. У той же час потокові шифри більше підходять для апаратної реалізації, хоча існують і алгоритми поточного шифрування, орієнтовані на програмну реалізацію.

4.2.2 Потокові шифри, орієнтовані на програмну реалізацію. Алгоритм RC4

Прикладом стандартного потокового алгоритму є алгоритм RC4 - це потоковий шифр з перемінним розміром ключа. Шифр розроблений в 1987 р. Рівестом (R. Rivest) для RSA Data Security, Inc (там же і тим же автором, що і RC5).

Він працює в режимі OFB: потік ключів не залежить від відкритого тексту. Використовується S-блок розміром 8x8: S_0 S_1, ..., S_{255}. Елементи представляють собою перестановку чисел від 0 до 255, а перестановка є функцією ключа змінної довжини. В алгоритмі застосовуються два лічильника, i та j, з нульовими початковими значеннями. Для генерації випадкового байта виконуються наступні обчислення:

$i = (i + 1) \bmod 256$;

$j = (j + S_i) \bmod 256$.

Поміняти місцями S_i та S_j.

$t = (S_i + S_j) \bmod 256$; $k = S_t$.

К використовується в операції $\oplus$ з відкритим текстом для отримання шифртекста або в операції $\oplus$ з шифротекста для отримання відкритого тексту. Шифрування виконується приблизно в 10 разів швидше, ніж в DES. Так само нескладна і ініціалізація S-блоку. Спочатку S-блок заповнюється за правилом: $S_0=0$, $S_1=1$,…., $S_{255}=255$. Після цього ключ записується в масив: k_0, k_1 ,. .., k_{255}. Потім при початковому значенні $j = 0$ в циклі виконуються наступні обчислення:

for $i = 0$ to 255;

$j = (j + S_t + k_i) \bmod 256$.

Поміняти місцями S_i та S_j.

Компанія RSA Data Security, Inc. стверджує, що алгоритм стійкий до диференціального і лінійного криптоаналізу і що він у високому ступені не лінійний. S-блок повільно змінюється при використанні: i та j забезпечують виконання довільного збільшення кожного елемента.

RC4 входить в десятки комерційних продуктів, включаючи Lotus Notes, AOCE компанії Apple Computer і Oracle Secure SQL. Цей алгоритм також є частиною специфікації стандарту стільникової цифрової пакетної передачі даних - CDPD (Cellular Digital Packet Data).

4.2.3 Алгоритм SEAL

SEAL - це ефективний потоковий шифр, розроблений в IBM Рогевеєм (P. Rogaway) і Копперсмітом (D. Coppersmith). Алгоритм оптимізований для 32-бітових процесорів. Для нормальної роботи йому потрібно вісім 32-бітових регістрів і пам'ять на кілька кілобайт.

У SEAL передбачений ряд попередніх дій з ключем зі збереженням результатів в кількох таблицях. Таблиці використовуються для прискорення процедур шифрування і дешифрування.

Особливість SEAL полягає в тому, що він насправді є не традиційним потоковим шифром, а являє собою сімейство псевдовипадкових функцій. При 160-бітовому ключі k і 32-бітовому регістрі n SEAL розтягує n в L-бітову рядок $k(n)$. L може приймати будь-яке значення, менше 64 Кбайт.

SEAL використовує наступне правило: якщо k вибирається випадковим чином, то $k(n)$ має бути не відрізняються від випадкової L-бітової функції n.

Практичний ефект того, що SEAL є сімейством псевдовипадкових функцій, полягає в тому, що він зручний в ряді програм, де непридатні традиційні потокові шифри.

При використанні більшості поточних шифрів створюється односпрямована послідовність біт: єдиним способом визначити i-й біт (знаючи ключ та позицію i) є генерування всіх бітів аж до i-го. Відмінність сімейства псевдовипадкових функцій полягає в тому. що можна легко отримати доступ до будь-якої позиції ключової послідовності.

Наприклад, для шифрування жорсткого диска, що складається з множини 512-байтових секторів, можна скористатися сімейством псевдовипадкових функцій, подібних SEAL, і виконати XOR кожного сектора з $k(n)$. Це те ж саме, як якщо б була виконана операція XOR всього диска з довгою псевдовипадковою функцією, причому будь-яка частина цієї довгої послідовності біт може бути обчислена незалежно.

Сімейство псевдовипадкових функцій також спрощує проблему синхронізації, котра трапляється нам в стандартних поточних шифрах, - можна зашифрувати x_n (n-є передане повідомлення) на ключі k, виконавши XOR x_n і $k(n)$. Одержувачу не потрібно зберігати стан шифру для відновлення x_n, йому не доводиться турбуватися і про втрачені повідомленнях, що впливають на процес дешифрування.

4.2.4 Потокові шифри, засновані на регістрах зрушення зі зворотним

зв'язком

Більшість реальних поточних шифрів засноване на регістрах зрушення зі *зворотним зв'язком* (рис. 4.1). Регістр зрушення застосовують для генерації ключової послідовності.

Регістр зрушення зі зворотним зв'язком складається з двох частин: регістра зрушення і функції зворотного зв'язку. Регістр зрушення являє собою послідовність бітів. (Кількість бітів визначається довжиною зрушеного регістру. Якщо довжина дорівнює *n* бітам, то регістр називається *n*-бітовим регістром зрушення).

Всякий раз, коли потрібно витягти біт, всі біти регістра зрушуються вправо на 1 позицію. Новий крайній лівий біт, отриманий від функції зворотного зв'язку, є функцією всіх інших бітів регістра. На виході регістра виявляється витіснений молодший значущий біт.

Рис. 4.1

Періодом регістра називається довжина одержуваної послідовності до початку її повторення.

Найпростішим видом регістра зрушення зі зворотним зв'язком є *регістр зрушення з лінійним зворотним зв'язком* (РЗЛЗЗ), (рис 4.2.)

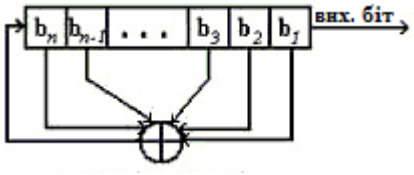


Рис. 4.2. РЗЛЗЗ

Зворотній зв'язок є $\oplus$ деяких (не всіх) бітів регістра; ці біти називаються *відвідною послідовністю*. Задати послідовність біт, що входять в зворотний зв'язок, можна за допомогою коефіцієнтів, рівних 0 або 1 для кожного біта.

Розглянемо приклад 4-бітного РЗЛЗЗ з відведенням від першого і четвертого бітів (рис. 4.3). Якщо його проініціалізувати значенням 1111, то до повторення регістр буде приймати наступний внутрішній стан:

1111 0111 1011 0101 1010 1101 0110 0011 1001 0100 0010 0001 1000 1100 1110.

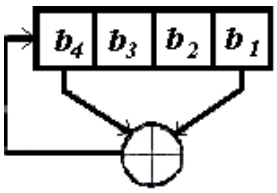


Рис. 4.3

Вихідною послідовністю буде рядок молодших значущих бітів:
1111 0101 1001 000 ...

РЗЛЗЗ (n-бітовий) може перебувати в одному з 2n-1 внутрішніх станів (значень бітів). Це означає, що теоретично такий регістр може генерувати псевдовипадкову послідовність з періодом 2n-1 бітів. Число внутрішніх станів і період рівні 2n-1, тому що заповнення РЗЛЗЗ нулями призведе до того, що зрушений регістр видаватиме нескінченну послідовність нулів, що абсолютно марно. Тільки за певних відвідних послідовностях РЗЛЗЗ циклічно пройде через всі внутрішні стани. Такі РЗЛЗЗ мають максимальний період. Одержаний результат називається М-послідовністю.

Формально *n*-бітовий РЗЛЗЗ можна описати многочленом ступеня *n* від формальної змінної *x*, де *i*-му біту відповідає член x^i з коефіцієнтом 0, якщо біт входить в відвідну послідовність, або з коефіцієнтом 1, якщо не входить. Вільний член многочлена завжди дорівнює 1. У нашому прикладі це буде:

x^4+x+1

Ступеня формальної змінної многочлена, за винятком 0-го, задають відвідну послідовність,

відлічувану від лівого краю зрушеного регістру. Тобто члени многочлена з меншим ступенем відповідають позиціям, розташованим ближче до правого краю регістра. Нульовий біт, якому в многочлені відповідає $x^0 = 1$, завжди входить в відвідну послідовність. Це гарантує, що всі біти регістра не стануть нульовими одночасно.

Для того щоб конкретний РЗЛЗЗ мав максимальний період, многочлен, асоційований з відвідної послідовністю, повинен бути примітивний по модулю 2, тобто не розкладатися на добуток двійкових многочленів меншого ступеня.

Наприклад, многочлен $x^{32} + x^7 + x^5 + x^3 + x^2 + x + 1$ примітивний по модулю 2. Розглянемо цей многочлен в термінах РЗЛЗЗ з максимальним періодом. Ступінь многочлена задає довжину РЗЛЗЗ.

Тоді для взятого 32-бітового зрушеного регістру новий біт генерується за допомогою $\oplus$ тридцять другого, сьомого, п'ятого, третього, другого і першого бітів (рис. 4.4); виходить РЗЛЗЗ матиме максимальну довжину, циклічно проходячи до повторення через 2^{32}-1 різних значень.

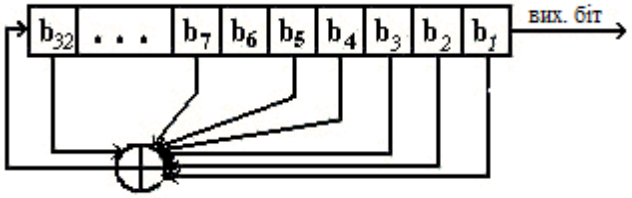


Рис. 4.4. 32-бітовий РСЛОС з максимальним періодом

Самі по собі РЗЛЗЗ є хорошими генераторами псевдовипадкових послідовностей, але вони мають деякі небажані невипадкові властивості. Для РЗЛЗЗ довжини n внутрішній стан являє собою попередні n вихідних бітів генератора. Навіть якщо схема зворотного зв'язку зберігається в секреті, вона може бути визначена по $2n$ вихідним бітам генератора за допомогою алгоритму Берлекемпа-Мессі.

Крім того, великі випадкові числа, що генеруються з використанням ідучи підряд біт цієї послідовності, сильно корельовані і для деяких типів додатків зовсім не є випадковими. Незважаючи на це, РЗЛЗЗ часто використовуються при розробці алгоритмів шифрування. Прикладом є шифр А5.

4.2.5 Алгоритм А5

А5 - це потоковий шифр, який використовується для шифрування GSM (Group Special Mobile) - європейського стандарту для цифрових стільникових телефонів, а саме, каналу «телефон - базова станція».

А5 складається з трьох РЗЛЗЗ довжиною 19, 22 і 23. Виходом є $\oplus$ трьох РЗЛЗЗ. В А5 використовується змінюване управління тактуванням (зрушенням на кожному кроці). Кожен регістр тактується в залежності від свого середнього біта, потім виконується $\oplus$ зі зворотною пороговою функцією середніх бітів всіх трьох регістрів. Зазвичай на кожному етапі тактується два РЗЛЗЗ.

Існує тривіальна атака на відкритому тексті, заснована на припущенні про зміст перших двох РЗЛЗЗ і спробі визначення третього РЗЛЗЗ за ключовою послідовністю. Проте ідеї, що лежать в основі А5, дозволяють проектувати надійні потокові шифри. Алгоритм ефективний і задовольняє всім відомим статистичним тестам, єдина його слабкість - короткі регістри. Варіанти А5 з довшими зрушеними регістрами і більш щільними многочленами зворотного зв'язку дозволяють протистояти силовій атаці.

Алгоритм А5/1 - одна з двох різновидів алгоритму А5.

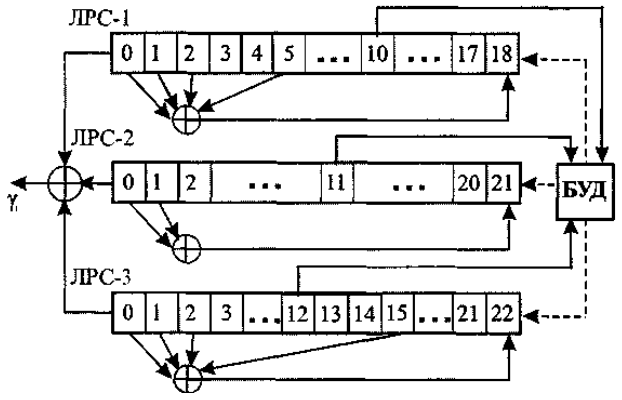


Рис. 4.5. Криптосхема генератора шифру A5/1

Генератор A5/1 складається з трьох АРС над GF (2) довжини 19, 22 і 23 (рис. 4.5) з характеристичними многочленами відповідно:

$x^{19} + x^5 + x^2 + x + 1$ $x^{22} + x + 1$ $x^{22} + x^{15} + x^2 + x + 1$

Сума бітів, що знімаються з трьох ЛРС, утворює гаму генератора. Нелінійність алгоритму досягається за рахунок нерівномірного руху всіх ЛРС, контрольованим блоком управління рухом (БУД).

Після обчислення знака гами відбувається зрушення деяких (не менше ніж двох) регістрів. Зрушення залежить від значення трьох бітів: 10-го біта ЛРС-1, 11-го біта ЛРС-2 і 12-го біта ЛРС-3 - і визначається за правилом: якщо всі три біта однакові, то зсовуються всі регістри, в іншому випадку зсовуються два регістра, чиї керуючі біти збігаються.

Відкритий текст являє собою набір 114-бітових блоків. Перед шифруванням кожного блоку відбувається перезавантаження станів регістрів, яке визначається однозначно переданим в лінію несекретним номером блоку і секретним 64-бітовим сеансовим ключем шифру. Визначення сеансового ключа шифру рівносильне визначенню стану регістрів після перезавантаження, так як сеансовий ключ відновлюється однозначно за станом регістрів і за номером блоку з вирішенням лінійної системи рівнянь від 64 змінних над полем Галуа GF (2), яке буде розглянуто далі.

Після перезавантаження регістрів здійснюється 100 тактів холостого прогону, тобто 100 перших знаків гами ігноруються. У наступні 114 тактів виробляються генератором біти, які використовуються для гамування блоку відкритого тексту.

4.3 Стандарт шифрування AES

Даний стандарт прийнятий в якості заміни широко поширеного, але недостатньо стійкого для захисту секретних даних стандарту DES. Перші критичні зауваження на адресу алгоритму DES з'явилися практично відразу після його публікації. Одними з найбільш суворих критиків поки ще не затвердженого стандарту були М.Хеллман і У.Діффі, рік по тому, які прославили себе винаходом асиметричної криптографії. Основним об'єктом критики була занадто мала довжина ключа - всього 56 біт. За оцінками М.Хеллмана, вартість пристрою для злому шифру шляхом повного перебору ключів, що містить мільйон вузлів, здатних випробувати мільйон ключів в секунду, не повинна перевищувати $20 млн. Така сума вже в той час була цілком під силу розвідувальній службі великої держави. З тих пір ця сума зменшилася на кілька порядків завдяки бурхливому розвитку мікроелектроніки.

Не слід забувати те, що стандарт визначає алгоритм шифрування несекретних даних - в сукупності з попереднім це виправдовує відсутність запасу міцності в ньому. Дослідники відзначили, що існуючої довжини ключа буде цілком достатньо на 10-15 років, і в цьому вони не помилилися. Про всяк випадок було вирішено переглядати стандарт кожні п'ять років. Такі перегляди були виконані в 1983, 1988 і 1993 роках, тоді

стандарт був залишений без змін. Однак незабаром слабкість шифру стала очевидною, і для підвищення стійкості фахівці рекомендували при його використанні шифрувати два або три рази з різними ключами. У 1998 році, коли стало остаточно ясно, що стандарт шифрування повинен бути замінений, Національний інститут стандартів і технологій (NIST), випустив запит, де описувався передбачуваний «Вдосконалений стандарт шифрування» (Advanced Encryption Standard - AES), який повинен прийти на зміну DES.

Відповідно до вимог NIST претендент на стандарт шифрування повинен бути симетричним блоковим шифром з розміром блоку 128 біт, ключем 256 біт (також повинні підтримуватися ключі довжиною 128 і 192 біта), мати стійкість, не меншу, ніж потрійний DES. Швидкість шифрування претендента повинен перевищувати швидкість шифрування потрійного DES-алгоритму. Шифр повинен мати досить прозору структуру для аналізу, можливість ефективної реалізації на платформі Pentium Pro, а також можливість ефективної апаратної реалізації.

Щоб бути затвердженим як стандарт, алгоритм повинен:

- реалізувати шифрування приватним ключем;
- являти собою блоковий шифр;
- працювати з 128-розрядними блоками даних і ключами трьох розмірів 128, 192 і 256 розрядів.

Додатково кандидатам рекомендувалося:

- використовувати операції, легко реалізовані як апаратно (в мікрочіпах), так і програмно (на персональних комп'ютерах і серверах);
- орієнтуватися на 32-розрядні процесори;
- не ускладнювати без необхідності структуру шифру для того, щоб всі зацікавлені сторони були в змозі самостійно провести незалежний криптоаналіз алгоритму і переконатися, що в ньому не закладено жодних недокументованих можливостей.

Перед проведенням першого туру конкурсу в NIST надійшло 21 пропозиція, з яких 15 задовольняли висунутим критеріям. Потім були проведені дослідження цих рішень, в тому числі пов'язані з дешифруванням і перевіркою продуктивності, і отримані експертні оцінки фахівців з криптографії. У серпні 1999 року NIST оголосив п'ять фіналістів. Це такі алгоритми:

- **MARS**, запропонований корпорацією IBM;
- **RC6**, запропонований компанією RSA Security;
- **Rijndael**, запропонований Д. Дайманом і В. Райманом;
- **Serpent**, запропонований Р. Андерсоном, Е. Біхам і Л. Кнудсеном;
- **Twofish**, запропонований Б. Шнайєром і іншими співробітниками компанії Counterpane Internet Security.

У жовтні 2000 року NIST оголосив про свій вибір - переможцем конкурсу став алгоритм RIJNDAEL (вимовляється як "райндол") бельгійських криптографів Вінсента Раймана і Джоана Даймана. Алгоритм зареєстрований в якості офіційного федерального стандарту як FIPS 197. Розробники Rijndael погодилися надати алгоритм для вільного використання без будь-яких авторських відрахувань.

Найвищу надійність AES NIST підтверджує астрономічними числами. 128-бітний ключ забезпечує 340 ундеціліонів ($340 \cdot 10^{36}$) можливих комбінацій, а 256-бітний ключ збільшує це число до $11 \cdot 10^{76}$. Для порівняння, старий алгоритм DES, дає загальне число комбінацій в $72 \cdot 10^{15}$.

4.4 Алгоритм RIJNDAEL

Rijndael - швидкий і компактний алгоритм з простою математичною структурою, завдяки чому він виявився простим для аналізу при оцінці рівня захисту, і ніяких претензій фахівці NIST при цьому не висловили. Атаки на версію з скороченим числом раундів показали, що Rijndael не має такого запасу міцності, як інші кандидати, а збільшення числа раундів уповільнює його роботу. Крім того, Rijndael продемонстрував хорошу стійкість до атак на реалізацію, при яких хакер намагається декодувати зашифроване повідомлення, аналізуючи зовнішні прояви алгоритму, в тому числі рівень енергоспоживання і час виконання. Rijndael можна легко захистити від таких атак, оскільки він спирається в основному на булеві операції.

Загальна продуктивність програмних реалізацій Rijndael виявилася найкращою. Він прекрасно пройшов всі тести зі смарт-картами і в апаратних реалізаціях. Алгоритму в значній мірі притаманний внутрішній паралелізм, що дозволяє без праці забезпечити ефективне використання процесорних ресурсів. Збільшення довжини ключа кілька уповільнює його роботу, оскільки при обробці ключів більшої довжини алгоритм передбачає виконання додаткових раундів шифрування.

NIST зупинив свій вибір на Rijndael, оскільки той поєднує в собі простоту і високу продуктивність. Хоча Rijndael володіє меншим запасом міцності, ніж інші алгоритми, це не несе в собі ніякого практичного ризику.

4.4.1. Опис алгоритму

Rijndael - це симетричний ітеративний оборотний блоковий шифр з змінними розмірами ключа, блоків інформації і числом циклів шифрування. Розмір блоку в Rijndael може бути довільним, кратним 32 бітам, але в стандарті AES зафіксовано розмір блоку 128.

Довжина ключа в AES дорівнює 128, 192 або 256 біт і назва стандарту відповідно буде - AES-128, AES-192 і AES-256.

Циклове перетворення шифру є однорідним і складається з трьох типів шарів. Під однорідністю перетворення розуміється те, що кожен біт стану обробляється аналогічним чином.

Вибір конструктивних параметрів шарів здійснений відповідно з певною стратегією (*Wide Trail Strategy*):

1) ***нелінійний шар*** реалізує паралельне застосування s-боксів з оптимальними (в гіршому випадку) нелінійними властивостями;

2) ***шар лінійного перемішування*** забезпечує хороші перемішуючі властивості алгоритму (дифузію) вже після декількох циклів шифрування;

3) ***шар додавання ключа*** реалізує підмішування ключа до проміжного стану за допомогою X0R-додавання.

Для досягнення оборотності шифру шар лінійного перемішування останнього циклу змінений у порівнянні з шарами лінійного перемішування інших циклів.

4.4.2. Стан, ключі і число циклів шифрування

Введемо наступні позначення:

N_b - число 32-бітових слів, що містяться у вхідному блоці
(В AES $N_b = 4$);
N_k - число 32-бітових слів, що містяться в ключі шифрування
(В AES $N_k = 4, 6, 8$);
Nr - кількість раундів шифрування як функція від N_b і N_k ($N_r = 10,12,14$).

Вхідні (*input*), проміжні (*state*) і вихідні (*output*) результати перетворень блоків

даних, називаються *станами* (*State*). Стану зручно представити у вигляді прямокутних масивів байтів, що мають 4 рядки і N_b стовпців, де N_b є довжина блоку, поділена на 32.

Рис. 3.6 демонструє таке уявлення, що носить назву архітектури «Квадрат», для 128-бітових вхідних блоків визначених у стандарті AES. Послідовність бітів стану записується по колонкам зверху вниз, у стовпчиках ідуть зліва направо. Стовпець матриці утворює 32-бітовий вектор (слово), вся матриця містить один блок даних і описує стан.

input			
in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	i_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

state			
s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}

output			
out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

Рис. 4.6. Приклад представлення блоку у вигляді матриці для $N_b = 4$

Ключ k також можна представити у вигляді прямокутного масиву байтів, який має 4 рядки і N_k стовпців, де N_k є довжина ключа, поділена на 32. Якщо колонку кожного з масивів розглядати як 4-байтове слово, то можна вважати, що ключ і всякий стан є послідовність 4-байтових слів.

Довжини блоку і ключа можуть вибиратися незалежно один від одного і складають, як правило, 128, 192 або 256 біт.

Ключ k може мати будь-яку довжину, кратну 4 байтам, але якщо його довжина відрізняється від 128, 192 або 256 біт, необхідно довизначити число N_r циклів шифрування.

Довжина блоку також може бути будь-якою, кратною 4 байтам, але не менше 16 байт. При цьому необхідно довизначити 3 константи зрушення.

Число N_r циклів шифрування (раундів) визначається на старті і залежить від N_b і N_k:

N_r	N_k=*4*	N_k=*6*	N_k = 8
N_b=*4*	10(AES-128)	12(AES-192)	14(AES-256)
N_b=6	12	12	14
N_b= 8	14	14	14

4.4.3. Математичні операції алгоритму

Байтові операції. Байти розглядаються як елементи кінцевого поля Галуа GF (2^8) близько 2^8, де байту $b = (b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0)$ відповідає поліном

$b(x) = b_7 x^7 + b_6 x^6 + \dots b_1 x + b_0$

У цьому полі визначені бінарні операції $\oplus$ і $\cdot$.

- операція $\oplus$ є по розрядне підсумовування байтів по модулю 2 (або X0R-підсумовування).
- операція $\cdot$ це множення поліномів, відповідних байтам, по модулю не зводжуваного довічного полінома $m(x)=x^8+x^4+x^3+x+1$.
- і існує зворотний елемент по множенню, який можна знайти за допомогою розширеного алгоритму Евкліда.

Добуток многочлена $b(x)$ на x дорівнює поліному:

$$b(x) \cdot x = b_7 x^8 + b_6 x^7 + b_5 x^6 + b_4 x^5 + b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x,$$

наведеним по модулю $m(x)$.

Зауваження. Приведення по модулю $m(x)$ означає обчислення залишку від ділення на $m(x)$, тобто результат повинен мати ступінь не вище 7. При цьому

- якщо $b_7 = 0$, результатом буде зрушений вліво на одну позицію байт b, до якого приписується справа 0:

  $b'=(b_6, b_5, b_4, b_3, b_2, b_1, b_0,0)$

- якщо $b_7 = 1$, то результат виходить за допомогою X0R-підсумовування поліномів $b(x)\cdot x$ та $m(x)$ довжиною по 9 біт з відкиданням старшого біта або, що рівносильно, байтам b' і байта ′0B′ (в шістнадцятирічному поданні), так як поліному $m(x)$ відповідають біти ′1 0000 1011′, дві молодші четвірки біт рівні шістнадцятирічним цифрам 0_{16} і B_{16} (11_{10}).

- Операцію множення x на многочлен $b(x)$ позначають $x$time(b). Множення многочлена $b(x)$ на x^k рівносильно k-кратній композиції операції $x$time(b).

Операції з 4-байтовими векторами. Стан, що представляє собою 4-байтові вектори можна представляти за допомогою поліномів ступеня не вище 3 з коефіцієнтами з поля Галуа GF (2^8), які самі є поліномами ступеня не вище 7. Складання 4-байтових векторів проводиться шляхом X0R-підсумовування, що рівносильно підсумовуванню в полі GF (2^8) коефіцієнтів при відповідних ступенях поліномів.

Множення двох поліномів проводиться з подальшим приведенням добутку по модулю полінома M (x) = x^4 + 1. При цьому, якщо:

$$c(x) = a(x) + b(x), \text{ где } \quad a(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0$$
$$b(x) = b_3x^3 + b_2x^2 + b_1x^1 + b_0$$
$$c(x) = c_3x^3 + c_2x^2 + c_1x^1 + c_0, \text{ то}$$

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Поліном M (x) не є не зводжуваним над GF (2^8), отже, множення на фіксований поліном не завжди є оборотним. Однак в шифрі *Rijndael* множення реалізується на фіксований поліном, для якого зворотний існує.

Добуток многочлена $b(x)$ на x дорівнює поліному:

$$b(x) \cdot x = b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x$$

наведеним по модулю M (x). Результатом приведення є поліном:

$b_2 x^3 + b_1 x^2 + b_0 x + b_3$

Таким чином, множення на x рівносильно лівому циклічному зрушенню байтів всередині вектора.

4.4.4. Функція шифрування

Шари, що складають кожен цикл (крім останнього циклу), реалізовані наступними чотирма перетвореннями:

I) *заміною байтів* - *SubBytes* (*S*) – побайтова нелінійна підстановка в *State*-блоках (S-Box) з використанням фіксованої таблиці замін (*affain map table*);

II) *зрушенням рядків* - *ShiftRows* (*S*) - циклічне зрушення рядків масиву *State* на різну кількість байт;

III) *перемішуванням стовпців* - *MixColumns* (*S*) - множення стовпців стану, що розглядаються як многочлени над $GF(2^8)$;

IV) *накладенням циклового ключа AddRoundKey* (*S*) - порозрядне *XOR* умісту *State* з поточним фрагментом розгорнутого ключа.

I) ***Заміна байтів*** (нелінійна підстановка *SubBytes*) діє на всі байти b стану незалежно і визначається функцією λ, що має вигляд:

$$\lambda(b)=((x^4+x^3+x^2+x+1)\cdot b(x)+x^6+x^5+x+1) \bmod (x^8+1)$$

Зворотне йому перетворення виглядає як:

$$\lambda^{-1}(b)=((x^6+x^3+x)\cdot b(x)+x^2+1) \bmod (x^8+1)$$

Результати всіх інших операцій над байтами наводяться не по модулю $x^8 + 1$, а по модулю $m(x)=x^8+x^4+x^3+x+1$.

Багаторазове обчислення в процесі зашифрування даного виразу надавало б невиправдане обчислювальне навантаження на виконуючу систему, тому для практичної реалізації найбільш прийнятним рішенням є використання попередньо обчисленої таблиці заміни *S-Box*. Її використання зводить операцію *SubBytes* () до найпростішої вибірки байта з масиву $\lambda(b)= Sbox(b)$. Логіка роботи *S-Box* при перетворенні байта $b=\{xy\}$ відображена в шістнадцятирічному вигляді на Рис. 4.7.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рис. 4.7 Таблиця *S-Box* заміни байт

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61

Рис. 4.8. Таблиця *S-Box* інверсної заміни байт

У функціях розшифрування застосовується операція, зворотна *SubBytes* () - *InvSubBytes* (), яка реалізується так само просто, як і попередня допомогою інверсної таблиці *S-Box* - $\lambda^{-1}(b)=InvSbox(b)$, її логіка роботи при перетворенні байта {*xy*} відображена в шістнадцятирічному вигляді на Рис. 4.8.

В *іншому варіанті* реалізації підстановки перетворення байта складається з двох операцій:

1) для кожного байта стану (представленого двійковим поліномом b (x) ступеня не вище 7) знаходимо зворотний елемент b^{-1} (x) в полі GF (2^8), при цьому '00' переходить в '00';

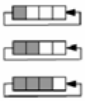
2) до отриманого байту, представленому двійковим вектором (b_0, b_1, ..., b_7), застосовуємо невироджене афінне перетворення A над GF (2):

$$A\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

II) ***Зрушення рядків*** стану *ShiftRows* (*S*) належить до шару лінійного перемішування і реалізується як циклічне зрушення вправо трьох останніх рядків матриці стану на C_1 C_2 і C_3 байт відповідно. Значення C_1 C_2 і C_3 залежать від величини N_b:

	C_1	C_2	C_3	
N_b =4	1	2	3	AES
N_b =6	1	2	3	
N_b = 8	1	3	4	

state			
s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}



III) ***Перемішування стовпців*** стану *MixColumns* (*S*) відноситься також до шару лінійного перемішування і реалізується так. Стовпці розглядаються як поліноми ступеня не вище 3 над полем GF (2^8) і множаться по модулю полінома M (x) = x^4 + 1 на фіксований поліном c (x):

$$c(x) = \text{'03'}\, x^3 + \text{'01'}\, x^2 + \text{'01'}\, x + \text{'02'}$$

Тут ′03′ означає запис константи довжиною 8 біт у вигляді двох шістнадцятирічних цифр від 0 до E, кожна довжиною 4 біта. Це перетворення може бути представлено в матричному вигляді наступним чином:

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}.$$

IV) ***Накладення i-го циклового ключа***. У даній операції раундовий ключ додається до стану за допомогою порозрядного *XOR*. Довжина ключа дорівнює числу біт в стані (N_b в 32-розрядних словах)

Далі, в залежності від величини N_k, масив *State* піддається раундовій трансформації N_p раз (в AES 10, 12 або 14), причому фінальний раунд є кілька укороченим - в ньому відсутнє перетворення *MixColumns* ().

Вихідними даними описаної послідовності операцій є шифртекст - результат дії функції шифрування AES.

4.4.5. Ключовий розклад

Генерація циклових (раундових) ключів з ключа шифру містить дві процедури:

1. розширення ключа (*Key Expansion*) і

2. вибір циклових ключів (*Round Key Selection*).

Генерація реалізується на основі наступних принципів:

- ключ шифру *k* перетворюється в розширений ключ (*Expanded Key*);
- загальна кількість біт циклових ключів дорівнює довжині блоку, помноженої на N_r+1, де N_r - кількість циклів шифрування;
- циклові ключі утворюються з розширеного ключа наступним чином:
  1-й цикловий ключ складається з перших N_b слів,
  2-й - з наступних N_b слів і т. п.

Розширення ключа шифру *k* відбувається з використанням одного з двох варіантів рекурентного закону. Уявімо розширений ключ як послідовність 4-байтових слів {w[*i*]}, *i* = 0,1,2,...

Перші N_k слів містять ключ шифрування. Кожне наступне слово *w*[*i*] виходить за допомогою *XOR* попереднього слова *w*[*i*-1] і слова на N_k позицій раніше (при $N_k < 6$):

$$w[i] = w[i-1] \oplus w[i-N_k], i \geq N_k, i \neq l \cdot N_k, l = 1,2,3,...$$

Для слів, позиція *i* яких кратна N_k, перед *XOR* застосовується перетворення до *w*[*i*-1], а потім ще додається раундова константа RC[*i*]:

$$w[i] = F(T(w[i-1])) \oplus w[i-N_k] \oplus C\left(\frac{i}{N}\right)$$

де T – циклічне зрушення байтової послідовності на 1 байт, C(*t*) є 4-байтовий вектор (RC[*t*], '00', '00', '00'), 1-й байт якого обчислюється за законом:

$$RC[t] = xtime(RC[t-1]); \quad RC[0] = '01'; \quad t = 1, 2,...$$

де *x*time (*b*) ми позначили операцію множення *x* на многочлен *b* (*x*),

RC[ ] - масив 32-бітових раундових констант; значення $RC[j] = 2^{j-1}$.

Якщо $N_k > 6$, то рекурентний закон відрізняється від раніше заданих, тільки при *i*, для яких *i*-4 кратна N_k, а саме:

$$w[i] = F(w[i-1]) \oplus w[i-N_k]$$

На вибір ключа *k* ніяких обмежень не накладається, але розширений ключ необхідно обчислювати з ключа шифру *k* і не можна визначати іншим способом.

Таким чином, алгоритм шифрування *rijndael* складається з початкового накладення циклового ключа, N_r -1 ідентичних циклів шифрування і останнього циклу, в якому відсутнє перемішування стовпців.

4.4.6. Функція розшифрування

У специфікації алгоритму AES пропонуються два види реалізацій функції розшифрування, що відрізняються один від одного послідовністю застосування перетворень, зворотних перетворень функції шифрування, і послідовністю планування ключів.

Так як алгоритм шифрування є композицією оборотних перетворень, то розшифрування здійснюється шляхом застосування зворотних перетворень до вихідного блоку (стану):

InvSubBytes () - зворотна заміна байтів- побайтова нелінійна підстановка в *State*-блоках з використанням фіксованої таблиці замін розмірністю 8x256 (inverse affain map);

InvShiftRows () - зворотне циклічне зрушення рядків масиву *State* на різну кількість байт;

InvMixColumns () - відновлення значень стовпців - множення стовпців стану, що розглядаються як многочлени над GF (2^8);

Функція зворотного розшифрування. Якщо замість *SubBytes* (), *ShiftRows* (), *MixColumns* () і *AddRoundKey* () в зворотній послідовності виконати інверсні їм перетворення, можна побудувати функцію зворотного розшифрування. При цьому порядок використання раундових ключів є зворотним по відношенню до того, який використовується при зашифровані.

При заміні байтів використовується *S-box*, зворотний до *S-box*, що використовується при шифруванні, при зрушенні рядків три останні рядки стану циклічно зрушуються відповідно на N_b-C_1, N_b-C_2 і N_b-C_3 байт, при перемішуванні колонок використовується множення на многочлен:

$$d(x) = '0B' x^3 + '0D' x^2 + '09' x + '0E'$$

зворотний до введеного вище многочлену $c(x)$.

Таким чином, цикл алгоритму зворотного розшифрування складається з наступних послідовно застосовуваних перетворень:

а) зворотної заміни байтів;

б) зворотного зрушення рядків;

в) зворотного перемішування стовпців;

г) накладення інверсного циклового ключа, одержуваного шляхом застосування зворотного замішування стовпців до відповідного циклового ключа.

Циклові ключі в алгоритмі зворотного розшифрування використовуються в зворотному порядку. В останньому циклі, як і в разі прямого перетворення, відсутнє перемішування стовпців.

Алгоритм прямого розшифрування. Алгоритм зворотного розшифрування, описаний вище, має порядок застосування операцій-функцій, зворотний порядку операцій в алгоритмі прямого шифрування, але використовує ті ж параметри розгорнутого ключа. Змінивши певним чином послідовність планування ключа, можна побудувати ще один алгоритм - алгоритм *прямого* розшифрування.

Два наступних властивості дозволяють зробити це:

- Порядок застосування функцій *SubBytes* () і *ShiftRows* () не грає ролі. Те ж саме вірно і для операцій *InvSubBytes* () і *InvShiftRows* (). Це відбувається тому, що функції *SubBytes* () і *InvSubBytes* () працюють з байтами, а операції *ShiftRows* () і *InvShiftRows* () зрушують цілі байти, не зачіпаючи їх значень.

- Операція *MixColumns* () є лінійною щодо вхідних даних, що означає *InvMixColumns* (*State XOR RoundKey*) = = *InvMixColumns* (*State*) *XOR InvMixColumns* (*RoundKey*)

Ці властивості функцій алгоритму шифрування дозволяють змінити порядок застосування функцій *InvSubBytes* () і *InvShiftRows* (). Функції *AddRounKey* () і *InvMixColumns* () також можуть бути застосовані в зворотному порядку, але за умови, що стовпчики (32-бітові слова) розгорнутого ключа розшифрування попередньо пропущені через функцію *InvMixColumns* ().

Таким чином, можна реалізувати більш ефективний спосіб розшифрування з тим же порядком додатка функцій, що і в алгоритмі шифрування.

Алгоритм розшифрування грає кілька менш важливу роль в конструкції шифру, так як в деяких режимах роботи розшифрування не використовується (CFB, гамування, обчислення вектора аутентифікації). Реалізація алгоритму допускає високий ступінь розпаралелювання. Всі перетворення в циклі шифрування можуть виконуватися паралельно над байтами, рядками або стовпцями стану. У тих додатках, де особливі вимоги до швидкості шифрування, розширення ключа доцільно виконати один раз і багаторазово використовувати розширений ключ для обробки вхідних блоків. Якщо ключ шифру необхідно часто змінювати, то розширення ключа можна виконувати паралельно з циклами шифрування.

На закінчення сформулюємо *основні особливості AES*:

- нова архітектура «квадрат», що забезпечує швидке розсіювання і перемішування інформації, при цьому за один раунд перетворення піддається весь вхідний блок;
- байт-орієнтована структура, зручна для реалізації на 8-розрядних мікроконтролерах;
- усі раундові перетворення суть операції в кінцевих полях, що допускають ефективну апаратну і програмну реалізацію на різних платформах.

4.5 Інші алгоритми - кандидати на AES

Об'єктивно порівняти гідності алгоритмів шифрування досить складно. Претенденти на роль AES мають багато спільного, але є і важливі відмінності, хоча і не існує прийнятного методу, що дозволяє надійно оцінити, які з цих відмінностей впливають на безпеку зашифрованих даних. Щоб вирішити це завдання, фахівці з шифрування розробили методику для дослідження алгоритмів шифрування. Один з підходів передбачав аналіз версій кожного алгоритму зі скороченим числом раундів. Оскільки у всіх п'яти кандидатів передбачено виконання серії окремих раундів, криптографи можуть вивчати спрощені версії кожного алгоритму, зменшуючи число виконуваних раундів. Наприклад, при шифруванні 128-розрядним ключем Rijndael виконує 10 раундів. Криптографи проаналізували рівень захисту Rijndael і виявили недоліки при виконанні семи або меншого числа раундів. Аналогічним чином були перевірені і інші кандидати на роль AES. В результаті було виявлено, що Rijndael стає досить стійким до атак вже починаючи з восьмого раунду і, до того ж, виконує після цього ще два раунди шифрування. Фахівці NIST прийшли до висновку, що дане рішення має адекватним запасом захисту, хоча інші кандидати мають навіть більший запас міцності.

Для оцінки продуктивності в NIST провели тестування програмних і апаратних реалізацій алгоритмів, а також реалізацій для смарт-карт (званих в NIST версіями з ресурсними обмеженнями - restricted space). При тестуванні програмного забезпечення розглядалися 32-розрядні реалізації на C, Java і для смарт-карт на базі ARM, а крім того, реалізації на 64 і 8-розряд-них процесорах і процесорах для обробки цифрових сигналів.

MARS - занадто повільно і складно. MARS виявився найскладнішим з усіх представлених кандидатів. У той час як інші алгоритми в усіх раундах використовують одну і ту ж функцію, в MARS застосовуються чотири різні функції. Як показало тестування варіантів зі скороченим числом раундів, це забезпечує даному алгоритму дуже високий запас міцності. Однак його складність змусила засумніватися в цих оцінках, а

деякі з тих хто бере участь в тестуванні фахівців порахували, що MARS вимагає більш ретельного аналізу, ніж той, який можна зробити у відведений для експертизи час.

У MARS використані множення, змінне чергування і великі таблиці даних. Все це значно ускладнює його захист від атак на реалізацію, при тому, що модифікація MARS з метою посилення захисту від таких атак серйозно знижує його продуктивність. MARS не отримав оцінок вище середніх. В цілому продуктивність його програмної реалізації знаходиться на середньому рівні, хоча результати значно варіюються в залежності від застосовуваних процесорів і компіляторів.

Оцінки апаратних реалізацій виявилися нижче середнього, незалежно від довжини ключа. MARS не дуже добре підходить для реалізацій в смарт-картах, оскільки вимагає оперативної пам'яті великої місткості. В кінцевому рахунку цей алгоритм виявився відкинутий через оцінку по продуктивності і виключно високою складністю.

RC6 - занадто багато оперативної пам'яті. RC6 - це простий алгоритм з адекватним запасом міцності. Він базується на RC5, розробленим раніше в RSA Security, застосування якого не виявило якихось серйозних проблем. Як і в MARS, в RC6 використовуються множення і змінне чергування, в силу чого RC6 важко захистити від атак на реалізацію, хоча і не настільки складно, як MARS.

Крім того, RC6 працює досить швидко. У деяких випадках, зокрема, в програмних реалізаціях на 32-розрядних процесорах, він випереджає Rijndael, але апаратні реалізації мають лише середню продуктивність. До того ж, RC6 потрібно багато оперативної пам'яті, в силу чого він не дуже добре підходить для середовищ з ресурсними обмеженнями. RC6 не став переможцем через низьку продуктивність при апаратній реалізації.

Serpent - надійний, але повільний. Serpent схожий на Rijndael, але замість виконання невеликого числа більш складних раундів Serpent виконує більше простих раундів. Завдяки своїй простій, надійній архітектурі Serpent повторює деякі характеристики DES і в цілому спирається на добре відомі операції. Через цю простоту і популярності оцінити надійність Serpent виявилося набагато простіше, і після вивчення версії зі скороченим числом раундів з'ясувалося, що він володіє високим запасом міцності. Serpent відноситься до тих алгоритмам, які найпростіше захистити від атак на реалізацію.

На жаль, програмні реалізації Serpent виявилися повільними серед фіналістів. З іншого боку, в деяких випадках тестери змогли організувати конвеєр апаратних реалізацій, що показав дуже високу продуктивність. Збільшення розміру ключа не впливало на швидкість роботи. Через низькі вимоги до пам'яті Serpent добре підходить для застосування в смарт-картах. Хоча Serpent пропонував найкраще поєднання простоти і запасу міцності, ніж Rijndael, він поступився останньому через низьку продуктивність програмних реалізацій.

Twofish - повільний і таємничий. Twofish використовує кардинально новий підхід, при якому половина ключа використовується для зміни роботи самого алгоритму шифрування, і в цьому підалгоритмі як власного ключа шифрування застосовується інша половина вихідного ключа. Ця особливість призводить до поділу ключа, що, на думку деяких аналітиків, може зробити алгоритм нестійким до атак, організованих за принципом "розділяй і володарюй". При подібній атаці хакер може спробувати визначити, який ключ був обраний в підалгоритмі, і відразу ж отримати половину значення ключа. Однак при аналізі тестерам не вдалося провести жодну з подібних атак.

Вивчення варіантів Twofish зі скороченим числом раундів показало, що він володіє високим запасом міцності. Однак, як і у випадку з MARS, його незвичайна структура породила певні сумніви в якості цих досліджень. Деякі тестери відзначали, що через складність Twofish проаналізувати його детально в відведені для цього терміни виявилося дуже складно.

Twofish вразливий для атак на реалізацію, але його можна модифікувати таким чином, щоб він виявився здатний ефективно протистояти деяким атакам. В цілому Twofish показав середню продуктивність. Продуктивність програмних реалізацій виявилася нижчою за середню, а час попередньої обробки ключа найбільшим. Продуктивність апаратних реалізацій була середньою. Завдяки обмеженим вимогам до пам'яті цей

алгоритм підходить для реалізації на смарт-картах. NIST не вибрав Twofish через його порівняно низьку продуктивність і складність алгоритму.

4.6 Інші відомі блокові шифри

Існує безліч інших блочних шифрів, які запатентовані і використовуються для вирішення різних практичних завдань, однак вони не є стандартами. Розглянемо деякі з цих алгоритмів.

Алгоритм FEAL

Криптоалгоритм FEAL був розроблений як альтернатива DES. Оригінальний криптоалгоритм (FEAL-4) був розрахований на програмну реалізацію і мав чотири цикли перетворення при обробці 64-бітного блоку і 64-бітний секретний ключ.

Криптоаналітичні дослідження продемонстрували слабкість криптоалгоритма - для розкриття ключа при атаці на основі вибіркового відкритого тексту досить 20 зразків. Успіхи в криптоаналізі FEAL-8 (вісім циклів перетворення) привели до появи модифікації зі змінним числом циклів перетворення FEAL-N. Опубліковані результати успішного диференціального криптоаналізу FEAL-N при $N < 31$. Для розкриття секретного ключа FEAL-8 методом лінійного криптоаналізу досить 2^{25} різних відкритих текстів.

Алгоритм SAFER

Криптоалгоритм SAFER (Secure And Fast Encryption routine) являє собою блоковий шифр, розроблений на замовлення корпорації Cylink. Довжина секретного ключа в одній з версій становить 64 біта. Криптоалгоритм орієнтований на байтову обробку блоків по 64 біта. Має змінне число циклів криптографічного перетворення (від 6 до 10). На відміну від більшості блокових шифрів має різні процедури шифрування і дешифрування.

Перша версія SAFER з довжиною ключа 64 біта відома під назвою SAFER K-64. Результати криптоаналізу продемонстрували криптостійкість SAFER K-64 по відношенню до диференціального і лінійного криптоаналізу при дотриманні однієї умови - число циклів перетворення повинно бути більше шести.

Друга версія - SAFER K-128 має 128-бітний ключ. В результаті дослідження було виявлено слабке місце в процедурі перетворення ключа. У нових модифікаціях SAFER SK-64 і SAFER SK-128 виявлений недолік усунуто. Модифікація SAFER SK-40 має 40-бітний ключ і п'ять циклів перетворення і вимагає більший обсяг обчислень при диференціальному і лінійному криптоаналізу в порівнянні з силовою атакою (вичерпним перебором на множині ключів).

Алгоритм Skipjack

Криптоалгоритм Skipjack розроблений Агентством національної безпеки США і є невід'ємною частиною мікросхеми Clipper. Розрахований на обробку 64-бітових вхідних блоків даних (32 циклу перетворення на кожен блок) і має 80-бітний ключ.

Мабуть, криптоалгоритм має високу криптостійкість -для порівняння: DES має 56-бітний ключ і 16 циклів перетворення на блок. Однак деталі криптоалгоритма тривалий час були засекречені. Таким чином, відкриті криптоаналітичні дослідження Skipjack відсутні. На замовлення уряду США незалежна група експертів провела обмежене дослідження криптостійкості Skipjack.

Алгоритм Blowfish

Криптоалгоритм Blowfish - варіант шифру Фейстеля - розрахований на обробку 64-бітових вхідних блоків. Кожен цикл криптографічного перетворення складається з серії перестановок (залежать від ключа) і підстановок (залежать від ключа і вхідних даних). Процедура криптографічного перетворення побудована на операціях підсумовування 32-розрядних чисел і підсумовування по модулю 2. Ключ має змінну довжину (максимум 448 біт) і використовується для побудови декількох допоміжних ключових таблиць. Криптоалгоритм розроблений спеціально для 32-бітових комп'ютерів. За ефективністю програмна реалізація Blowfish значно перевершує аналогічну реалізацію DES. Відомий ряд успішних атак на Blowfish з трьома циклами перетворення. Дослідження

криптостійкості алгоритму тривають.

Алгоритм REDOC

Криптоалгоритм REDOC розроблений для компанії Cryptech, Inc. У ньому використовується 20-байтовий (160-бітний) ключ і 80-бітний блок (по 10 циклів криптографічного перетворення на кожен блок). REDOC орієнтований на байтові операції і ефективний при програмній реалізації. REDOC використовує мінливі табличні функції. На відміну від DES, що має фіксований набір таблиць підстановок і перестановок REDOC II використовує залежні від ключа і відкритого тексту набори таблиць.

Особливість криптоалгоритма - використання спеціальних масок - чисел, отриманих з ключів і необхідних для вибору табличних функцій. З точки зору силової атаки REDOC дуже надійний: для розкриття ключа потрібно 2^{160} спроб дешифрування. Використовуючи диференційний криптоаналіз, Біхам і Шамір досягли успіху в криптоаналізі одного циклу криптографічного перетворення REDOC за допомогою 2300 обраних відкритих текстів. Спроби криптоаналіза двох і більше циклів перетворення закінчилися невдачею.

Алгоритми Khufu і Khafre

Криптоалгоритми Khufu і Khafre розроблені Р. Меркле (R. Merkl). (Khufu (Хуфу) і Khafre (Хафр) імена єгипетських фараонів.)

Khufu є 64-бітний шифр з 512-бітовим ключем і змінним числом циклів криптографічного перетворення. Автор криптоалгоритма зазначив, що Khufu з 8 циклами менш криптостійкий при атаці на основі вибіркового відкритого тексту, ніж варіант з 16, 24 або 32 циклами. Відомо, що Khufu стійкий до атаки методом диференціального криптоаналізу. Відомий результат успішної атаки на Khufu з 16 циклами. Для здійснення атаки знадобилося 231 зразків вибіркового відкритого тексту. Результат, однак, не вдалося поширити на більше число циклів перетворення. Силова атака на Khufu безнадійна: 2^{512} спроб дешифрування - такий обсяг перебору не реалізуємо ні за яких умов.

Криптоалгоритм Khafre по конструкції схожий на Khufu. При реалізації Khafre менш ефективний, ніж Khufu, за рахунок використання 64 або 128-бітних ключів і більшого числа циклів перетворення. Атака на основі методу диференціального криптоаналізу дозволила розкрити ключ Khafre з 16 циклами після години роботи на персональному комп'ютері.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Брюс Шнайдер. Криптографія: принципи і практика. – Київ: ВД "Професіонал", 2017. – 432 с.
2. Мороз, В. В. Основи криптографії та захисту інформації. – Київ: Кондор, 2012. – 200 с.
3. Шевченко, В. І. Криптографія та інформаційна безпека. – Харків: ХНУРЕ, 2016. – 300 с.
4. Камінський, О. О. Захист інформації: теорія та практика. – Київ: Либідь, 2015. – 400 с.
5. Лисенко, В. В. Методи і засоби криптографії. – Одеса: Одеська національна академія харчових технологій, 2018. – 250 с.
6. Романенко, А. В. Теоретичні основи криптографії. – Львів: Львівська політехніка, 2020. – 220 с.
7. Stallings, W. Cryptography and Network Security: Principles and Practice. – 7th ed. – Boston: Pearson, 2017. – 672 p.
8. Paar, C., & Pelzl, J. Understanding Cryptography: A Textbook for Students and Practitioners. – Berlin: Springer, 2010. – 362 p.
9. Stinson, D. R. Cryptography: Theory and Practice. – 3rd ed. – Boca Raton: Chapman & Hall/CRC, 2006. – 688 p.
10. NIST Special Publication 800-131A. Transitioning the Use of Cryptographic Algorithms and Key Lengths. – National Institute of Standards and Technology, 2019. – 31 p.
11. FIPS PUB 197. Announcing the Advanced Encryption Standard (AES). – National Institute of Standards and Technology, 2001. – 60 p.