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

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

Одеса – 2025

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

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 № 1 від 27.08.2025

Одеса – 2025

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» / Укл.: А. С. Коляда, О. С. Лопаков, В. В. Космачевський – Одеса : Національний університет «Одеська політехніка», 2025. – 152 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

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.

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.

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.

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:

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.

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.

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):

порушення повноважень;

маскарад;

обхід захисту;

троянські програми або потаємні ходи;

«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.

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.

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.

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:

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;

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

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

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

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

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.

Приклади завдань

1. Скласти коротку модель загроз для умовної інформаційної системи (активи, порушник, канали впливу) та віднести загрози до конфіденційності/цілісності/доступності.
2. Навести приклад санкціонованого і несанкціонованого доступу для одного об’єкта (файл/БД/канал) і описати, які правила розмежування доступу було порушено.

3. Для заданого сценарію визначити суб’єкти й об’єкти доступу та побудувати фрагмент матриці доступу (читання/запис/виконання/делегування).
4. Запропонувати набір заходів політики безпеки, що одночасно зменшує ризик витоку, підміни даних і відмови в обслуговуванні.
5. Описати мінімальний склад подій для протоколювання й аудиту в ІС та пояснити, як ці журнали допоможуть розслідувати інцидент “маскарад + обхід захисту”.

Контрольні питання

1. Чим відрізняються поняття “загроза”, “уразливість” і “атака” в контексті безпеки інформаційних систем?
2. Як формулюються та практично забезпечуються конфіденційність, цілісність і доступність (CIA) в ІС?
3. У чому різниця між санкціонованим і несанкціонованим доступом та яку роль відіграють правила розмежування доступу?
4. Що таке політика безпеки, які її ключові елементи та чим відрізняються довільне і примусове керування доступом?
5. Для чого потрібні протоколювання й аудит, і які типи подій є найбільш важливими для аналізу інцидентів?

Література

1. ISO/IEC 27000:2018. Information technology — Security techniques — Information security management systems — Overview and vocabulary. Geneva : ISO, 2018.
2. ISO/IEC 27001:2022. Information security, cybersecurity and privacy protection — Information security management systems — Requirements. Geneva : ISO, 2022.
3. Stallings W., Brown L. Computer Security: Principles and Practice. 4th ed. Boston : Pearson, 2018.
4. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems. 3rd ed. Hoboken : Wiley, 2020.
5. NIST. Security and Privacy Controls for Information Systems and Organizations. NIST Special Publication 800-53, Rev. 5. Gaithersburg : National Institute of Standards and Technology, 2020.

Лекція №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.

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.

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:

- симетричні криптосистеми (класична криптографія);
- асиметричні алгоритми шифрування;
- системи електронного підпису;
- 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.

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.

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.

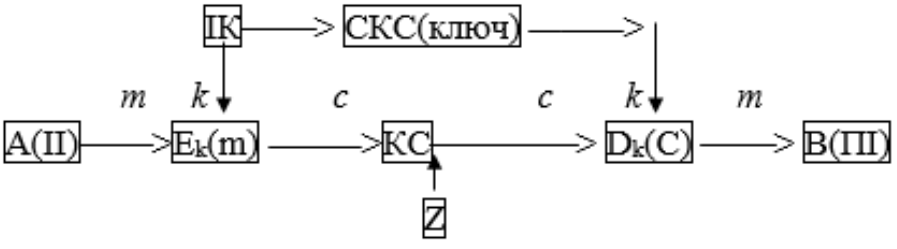


Рис. 2.1. Загальна схема захищеної передачі інформації. Тут:

А (*відправник*, джерело інформації (ІІ), в англомовній літературі зазвичай позначається ім'ям Alice) і В (*одержувач*, приймач інформації (ІП), 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.

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.

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 *ci* і добувати відповідні їм відкриті тексти *mi*. 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:

чи є вона для противника ціннішою, ніж вартість атаки;
чи є вона для вас ціннішою, ніж вартість захисту.

Вимоги до криптосистем

Для сучасних криптографічних систем захисту інформації сформульовані наступні загальноприйняті вимоги:

зашифроване повідомлення повинно піддаватися читанню тільки при наявності ключа;

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;

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);

знання алгоритму шифрування не повинно впливати на надійність захисту;

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

структурні елементи алгоритму шифрування повинні бути незмінними;

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

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

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

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

Приклади завдань

1. Пояснити на прикладі “Сцітали” і “шифру Цезаря”, у чому різниця між шифрами перестановки та шифрами заміни.
2. Скласти коротку часову шкалу (5–7 подій) розвитку криптографії від античних шифрів до ери наукової криптології після робіт К. Шеннона.
3. Для заданого сценарію обміну повідомленнями описати ролі A, B, Z, відкритий текст m, шифртекст c і ключ k в узагальненій схемі криптосистеми з секретним ключем.
4. Для одного простого шифру сформулювати, яка з атак (ciphertext-only, known-plaintext, chosen-plaintext, chosen-ciphertext) є для нього найбільш небезпечною, і чому.
5. Підібрати приклад вимоги до криптосистеми (з переліку в лекції) та показати, як її порушення проявляється на практиці (на рівні ключа, алгоритму або реалізації).

Контрольні питання

1. Що таке криптологія, криптографія і криптоаналіз, і чому ці напрями є взаємопов’язаними?
2. Які ознаки відрізняють “донаукову” криптологію від “наукової” і яку роль відіграли роботи Клода Шеннона?
3. У чому полягає принцип Керкгофса (припущення про відомість алгоритму противнику) і як він пов’язаний із роллю ключа?
4. Які цілі пасивних і активних криптоаналітичних атак та які вихідні дані має противник у кожному типі атак?
5. Чому абсолютно стійкий шифр є непрактичним у більшості систем і які підходи використовують для оцінювання криптостійкості на практиці?

Література

1. Shannon C. E. Communication Theory of Secrecy Systems // Bell System Technical Journal. 1949. Vol. 28, No. 4. P. 656–715.
2. Kahn D. The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. New York : Scribner, 1996.
3. Singh S. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. New York : Anchor Books, 2000.
4. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton : CRC Press, 1996.
5. Kerckhoffs A. La cryptographie militaire // Journal des sciences militaires. 1883. T. 9. P. 5–38; 161–191.

Лекція №3 Класичні симетричні криптосистеми. Шифри перестановки. Стеганографічні шифри. Шифри простої і складної заміни

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.

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 5x7 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 4х4.

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 3х3 (якщо не враховувати його повороти). Кількість магічних квадратів 4х4 становить вже 880, а кількість магічних квадратів 5х5 - 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, то ця кількість дорівнює воно має порядок десять у п'ятнадцятій ступені. 4^{N-1}. При $N = 10$

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.

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.

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.

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 *x0* вихідного повідомлення замінюється символом з алфавіту B0, символ *x1* - символом з алфавіту B1 і так далі, символ *xr-1* замінюється символом з алфавіту Br-1, символ *xr* замінюється символом знову з алфавіту B0 і т.п.

Приклад багатоалфавітної підстановки ($r = 4$): Вхідний символ *x0 x1 x2 x3 x4 x5 x6 x7 x8 x9* Алфавіт підстановки B0 B1 B2 B3 B0 B1 B2 B3 B0 B1

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 Трітемієм, 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	щ	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	щ	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	щ	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.

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

$$P\ \{(K0, K1, ..., Kn\text{-}1) = (k0, k1, ..., kn\text{-}1)\} = (1/m)^n$$

Одноразова система шифрування перетворює вихідний текст ($x0$, $x1$, ..., xn-1) в шифрований текст ($y0$, $y1$, ..., yn-1) за допомогою підстановки Цезаря:

$yi = Ek\ (xi) = (ki+xi) \bmod m,\ i = 0, ..., n\text{-}1$

Для такої системи підстановки використовують також термін "одноразова стрічка" і "одноразовий блокнот". Простір ключів K системи одноразовою підстановки складається з векторів ($k0$, $k1$, ..., kn-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:

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

рівність довжини ключа і довжини відкритого тексту;
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.

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:

$yi = xi \oplus ki$, $i = 1, ..., n$

Тут *x1 x2 ... xn* - відкритий текст, *k1 k2 ... kn* - ключ, *y1 y2 ... yn* - шифрований текст.

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

$yi \oplus ki = xi \oplus ki \oplus ki = xi,\ i=1,...,n$

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.

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 *mi*, *i*

= 1, ..., M однакової довжини, зазвичай по 64 біта. Гамма шифру виробляється у вигляді послідовності блоків γi, $i = 1, ..., M$ аналогічної довжини. Рівняння *зашифрування* можна записати у вигляді:

$ci = \gamma i \oplus mi$, $i = 1, ..., M$

де *ci* - *i*-й символ (блок) шифртекста; γi - *i*-й символ (блок) гами шифру; *mi* - *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:

$mi = \gamma i \oplus ci$, $i = 1, ..., M$

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.

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 γi, описувані співвідношенням:

$\gamma i+1 = (a\gamma i + b) \bmod m$, $i = 1,2, ..., M$

де *a* і *b* - константи, $\gamma 0$ - 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* mod 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.

Приклади завдань

1. Виконати шифрування короткого повідомлення методом табличної перестановки (проста перестановка або одиночна перестановка по ключу) та показати кроки формування шифртекста.
2. Побудувати магічний квадрат 4×4 (або взяти заданий) і зашифрувати ним фразу, пояснивши, як нумерація клітин задає перестановку.
3. Для шифру простої заміни (Цезар/полібіанський квадрат/Цезар з ключовим словом) оцінити кількість ключів і пояснити, чому частотний аналіз є ефективним.
4. Скласти невелику таблицю омофонів для 5–7 найчастотніших літер української мови та пояснити, як це зменшує інформативність односимвольних частот.
5. Порівняти шифр Віженера з коротким періодом і одноразову стрічку: вказати умови абсолютної стійкості та продемонструвати, що повторне використання “одноразового” ключа робить шифр вразливим.

Контрольні питання

1. У чому полягає принципова різниця між шифрами перестановки та шифрами заміни, і чому їх комбінація лежить в основі сучасних симетричних шифрів?
2. Які параметри виступають ключем у табличних шифрах перестановки та як впливають розмір таблиці і ключове слово на стійкість?
3. Чому шифри простої (одноалфавітної) заміни легко піддаються криптоаналізу на основі частот, і як ідея омофонів намагається це компенсувати?
4. Що таке багатоалфавітна (складна) заміна, як працює період r, і чому короткий період у Віженера є слабким місцем?
5. Які три умови забезпечують абсолютну стійкість одноразової стрічки, і чому метод гамування є практичним компромісом із власними ризиками?

Література

1. Shannon C. E. Communication Theory of Secrecy Systems // Bell System Technical Journal. 1949. Vol. 28, No. 4. P. 656–715.
2. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton : CRC Press, 1996.
3. Kahn D. The Codebreakers: The Comprehensive History of Secret Communication from Ancient Times to the Internet. New York : Scribner, 1996.
4. Singh S. The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. New York : Anchor Books, 2000.
5. Kerckhoffs A. La cryptographie militaire // Journal des sciences militaires. 1883. T. 9. P. 5–38; 161–191.

Лекція №4 Сучасні симетричні криптосистеми: стандарт шифрування даних DES; Основні режими роботи алгоритму DES

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.

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.

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 $2^{64} \approx 10^{19}$ "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 *fi*, 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* ⊕ *f* (L*i*, K*i*), *i* = 1,2, ..., 16

Структура алгоритму DES

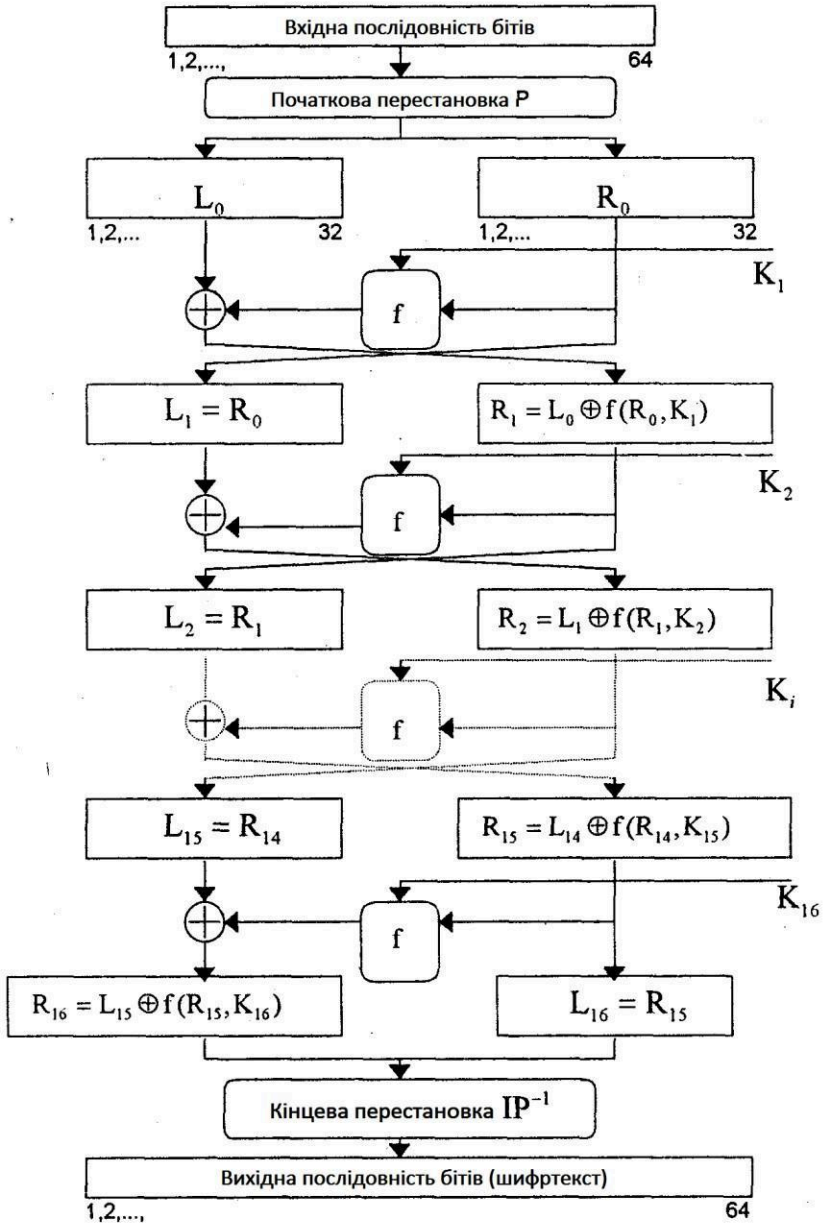


Рис. 4.1. Структура алгоритму DES

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.

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

Таблиця 4.1. Матриця початкової перестановки IP

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. 4.1), 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 T0 розділяється на дві по 32 біта: L0 - ліві (старші) і R0-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* = t1 t2 ... t32 (перші 32 біта); R*i* = t33 t34 ... t64 (останні 32 біта). Тоді результат *i*-й ітерації описується наступними формулами:

L*i* = R*i-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.)

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

Таблиця 4.2. Матриця зворотної перестановки

На останньому кроці ітерації отримують послідовності R16 і L16 (без перестановки місцями), які конкатенуються в 64-бітову послідовність R16 L16.

Після закінчення шифрування здійснюється відновлення позицій бітів за допомогою матриці зворотної перестановки IP^{-1} (таблиця 4.2).

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* ⊕ *f* (L*i*, K*i*), *i* = 16, ..., 2, 1

Таким чином, для процесу розшифрування з переставленим вхідним блоком R16L16 на першій ітерації використовується ключ K16, на другий ітерації - K15 і т.п. На 16-й ітерації використовується ключ K1. На останньому кроці ітерації будуть отримані послідовності L0 і R0, які конкатенуються в 64-бітову послідовність L0R0. Потім в цій послідовності 64 біта переставляються відповідно з матрицею IP^{-1}. Результат такого перетворення - вихідна послідовність бітів (розшифроване 64-бітове значення).

Функція шифрування

Тепер розглянемо, що ховається під перетворенням, позначеною буквою *f* (рис.4.2).

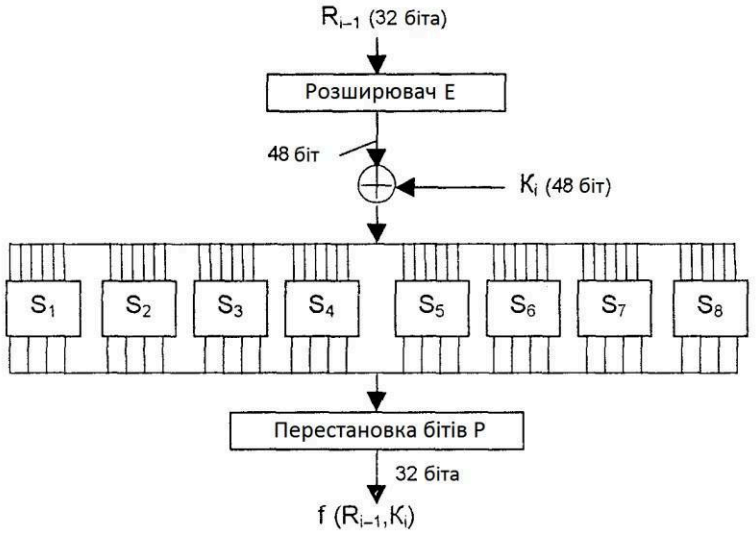


Рис. 4.2. Схема обчислення функції шифрування *f*

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* використовуються:

- функція E (розширення 32 біт до 48);
- функції S1, S2, ..., S8 (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 4.3.

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

Таблиця 4.3. Функція Е.

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 Е (R*i*-1)) складається по модулю 2 з поточним значенням ключа K*i* і потім розбивається на вісім 6-бітових блоків B1, ..., B8:

Е (R*i*-1) ⊕ K*i* = B1B2 ... B8

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	
	1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S1
	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	
	1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S2
	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	
	1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S3
	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	
	1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S4
	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	
	1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S5
	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	
	1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S6
	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	
	1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S7
	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	

	1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S8
	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 4.4. Функції перетворення (вузли заміни) S1, ..., S8

Кожен 6-бітовий блок B*j* перетворюється в 4-бітовий блок за допомогою відображення, заданого таблицями S1, ..., S8. 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 4.4).

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 B1, ..., B8 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 ... *b*6, тоді:

- двохбітове число *b*1*b*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:
S1 (B1) ... S8 (B8)

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 4.5. Функція перестановки бітів P

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.4.3).

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 4.6. Функція G

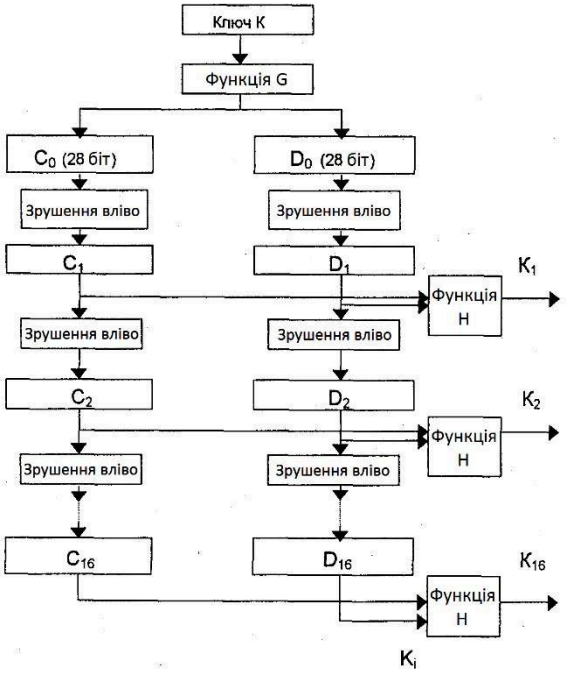


Рис. 4.3. Схема алгоритму обчислення підключів

Для видалення контрольних бітів і підготовки ключа до роботи використовується функція G (K) початкової підготовки ключа (табл. 4.6). Таблиця 3.6 розділена на дві частини. Результат перетворення G (K) розбивається на дві половини C0 і D0 по 28 біт кожен. Перші чотири рядки матриці G визначають, як вибираються біти послідовності C0.(першим бітом C0 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 D0 (тобто послідовність D0 буде складатися з бітів 63, 55, 47, ..., 12, 4 ключа шифру).

Після визначення C0 і D0 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 4.7. Операції зрушення виконуються для послідовностей C*i*, і D*i* незалежно.

Таблиця 4.7.

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 4.8.

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 4.8. Функція H

K*i* = H (C*i* D*i*)

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.

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.

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 = DESk1 (DESk2 (m))

Не забезпечує збільшення в 2^{56} разів обсягу перебору, необхідного для визначення ключа, оскільки при атаці з відомим повідомленням можна підбирати паралельно ключі $k1$ і $k2$, 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$k1$ (DESk^{-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.

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.

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.

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 $m1$ складається по модулю два з 64-бітовим секретним початковим вектором $c0$, званим вектором ініціалізації. Отримана сума шифрується за схемою DES. Отриманий 64-бітовий блок шифртекста $c1$ складається по модулю два з другим блоком відкритого тексту, результат шифрується і виходить другий 64-бітовий блок шифртекста $c2$ і так до кінця файлу. Таким чином, результат шифрування ci визначається наступним чином:

$ci = DES (mi \oplus ci-1), \forall i = 1, ..., n$

де n - число блоків вихідного файлу, а $c0$ - 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:

$mi = ci\text{-}1 \oplus \mathrm{DES}^{-1}(ci)$

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 *mi* є функцією тільки *ci* і *ci*-1 Якщо при передачі буде викривлений один з блоків шифрограми *ci*, то при розшифровці блок *mi* виявиться повністю викривленим, а в блоці *mi* + 1 будуть спотворені тільки біти, що змінилися в блоці *ci*. 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 ⊕ 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 *mn*, операції >> і << означають, як і в мові програмування С, арифметичні зрушення вправо і вліво на задане число розрядів):

●неповний блок *mn* відкритого тексту доповнюється фіксованим доповненням *p*64- *j* до 64 бітів, наприклад пропусками:

$cn = \mathrm{DES}(((mn << (64\text{-}j)) + p64\text{-}j) \oplus cn\text{-}1)$

●неповний блок *mn* відкритого тексту складається побітно зі старшими *j* розрядами DES перетворення попереднього блоку шифртекста:$cn = mn \oplus (\mathrm{DES}(cn\text{-}1) >> (64\text{-}j))$.

В останньому випадку зберігається довжина повідомлення.

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:

$ci = mi \{ pi\text{-}1, \text{ⓐ } i = 1, ..., n$

де *pi*-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 *ci* в регістр (рис.4.4).

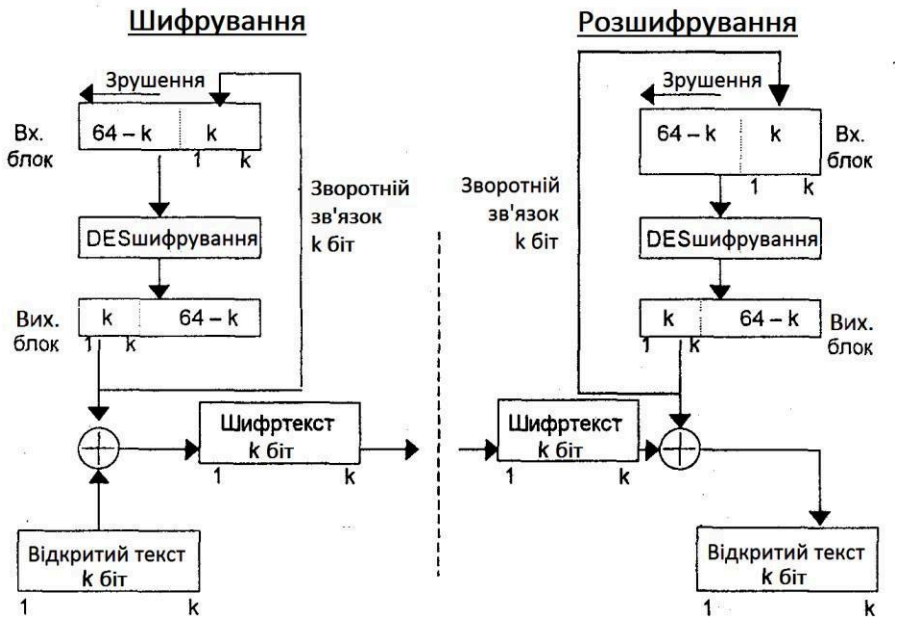


Рис. 4.4. Схема DES шифрування/дешифрування в режимі CFB Відновлення зашифрованих даних також виконується відносно просто: *pi*-1

та *ci*, обчислюється аналогічним чином: *mi* = *ci* ⊕ *pi*-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.

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 ... *mn*. Тоді блоки шифртекста:

ci = *mi* ❲ *pi-1*֍ *i* = *1* ... *n*

де *pi*-1- старші *k* бітів результату операції DES (*ci*-1).

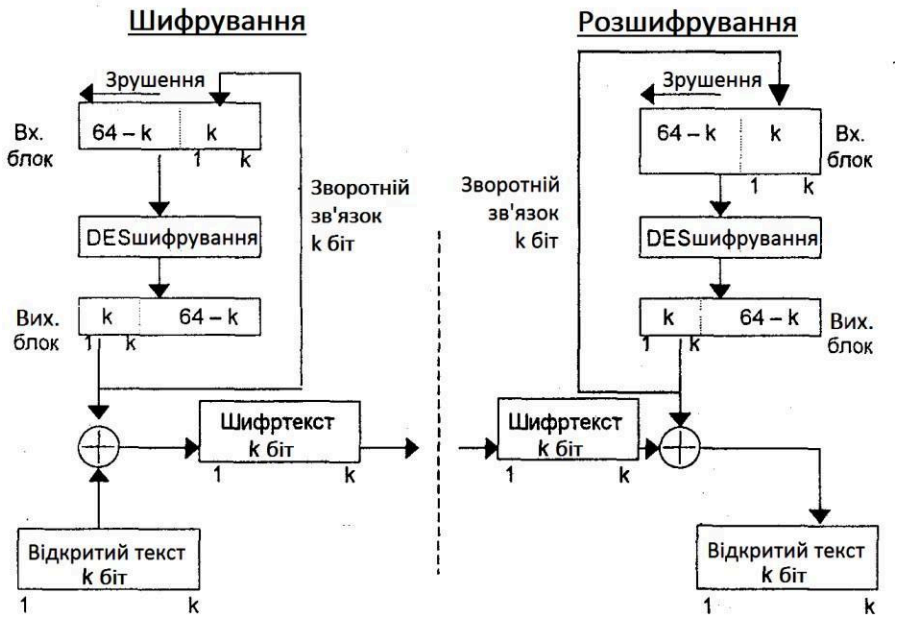


Рис. 4.5. Схема DES шифрування/дешифрування в режимі OFB

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.

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.

Приклади завдань

1. Пояснити на прикладі, як принципи Шеннона розсіювання та перемішування реалізуються в DES (через мережу Фейстеля, S-блоки та перестановки).
2. Для одного 64-бітового блоку виконати схему DES на рівні етапів (IP → 16 раундів → IP^{-1}) і вказати, де саме застосовуються XOR, розширення E та перестановка P.
3. Порівняти ECB і CBC для повідомлення з повторюваними 8-байтовими блоками та зробити висновок, який режим “видає” структуру відкритого тексту.
4. Змоделювати поширення помилки в шифртексті для режимів CBC, CFB і OFB (які блоки/біти відкритого тексту спотворяться після розшифрування).
5. Оцінити практичну стійкість DES за рахунок розміру ключа 56 біт і пояснити, чому 3DES підвищує стійкість, а 2DES уразливий до meet-in-the-middle.

Контрольні питання

1. У чому полягають принципи розсіювання та перемішування і чому їх поєднання є базовим для сучасних симетричних шифрів?
2. Що таке мережа Фейстеля, які формули раунду DES, і чому дешифрування можливе навіть якщо f-функція не є взаємно-однозначною?
3. Яку роль у DES відіграють S-блоки (нелінійність), і чому саме вони є ключовими для криптостійкості?
4. Як працює key schedule у DES (56-бітовий ключ, зсуви C_i/D_i, компресія до 48 біт), і навіщо потрібні підключі $K_1 \ldots K_{16}$?
5. Порівняти режими ECB, CBC, CFB, OFB за трьома критеріями: маскування повторів, поширення помилок, придатність для потокових даних/аутентифікації.

Література

1. Shannon C. E. Communication Theory of Secrecy Systems // Bell System Technical Journal. 1949. Vol. 28, No. 4. P. 656–715.

2. Data Encryption Standard (DES) : FIPS PUB 46-3. Gaithersburg : National Institute of Standards and Technology, 1999.
3. DES Modes of Operation : FIPS PUB 81. Washington, DC : National Bureau of Standards, 1980.
4. Computer Data Authentication : FIPS PUB 113. Washington, DC : National Bureau of Standards, 1985.
5. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton : CRC Press, 1996.

Лекція №5 Сучасні симетричні криптосистеми: алгоритм Магма (ДСТУ 28147:2009) та його порівняння з алгоритмом DES

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.

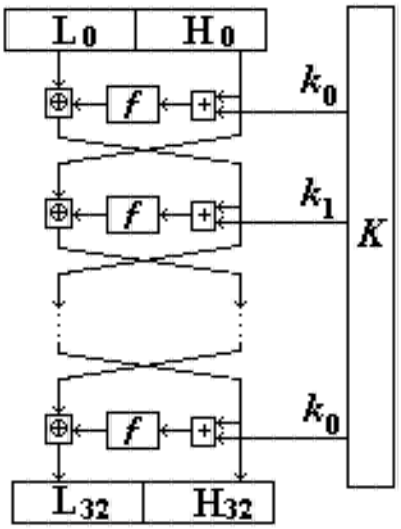


Рис. 5.1. Схема алгоритму ДСТУ 28147:2009

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: + - складання по модулю 2^{32},

⊕ - побітове додавання по модулю 2.

Для додавання ключа і складання підблоків в мережі Фейстеля використовуються операції + і ⊕. 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:

$k = k(0) \| k(1) \| ... \| k(7)$

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.

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)mod8)) ⊕ L*i*-1 для *i*=1, 2, ..., 24 L*i* = H*i*-1

H*i* = *f*(H*i*-1 [+] *k*(32-*i*)) ⊕ L*i*-1 для *i*=25, 26, ..., 31 L*i* = H*i*-1

H32 = H31 L32 = *f*(H31 [+] *k*0) ⊕ L31

64-розрядний блок зашифрованих даних C представляється у вигляді: C = H32L32

Решта блоків відкритих даних в режимі простої заміни зашифровуються аналогічно.

Підключи *ki*, 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.

Функція шифрування

Функція шифрування включає дві операції над 32-розрядним аргументом. Схема функції шифрування в алгоритмі ДСТУ 28147:2009 показана на Рис. 5.2.

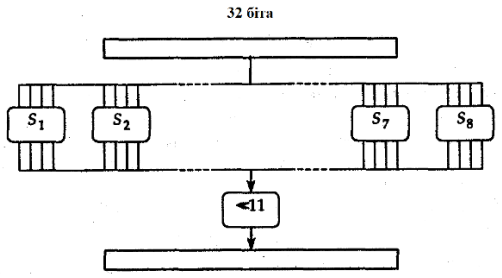


Рис. 5.2. Функція шифрування в алгоритмі ДСТУ 28147:2009

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.

Режим гамування

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

Відкриті дані, розбиті на 64-розрядні блоки *mi* (*i* = 1,2, ..., *n*) (*n* визначається обсягом шифрованих даних), зашифровуються в режимі гамування шляхом порозрядного додавання за модулем 2 з гамою шифру ■ш=(■1, ■2,...., ■m), яка виробляється блоками по 64 біта. *ci* (64-розрядний блок зашифрованого тексту) виходить з рівняння

ci= *mi* ⊕ ■*i*

де гамма шифру: ■*i* = F (*yi*-1, *zi*-1).

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

Величини *yi* і *zi* визначаються ітераційно по мірі формування гами наступним чином:

(*y*0, *z*0)=F(*s*)

(*yi* , *zi*)=(*yi*-1+ *a*2, *zi*-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 = 0101010116 і *a*2 = 0101010416 - 32-розрядні двійкові константи, задані в ДСТУ 28147:2009.

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 *mi*, 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:

$c1=F(s) \oplus m1=$ [redacted]$1 \oplus m1$, s - синхропосилка,

$ci = F(mi\text{-}1) \oplus mi =$ [redacted]$i \oplus mi$, $i=2, 3, ..., n$.

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.

Вироблення імітовставки

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

Імітовставка - це блок з p біт (імітовставка Ip), 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 p 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 $m1$ 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 $m2$, 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 Ip 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 Ip передається по каналу зв'язку після зашифрованих даних:

$c1$, ..., cm, Ip

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.

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.

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. 5.3. 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 $x1$, $x2$, $x3$, $x4$, підблоки вихідного відображення (шифртекста) позначимо через $y1$, $y2$, $y3$, $y4$, ключові підблоки i-го циклу позначимо через $k1^{(i)}$, $k2^{(i)}$, $k3^{(i)}$, …,$k6^{(i)}$, $i = 1,2, ..., 8$, ключові підблоки вихідного відображення $k1^{(9)}$, $k2^{(9)}$, $k3^{(9)}$, $k4^{(9)}$.

Опишемо перший цикл роботи алгоритму (інші цикли реалізуються аналогічно). Протягом першого циклу шифрування реалізуються наступні кроки обчислень (для простоти замість позначення $k1^{(i)}$, будемо використовувати ki, $i = 1,2, ..., 6$).

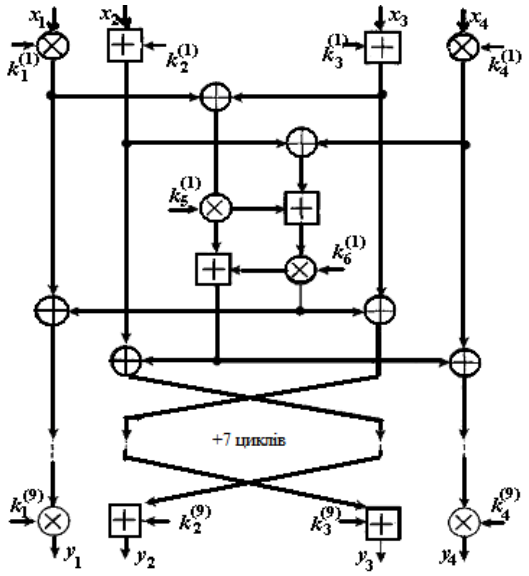


Рис. 5.3. Блок-схема алгоритму шифрування IDEA

Крок 1. $c1= x1 \otimes k1$ Крок 2. $c2= x2 + k2$ Крок 3. $c3= x3 + k3$

Крок 4. $c4= x4 \otimes k4$ Крок 5. $c5= c1 \oplus c3$ Крок 6. $c6= c2 \oplus c4$

Крок 7. $c7= c5 \otimes k5$ Крок 8. $c8= c6 + k6$ Крок 9. $c9= c8 \otimes c7$

Крок 10. $c10= c9+c7$ Крок 11. $c11= c1 \oplus c9$ Крок 12. $c12= c3 \oplus c9$

Крок 13. $c13= c2 \oplus c10$ Крок 14. $c14= c4 \oplus c10$

Вхідні підблоки наступного циклу є $c11$, $c13$, $c12$, $c14$. Після 8-го циклу реалізується вихідне відображення:

Крок 1. $y1= x1^{(9)} \otimes k1^{(9)}$ Крок 2. $y2= x2^{(9)} + k2^{(9)}$ Крок 3. $y3= x3^{(9)} + k3^{(9)}$ Крок 4. $y4= x4^{(9)} \otimes k4^{(9)}$ Шифрованим текстом є блок ($y1$, $y2$, $y3$, $y4$).

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:

$k1^{(1)}, k2^{(1)}, k3^{(1)}, k4^{(1)}, k5^{(1)}, k6^{(1)}, k1^{(2)}, k2^{(2)}$

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 $qi^{(r)}$) визначаються наступним чином: перші чотири

- При $r = 2,3, ..., 8$: $(q^{(r)}_1, q^{(r)}_2, q^{(r)}_3, q^{(r)}_4) = ([k^{(10-r)}_1]^{-1}, -k^{(10-r)}_2, -k^{(10-r)}_3, [k^{(10-r)}_4]^{-1})$,

1 2 3 4 1 2 3 4

- При $r = 1,9$: $(q^{(r)}_1, q^{(r)}_2, q^{(r)}_3, q^{(r)}_4) = ([k^{(10-r)}_1]^{-1}, -k^{(10-r)}_2, -k^{(10-r)}_3, [k^{(10-r)}_4]^{-1})$,

і два останніх $(q^{(r)}_5, q^{(r)}_6) = (k^{(r)}_5, k^{(r)}_6)$, $r = 1,2, ..., 8$;

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).

Атаки на блокові шифри

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.

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.

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.

Приклади завдань

1. Реалізувати (або промоделювати) один раунд алгоритму Магма за ДСТУ 28147:2009: підстановка 8 S-блоками по 4 біти та циклічний зсув на 11 біт.
2. Для заданого 256-бітового ключа сформувати послідовність підключів на 32 цикли (3 проходи k(0)…k(7) і 1 — у зворотному порядку) та пояснити логіку такого розкладу.
3. Порівняти DES і Магма за параметрами: довжина ключа, кількість раундів, структура раунду, роль S-блоків та очікуваний лавинний ефект (короткий висновок у 6–8 реченнях).
4. Побудувати схему режиму гамування (на основі Магма) з синхропосилкою s та показати, як формується γ(i) і як виконується шифрування ci = mi ⊕ γi.
5. Для повідомлення, розбитого на 64-бітові блоки, описати алгоритм вироблення імітовставки Ip (p на вибір) та оцінити ймовірність успішного нав’язування як 1/2^p.

Контрольні питання

1. Які ключові відмінності Магма (ДСТУ 28147:2009) від DES з точки зору ключового простору, кількості циклів і раундових перетворень?
2. Яку роль відіграє комбінація операцій “додавання mod 2^32” та “XOR” у стійкості Магма проти аналітичних атак?
3. Чим режим простої заміни в Магма обмежений з точки зору практичної безпеки та де його застосування є виправданим?
4. Як працюють режими гамування та гамування зі зворотним зв’язком у Магма і чим вони концептуально подібні до OFB/CFB?
5. Що таке імітовставка, як вона обчислюється в ДСТУ 28147:2009 і як параметр p впливає на рівень імітозахисту?

Література

1. ДСТУ 28147:2009. Захист інформації. Криптографічний захист. Алгоритм криптографічного перетворення : нац. стандарт України. — Київ : Держспоживстандарт України, 2009.
2. National Institute of Standards and Technology. Data Encryption Standard (DES) : FIPS PUB 46-3. — Gaithersburg : NIST, 1999.
3. National Institute of Standards and Technology. Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher : NIST Special Publication 800-67. — Gaithersburg : NIST, 2012.
4. Menezes, A. J., van Oorschot, P. C., Vanstone, S. A. Handbook of Applied Cryptography. — Boca Raton : CRC Press, 1996.
5. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. — 2nd ed. — New York : John Wiley & Sons, 1996.

Лекція №6 Сучасні симетричні криптосистеми: шифри зі змінною довжиною ключа; потокові шифри

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 Ek (m) і дешифрування Dk

(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 *t1*, а час, який зловмисник може дозволити собі витратити на атаку, так само *t2*. При швидкості обчислень, що дорівнює *v* операцій в одиницю часу, отримуємо такі обмеження на довжину ключа:

log$a$ $c2^{-1}$ v $t2 <$ | k | $<(c1^{-1}$ v $t1)^{1/m}$

Тут $c1$ і $c2$ - 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.

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.

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.

Потокові шифри

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: $k1, k2, k3, \ldots, ki$. Цей потік бітів (званий біжучим ключем) і потік бітів відкритого тексту $x1, x2, x3, \ldots, xi$ підсумовуються по модулю 2 (операція $\oplus$ або XOR- «виключне АБО»), і в результаті виходить потік бітів шифртекста:

$ci = xi \oplus ki.$

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

$xi = ci \oplus ki$

Відзначимо, що для отримання ключа за формулою:

$ki = ci \oplus xi$

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.

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: S0 S1, ..., S255. 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:

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

$j = (j + Si) \bmod 256$. Поміняти місцями Si та Sj. $t = (Si + Sj) \bmod 256$; $k = St$.

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: S0=0, S1=1,…., S255=255. Після цього ключ записується в масив: $k0$, $k1$,. .., $k255$. Потім при початковому значенні $j = 0$ в циклі виконуються наступні обчислення:

for $i = 0$ to 255;

$j = (j + St + ki) \bmod 256$.

Поміняти місцями Si та Sj.

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).

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 xn (n-є передане повідомлення) на ключі k, виконавши XOR xn і $k(n)$. Одержувачу не потрібно зберігати стан шифру для відновлення xn, йому не доводиться турбуватися і про втрачені повідомленнях, що впливають на процес дешифрування.

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 6.1

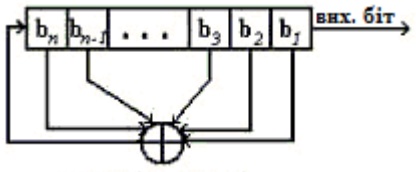


Рис 6.2 РЗЛЗЗ

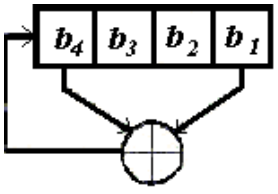


Рис. 6.3

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 6.2.)

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. 6.3). 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.

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 і-му біту відповідає член xi 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 x0 = 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 x32 + x7 + x5 + x3 + x2 + 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 ⊕ тридцять другого, сьомого, п'ятого, третього, другого і першого бітів (рис. 6.4); виходить РЗЛЗЗ матиме максимальну довжину, циклічно проходячи до повторення через 232-1 різних значень.

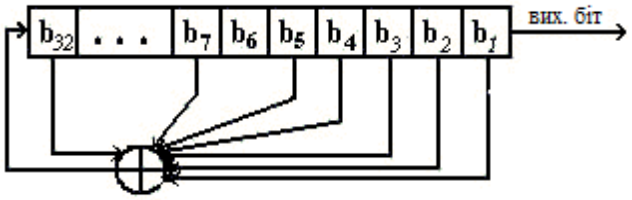


Рис. 6.4. 32-бітовий РСЛОС з максимальним періодом

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 2*n* 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 A5.

Алгоритм A5

A5 - це потоковий шифр, який використовується для шифрування GSM (Group Special Mobile) - європейського стандарту для цифрових стільникових телефонів, а саме, каналу «телефон - базова станція».

A5 складається з трьох РЗЛЗЗ довжиною 19, 22 і 23. Виходом є ⊕ трьох РЗЛЗЗ. В A5 використовується змінюване управління тактуванням (зрушенням на кожному кроці). Кожен регістр тактується в залежності від свого середнього біта, потім виконується ⊕ 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.

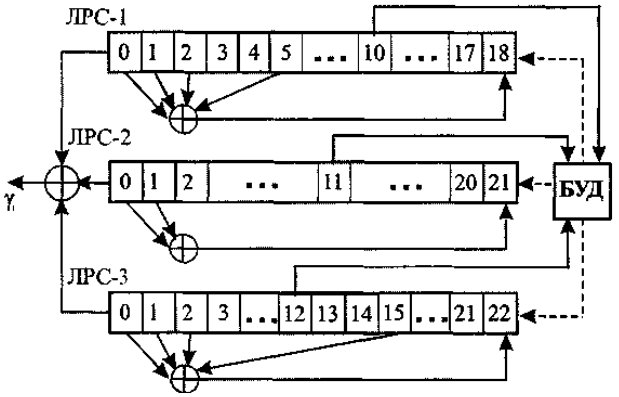


Рис. 6.5. Криптосхема генератора шифру А5/1

Генератор А5/1 складається з трьох АРС над GF (2) довжини 19, 22 і 23 (рис. 6.5) з характеристичними многочленами відповідно:

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

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.

Приклади завдань

1. Для заданих t_1,t_2,v оцінити допустимий інтервал довжини ключа |k| за наведеними нерівностями та пояснити, як зміна v впливає на вибір |k|.
2. Вибрати параметри RC5-w/r/b для двох сценаріїв (вбудований пристрій і сервер) та обґрунтувати компроміс «швидкодія/стійкість» (врахувати w, r, b).
3. Реалізувати генерацію одного байта потоку ключів RC4 (оновлення i,j, swap S_i,S_j, обчислення t, вибір k=S_t) і перевірити шифрування/дешифрування на короткому повідомленні.
4. Побудувати приклад потокового шифрування для мультимедійного потоку: сформувати біжучий ключ k_i і показати, як помилка в одному біті шифртексту вплине на

відновлений відкритий текст для синхронного поточного шифру.
5. Для заданого примітивного многочлена над GF(2) побудувати РЗЛЗЗ з максимальним періодом і експериментально підтвердити період 2^n-1 для малого n (наприклад, n=5 або n=6).

Контрольні питання

1. Навіщо потрібні шифри зі змінною довжиною ключа і які вимоги ставляться до складності легальних алгоритмів шифрування/дешифрування та атак?
2. У чому полягає параметризація RC5 (w/r/b) і чому не всі комбінації параметрів є криптостійкими?
3. Чим потокові шифри принципово відрізняються від режимів гамування блочних шифрів (на рівні залежності потоку ключів від відкритого тексту та синхронізації)?
4. Які ключові ідеї лежать в основі RC4 та які ризики виникають, якщо повторно використовувати один і той самий потік ключів/стан (keystream reuse)?
5. Що таке РЗЛЗЗ, що означає «максимальний період», навіщо потрібен примітивний многочлен і чому РЗЛЗЗ сам по собі не гарантує криптостійкість?

Література

1. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. — 2nd ed. — New York : John Wiley & Sons, 1996.
2. Menezes, A. J., van Oorschot, P. C., Vanstone, S. A. Handbook of Applied Cryptography. — Boca Raton : CRC Press, 1996.
3. Rivest, R. The RC5 Encryption Algorithm. — In: Fast Software Encryption. Lecture Notes in Computer Science. — Berlin : Springer, 1995.
4. Kaliski, B., Robshaw, M. RFC 2268: A Description of the RC2(r) Encryption Algorithm. — Internet Engineering Task Force, 1998.
5. Rogaway, P., Coppersmith, D. A Software-Optimized Encryption Algorithm (SEAL). — In: Fast Software Encryption. Lecture Notes in Computer Science. — Berlin : Springer, 1994.

Лекція №7 Сучасні симетричні криптосистеми: стандарт шифрування AES; алгоритм RIJNDAEL

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 $72 \cdot 10^{15}$.

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.

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*):

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

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

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.

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 Nb = 4

Ключ k також можна представити у вигляді прямокутного масиву байтів, який має 4 рядки і Nk стовпців, де Nk 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 Nr 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 Nr циклів шифрування (раундів) визначається на старті і залежить від Nb і Nk:

Nr	Nk=4	Nk=6	Nk = 8
Nb=4	10(AES-128)	12(AES-192)	14(AES-256)
Nb=6	12	12	14
Nb= 8	14	14	14

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

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

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

2. якщо $b_7 = 1$, то результат виходить за допомогою X0R-підсумовування поліномів $b(x)x$ 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}).

3. 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.

Функція шифрування

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

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

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

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

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

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 $S(b)=Sbox(b)$. Логіка роботи *S-Box* при перетворенні байта $b=\{xy\}$ відображена в шістнадцятирічному вигляді на Рис. 7.1.

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. 7.1 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
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Рис. 7.2. 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. 7.2.

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

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';

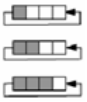
до отриманого байту, представленому двійковим вектором ($b0$, $b1$, ..., $b7$), застосовуємо невироджене афінне перетворення A над GF (2):

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

Зрушення рядків стану *ShiftRows* (*S*) належить до шару лінійного перемішування і реалізується як циклічне зрушення вправо трьох останніх рядків матриці стану на C1 C2 і C3 байт відповідно. Значення C1 C2 і C3 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}



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):

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

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:

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

Накладення і-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.

Ключовий розклад

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

розширення ключа (*Key Expansion*) і

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.

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*-Cl, N*b*-C2 і N*b*-C3 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.

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.

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.

Приклади завдань

1. Для AES-128, AES-192 і AES-256 визначити параметри N_b, N_k, N_r та пояснити, як зміна N_k впливає на кількість раундів і продуктивність.
2. Задано 16-байтовий стан (State) у вигляді матриці 4×4: виконати один раунд AES (SubBytes → ShiftRows → MixColumns → AddRoundKey) і показати проміжні матриці після кожного кроку.
3. Реалізувати операцію xtime(b) у GF(2^8) з модульним поліномом m(x)=x^8+x^4+x^3+x+1 та перевірити множення байта на \{02\} і \{03\} (як у MixColumns).
4. Побудувати S-Box двома способами: через знаходження мультиплікативного оберненого в GF(2^8) (з урахуванням $00 \mapsto 00$) та через афінне перетворення, і порівняти з табличним S-Box.
5. Задано 128-бітний ключ: виконати Key Expansion (обчислення слів w[i], застосування RotWord/SubWord і Rcon), а потім сформувати перші два раундові ключі для AddRoundKey.

Контрольні питання

1. Чому AES (Rijndael) був обраний замість DES і які ключові вимоги NIST висував до кандидата AES (блок, ключі, стійкість, продуктивність, прозорість структури)?
2. У чому полягає архітектура State як матриці $4 \times N_b$ і як вона впливає на дифузію та паралелізм реалізації?
3. Які ролі виконують перетворення SubBytes, ShiftRows, MixColumns, AddRoundKey, і чому в останньому раунді відсутній MixColumns?
4. Як визначаються операції в GF(2^8) для AES (XOR, множення по модулю m(x)) та для чого потрібна операція xtime?
5. Як працює розширення ключа (Key Expansion), що таке Rcon і чому порядок/підготовка раундових ключів важливі для ефективного розшифрування?

Література

1. Advanced Encryption Standard (AES). — (FIPS PUB 197). — Gaithersburg : National Institute of Standards and Technology, 2001.
2. Daemen, J. The Design of Rijndael: AES — The Advanced Encryption Standard / J. Daemen, V. Rijmen. — Berlin ; Heidelberg : Springer, 2002.
3. National Institute of Standards and Technology. Report on the Development of the Advanced Encryption Standard (AES). — Gaithersburg : NIST, 2001.
4. Menezes, A. J. Handbook of Applied Cryptography / A. J. Menezes, P. C. van Oorschot, S. A. Vanstone. — Boca Raton : CRC Press, 1996.
5. Schneier, B. Applied Cryptography: Protocols, Algorithms, and Source Code in C / B. Schneier. — 2nd ed. — New York : John Wiley & Sons, 1996.

Лекція №8 Асиметричні криптосистеми: криптосистема RSA.

Причини виникнення асиметричних криптосистем

Появі нового напряму в криптології - асиметричної криптографії з відкритим ключем - сприяли дві проблеми, які не вдавалося вирішити в рамках класичної симетричної одноключової криптографії.

Перша з цих проблем пов'язана з *поширенням секретних ключів*. Як передати учасникам обміну інформацією змінювані секретні ключі, які потрібні їм для здійснення цього обміну? У загальному випадку для передачі ключа знову ж потрібне використання якоїсь криптосистеми, тобто завдання в рамках симетричної криптографії нерозв'язна.

Друга з цих проблем пов'язана з поширенням електронного документообігу. Виникла проблема забезпечення достовірності і авторства електронних документів. У звичайному, паперовому документообігу ця проблема вирішується за допомогою підпису на папері. Підробити підпис людини на папері зовсім не просто, а скопіювати ланцюжок цифр на ЕВМ - нескладна операція. Виникла проблема цифрового підпису, яка б виконувала всі ті завдання, які виконує підпис, поставлена на документі рукою. Обидві ці проблеми були успішно вирішені за допомогою криптографії з відкритими ключами. В опублікованій в 1976 р статтею "Нові напрямки в криптографії" У.Діффі і М.Хеллман вперше показали, що секретний зв'язок можливий без передачі секретного ключа між відправником і отримувачем.

На основі результатів, отриманих класичною та сучасною алгеброю, були запропоновані *системи з відкритим ключем*, називаються також *асиметричними криптосистемами*.

Суть їх полягає в тому, що кожним адресатом ІС генеруються два ключі, пов'язані між собою за певним правилом. Один ключ оголошується *відкритим*, а інший *закритим*. Відкритий ключ публікується і доступний кожному, хто бажає послати повідомлення адресату. Секретний ключ зберігається в таємниці. Якщо генератор ключів розташувати на стороні одержувача, то відпадає необхідність пересилання секретного ключа по каналу зв'язку.

Вихідний текст шифрується відкритим ключем адресата і передається йому. Зашифрований текст *в принципі не може бути розшифрованим* тим же відкритим ключем. Дешифрування повідомлення можливо тільки з використанням закритого ключа, який відомий тільки самому адресату. Таким чином, не потрібен секретний канал зв'язку для передачі ключа, але необхідно забезпечити справжність відкритого ключа, так як його викривлення або підміна не дозволить розшифрувати інформацію на парному з ним закритому ключі. Крім того, заміна зловмисником законного відкритого ключа на свій відкритий ключ надає йому повний доступ до шифруємої інформації.

Узагальнена схема асиметричної криптосистеми

Нижче (рис.8.1) наведена *узагальнена схема асиметричної криптосистеми*.

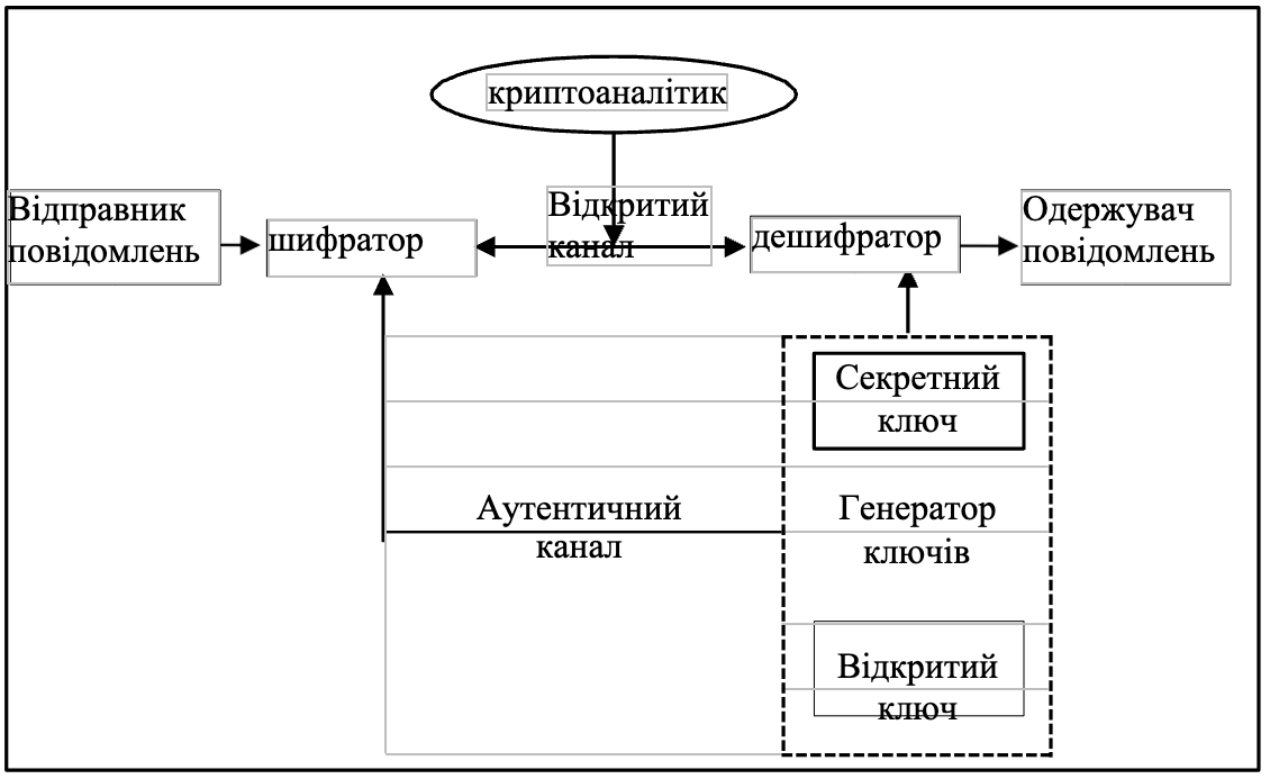


Рис. 8.1. Узагальнена схема асиметричної криптосистеми

Тут для передачі ключа використовується відкритий канал зв'язку, що забезпечує автентичність інформації, що передається.

Щоб гарантувати надійний захист інформації, до систем з відкритим ключем (СВК) пред'являються дві важливі і очевидні вимоги:

Перетворення вихідного тексту повинно бути необоротним і виключати його відновлення на основі відкритого ключа.

Визначення закритого ключа на основі відкритого також повинно бути неможливим на сучасному технологічному рівні. При цьому бажана точна нижня оцінка складності розкриття шифру.

Алгоритми шифрування з відкритим ключем набули широкого поширення в сучасних інформаційних системах. Їх використовують в наступних основних напрямках:

Як самостійні засоби захисту переданих і збережених даних.

Як засоби для розподілу криптографічних ключів. Алгоритми асиметричних криптосистем більш трудомісткі, ніж традиційні криптосистеми. Тому часто на практиці має сенс за допомогою асиметричних криптосистем розподіляти ключі, обсяг яких незначний. А потім за допомогою менш трудомістких симетричних алгоритмів здійснювати обмін великими інформаційними потоками.

Як засоби аутентифікації користувачів інформаційних систем, в тому числі для вирішення проблеми електронного підпису.

Як засоби для побудови складних криптографічних протоколів для вирішення різних завдань захисту інформації в сучасних інформаційних системах.

Алгебраїчна узагальнена модель шифру

Раніше ми розглядали алгебраїчну модель шифру К.Шеннона як трьохосновну універсальну алгебру A = (M, K, C, E), де

M - множина відкритих текстів, K - множина ключів,

C - множина криптограми і

E - ін'єктивна (взаємно однозначна) функція шифрування:

E_k: MxK → C $E_k(m) = c$, де $m \in M$, $k \in K$, $c \in C$

Для того, щоб ця модель могла бути застосовна до асиметричних криптографічних систем, необхідно її розширення. Основна концептуальна ідея побудови такої моделі, природна і очевидна. Вона складається в окремому описі моделей двох шифрів: шифру шифрування і шифру розшифрування, сукупність яких і складає узагальнену алгебраїчну модель шифру.

Шифром шифрування (алгеброю зашифрування) назвемо алгебру: $A_ш = (M, M_c, K_ш C, C_c, E)$, де

множина $M_c \subseteq M$ трактується як підмножина всіх змістовних текстів з множини «відкритих текстів» M.

функція шифрування E здійснює відображення $MxK_ш$ на C:

$E: MxK_ш \rightarrow C$, $E_к(m) = c$

тобто є сюр'єктивною, причому $\forall k \in K_ш$ відображення $E_k(m)$ ін'єктивне (образи двох різних елементів різні), а множина C_c складається з шифрограм, які можуть бути отримані в результаті шифрування змістовних текстів: $C_c = E(M_c x K_ш)$, тобто результатів шифрування тих відкритих текстів m, для яких визначено значення $E_k(m)$ для всіх ключів шифру $k \in K_ш$.

Введення підмножини $M_c \subseteq M$ як множини змістовних текстів дозволяє коректно вводити критерії на змістовні тексти.

Таким чином, шифр шифрування є деяке уточнення моделі шифру Шеннона $A = (M, K_ш, C, E)$.

Шифром розшифрування (алгеброю розшифрування) для $A_ш$ назвемо алгебру: $A_p = (M_p, K_p, C_p, D)$, де

$C \subseteq C_p$ і введення множини C_p - шифртекст «правильних» повідомлень (а точніше, $C_p \backslash C$ - множини перекручених шифртекстів) забезпечує можливість опису реакції приймальної сторони на вступ викривленого шифрованого повідомлення $c_p \notin C$;

$M \subseteq M_p$, і введення множини $M_P \backslash M$ забезпечує можливість опису результату розшифрування приймальною стороною викривленого шифрованого повідомлення $c_p \notin C$;

функція дешифрування D - сюр'єктивне відображення: $D: C_p x K_p \rightarrow M_p$, $D_k(c) = m$,

для якого виконуються наступні умови:

існує бієкція f: $K_ш \rightarrow K_p$;

для будь-яких $m \in M$, $k \in K_ш$ з умови $E_k(m) = c$ випливає: $D(c, f(k)) = m$

При відсутності викривлень в каналі зв'язку функція розшифрування D повністю визначена на всій множині CxK_p.

Відзначимо, що в визначенні шифру розшифрування не міститься вимог ін'єктивності функції f по змінній $k \in K_p$.

Алгебраїчною узагальненою моделлю шифру назвемо трійку:

$(A_ш, A_p, f)$

До позитивних властивостей цієї моделі належить можливість моделювання шифрів як з симетричним, так і з асиметричним ключем.

При цьому враховуються такі міркування:

ключ $k_ш \in K_ш$ не є таємним, а ключ $k_p = f(k_ш) \in K_p$ є секретним;

визначення значення k пов'язано з вирішенням складних проблем;

синтез пар ключів $(k_ш, k_p)$ проводиться досить просто.

Зауважимо, що тут проявляється можливість класифікації шифрів по параметру складності обчислення значення $f(k_ш)$ ключа розшифрування, що визначає основний параметр криптографічного стійкості шифрів з асиметричним ключем.

Односторонні функції

Концепція асиметричних криптографічних систем з відкритим ключем засновано на застосуванні односпрямованих або односторонніх функцій. Остання назва була дана по асоціації з одностороннім рухом, коли легко проїхати в одну сторону і не можна в іншу. При цьому в криптографії, як в житті, «не можна» не означає «неможливо ні за яких умов», але говорить про те, що це пов'язане з серйозними труднощами.

Визначення. Функція f: $X \to У$ називається односторонньою (*oneway function*), якщо існує ефективний алгоритм для обчислення $f(x)$ $\forall x$, але не існує ефективного алгоритму для обчислення хоча б одного елемента прообразу $f^{-1}(y)$.

Ніхто не знає, чи існують взагалі односторонні функції. Основним критерієм віднесення функції f до класу односторонніх або незворотних є відсутність ефективних з *обчислювальної точки зору* алгоритмів зворотного перетворення $Y \to X$. У криптографії під *необоротністю* розуміється не теоретична необоротність функції, а практична неможливість обчислити зворотнє значення, використовуючи сучасні обчислювальні засоби за заданий інтервал часу. Таким чином, проблеми побудови односторонніх функцій пов'язані з теоретико-ймовірнісною складністю алгоритмів і алгоритмічними питаннями теорії чисел.

Безліч класів необоротних функцій і породжує все розмаїття систем з відкритим ключем. Більшість пропонованих сьогодні криптосистем з відкритим ключем спираються на один з наступних типів необоротних перетворень:

Розпад великих цілих чисел на прості множники.

Обчислення логарифма в кінцевому полі.

Обчислення коренів алгебраїчних рівнянь.

Факторизація

Як перший приклад односпрямованої функції розглянемо цілочислове множення. Пряма задача - обчислення добутку двох дуже великих цілих чисел p і q, тобто знаходження значення $n = p \cdot q$, є відносно нескладним завданням.

Зворотнє завдання, звана завданням факторизації, - розкладання на множники великого цілого числа, тобто знаходження дільників p і q великого цілого числа

$n = p/q$

є практично нерозв'язним завданням при досить великих значеннях n. За сучасними оцінками теорії чисел при цілому $n \approx 2^{664}$ і $p \approx q$ для розкладання числа n буде потрібно близько 10^{23} операцій, тобто задача практично нерозв'язна на сучасних ЕВМ.

Якщо прості множники мають спеціальний вид, відомі більш ефективні алгоритми факторизації. Йдеться про співмножників p, таких, у яких величини p-1 або p+1 є «гладкими», тобто мають тільки малі прості дільники.

Однак з появою алгоритму факторизації з використанням еліптичних кривих клас чисел, що допускають швидку факторизацію, розширився і прості критерії перевірки приналежності даного класу втратили свою важливість. Тому, як правило, єдиним розумним критерієм може слугувати розмір простих множників, оскільки зі збільшенням розміру зменшується ймовірність вибрати число спеціального виду.

Дискретний логарифм

Інший приклад односпрямованої функції - це модульна експонента з фіксованою підставою і модулем. Нехай a і n - цілі числа, такі, що $1 \le a \le n$. Тоді модульна експонента з підставою a за модулем n являє собою функцію:

$y = a^x \bmod n$

де x - ціле число. Звичайно записати $x = \log_a (y)$.

Завдання звернення цієї функції в множині цілих чисел називають завданням знаходження дискретного логарифма.

Визначення. Число x називають дискретним логарифмом числа y за основою a і модулю n, якщо для всіх $a \in Z_n$ знайдеться таке ціле y, що

$y = a^x \bmod n$

Обчислення дискретних логарифмів (коли задані a, y і n) приблизно така ж важко вирішуване завдання, як і розкладання на множники.

Визначення. Одностороння функція f: X → Y називається *односторонньою функцією з пасткою*, якщо f^1 (y) можна обчислити за поліноміальний час, маючи деяку додаткову інформацію, тобто існує функція g (y, t), обчислювана за поліноміальний час і така , що g (y, t) = f^{-1} (y) для деякої пастки t.

Ефективне обчислення оберненої функції можливо, якщо відомий "потайний хід" (секретне число, рядок або інша інформація, що асоціюється з цією функцією). Як приклад односпрямованої функції з "потайним ходом" можна привести використання функції Ейлера в криптосистемі RSA.

Криптосистема RSA

Алгоритм RSA став першим повноцінним алгоритмом з відкритим ключем, який може працювати як в режимі шифрування даних, так і в режимі електронного цифрового підпису. Заснована на цьому алгоритмі популярна криптосистема RSA розроблена в 1977 році і отримала назву на честь її творців: Рональда Рівеста (в даний час він очолює компанію RSA Data Security), Аді Шаміра і Леонарда Ейдельмана.

В даний час RSA є найбільш поширеною криптосистемою з відкритим ключем - стандартом де-факто для багатьох криптографічних додатків. Статус де-факто став причиною включення криптосистеми RSA в прийняті раніше криптографічні стандарти, наприклад в фінансові стандарти США і Франції, австралійський стандарт управління ключами і багато інших. Криптосистема RSA застосовується в різних протоколах Internet. У криптографічні стандарти, що діють на території СНГ, RSA не входить, що ускладнює її застосування з точки зору правових норм.

Основні визначення і теореми

Надійність алгоритму ґрунтується на важко обчислюваних завданнях факторизації (розкладання на множники) великих чисел і обчислення дискретних логарифмів. Визначення та теореми з алгебри, використані при створенні даної криптосистеми розглянуті в додатку. Наведемо тут лише основні твердження.

●Криптографічні системи є стійкими, якщо певні їх параметри є ***простими числами***. Число a називається простим, якщо воно не має цілих дільників, крім одиниці. Числа a і b називаються взаємно простими, якщо їх найбільший спільний дільник НСД (a, b) = 1.

●***Функцією Ейлера*** φ (n) називається число позитивних цілих чисел менших n і взаємно простих з n.

Обчислення функції Ейлера φ (n) для великих n в загальному випадку являє собою трудомістку процедуру перебору всіх чисел менших n і перевірки для кожного взаємної простоти з n. Однак, ця функція має такі властивості:

φ (p) = p - 1 ∀p - простого числа.

φ (a, b) = φ (a) φ (b) для будь-яких натуральних взаємно простих a й b,

які дозволяють легко обчислити значення функції Ейлера φ (n) за допомогою трьох арифметичних дій, якщо відомо розкладання числа n на прості множники p і q:

φ (n) = (p-1) (q-1).

У RSA використовується теорема, яка носить назву ***китайської теореми про залишки***, так як цей результат був відомий ще в древньому Китаї, де теорема була запропонована китайським математиком першого століття Сун Це. Вона стверджує, що будь-яке невід'ємне

ціле число, яке не перевищує добутку модулів, можна однозначно відновити, якщо відомі його відрахування по цих модулям, і названа так тому, що результатом приведення числа a по модулю n є залишок від ділення a на n. Фактично в RSA використовується наслідок з цієї теореми, яке стверджує, що якщо відомо розкладання числа n на прості множники $n=n_1n_2...n_k$, де все n_i попарно взаємно прості, і результат приведення числа x по модулю n_i $\forall i=1,\ldots,k$ однаковий, то результатом приведення числа x по модулю n буде те саме число. Тобто $\forall x$, a - цілих чисел:

$x \equiv a \bmod n \Leftrightarrow x \equiv a \bmod n_i \; \forall i=1,\ldots,k$

Теорема Ейлера. Якщо $n > 1$, то $\forall x \in Z_n^*$ (x взаємно простого з n), виконується порівняння

$x^{\varphi(n)} \equiv 1 \bmod n$

Слідство. $\forall x < n$ і взаємно простого з n можна легко обчислити зворотний елемент x^{-1} в кільці відрахувань Z_n з порівняння:

$x^{-1} \equiv x^{\varphi(n)-1} \bmod n$

Мала теорема Ферма. $\forall x \in GF(p), x \neq 0$, виконується порівняння:

$x^{p-1} \equiv 1 \bmod p$

Мала теорема Ферма є наслідком з теореми Ейлера, хоча історично вона була доведена раніше, потім Ейлер її узагальнив.

Слідство 1: Якщо p - просте число, то $\forall x$, взаємно простого з p: $x^p = x \bmod p$

Слідство 2: якщо НСД (e, φ (n)) = 1 (e - просте відносно φ (n)) то ∃d-ціле, таке, що:

$ed = 1 \bmod n$

На цих математичних фактах заснований алгоритм RSA.

Алгоритм RSA

У криптосистемі RSA повідомлення m, криптограма c, відкритий ключ K_o, і секретний ключ K_c, належать множині цілих чисел $Z_n = \{0, 1, 2, ..., n\text{-}1\}$. Безліч Z_n з операціями додавання і множення по модулю n утворює кільце.

Модуль n визначається як складене число, яке дорівнює добутку $n = p \cdot q$ двох великих простих чисел p і q. Модуль n є відкритим параметром алгоритму, а числа p і q - секретними параметрами. Тобто множники p і q зберігають в секреті, а їх добуток n відомо всім, хто користується цією криптосистемою. Тут використовується одностороння функція з пасткою. Знаючи секретні параметри алгоритму p і q можна легко обчислити функцію Ейлера $\phi(n)$ за формулою:

$\phi(n) = (p\text{-}1)(q\text{-}1)$

тоді як обчислення $\phi(n)$ тільки по великому числу n є важко обчислюваним завданням.

Функція Ейлера використовується в RSA при обчисленні ключів. Відкритий ключ $K_o = e$ вибирають випадковим чином з множини: $Z^*_{\phi(n)}$ - чисел менших $\phi(n)$ і взаємно простих з $\phi(n)$.

Інакше умова $e \in Z^*$ рівносильне виконанню двох умов:

$1 < K_o \leq \phi(n)$, и НОД (K_o, $\phi(n)$) = 1

яким повинно задовольняти випадково вибране число $K_o = e$, щоб воно могло служити відкритим ключем в RSA.

Секретний ключ $K_c = d$ обчислюють так, щоб виконувалася умова: $K_c \cdot K_o = e \cdot d \equiv 1 \bmod \phi(n)$ (*)

Тобто, секретний ключ d є зворотним елементом до відкритого ключа e в множині $Z_{\phi(n)}$: $d = e^{-1} \bmod \phi(n)$. Рішення порівняння (*) можна знайти за допомогою розширеного алгоритму Евкліда.

Зауважимо, що d і n також взаємно прості числа. Обчислення секретного ключа з відкритого є важко обчислюваним завданням, якщо невідомі секретні параметри алгоритму p і q, так як при цьому важко обчислити значення функції Ейлера, тобто модуля, за яким наводяться результати операцій при обчисленні ключів.

При виконанні шифрування і дешифрування обчислення наводяться по модулю n. Відкритий ключ K_o і модуль n повідомляють всім, з ким припускають обмінюватися повідомленнями і використовують для шифрування даних, а секретний ключ K_c зберігають в секреті на стороні одержувача і використовують для дешифрування.

Перетворення шифрування визначає криптограму c через пару (відкритий ключ K_o, повідомлення m) у відповідності з наступною формулою:

$c = E_{K_o}(m) = m^{K_o} \bmod n$

Як алгоритм швидкого обчислення значення c використовують ряд послідовних зведень в квадрат цілого m і множень на m з приведенням по модулю n.

Звернення функції $c = m^{K_o} \bmod n$, тобто визначення значення m за відомими значеннями c, K_o і n, є завданням дискретного логарифмування і практично нездійсненно

при $n \approx 2^{512}$.

Однак завдання розшифрування криптограми c, можна легко вирішити, використовуючи секретний ключ K_c, за такою формулою:

$m = D_{K_c}(c) = c^{K_c} \bmod n$

Доведемо, що в результаті зведення криптограми c в ступінь секретного ключа K_c виходить вихідний текст m. Процес розшифрування можна записати так:

$D_{K_c}(E_{K_o}(m)) = D_{K_c}(m^e) \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n$

За умовою вибору ключів:

$e \cdot d \equiv 1 \bmod \phi(n)$ (*)

ми можемо написати, що $\exists\, k$ таке що, з урахуванням властивостей $\phi(n)$:

$e \cdot d = k \cdot \phi(n) + 1 = k\,(p-1)(q-1) + 1$ Підставимо цей вираз в показник ступеня: $m^{ed} \equiv m\,(m^{(p-1)})^{k(q-1)} = (m^{(q-1)})^{k(p-1)}$

За малої теореми Ферма ($x^{p-1} \equiv 1 \bmod p$, p - просте): $m^{ed} \bmod p \equiv m\,(1)^{k(q-1)} \bmod p = m \bmod p$

Аналогічно, замінивши p на q, отримаємо:

$m^{ed} \bmod q \equiv m\,(1)^{k(p-1)} \bmod q = m \bmod q$

Далі, по слідству з китайської теореми про залишки так як $n = p \cdot q$: $m^{ed} \bmod n = m \bmod n$, ч.т.д.

Таким чином, якщо криптограму з звести в ступінь K_c:

$c^{K_c} \bmod n = m$

то в результаті відновлюється вихідний відкритий текст m.

Саме тому для обчислення секретного ключа K_c використовують співвідношення (*).

Процедури шифрування і розшифрування в криптосистемі RSA

У реальних системах алгоритм RSA реалізується в такий спосіб. Припустимо, що

користувач А хоче передати користувачеві В повідомлення в зашифрованому вигляді, використовуючи криптосистему RSA. У такому випадку користувач А виступає в ролі відправника повідомлення, а користувач В - в ролі одержувача. Криптосистему RSA повинен сформувати одержувач повідомлення, тобто користувач В, так як в цьому випадку не буде необхідності в передачі секретного ключа. Розглянемо послідовність дій користувача В і користувача А.

Формування криптосистеми (на стороні одержувача інформації) полягає у виборі параметрів алгоритму та обчисленні пари ключів:

Користувач В вибирає два великих простих числа p і q. Це секретні параметри алгоритму, вони зберігаються в секреті на стороні одержувача.

Користувач В обчислює значення модуля n криптосистеми як результат множення перших двох чисел:

$n = p \cdot q$

Це *загальнодоступний параметр* криптосистеми. Іноді його включають у відкритий ключ.

Користувач В, знаючи секретні параметри алгоритму p і q, обчислює функцію Ейлера:

$\phi(n) = (p-1)(q-1)$

і вибирає випадковим чином просте число e як значення відкритого ключа $К_о$ з урахуванням виконання умов:

$К_о < \phi(n)$ и НОД $(К_о, \phi(n)) = 1$

Користувач В обчислює значення секретного ключа $К_с = d$ при вирішенні порівняння

$К_с \equiv К_о^{-1} \bmod \phi(n)$

Для обчислення зворотного елемента в кільці $Z_{\phi(n)}$ використовують приватний режим роботи розширеного алгоритму Евкліда для визначення НОД. Це можна здійснити, так як одержувач В знає пару простих чисел (p, q) і може легко знайти $\phi(n)$. Зауважимо, що $К_с$ і n повинні бути взаємно простими.

(В принципі відкриті і закриті ключі можна поміняти місцями, головне, щоб виконувалося співвідношення $e \cdot d \equiv 1 \bmod \phi(n)$).

Користувач В пересилає користувачеві А пару чисел $\{e, n\}$ по незахищеному каналу.

Шифрування інформації (на стороні відправника). Якщо користувач А хоче передати користувачеві В повідомлення m, він виконує наступні кроки.

Користувач А розбиває вихідний відкритий текст m на блоки m_i, $i = 1, ..., N$,

кожен з яких може бути представлений у вигляді числа меншого n $m_i \in \{0, 1, 2, ... , n-1\}$

Користувач А шифрує текст, представлений у вигляді послідовності чисел m_i за допомогою ключа $К_о = e$ за формулою:

$c_i = m_i^e \bmod n$

і відправляє криптограму $c_1, ... , c_i$ користувачу В.

Дешифрування інформації (на стороні одержувача):

Користувач В розшифровує прийняту криптограму $c_1, ... , c_i$, використовуючи секретний ключ $К_с = d$, за формулою

$m_i = c_i^d \bmod n$

В результаті буде отримана послідовність чисел, які представляють собою вихідне повідомлення m.

Таким чином, коли одержувач В, який створює криптосистему, він захищає наступні параметри:

секретний ключ Кс;

пару чисел (p, q);

Відкритий ключ $К_о$ і значення модуля n публікуються і доступні кожному, хто бажає послати власнику ключа повідомлення, яке зашифровується зазначеним алгоритмом. Після шифрування повідомлення неможливо розкрити за допомогою відкритого ключа. Власник закритого ключа без праці може розшифрувати прийняте повідомлення. Противнику ж для того,

щоб визначити значення секретного ключа K_c потрібно зуміти розкласти число n на множники p і q, щоб дізнатися про "потайний хід" і обчислити значення функції Ейлера як $\phi(n) = (p-1)(q-1)$.

Приклад. Розглянемо невеликий приклад, який ілюструє застосування алгоритму RSA. Зашифруємо повідомлення "CAB". Для простоти будемо використовувати маленькі числа (на практиці застосовуються набагато більші).

Виберемо $p = 3$ і $q = 11$.

Визначимо $n = 3 \cdot 11 = 33$.

3. Знайдемо $\phi(n) = (p-1)(q-1) = 2 \cdot 10 = 20$.

Виберемо в якості секретного ключа d довільне число взаємно просте з $\phi(n) = 20$, наприклад, $d = 3$.

Виберемо відкритий ключ $e = 7$. В якості такого числа може бути взято будь-яке число, для якого задовольняється співвідношення

$e \cdot d \mod \phi(n) = 7 \cdot 3 \mod 20 = 1$

Уявімо шифруєме повідомлення як послідовність цілих чисел за допомогою відображення: A = 1, B = 2, C = 3. Тоді повідомлення приймає вид (3,1,2). Зашифруємо повідомлення за допомогою ключа $\{e = 7, n = 33\}$:

ШТ1 = $(3^7) \mod 33 = 2187 \mod 33 = 9$,

ШТ2 = $(1^7) \mod 33 = 1 \mod 33 = 1$,

ШТ3 = $(2^7) \mod 33 = 128 \mod 33 = 29$.

Розшифруємо отримане зашифроване повідомлення (9, 1, 29) на основі закритого ключа $\{d = 3, n = 33\}$:

ІТ1 = $(9^3) \mod 33 = 729 \mod 33 = 3$,

ІТ2 = $(1^3) \mod 33 = 1 \mod 33 = 1$,

ІТ3 = $(29^3) \mod 33 = 24389 \mod 33 = 2$.

Надійність RSA. RSA багато років протистоїть інтенсивному криптоаналізу. Доведено, що розкриття шифру RSA еквівалентно рішенню завдання факторизації великих (100-200 двійкових розрядів) чисел. Важливо, що в цьому завданні для будь-якої довжини ключа можна дати нижню оцінку числа операцій для розкриття шифру, а з урахуванням продуктивності сучасних комп'ютерів оцінити і необхідний на це час.

Можливість гарантовано оцінити захищеність алгоритму RSA стала однією з причин популярності цієї СВК на фоні десятків інших схем. Тому алгоритм RSA використовується в банківських комп'ютерних мережах, особливо для роботи з віддаленими клієнтами (обслуговування кредитних карток). В даний час алгоритм RSA активно реалізується як у вигляді самостійних криптографічних продуктів, так і в якості вбудованих засобів в популярних додатках, а також використовується в багатьох стандартах.

Приклади завдань

1. Пояснити, які саме дві практичні проблеми симетричної криптографії привели до появи асиметричних криптосистем (розподіл ключів і цифровий підпис) та навести по одному прикладу для кожної.
2. Для заданих простих p, q обчислити n=pq, $\varphi(n) = (p-1)(q-1)$, вибрати e так, щоб $\gcd(e, \varphi(n)) = 1$, і знайти $d = e^{-1} \bmod \varphi(n)$ розширеним алгоритмом Евкліда.
3. Для заданого повідомлення m<n виконати шифрування $c = m^e \bmod n$ та розшифрування $m = c^d \bmod n$, використовуючи швидке піднесення до степеня за модулем.
4. Пояснити, чому без автентифікації відкритого ключа можливе перехоплення обміну (підміна ключа), і запропонувати загальну ідею захисту через сертифікати/PKI.
5. Порівняти роль RSA у «гібридній схемі» (RSA для обміну ключем + симетричний шифр для даних) з використанням лише RSA для шифрування великих повідомлень за критеріями швидкодії та практичності.

Контрольні питання

1. У чому різниця між симетричною та асиметричною криптографією з погляду ключів, каналів розподілу ключів і типових сценаріїв застосування?
2. Що таке одностороння функція та одностороння функція з «пасткою» (trapdoor), і яку роль у RSA відіграють p, q як «пастка»?
3. Чому в RSA обчислення $\varphi(n)$ легко виконати, якщо відомі p і q, але важко, якщо відомий лише n?
4. Навіщо потрібна умова $\gcd(e, \varphi(n)) = 1$ і як пов'язане рівняння $ed \equiv 1 \pmod{\varphi(n)}$ з коректністю розшифрування?
5. Які математичні факти (теорема Ейлера/мала теорема Ферма та китайська теорема про залишки) використовуються для обґрунтування $(m^e)^d \equiv m \pmod{n}$?

Література

1. Diffie W., Hellman M. E. New directions in cryptography // IEEE Transactions on Information Theory. — 1976. — Vol. 22, No. 6. — P. 644–654.
2. Rivest R. L., Shamir A., Adleman L. A method for obtaining digital signatures and public-key cryptosystems // Communications of the ACM. — 1978. — Vol. 21, No. 2. — P. 120–126.
3. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. — Boca Raton : CRC Press, 1996.
4. Katz J., Lindell Y. Introduction to Modern Cryptography. — 2nd ed. — Boca Raton : CRC Press, 2014.
5. Moriarty K., Kaliski B., Jonsson J., Rusch A. PKCS #1: RSA Cryptography Specifications Version 2.2 (RFC 8017). — Internet Engineering Task Force, 2016.

Лекція №9. Асиметричні криптосистеми: криптосистема Ель-Гамаля; метод експоненціального ключового обміну Діффі-Хеллмана

Дана система є альтернативою RSA і при рівному значенні ключа забезпечує ту ж криптостійкість. Однак спільної думки з приводу переваги того чи іншого методу немає.

На відміну від RSA метод Ель-Гамаля заснований на проблемі дискретного логарифма. Цим він схожий на *алгоритм Діффі-Хелмана*. Якщо зводити число в ступінь в кінцевому полі досить легко, то відновити аргумент за значенням (тобто знайти логарифм в множині цілих чисел) досить важко.

Основу системи складають параметри p і g - числа, перше з яких p - просте, а друге ($g \in Z_p$) - ціле. Дані параметри не є секретними і можуть бути загальними для групи користувачів. У реальних схемах шифрування необхідно використовувати в якості модуля велике ціле просте число, що має в двійковому поданні довжину 512 ... 1024 біт.

Секретний ключ $x \in Z_{p-1}$ генерується випадковим чином, а відкритий ключ y обчислюється за формулою:

$y = g^x \bmod p$

Для шифрування повідомлення m спочатку вибирається випадкове число k, взаємно просте з p-1, яке називають також сеансовим ключем.

Шифртекст є пара чисел (a, b), що обчислюється за формулами: $a = g^k \bmod p$

$b = y^k m \bmod p$

Таким чином, шифртекст в два рази довший відкритого тексту. Для дешифрування обчислюється:

$$m = \frac{b}{a^x} \bmod p$$

Перетворення можна зупинити, так як:

ax= gkx mod p

$$\frac{b}{a^x} \equiv \frac{y^k m}{a^x} \equiv \frac{g^{xk} m}{a^x} = m \mod p$$

Число k називають також рандомізатором. Його використання означає, що тут реалізований багатозначний шифр заміни. При цьому для шифрування різних блоків (чисел) відкритого тексту необхідно використовувати різні значення рандомізатора. При використанні одного і того ж значення відповідні шифртекст (a, b) і (a', b'), отримані для блоків відкритих текстів m і m', пов'язані співвідношенням:

$b(b')^{-1} = m(m')^{-1}$

і текст m' можна обчислити, якщо відомий текст m.

Стійкість криптосистеми Ель Гамаля заснована на складності завдання логарифмування в мультиплікативній групі кінцевого простого поля. Ця криптосистема може бути узагальнена для застосування в будь-якій кінцевій циклічної групи. В якості такої групи, крім розглянутої Z^*_p, найчастіше використовується мультиплікативна група кінцевого поля GF (2m) і група точок на еліптичній кривій над кінцевим полем.

Алгоритм не запатентований, але потрапляє під дію патенту на метод експоненціального ключового обміну Діффі-Хеллмана, розглянутий нижче. Перетворення шифрування/дешифрування Ель Гамаля по суті те ж саме, що ключовий обмін по Діффі-Хеллману, за винятком того, що y - це частина ключа, а при шифруванні повідомлення множиться на y^k. Алгоритм цифрового підпису DSA, розроблений NIST (National Institute

of Standard and Technology USA) і є частиною стандарту DSS частково спирається на розглянутий метод.

Комбінований метод шифрування

Головною перевагою криптосистем з відкритим ключем є їх потенційно висока безпека: немає необхідності ні передавати, ні повідомляти будь-кому значення секретних ключів, ні переконуватися в їх справжності. У симетричних криптосистемах існує небезпека розкриття секретного ключа під час передачі.

Однак алгоритми, що лежать в основі криптосистем з відкритим ключем, мають такі недоліки:

генерація секретних і відкритих ключів заснована на генерації великих простих чисел, а перевірка простоти чисел займає багато процесорного часу;

процедури шифрування і розшифрування, пов'язані зі зведенням до ступеня багатозначного числа, досить трудомісткі.

Тому швидкодію (швидкість шифрування і дешифрування) в криптосистемах з відкритим ключем зазвичай в сотні і тисячі разів менше швидкодії симетричних криптосистем з секретним ключем.

Комбінований метод шифрування дозволяє поєднувати переваги високої секретності, що надаються асиметричними криптосистемами з відкритим ключем, з перевагами високої швидкості роботи, властивими симетричним криптосистемам з секретним ключем. При такому підході асиметричні алгоритми шифрування застосовуються для шифрування, передачі і подальшого розшифрування тільки секретного ключа симетричної криптосистеми. А симетрична криптосистема застосовується для шифрування і передачі вихідного відкритого тексту. В результаті асиметричні алгоритми шифрування не замінює симетричну криптосистему з секретним ключем, а лише доповнює її, дозволяючи підвищити в цілому захищеність переданої інформації.

Користувачі А і В, які використовують комбінований метод шифрування, мають кожен по парі асиметричних ключів шифрування

($К_{Ао}$, $К_{Ас}$) и ($К_{Во}$, $К_{Вс}$)

Якщо користувач А хоче передати зашифроване комбінованим методом повідомлення m користувачеві В, то порядок його дій буде такий.

Створити (наприклад, згенерувати випадковим чином) симетричний ключ, званий в цьому методі сеансовим ключем K_s.

Зашифрувати повідомлення *m* на сеансовому ключі K_s.

Зашифрувати сеансовий ключ K_s на відкритому ключі K_{Bo} користувача B і своєму таємному ключі K_{Ac}.

Передати по відкритому каналу зв'язку на адресу користувача B зашифроване повідомлення разом із зашифрованим сеансовим ключем.

Вирішення B при отриманні зашифрованого повідомлення і зашифрованого сеансового ключа повинні бути зворотними:

Розшифрувати на своєму таємному ключі K_{Bc} і відкритий ключ K_{Ao} користувача A сеансовий ключ K_s.

За допомогою отриманого сеансового ключа K_s розшифрувати і прочитати повідомлення *m*.

При використанні комбінованого методу шифрування можна бути впевненим в тому, що тільки користувач B зможе правильно розшифрувати ключ K_s і прочитати повідомлення *m*.

Вибір довжини ключів в комбінованому методі шифрування. Таким чином, при комбінованому методі шифрування застосовуються криптографічні ключі як симетричних, так і асиметричних криптосистем. Очевидно, вибір довжин ключів для кожного типу криптосистеми слід здійснювати таким чином, щоб зловмисникові було однаково важко атакувати будь-який механізм захисту комбінованої криптосистеми.

У таблиці 9.1. наведені поширені довжини ключів симетричних і асиметричних криптосистем, для яких труднощі атаки повного перебору приблизно дорівнює складності факторизації відповідних модулів асиметричних криптосистем.

Таблиця 9.1. Довжини ключів для симетричних і асиметричних криптосистем при однаковій їх криптостійкості

Симетричні криптосистеми	56	64	80	112	128
Асиметричні криптосистеми	384	512	786	1792	2304

Метод експоненціального ключового обміну Діффі-Хеллмана

Одні з авторів ідеї криптосистем з відкритим ключем, Діффі і Хеллмана запропонували нову ідею - відкритий розподіл ключів.

Вони задалися питанням: чи можна організувати таку процедуру взаємодії абонентів A і B по відкритих каналах зв'язку, щоб вирішити такі завдання:

спочатку у A і B немає ніякої спільної секретної інформації, але в кінці процедури така загальна секретна інформація (загальний ключ) у A і B з'являється, т. Е. Виробляється;

пасивний супротивник, який перехоплює все передачі інформації і знає, що хочуть отримати A і B, проте не може відновити вироблений загальний ключ A і B.

Метод отримав назву методу експоненціального ключового обміну. Він був першою криптосистемою з відкритим ключем, хоча для обміну ключами можна використовувати будь-які асиметричні алгоритми шифрування, наприклад, той же алгоритм RSA.

Крипостійкість даного методу визначається трудомісткістю обчислення дискретного логарифма:

$f(x) = GX \bmod p$,

де p - просте число, x - довільне натуральне число, g - деякий примітивний елемент поля Галуа GF (p). Загальновизнано, що інвертування функції GX mod p, тобто дискретного логарифмування, є важкою математичної завданням.

Сама процедура або протокол вироблення загального ключа полягає в наступному. Значення p і g є загальнодоступними параметрами протоколу. Абоненти A і B незалежно один від одного випадково вибирають по одному великому натуральному числу Xa і xB. Це їх секретні ключі. Далі кожен з них обчислює відкриті ключі:

ya = gxA mod p і УВ = gxB mod p.

Потім вони обмінюються цими елементами по каналу зв'язку. Тепер абонент А, отримавши Yв і знаючи свій секретний елемент ХА, обчислює новий елемент:

yBxA mod p = (gxB) Xa mod p. Аналогічно надходить абонент В: yAxB mod p = (gxA) xB mod p.

Тим самим у А і В з'явився спільний елемент поля, рівний gxA xB. Цей елемент і оголошується загальним ключем абонентів А і В.

Незворотність перетворення в цьому випадку забезпечується тим, що досить легко обчислити показову функцію в кінцевому поле Галуа, що складається з р елементів (р - просте число). Зворотній завдання обчислення х з у буде досить складною. Якщо р вибрано досить правильно, то витяг логарифма потребують обчислень, пропорційних L (p) = exp {(ln p ln ln p) 0.5}.

При всій простоті алгоритму Діффі-Хелмана його недоліком в порівнянні з системою RSA є відсутність гарантованої нижньої оцінки трудомісткості розкриття ключа.

Алгоритми практичної реалізації криптосистем з відкритим ключем

Зведення в ступінь за модулем *m*

У криптографії використовуються обчислення за mod *m*, так як алфавіт будь-якої криптографічного системи є кінцева множина цілих чисел. Арифметика відрахувань до того ж легше реалізується на комп'ютерах, оскільки вона обмежує діапазон проміжних значень і результату. Для *k*-бітових відрахувань *m* проміжні результати будь-якого додавання, віднімання або множення будуть не довше, ніж 2*k* біт. Тому в арифметиці відрахувань ми можемо виконати зведення в ступінь без величезних проміжних результатів. Обчислення ступеня деякого числа по модулю іншого числа,

$a^d \bmod m$

є просто послідовність множень і поділок, але існують прийоми, що прискорюють цю дію. Один з таких прийомів прагне мінімізувати кількість множень по модулю, інший - оптимізувати окремі множення по модулю. Так як операції дистрибутивні, швидше виконати зведення в ступінь як потік послідовних множень, щоразу отримуючи відрахування.

Наприклад, для того, щоб обчислити $a^8 \bmod m$, не виконуйте сім множень і одне приведення по модулю; замість цього виконайте три менших множення і три менших приведення по модулю:

$a^8 \bmod m = ((a^2 \bmod m)^2 \bmod m)^2 \bmod m$

Точно також, $a^{16} \bmod m = (((a^2 \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m$.

Обчислення $a^d \bmod m$, де *d* не є ступенем 2, не набагато важче. Двійковий запис являє *x* у вигляді суми ступенів 2: число 25 - це бінарне 11001, тому 25 = 16 + 8 + 1. Тоді:

$a^{25} \bmod m = (a\, a^8\, a^{16}) \bmod m = (a*(((\, a*a^2)^{\,2})^{\,2})^{\,2}) \bmod m$

З продуманим збереженням проміжних результатів нам знадобиться тільки шість множень:

$(((((((a^2 \bmod m)*a)^2 \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m)^2 \bmod m\ *a) \bmod m$

Такий прийом називається методом двійкових квадратів і множень. Він використовує простий і очевидний ланцюжок складань, в основі якої лежить двійкове уявлення числа. Збільшення швидкості обчислень при множенні 200-бітових чисел буде дуже помітним.

Алгоритм обчислення $a^d \bmod m$

Нехай натуральні числа *a* і *d* не перевищують за величиною *m*.

Уявімо *d* в двійковій системі числення:

$d = d_o 2^r + \ldots + d_{r-1} 2 + d_r$

де *r* - число двійкових розрядів, d_i - цифри в двійковому поданні, рівні 0 або 1, і $d_o = 1$.

Покладемо $a_o = a$ і потім для $i = 1, \ldots r$ обчислимо:

$a_i \equiv a^2 \quad a^{di} \bmod m$

Тоді *ar* є бажаємий лишок $a^d \bmod m$.

Справедливість цього алгоритму випливає з легко доказуваного індукцією по *i*

порівняння:

$a_i \equiv a^{do2i+\ldots+di} \bmod m$

Так як кожне обчислення на кроці 2 вимагає не більше трьох множень по модулю m і цей крок виконується $r \le \log_2 m$ раз, то складність алгоритму можна оцінити величиною 0 (ln m). Кажучи про складність алгоритмів, ми маємо на увазі кількість арифметичних операцій.

Другий алгоритм - це класичний алгоритм Евкліда обчислення найбільшого загального дільника цілих чисел. Ми припускаємо заданими два натуральних числа a і b і обчислюємо їх найбільший спільний дільник НСД (a, b).

Алгоритм Евкліда обчислення НСД

Одним із способів обчислення НСД двох чисел є алгоритм Евкліда, який описав його з своїй книзі «Начала» близько 300 років до н.е. Вважають, що він не винайшов його, а сам алгоритм ще старше років на 200. Це найдавніший нетривіальний алгоритм, який актуальний і в наш час.

●Обчислимо r - залишок від ділення числа a на b ($a > b$):

$a = b\,q + r,\ 0 \le r < b$

Якщо $r = 0$, то b є шукане число.

Якщо $r \neq 0$, то замінимо пару чисел (a, b) парою (b, r) і перейдемо до кроку 1.

Зупинка гарантується, оскільки залишки від поділів утворюють строго спадну послідовність.

Оцінку складності цього алгоритму дає наступна теорема.

Теорема. При обчисленні найбільшого загального дільника НЗД (a, b) за допомогою алгоритму Евкліда буде виконано не більше 5p операцій поділу із залишком, де p є кількість цифр в десяткового запису меншого з чисел a і b.

Обчислення зворотних величин в кільці цілих чисел

При обчисленні ключів в асиметричних криптографічних системах необхідно знаходити зворотні елементи в кільці цілих чисел Z_m. Елемент $x \in Z_m$ є зворотним до $a \in Z_m$, ($x = a^{-1}$), якщо

$x \cdot a \equiv 1 \bmod m$ или $a^{-1} \equiv x \bmod m$

У загальному випадку це порівняння може не мати рішень або мати кілька рішень. Воно має єдине рішення тоді і тільки тоді, якщо числа a і m взаємно прості, тобто НСД (a, m) = 1.

Розглянемо основні способи знаходження зворотних величин в Z_m.

Метод перебору. Перевірити по черзі значення $x \in$ (1, 2, ... m-1), поки не виконається порівняння $x \cdot a \equiv 1 \bmod m$.

Якщо відома функція Ейлера (для RSA) $\phi(m)$, то можна обчислити $a^{-1} \bmod m \equiv a^{\phi(m)-1} \bmod m$, використовуючи алгоритм швидкого зведення в ступінь. (Це випливає з твердження теореми Ейлера: якщо a і m взаємно прості, то $a^{\varphi(m)} \equiv 1 \bmod m$.)

Знаходження зворотної величини $a^{-1} \bmod m$ за допомогою розширеного алгоритму Евкліда.

Алгоритм Евкліда можна узагальнити способом, який має велике практичне значення. При цьому способі під час обчислення НСД можна водночас обчислити такі цілі числа x і y, що

$a \cdot x + b \cdot y =$ НСД (a, b)

Цей варіант називається розширеним алгоритмом Евкліда. Для обчислення зворотної величини використовується приватний режим роботи алгоритму Евкліда, при якому:

$b = m$ и НСД (a, m) = 1

Справді порівняння $x \cdot a \equiv 1 \bmod m$ означає, що $\exists$ ціле k, що вірно:

$a\,x + m\,k = 1 \bmod m$

Це завдання рівносильне пошуку цілих рішень рівняння

$ax + mk = 1$

Алгоритм пошуку цілих рішень рівняння a x + m k = 1

Трохи підправив алгоритм Евкліда, можна досить швидко вирішувати порівняння

$ax + mk = 1 \bmod m$

за умови, що НСД (a, m) = 1.

Визначимо матрицю E = I - одинична матриця розміром 2x2.

Обчислимо r - залишок від ділення числа a на m:

$a = m\,q + r,\ 0 \le r < m$

Якщо $r = 0$, то другий стовпець матриці E дає вектор $[x, k]^T$ рішень рівняння.

Якщо $r \neq 0$, то замінимо матрицю E матрицею

$$E = E \bullet \begin{pmatrix} 1 & 0 \\ 0 & -q \end{pmatrix}$$

Замінимо пару чисел (a, m) парою (m, r) і перейдемо до кроку 1.

Три наведених вище алгоритму відносяться до розряду так званих поліноміальних алгоритмів. Ця назва носить алгоритми, складність яких оцінюється зверху ступеневим чином в залежності від довжини записи вхідних чисел. Якщо найбільше з чисел, що подаються на вхід алгоритму, не перевищує m, то складність алгоритмів цього типу оцінюється величиною O ($ln^c m$), де c - деяка абсолютна постійна. У всіх наведених вище прикладах c = 1.

Поліноміальні алгоритми в теорії чисел - велика рідкість. Та й оцінки складності алгоритмів найчастіше спираються на будь-які недоведені, але правдоподібні гіпотези, зазвичай пов'язані з аналітичною теорією чисел.

Генерація простих чисел

Для алгоритмів з відкритими ключами потрібні прості числа. Їх потрібно безліч для будь-якої досить великої мережі, чи вистачить їх?

Так, існує приблизно 10^{151} простих чисел довжиною до 512 біт включно. Для чисел, близьких m, ймовірність того, що випадково вибране число виявиться простим, дорівнює $1/\ln m$. Тому повне число простих чисел, менших m, так само $m/(\ln m)$. При виборі з 10^{151} простих чисел ймовірність збігу вибору практично дорівнює нулю.

Але якщо так складно розкладання на множники, як може бути простим генерація простих чисел? Справа в тому, що відповісти "так" або "ні" на питання "чи є число простим?" набагато простіше, ніж відповісти на більш складне питання "які множники m?"

Генерація випадкових чисел з подальшою спробою розкладання їх на множники - це неправильний спосіб пошуку простих чисел. Існують різні імовірнісні перевірки на простоту чисел, що визначають, чи є число простим, із заданою ступенем достовірності. За умови, що ця «ступінь достовірності» досить велика, такі способи перевірки досить хороші.

Пропонуються наступні тести перевірки чисел на простоту:

Тест Леманна (Lehmann)

Ось послідовність дій при перевірці простоти числа p;

Виберіть випадково число a, менше p.

Розрахуйте $a^{(p-1)/2} \bmod p$.

Якщо $a^{(p-1)/2} \neq 1$ або $-1 \bmod p$, то p не є простим.

Якщо $a^{(p-1)/2} \equiv 1$ або $-1 \bmod p$, то ймовірність того, що число p не є простим, не більше ніж 50 відсотків.

Число a, яке не показує, що число p не є простим числом, називається свідком. Якщо p-складене число, ймовірність випадкового числа a бути свідком складовою природи числа p не менше 50 відсотків. Повторіть цю перевірку t раз з t різними значеннями a. Якщо результат обчислень дорівнює 1 або -1, але не завжди дорівнює 1, то p є простим числом з ймовірністю помилки $1/2^t$.

Алгоритм Рабіна-Міллера (Rabin-Miller)

Повсюди використовується простий алгоритм, розроблений М. Рабіном, частково

заснованим на ідеях Міллера.

Виберіть для перевірки випадкове число p. Обчисліть b - число поділок p-1 на 2 (тобто $2b$ - це найбільша ступінь числа 2, на яку ділиться p-1). Потім обчисліть m, таке, що $p = 1 + 2b \cdot m$.

Виберіть випадкове число a, менше p.

Встановіть $j = 0$ і $z = a \cdot m \bmod p$.

Якщо $z = 1$ або якщо $z = p$-1, то p проходить перевірку і може бути простим числом.

Якщо j > 0 і Z = 1, то p не є простим числом.

Встановіть $j = j + 1$. Якщо $j < b$ і $z < p$ - 1, встановіть $z = z^2 \bmod p$ і поверніться на етап (4). Якщо $z = p$ - 1, то p проходить перевірку і може бути простим числом.

Якщо $j = b$ і $z \neq p$ - 1, то p не є простим числом.

У цьому тесті ймовірність проходження перевірки складовим числом спадає швидше, ніж в попередніх. Гарантується, що три чверті можливих значень a виявляться свідками. Це означає, що складене число прослизне через t перевірок з ймовірністю, що не більшою $1/4^t$, де t - число ітерацій. Насправді і ці оцінки дуже песимістичні. Для більшості випадкових чисел близько 99.9 відсотків можливих значень є свідками.

Практичні міркування

У реальних додатках генерація простих чисел відбувається швидко. Алгоритм наступний:

Згенеруйте випадкове n-бітове число p.

Встановіть старші і молодші біти рівними 1. (Старший біт гарантує необхідну довжину простого числа, а молодший біт забезпечує його непарність.)

Переконайтеся, що p не ділиться на невеликі прості числа: 3, 5, 7, 11 і т.д. У багатьох реалізаціях перевіряється подільність p на всі прості числа, менші 256. Найбільш ефективною є перевірка на подільність для всіх простих чисел, менших 2000.

Виконайте тест Рабіна-Міллера для деякого випадкового a. Якщо p проходить тест, згенеруйте інше випадкове a й повторіть перевірку. Обирайте невеликі значення a для прискорення обчислень. Виконайте п'ять тестів. (Одного може здатися достатнім, але виконайте п'ять.) Якщо p не проходить однієї з перевірок, згенеруйте інше p і спробуйте знову.

Інакше, можна не генерувати p випадковим чином кожен раз, але послідовно перебирати числа, починаючи з випадково обраного до тих пір, поки не буде знайдено просте число.

Етап 3 не є обов'язковим, але це хороша ідея. Перевірка, що випадкове непарне p не ділиться на 3, 5 і 7 відсікає 54% непарних чисел ще до етапу 4. Перевірка подільності на всі прості числа, менші 100, прибирає 76% непарних чисел, перевірка подільності на всі прості числа, менші 256, прибирає 80% непарних чисел. У загальному випадку, частка непарних кандидатів, які не діляться ні на одне просте число, менше m, дорівнює $1.12/1n$

Чим більше перевіряється m, тим більше попередніх обчислень потрібно виконати до тесту Рабіна-Міллера.

Сильні прості числа

Якщо m - добуток двох простих чисел p і q, то може знадобиться використовувати в якості і *q сильні прості числа*.

Такі прості числа мають ряд властивостей, які ускладнюють розкладання добутку m певними методами розкладання на множники. Серед таких властивостей були запропоновані:

Найбільший спільний дільник p-1 і q-1 повинен бути невеликим.

І p-1, і q-1 повинні мати серед своїх множників великі прості числа, відповідно p' *і* q'

І p'-1, і q'-1 повинні мати серед своїх множників великі прості числа.

І $p + 1$, і $q + 1$ повинні мати серед своїх множників великі прості числа.

І (p-1)/2, і (q-1)/2 повинні бути простими (зверніть увагу, при виконанні цієї умови виконуються і два перших).

Наскільки суттєво застосування саме сильних простих чисел залишається предметом триваючих суперечок. Ці властивості були розроблені, щоб затруднити виконання ряду старих алгоритмів розкладання на множники. Однак найшвидші алгоритми однаково швидкі при розкладанні на множники будь-яких чисел, як задовольняють наведеним умовам, так і ні.

Деякі автори виступають проти спеціальної генерації сильних простих чисел, стверджуючи, що довжина простих чисел набагато важливіше їх структури. Більш того, сама структура зменшує випадковість числа і може знизити стійкість системи.

Але все може змінитися. Можуть бути створені нові методи розкладання на множники, які краще працюють з числами, що володіють певними властивостями. У цьому випадку знову можуть знадобитися сильні прості числа.

Приклади завдань

1. Для заданих p, g, x обчислити відкритий ключ Ель-Гамаля $y = g^x \bmod p$ та пояснити, чому x не відновлюється практично з y (проблема дискретного логарифма).
2. Виконати шифрування Ель-Гамаля для повідомлення m з випадковим k: знайти $a = g^k \bmod p, b = y^k \cdot m \bmod p$, а потім розшифрувати $m = b \cdot (a^x)^{-1} \bmod p$.
3. Показати на прикладі, що повторне використання одного й того самого k для двох повідомлень у Ель-Гамалі призводить до витоку співвідношення між відкритими текстами (через $b \cdot (b')^{-1}$).
4. Реалізувати обмін ключем Діффі-Хеллмана: обчислити $y_A = g^{x_A} \bmod p, y_B = g^{x_B} \bmod p$ і спільний ключ $K = y_B^{x_A} \bmod p = y_A^{x_B} \bmod p$.
5. Скласти схему комбінованого шифрування: згенерувати сеансовий симетричний ключ $K_я$, зашифрувати ним повідомлення, а $K_я$ — передати/захистити за допомогою Ель-Гамаля або DH-узгодження.

Контрольні питання

1. На якій математичній задачі базується криптостійкість Ель-Гамаля та Діффі-Хеллмана, і чим ця основа відрізняється від RSA?
2. Чому в Ель-Гамалі шифртекст має вигляд пари (a,b) і чому це робить його «рандомізованим» (ймовірнісним) шифром?
3. Які наслідки має повторне використання рандомізатора k в Ель-Гамалі, і як цього уникати на практиці?
4. У чому полягає вразливість протоколу Діффі-Хеллмана до атаки «людина посередині» без автентифікації, і який загальний механізм її усуває?
5. Чому комбінований (гібридний) підхід є стандартним у реальних системах: які операції «дорогі» в асиметрії та що саме асиметрія захищає в гібридній схемі?

Література

1. Diffie W., Hellman M. E. New directions in cryptography // IEEE Transactions on Information Theory. — 1976. — Vol. 22, No. 6. — P. 644–654.
2. ElGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE Transactions on Information Theory. — 1985. — Vol. 31, No. 4. — P. 469–472.
3. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. — Boca Raton : CRC Press, 1996.
4. National Institute of Standards and Technology. Digital Signature Standard (DSS) : FIPS PUB 186-5. — Gaithersburg : NIST, 2023.
5. Boneh D., Shoup V. A Graduate Course in Applied Cryptography. — 2020. — (електронний навчальний посібник).

Лекція №10 Електронний цифровий підпис. Хеш-функції. Алгоритми електронного цифрового підпису (RSA, EGSA, DSA)

В останні роки повсюдно простежується тенденція перекладу всього документообігу в електронну форму. Основна проблема, яка при цьому виникає - це встановлення автентичності автора і відсутності змін в отриманому документі тобто проблема аутентифікації як автора документа так і самого документа. У звичайному паперовому діловодстві ці проблеми вирішуються за рахунок того, що інформація в документі і рукописний підпис автора жорстко пов'язані з фізичним носієм, тобто папером. В електронних документах на машинних носіях такого зв'язку немає.

Принципово новим рішенням є електронний цифровий підпис (ЕЦП), який використовується для аутентифікації документа, переданого телекомунікаційними каналами. Функціонально він аналогічний звичайному рукописному підпису і володіє його основними перевагами:

гарантує цілісність підписаного тексту, тобто відсутність виправлень і підчисток;

засвідчує, що підписаний текст виходить від особи, яка його поставила;

не дає цього самого обличчя можливості відмовитися від зобов'язань, пов'язаних з підписаним текстом.

Інакше кажучи, вона забезпечує аутентифікацію і цілісність повідомлень, тобто гарантує, що повідомлення надходять від достовірного відправника і в неперекрученому вигляді. Більш того, одержувач повідомлення не тільки переконується в його достовірності, а й отримує електронний підпис, який в подальшому може використовуватися як доказ достовірності повідомлення третім особам (арбітру) в тому випадку, якщо відправник згодом спробує відмовитися від свого підпису. Тут є повна аналогія звичайного підпису на папері, за винятком того, що під арбітром зазвичай розуміється технічний експерт, який дає висновок про достовірність електронного підпису, тобто виконує функцію, аналогічну функції графолога в разі звичайного підпису. Застосуванням схеми електронного підпису може бути наприклад платіжне доручення при безготівкових розрахунках з пластикових карт.

В якості підписаного документа може бути використаний будь-який файл. Підписаний файл створюється з непідписаного шляхом додавання в нього однієї або більше електронних підписів.

Кожен підпис містить наступну інформацію:

дату підпису і термін закінчення дії ключа даного підпису;

ідентифікатор, хто саме підписав (ім'я відкритого ключа);
інформацію про особу, яка підписала файл;
власне цифровий підпис.

Цифровий підпис на основі симетричних криптосистем

Для реалізації схеми цифрового підпису зазвичай вибирається будь-яка криптосистема, частіше асиметрична, але можна використовувати і симетричну. Такі криптосистеми використовуються рідше в протоколах цифрового підпису, так як вимагають залучення так званого посередника - абонента, який заслуговує на безумовну довіру всіх сторін. Розглянемо спочатку таку схему.

Нехай відправник - абонент А хоче підписати цифрове повідомлення і послати його адресату - абоненту В. Використовуючи симетричний шифр $E = \{E_k\}$, він може це зробити за допомогою посередника - абонента С, що має ключ k_1 для секретного зв'язку з відправником А, і ключ k_2 для секретного зв'язку з адресатом В. Ці ключі повинні бути встановлені заздалегідь до початку протоколу і можуть використовуватися багаторазово для декількох підписів.

Викладемо протокол по крокам.

Відправник А шифрує з використанням ключа k_1 повідомлення m для адресата В і відправляє криптограму c посереднику, де

$E_{k1}(m) = c$

Посередник розшифровує криптограму c, використовуючи ключ k_1: $E_{k1}^{-1}(c) = m$

Посередник С формує блок інформації $I_A = [m, c, p_A]$, що складається з розшифрованого повідомлення m, копії криптограми c і інформаційного блоку p_A, що підтверджує, що повідомлення виходило від А (ідентифікатор абонента А). Використовуючи ключ k_2, він шифрує блок I_A і посилає криптограму d одержувачу В, де $E_{k2}(I_A) = d$.

Абонент В розшифровує криптограму d, використовуючи ключ k_2: $E_{k2}^{-1}(d) = I_A = [m,c,p_A]$

Тепер він може прочитати повідомлення m та сертифікат посередника p_A про те, що його відправив А.

До переваг протоколу слід віднести наступне:

Тільки посередник і відправник А спільно володіють секретним ключем k_1, тому тільки А міг відправити посереднику повідомлення, зашифроване ключем k_1 (підпис не підробляється), і підтвердження посередником цього підпису достовірно. Якщо активний противник спробує видати себе за відправника А, посередник виявить це на кроці 2 і не відправить повідомлення адресату В.

Цей підпис не тиражується і підписаний документ не змінюється. Якби В спробував сертифікат посередника p_A об'єднати з неправдивим повідомленням m', тобто заявити, що він отримав блок з повідомленням m' і сертифікатом p_A, то посередник виявив би цю невідповідність. Він запросив би у В це неправдиве повідомлення m' і криптограму

c. Зашифрувавши неправдиве повідомлення m' на ключі k_1, посередник виявив би, що $E_{k1}(m') \neq c$, тобто обчислена криптограма не збігається з криптограмою c, яку йому надіслав В. Звичайно, В не міг би пред'явити криптограму c', яка узгоджується з m', тому що він не знає ключа k_1.

Від цього підпису не можна відректися. Якщо відправник А пізніше заявить, що він не посилав повідомлення m, то те що зберігається у В сертифікат посередника p_A разом з повідомленням m засвідчать зворотне.

Таким чином, даний протокол усуває недоліки, властиві традиційному підпису на паперовому документі.

Розглянемо ще один протокол за участю посередника, завдання якого - забезпечити пред'явлення одержувачем В третій особі D підписаного документа m, отриманого ним від відправника А.

Абонент В об'єднує в блок $I = [m, p_A]$ отримане повідомлення m з сертифікатом посередника p_A і шифрує блок I на ключі k_2: $E_{k2}(I_A) = w$,

після цього В посилає криптограму w посереднику С (для пересилки третій особі

D).

Посередник, використовуючи ключ k_2, розшифровує криптограму w: $E_{k2}^{-1}(w) = I$

Посередник С шифрує блок I на секретному ключі k_3, що використовується для зв'язку з D, і посилає D криптограму v: $E_{k3}(I) = v$.

D розшифровує криптограму v, використовуючи k_3: $E_{k3}^{-1}(v) = I = [m, p_A]$

Тепер В може прочитати і повідомлення m, і сертифікат посередника p_A, який свідчить, що його послав А.

Ці протоколи в принципі здійсненні, але вони вимагають високопродуктивної та безпомилкової роботи посередника. Він повинен постійно розшифровувати і зашифровувати повідомлення між кожною парою абонентів, які бажають відправити один одному підписаний документ. Посередник є вузьким місцем в такій системі зв'язку, навіть якщо він - бездушна комп'ютерна програма. Посередник повинен бути непогрішний, бо навіть єдина помилка в мільйон підписів підриває довіру до нього. Крім того, він повинен працювати в цілковитій безпеці. Якщо його база даних по секретним ключам стане доступною зловмисникові або хтось спотворить його ключі, функціонування системи буде порушено. Цей протокол хороший швидше в теорії, ніж на практиці.

Щоб зловмисник не зміг скористатися багаторазово підписаним документом (наприклад, грошовим переказом), призначеним для разового використання, в документ слід додати інформаційний блок, де зафіксований час і дата підписання документа, так звана мітка часу. Крім того, мітка часу на документі знижує ризик відмови учасника від свого підпису (якщо той хто підписав заявить, що у нього вкрали закритий ключ і документ підписаний не їм). Відповідний протокол з посередником передбачає наступні кроки:

Абонент А підписує повідомлення.

Абонент А генерує заголовок, що містить таку нормативну інформацію, потім додає заголовок до підписаного повідомлення, підписує всі разом і відправляє посереднику.

Посередник перевіряє справжність зовнішньої підписи і підтверджує таку нормативну інформацію. Далі він проставляє позначку часу на повідомленні і на ідентифікаційній інформації. Потім він підписує весь пакет і відсилає його як автору підпису А, так і адресату В.

Абонент В перевіряє справжність підпису посередника, ідентифікаційну інформацію і справжність підпису А.

Абонент А перевіряє справжність повідомлення, яке посередник послав В. Якщо А не визнає свого авторства, він негайно заявляє про це.

Цифровий підпис на основі асиметричних криптосистем

Розглянемо більш практичні протоколи цифрового підпису, що використовують асиметричні криптосистеми.

Взагалі кажучи, в криптосистемах з відкритим ключем в якості цифрового підпису може використовуватися результат шифрування документа на закритому ключі. Кожен, хто має відкритий ключ, може скористатися ним для розшифрування і тим самим перевірити достовірність підпису. Такий протокол відповідає всім вимогам цифрового підпису та не потребує посередника.

Але головна проблема полягає в тому, що підписання великих документів за допомогою їх шифрування асиметричним алгоритмом неефективно через невисоку швидкість шифрування. Тим більше, що в багатьох випадках сам документ в шифруванні не потребує, так як не містить секретної інформації, проте необхідно забезпечити його цілісність і підтвердження авторства. Тому, як правило, підписують шляхом асиметричного шифрування не сам документ, а отриману з нього згортку, звану хеш- функцією. Крім економії часу на постановку і перевірку підпису, в даному варіанті ще й зберігання підпису вимагає менше пам'яті і може здійснюватися окремо від зберігання документа.

Одержувачу відправляється сам документ і його зашифрована згортка як підпис під цим документом. Одержувач, володіючи тільки відкритим ключем, може перевірити підпис, але не може його підробити, так для створення підпису необхідний секретний ключ. При цьому важливим питанням є захист загальнодоступної бази відкритих ключів.

Відмінність схеми цифрового підпису від схеми передачі зашифрованої інформації (див. Рис. 5.1) при асиметричному шифруванні полягає в тому, що в цій схемі джерело ключів

повинен бути перенесений на сторону відправника. Отже, цифровий підпис є відносно невелика кількість додаткової цифрової інформації, що передається разом з підписаним текстом. Протокол електронного цифрового підпису (ЕЦП) включає в себе дві процедури:

процедуру поставлення підпису; в ній використовується секретний ключ відправника повідомлення;

процедуру перевірки підпису, в ній використовується відкритий ключ відправника.

При формуванні ЕЦП відправник насамперед обчислює хеш-функцію h (M) від підписаного тексту M. Обчислення значення хеш-функції $m = h$ (M) являє собою один короткий блок інформації m, що залежить від усього тексту M в цілому. Потім число m шифрується секретним ключем відправника і результат шифрування є ЕЦП для даного тексту M.

При перевірці ЕЦП одержувач повідомлення також обчислює хеш-функцію $m = h$ (M) прийнятого по каналу тексту M за відомим алгоритмом хешування. Після цього він розшифрував отриману разом з текстом підпис за допомогою відкритого ключа відправника і порівнює, чи відповідає отриманий результат обчисленому значенню m хеш- функції.

Неможливість підробки ЕЦП гарантується двома моментами: неможливістю визначити секретний ключ по відкритому в асиметричних криптосистемах і неможливістю зміни документа таким чином, щоб хеш-функція залишилася без зміни.

Хеш-функції

Хеш-функція призначена для стиснення підписаного документа M до декількох десятків або сотень біт. Значення хеш-функції h (M) складним чином залежить від документа M і не дозволяє відновити сам документ M. Хеш-функція h (·) приймає в якості аргументу повідомлення (документ, файл) M довільної довжини і повертає хеш-значення h (M) = m фіксованої довжини. Таким чином, інформація хешування є стислим поданням основного повідомлення довільної довжини. Іноді результат хешування називають дайджестом повідомлення.

Хеш-функція повинна відповідати таким вимогам:

хеш-функція повинна бути однозначна;

хеш-функція повинна бути чутлива до всіляких змін в тексті M, таким, як вставки, викиди, перестановки і ін.

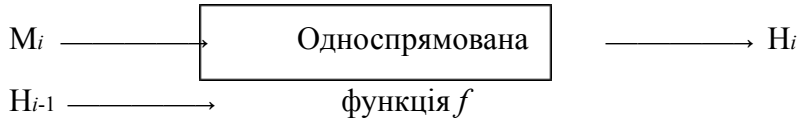
хеш-функція повинна мати властивість незворотності, тобто завдання підбору документа M', який володів би необхідним значенням хеш-функції, повинна бути обчислювально нерозв'язна;

ймовірність того, що значення хеш-функції двох різних документів (незалежно від їх довжин) співпадуть, повинна бути мізерно мала.

З визначення випливає, що для будь-якої хеш-функції є тексти-близнюки – які мають однакове значення хеш-функції, так як потужність множини аргументів необмежено більше потужності множини значень. Такий факт отримав назву "ефект дня народження".

Для побудови хеш-функцій, як правило, використовується одностороння функція f (·), Яка утворює вихідне значення довжиною n при завданні двох вхідних значень довжиною n.

Цими входами є блок вихідного тексту M_i і хеш-значення H_{i-1} попереднього блоку тексту: $H_i = f(M_i H_{i-1})$.



Хеш-значення, що обчислюється при введенні останнього блоку тексту, стає хеш-значенням всього повідомлення M.

В результаті односпрямована хеш-функція завжди формує вихід фіксованої довжини незалежно від довжини вхідного тексту.

Хеш-функції на основі симетричних блокових алгоритмів

Односпрямованою хеш-функцію можна побудувати, використовуючи в якості опції f (·) симетричний блоковий алгоритм. Найбільш очевидний підхід полягає в тому, щоб шифрувати

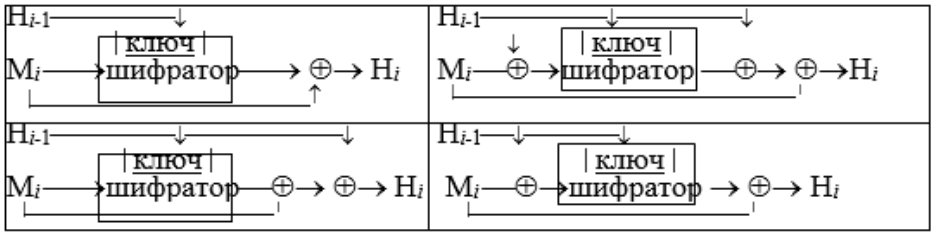
повідомлення M за допомогою блочного алгоритму в режимі CBC або CFB за допомогою фіксованого ключа і деякого вектора ініціалізації. Останній блок шифртекста можна розглядати в якості хеш-значення повідомлення M. При такому підході не завжди можливо побудувати безпечну односторонню хеш-функцію, але завжди можна отримати

код аутентифікації повідомлення КАП.

Більш безпечний варіант хеш-функції можна отримати, використовуючи блок повідомлення в якості входу, попереднє значення - як ключ, а поточне хеш-значення - як вихід. Реальні хеш-функції проектуються ще більш складними. Довжина блоку зазвичай визначається довжиною ключа, а довжина хеш-значення збігається з довжиною блоку. Оскільки більшість блокових алгоритмів є 64-бітовими, схеми хешування проектують так, щоб хеш-значення мало довжину, рівну одинарній або подвійній довжині блоку.

Якщо прийняти, що отримувана хеш-функція коректна, безпеку схеми хешування базується на безпеці, що лежить в основі цього блочного алгоритму.

Нижче наведено чотири схеми безпечного хешування, у яких довжина хеш- значення дорівнює довжині блоку.



Алгоритм безпечного хешування SHA

Алгоритм безпечного хешування SHA (Secure Hash Algorithm) розроблений НІСТ і АНБ США в рамках стандарту безпечного хешування SHS (Secure Hash Standard) в 1992 р. Він розроблений для використання спільно з алгоритмом цифрового підпису DSA.

При введенні повідомлення M довільної довжини менше 2^{64} біт алгоритм SHA виробляє 160-бітове вихідне повідомлення, зване дайджестом повідомлення MD (Message

Digest). Потім цей дайджест повідомлення використовується в якості входу алгоритму DSA, який обчислює цифровий підпис повідомлення M. Формування цифрового підпису для дайджесту повідомлення, а не для самого повідомлення підвищує ефективність процесу підписання, оскільки дайджест повідомлення зазвичай набагато коротший самого повідомлення. Такий же дайджест повідомлення повинен обчислюватися користувачем, перевіряючий отриманий підпис, при цьому в якості входу в алгоритм SHA використовується отримане повідомлення M.

Алгоритм SHA побудований на основі кінцевих автоматів Мілі. Головний цикл алгоритму обробляє по 512 біт повідомлення по черзі для всіх блоків, наявних в повідомленні. Він містить чотири цикли по 20 операцій кожен. Кожна операція реалізує нелінійну функцію, а потім виробляє зрушення і складання.

Алгоритм хешування SHA названий безпечним, тому що його творці стверджують, що обчислювально неможливо відновити повідомлення, відповідне даному дайджесту, а також знайти два різних повідомлення, які дадуть однаковий дайджест. Будь-яка зміна повідомлення при передачі з дуже великою ймовірністю спричинить зміну дайджесту, і прийнятий цифровий підпис не пройде перевірку. Також вважається, що алгоритм SHA більш стійкий до атак повного перебору і атакам "дня народження", оскільки видає 160- бітове хеш-значення, тоді як більшість інших алгоритмів хешування формують 128-бітові хеш-значення.

Алгоритми електронного цифрового підпису

Технологія застосування системи ЕЦП передбачає наявність мережі абонентів, що посилають один одному підписані електронні документи. Для кожного абонента генерується

пара ключів: секретний і відкритий. Секретний ключ зберігається абонентом в таємниці і використовується ним для формування ЕЦП. Відкритий ключ відомий всім користувачам і призначений для перевірки одержувачем дійсності підписаного електронного документа і автора підпису.

Для генерації пари ключів в алгоритмах ЕЦП, заснованих на асиметричних алгоритмах шифрування, використовуються математичні схеми, засновані на відомих складних обчислювальних завданнях: факторизації (розкладання на множники) великих цілих чисел і дискретного логарифмування.

Визначення. *Схема цифрового підпису* є конструкцією виду: (K, M, S, P_K, V_K), де

K - простір ключів, його елементи мають вигляд $k = (k_o, k_c)$, де

k_o називається *відкритим ключем* (*public key*) або *відкритим компонентом ключа*,

k_c називається *секретним ключем* (*secret key*) або *секретним компонентом ключа*;

M - простір повідомлень;

S - простір підписів;

функція побудови підпису P_K: M → S

ефективно будується по закритому ключу k_c;

функція перевірки підпису V_K: {M x S} → {0; 1}

ефективно будується по відкритому ключу k_o; і $\forall m \in M$ і $\forall s \in S$ $V_K(m, s) = 1 <=> s = P_K(m)$;

інакше $V_K(m, s) = 0$.

не існує ефективного способу знайти без знання закритого ключа значення s для заданого m так, щоб $V_K(m, s) = 1$.

Остання вимога до цифрового підпису може бути посилено, а саме: потрібна неможливість знайти хоча б одну пару (m, s) таку, що:

$V_K(m, s) = 1$ («екзистенціальна підробка»)

Цифровий підпис дозволяє учаснику A передати учаснику B повідомлення в вигляді $\left(m, P_{K_A}(m)\right)$

Тоді одержувач може бути впевнений, що повідомлення надіслано саме власником закритого ключа, і навіть довести це в суді.

Якщо цифровий підпис використовується спільно з шифруванням, то слід обчислювати підпис під відкритим текстом, а потім виконувати шифрування. Тобто передавати повідомлення у вигляді:

$E_K\left(m, P_K(m)\right)$, а не $\left(E_{K_B}(m), P_{K_A}\left(E_{K_B}(m)\right)\right)$

Підписувати шифртекст не рекомендується, так як при цьому не вдається зберігати розшифровані повідомлення разом з підписами.

Крім того, в цьому випадку від використовуваної криптосистеми потрібно стійкість до атак з відомим повідомленням. В іншому випадку, якщо по заданій шифрограмі c і заданому повідомленню m можна підібрати ключ k такий, що $E_k(m) = c$, зловмисник Z

може перехопити підпис $s = P_K\left(E_K(m)\right)$, для будь-якого повідомлення m' підібрати ключ k такий, що $E_k(m') = E_K(m)$

, зареєструвати цей ключ як свого і стверджувати, що A послав йому повідомлення c $s = P$

$(E_k(m'))$.

A

Алгоритм цифрового підпису RSA

Першою і найбільш відомою у всьому світі конкретною системою ЕЦП стала система RSA. Узагальнена схема формування і перевірки цифрового підпису RSA показана на Рис. 10.1.

Криптосистему формує відправник (автор) електронних документів. Для цього обчислює два великих простих числа p і q, потім знаходить їх добуток $n = p \cdot q$ і значення функції Ейлера

$\phi(n) = (p-1)(q-1)$

Далі він обчислює пару ключів (секретний ключ d і відкритий ключ e) з умов:

$e \le \phi(n), \text{НОД}\ (e, (\phi(n)) = 1$ і

$d < n, \quad e \cdot d = 1 \bmod \phi\ (n)$

Пару чисел (e, n) автор передає партнерам по листуванню для перевірки його цифрових підписів. Число d зберігається автором як *секретний ключ* для підписування.

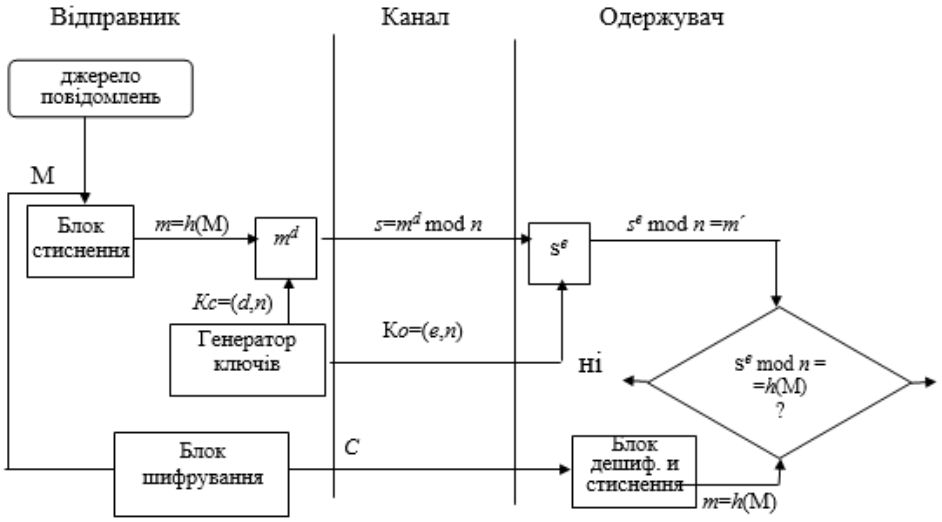


Рис. 10.1. Схема формування і перевірки цифрового підпису RSA

Припустимо, що відправник хоче підписати повідомлення M перед його відправкою. Спочатку повідомлення M (блок інформації, файл, ...) стискають за допомогою хеш-функції $h\ (\cdot)$ в ціле число $m = h$ (M). Потім обчислюють цифровий підпис s під електронним документом M, використовуючи хеш-значення m і секретний ключ d:

$s = m^d \bmod n$

Пара (M, s) передається одержувачу як електронний документ M, підписаний цифровим підписом s, причому підпис s сформований володарем секретного ключа d.

Після прийому пари (M, s) одержувач обчислює хеш-значення повідомлення M двома різними способами. Перш за все він відновлює хеш-значення m', застосовуючи криптографічне перетворення підпису s з використанням відкритого ключа e:

$m' = s^e \bmod n$

Крім того, він знаходить результат хешування прийнятого повідомлення M за допомогою такої ж хеш-функції $h\ (\cdot)$: $m = h$ (M).

Якщо дотримується рівність обчислених значень, тобто

$s^e \bmod n = h$ (M)

то одержувач визнає пару (M, s) справжньої. Доведено, що тільки власник секретного ключа d може сформувати цифровий підпис s по документу M, а визначити секретне число d з відкритого числу e не легше, ніж розкласти модуль n на множники.

Можна строго математично довести, що результат перевірки цифрового підпису *s* буде позитивним тільки в тому випадку, якщо при обчисленні *s* було використано секретний ключ *d*, відповідний відкритому ключу *e*. Тому *s* називають «ідентифікатором підписавшого».

Недоліки алгоритму цифрового підпису RSA

При обчисленні модуля *n*, ключів *e* і *d* для системи цифрового підпису RSA необхідно перевіряти велику кількість додаткових умов, що зробити практично важко. Невиконання будь-якої з цих умов робить можливим фальсифікацію цифрового підпису з боку того, хто виявить таке невиконання. При підписанні важливих документів не можна допускати таку можливість навіть теоретично.

Для забезпечення криптостійкості цифрового підпису RSA по відношенню до спроб фальсифікації на рівні, наприклад, національного стандарту США, необхідно використовувати при обчисленнях *n*, *d* і *e* цілі числа не менше 2^{512} (або близько 10^{154}) кожне, що вимагає великих обчислювальних витрат, що перевищують на 20-30%

обчислювальні витрати інших алгоритмів цифрового підпису при збереженні того ж рівня криптостійкості.

Цифровий підпис RSA уразливий до так званої мультиплікативної атаки. Інакше кажучи, алгоритм цифрового підпису RSA дозволяє зловмисникові без знання секретного ключа *d* сформувати підписи під тими документами, у яких результат хешування можна обчислити як добуток результатів хешування вже підписаних документів.

Алгоритм цифрового підпису Ель Гамаля (EGSA)

Більш надійний і зручний для реалізації на персональних комп'ютерах алгоритм цифрового підпису був розроблений в 1984 році американцем арабського походження Ель Гамалем. У 1991р. НІСТ США обґрунтував перед комісією Конгресса США вибір алгоритму цифрового підпису Ель Гамаля в якості основи для національного стандарту.

Ідея EGSA (El Gamal Signature Algorithm) заснована на тому, що для обґрунтування практичної неможливості фальсифікації цифрового підпису може бути використана більш складна обчислювальна задача, ніж розкладання на множники великого цілого числа - завдання дискретного логарифмування. Крім того, Ель Гамалю вдалося уникнути очевидній слабкості алгоритму цифрового підпису RSA, пов'язаної з можливістю підробки цифрового підпису під деякими повідомленнями без визначення секретного ключа.

Алгоритм цифрового підпису Ель Гамаля:

Вибирають *несекретні параметри* алгоритму: два великих цілих числа p — просте і $g < p$: p (~ 10^{308} або ~ 2^{1024}) і g (~ 10^{154} або ~ 2^{512});

Відправник вибирає *секретний ключ x* для підписування документів, яким є випадкове ціле число $x \in Z_p$, тобто $1 < x < p$;

Відправник обчислює *відкритий ключ y*, який використовується для перевірки свого підпису. Він і обчислює:

$y = g^x \bmod p$

і передає його всім потенційним одержувачам документів;

Відправник хешує повідомлення M за допомогою хеш-функції $h(\cdot)$ в ціле число m:

$m = h(M)$, $1 < m < (p\text{-}1)$

Відправник генерує секретний параметр алгоритму: випадкове ціле число k, $1 < k < (p\text{-}1)$, таке, що k і $(p\text{-}1)$ є взаємно простими;

Нарешті відправник обчислює цифровий підпис S = (a, b), що складається з двох цілих чисел a і b, що обчислюються наступним чином:

$a = g^k \bmod p$

і число b обчислюється за допомогою секретного ключа x з рівняння:

$m = x \cdot a + k \cdot b \bmod (p - 1)$

використовуючи розширений алгоритм Евкліда.

Трійка чисел (M, a, b) передається одержувачу, в той час як пара чисел (x, k) тримається в секреті.

Після прийому підписаного повідомлення (M, a, b) одержувач повинен перевірити, чи відповідає підпис S = (a, b) з повідомленням M. Для цього одержувач:

Хеширує прийняте повідомлення M, тобто обчислює $m = h$ (M).

Обчислює значення $A = g^m \bmod p$.

Визнає повідомлення M справжнім тільки, якщо воно співпало з: $A = y^a a^b \bmod p$

Інакше кажучи, він перевіряє справедливість співвідношення: $A = y^a a^b \bmod p = g^m \bmod p$

Можна строго математично довести, що остання рівність буде виконуватися тоді і тільки тоді, коли підпис S = (a, b) під документом M отриманий за допомогою саме того секретного ключа x, з якого був отриманий відкритий ключ y. Таким чином, можна надійно упевнитися, що відправником повідомлення M був володар саме даного секретного ключа x, не розкриваючи при цьому сам ключ, і що відправник підписав саме цей конкретний документ M.

Слід зазначити, що виконання кожного підпису за методом Ель Гамаля вимагає нового значення k, причому це значення повинно вибиратися випадковим чином. Якщо порушник розкриє коли-небудь значення k, повторно використовуване відправником, то він зможе розкрити секретний ключ x відправника.

Схема Ель Гамаля є характерним прикладом підходу, який допускає пересилання повідомлення M у відкритій формі разом з приєднаним аутентифікатором (a, b). У таких випадках процедура встановлення автентичності прийнятого повідомлення складається в перевірці відповідності аутентифікатора повідомлення.

Приклад. Виберемо: числа $p = 11$, $g = 2$ і секретний ключ $x = 8$. Обчислюємо значення відкритого ключа:

$y = g^x \bmod p = y = 2^8 \bmod 11 = 3$

Припустимо, що вихідне повідомлення M характеризується хеш-значенням $m = 5$. Для того щоб обчислити цифровий підпис для повідомлення M, що має хеш-значення $m = 5$, спочатку виберемо випадкове ціле число $k = 9$.

Переконаємося, що числа k і (p-1) є взаємно простими. Дійсно, НСД (9,10) = 1. Далі обчислюємо елементи a і b підпису:

$a = g^k \bmod p = 2^9 \bmod 11 = 6$

елемент b визначаємо, використовуючи розширений алгоритм Евкліда:

$m = x \cdot a + k \cdot b \bmod (p\text{-}1)$

При $m = 5$, $a = 6$, $x = 8$, $k = 9$, $p = 11$ отримуємо $b = 3$ як рішення рівняння

$5 = (6 \cdot 8 + 9 \cdot b) \bmod 10$ або $9 \cdot b \equiv -43 \bmod 10$

Отже, цифровий підпис представляє собою пару: $a = 6$, $b = 3$.

Далі відправник передає підписане повідомлення. Прийнявши підписане повідомлення і відкритий ключ $y = 3$, одержувач обчислює хеш-значення для повідомлення M: $m = 5$, а потім обчислює два числа:

$y^a a^b \bmod p = 3^6 \cdot б^3 \bmod 11 = 10 \bmod 11$

$g^m \bmod p = 2^5 \bmod 11 = 10 \bmod 11$

Так як ці два цілих числа рівні, прийняте одержувачем повідомлення визнається справжнім.

Схема цифрового підпису Ель Гамаля має ряд переваг в порівнянні зі схемою цифрового підпису RSA:

При заданому рівні стійкості алгоритму цифрового підпису цілі числа, які беруть участь в обчисленнях, мають запис на 25% коротший, що зменшує складність обчислень майже в два рази і дозволяє помітно скоротити обсяг використовуваної пам'яті.

При виборі модуля p досить перевірити, що це число є простим і що у числа (p-1) є великий простий множник (тобто всього дві умови, що досить просто перевіряються).

Процедура формування підпису за схемою Ель Гамаля не дозволяє обчислювати цифрові підписи під новими повідомленнями без знання секретного ключа (як в RSA).

Однак алгоритм цифрового підпису Ель Гамаля має і деякі, *недоліки* в порівнянні зі схемою підпису RSA. Зокрема, довжина цифрового підпису виходить в 1,5 рази більшою, що в свою чергу збільшує час її обчислення.

Алгоритм цифрового підпису DSA

Алгоритм цифрового підпису DSA (Digital Signature Algorithm) запропонований в 1991 р. в НІСТ США для використання в стандарті цифрового підпису DSS (Digital Signature Standard). Алгоритм DSA є розвитком алгоритмів цифрового підпису Ель Гамаля і К.Шнорра. Він складається з наступних кроків:

Відправник і одержувач електронного документа генерують використовувані при обчисленнях великі цілі числа g, p, q, що є відкритими параметрами криптосистеми:

g і p - прості числа, L біт кожне ($512 < L \leq 1024$);

q - просте число довжиною 160 біт (дільник числа (p -1)).

Вони можуть бути загальними для всіх користувачів мережі.

Відправник вибирає випадкове ціле число x ($1 < x < q$), що є його секретним ключем.

Потім відправник обчислює значення відкритого ключа:

$y = g^x \bmod p$

яке передається всім одержувачам документів.

Для того щоб підписати документ M, відправник хеширує його в ціле хеш- значення m з використанням односторонньої функції хешування $h(\cdot)$, визначеної в алгоритмі безпечного хешування SHA:

$m = h(M)$, $1 < m < q$

Потім відправник генерує випадкове ціле число k, $1 < k < q$, і обчислює:

$r = (g^k \bmod p) \bmod q$

Далі відправник обчислює за допомогою секретного ключа x число:

$s = (m + r*x)/k \bmod q$

Пара чисел (r, s) = S утворює цифровий підпис S під документом M. Таким чином, підписане повідомлення являє собою трійку чисел [M, r, s].

Одержувач підписаного повідомлення [M, r, s] перевіряє виконання умов $0 < r < q$, $0 < s < q$ і відкидає підпис, якщо хоча б одна з цих умов не виконана.

Потім одержувач обчислює хеш-функцію $m = h$ (M), значення:

$w = 1/s \bmod q$

і числа $U_1 = (m * w) \bmod q$

$U_2 = (r * w) \bmod q$

Далі одержувач за допомогою відкритого ключа y обчислює значення

$v = ((g^{U1} * y^{U2}) \bmod p) \bmod q$

і перевіряє виконання умови $v = r$.

Якщо умова $v = r$ виконується, тоді підпис S = (r, s) під документом M визнається одержувачем справжнім.

Можна строго математично довести, що остання рівність буде виконуватися тоді і тільки тоді, коли підпис S = (r, s) під документом M отриманий за допомогою саме того секретного ключа x, з якого був отриманий секретний ключ y.

Таким чином, можна надійно упевнитися, що відправник повідомлення володіє саме даними секретним ключем x (не розплющуючи при цьому значення ключа x) і що відправник підписав саме даний документ M.

У порівнянні з алгоритмом цифрового підпису Ель Гамаля алгоритм DSA має наступні основні *переваги*:

При будь-якому допустимому рівні стійкості, тобто при будь-якій парі чисел g і p (від 512 до 1024 біт), числа q, x, r, s мають довжину по 160 біт, скорочуючи довжину підпису до 320 біт.

Більшість операцій з числами як при обчисленні підписи, так і при її перевірці проводиться по модулю числа q довжиною 160 біт, що скорочує обсяг пам'яті і час обчислення підпису.

Недоліком алгоритму DSA є те, що при підписуванні і при перевірці підпису доводиться виконувати складні операції ділення по модулю q:

$s = (m + r\,x)/k \bmod q$, $w = 1/s \bmod q$

що не дозволяє отримувати максимальну швидкодію.

Але цей недолік може бути усунутий за допомогою виконання попередніх обчислень. Зауважимо, що значення *r* не залежить від повідомлення M і його хеш- значення *m*. Можна заздалегідь створити рядок випадкових значень *k* і потім для кожного з цих значень обчислити значення *r*. Можна також заздалегідь обчислити зворотні значення *k*-1 для кожного із значень *k*. Потім, під час вступу повідомлення M можна обчислити значення *s* для даних значень *r* і *k*-1. Ці попередні обчислення значно прискорюють роботу алгоритму DSA.

Приклади завдань

1. Для заданого файлу обчислити хеш (SHA-256 або SHA-3-256) і пояснити, як зміна 1 біта у файлі вплине на дайджест.
2. Побудувати схему обміну підписаним документом: (документ + підпис), де підпис формується як підпис від хешу, і описати етапи перевірки.
3. Порівняти RSA-підпис і DSA/EGSA-підпис за: розміром підпису, кількістю модульних експоненціацій та вимогами до випадкового числа k.
4. На прикладі EGSA/DSA змоделювати наслідки повторного використання одного й того ж k для двох різних повідомлень (які саме дані «витікають» і чому це критично).
5. Запропонувати політику зберігання і розповсюдження відкритих ключів (PKI/сертифікати/довіра) для навчальної мережі обміну підписаними документами.

Контрольні питання

1. Які властивості має задовольняти криптографічна хеш-функція (прообразостійкість, стійкість до колізій, лавинний ефект) і чому «ефект дня народження» важливий?
2. Чому на практиці підписують не весь документ, а його хеш, і які ризики виникають при «неправильному» зв’язуванні підпису з контекстом (алгоритм/параметри/формат)?
3. У чому принципова різниця між RSA-підписом і (E)GSA/DSA-підписом з точки зору базової складної задачі (факторизація vs дискретний логарифм)?
4. Яка роль випадкового (сеансового) параметра k у EGSA/DSA, які вимоги до нього, і чому його повтор/передбачуваність компрометує секретний ключ?
5. Що таке перевірка підпису (VK) у формальній моделі схеми підпису та що означає вимога «екзистенціальної непідроблюваності» (інтуїтивно)?

Література

1. Digital Signature Standard (DSS) : FIPS PUB 186-5. — Gaithersburg : National Institute of Standards and Technology, 2023. — Режим доступу: (офіц. публікації NIST). — Дата звернення: 08.01.2026.
2. Secure Hash Standard (SHS) : FIPS PUB 180-4. — Gaithersburg : National Institute of Standards and Technology, 2015. — Режим доступу: (офіц. публікації NIST). — Дата звернення: 08.01.2026.
3. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions : FIPS PUB 202. — Gaithersburg : National Institute of Standards and Technology, 2015. — Режим доступу: (офіц. публікації NIST). — Дата звернення: 08.01.2026.
4. Moriarty K., Kaliski B., Jonsson J., Rusch A. PKCS #1: RSA Cryptography Specifications Version 2.2 : RFC 8017. — IETF, 2016. — Режим доступу: (RFC Editor). — Дата звернення: 08.01.2026.
5. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. — Boca Raton : CRC Press, 1996.

Лекція №11 Криптографічні протоколи: інтерактивна система доказу; протоколи аутентифікації; захищені обчислення

Специфіка взаємодії віддалених абонентів

Успіхи, досягнуті в розробці схем цифрового підпису та відкритого розподілу ключів, дозволили застосувати ці ідеї також і до інших місій взаємодії віддалених абонентів. Так виник великий новий напрямок теоретичної криптографії - криптографічні протоколи.

У класичній шенновській моделі системи секретного зв'язку є два повністю довіряючих один одному учасника, яким необхідно передавати між собою інформацію, не призначену для третіх осіб. Вирішується завдання захисту секретної інформації від зовнішнього противника.

Об'єктом вивчення теорії криптографічних протоколів є віддалені абоненти, які взаємодіють, як правило, по відкритим каналам зв'язку. Мета взаємодії абонентів полягає у вирішенні конкретного завдання. У цій ситуації противником може виявитися будь-який з абонентів або кілька абонентів, що вступили в змову, і переслідують власні цілі. При цьому противник в різних завданнях може мати різні можливості: наприклад, може взаємодіяти з абонентами від імені інших абонентів або втручатись в обміни інформацією між абонентами та т. п. Тому криптографічні протоколи повинні захищати їх учасників не тільки від зовнішнього противника, а й від нечесних дій партнерів.

Є відмінності криптографічних протоколів від криптосистем, з яких можна виділити наступне:

учасники протоколу, взагалі кажучи, не довіряють один одному;

в протоколі може бути більше двох учасників;

протоколи можуть бути інтерактивними, тобто припускати багатораундовий обмін повідомленнями між учасниками.

На жаль, поняття криптографічного протоколу, мабуть, неможливо формалізувати.

Визначення. Під протоколом (не обов'язково криптографічним) зазвичай розуміють розподілений алгоритм, що включає в себе:

сукупність алгоритмів для кожного з учасників,

специфікації форматів повідомлень, що пересилаються між учасниками,

специфікації синхронізації дій учасників,

опис дій при виникненні збоїв.

На останній елемент цього списку слід звернути особливу увагу, оскільки його часто не беруть до уваги, а некоректний повторний пуск може повністю зруйнувати безпеку учасників навіть в стійкому криптографічному протоколі.

Всі типи криптографічних протоколів можна умовно розділити на дві групи: прикладні протоколи і примітивні.

Прикладний протокол вирішує конкретну задачу, яка виникає (або може виникнути) на

практиці.

Примітивні протоколи використовуються як своєрідні «будівельні блоки» при розробці прикладних протоколів.

За останнє десятиліття криптографічні протоколи перетворилися в основний об'єкт досліджень в теоретичній криптографії. Одна з основних областей додатків криптографічних протоколів - банківські платіжні системи, де замість платіжних доручень на папері використовується їх електронна форма. Вигоди від такої заміни настільки відчутні, що, мабуть, банки від неї вже ніколи не відмовляться, які б технічні та криптографічні труднощі при цьому не виникали. Але платіжні доручення - лише один з численних типів документів, що знаходяться в обороті в сфері бізнесу. Але ж існують ще документи, з якими працюють державні органи і громадські організації, юридичні документи і т. п. В останні роки чітко простежується тенденція перекладу всього документообігу в електронну форму.

У дослідженнях багатьох типів криптографічних протоколів зроблені тільки перші кроки і ще багато математичні проблеми треба буде розв'язати, перш ніж криптографічні протоколи увійдуть в повсюдне використання.

Приклади криптографічних протоколів

Познайомимося з деякими типами криптографічних протоколів, а також довкруги математичних задач, що виникають при дослідженні їх стійкості. Розглянемо кілька прикладів завдань, що вирішуються віддаленими абонентами.

Протокол підписання контракту. Взаємодіють два, які не довіряють один одному абонента. Вони хочуть підписати контракт. Це треба зробити так, щоб не допустити таку ситуацію: один з абонентів отримав підпис іншого, а сам не підписався.

Протокол ідентифікації абонента. Взаємодіють два абонента А і В. Абонент А хоче довести абоненту В, що він саме А, а не противник. Типовий приклад: А - власник інтелектуальної смарт-карти (кредитної картки, карти доступу в закрите приміщення, комп'ютерний ключ), В - банк, комп'ютер охорони, адміністратор мережі.

Протокол візантійської угоди. Взаємодіють кілька віддалених абонентів, які отримали накази з одного центру. Частина абонентів, включаючи центр, можуть бути противниками. Необхідно виробити єдину стратегію дій, виграшну для абонентів.

Це завдання прийнято називати завданням про візантійських генералів. Наведемо приклад, з яким ця задача зобов'язана своєю назвою. Візантія. Ніч перед великою битвою. Візантійська армія складається з n легіонів, кожен з яких підпорядковується своєму генералу. Крім того, у армії є головнокомандувач, який керує генералами. Однак імперія перебуває в занепаді і до однієї третини генералів, включаючи головнокомандувача, можуть бути зрадниками. Протягом ночі кожен з генералів отримує від головнокомандувача наказ про дії на ранок, причому можливі два варіанти наказу:

«атакувати» або «відступати». Якщо всі чесні генерали атакують, то вони перемагають. Якщо всі вони відступають, то їм вдається зберегти армію. Але якщо частина з них атакує, а частина відступає, то вони зазнають поразки. Якщо головнокомандувач виявиться зрадником, то він може дати різним генералам різні накази, тому накази головнокомандуючого не варто виконувати беззаперечно. Якщо кожен генерал буде діяти незалежно від інших, результати можуть виявитися сумними. Очевидно, що генерали потребують в обміні інформацією один з одним (щодо отриманих наказів) з тим, щоб прийти до угоди.

Протокол підкидання монети по телефону. Взаємодіють два, які не довіряють один одному абонента. Вони хочуть кинути жереб за допомогою монети. Це треба зробити так, щоб абонент, який підкидає монету, не міг змінити результат підкидання після отримання здогадки від абонента, який вгадує цей результат.

Наведемо один з найпростіших протоколів підкидання монети по телефону (так звана схема Блюма-Мікалі). Для його реалізації у абонентів А і В має бути одностороння функція f: X —> Y, яка задовольняє таким умовам:

X - кінцева множина цілих чисел, яка містить однакову кількість парних і непарних чисел.

будь-які числа $x_1, x_2 \in X$, мають один образ $f(x_1) = f(x_2)$, мають одну парність;

по заданому образу $f(x)$ «важко» обчислити парність невідомого аргументу x.

Роль підкидання монети грає випадковий і рівноймовірний вибір елемента $x \in X$, а роль орла і решки - парність і непарність x відповідно. Нехай А - абонент, підкидає монету, а В - абонент, що вгадує результат. Протокол складається з наступних кроків:

А вибирає x («підкидає монету»), зашифровує x, тобто обчислює $y = f(x)$, і посилає y абоненту В;

В отримує y, намагається вгадати парність x і посилає свою здогадку абоненту А;

А отримує здогадку від В і повідомляє В, вгадав чи він, посилаючи йому вибране число x;

В перевіряє, чи не обманює А, обчислюючи значення $f(x)$ і порівнюючи його з отриманим на другому кроці значенням y.

Інтерактивна система доказу

Осмислення різних протоколів і методів їх побудови призвело в 1985-1986 рр. до появи двох плідних математичних моделей - *інтерактивної системи доказів* і *доказів з нульовим розголошенням*. Математичні дослідження цих нових об'єктів дозволили довести багато тверджень, досить корисних при розробці криптографічних протоколів.

Визначення. Під *інтерактивною системою доказу* (P, V, S) розуміють протокол взаємодії двох абонентів:

P (prover) – який доводить і

V (verifier) – який перевіряє.

Абонент P хоче довести абоненту V, що твердження S істинно. При цьому абонент V самостійно, без допомоги абонента P, не може перевірити твердження S (тому V і називається перевіряючим). Абонент P може бути і противником, який хоче довести абоненту V, що твердження S істинно, хоча насправді воно помилкове. Протокол може складатися з багатьох раундів обміну повідомленнями між абонентами P і V і повинен задовольняти двом умовам:

повнота - якщо S дійсно істинно, то абонент P переконає абонента V визнати це;

коректність - якщо S помилково, то абонент P «навряд чи» переконає абонента V, що S істинно.

Тут словами «навряд чи» ми для простоти замінили точне математичне формулювання.

Підкреслимо, що у визначенні системи (P, V, S) не допускалось, що абонент V може бути противником. А якщо це так і перевіряючий хоче «випитати», у того хто доводить якусь нову корисну для себе інформацію про затвердження S? В цьому випадку абонент P, звичайно, може не хотіти, щоб це сталося в результаті роботи даного протоколу. Протокол (P, V, S), який вирішує таке завдання, називається *доказом з нульовим розголошенням* і повинен задовольняти, крім умовам 1) і 2), ще й наступній умові:

3) *нульове розголошення* - в результаті роботи протоколу (P, V, S) абонент V не збільшить свої знання про затвердження S або, іншими словами, не зможе випитати ніякої інформації про те, чому S істинно.

Протоколи аутентифікації

Одне із застосувань інтерактивних систем доказу - протоколи аутентифікації, коли один з абонентів повинен довести іншому свій статус, але при цьому зробити так, щоб ніхто інший, який спостерігає цей протокол, не зміг зробити те ж саме. Такий, один з найбільш важливих і поширених типів криптографічних протоколів називають схемами аутентифікації.

Прикладом протоколу аутентифікації може служити всім відома казка про вовка і семеро козенят. У цьому протоколі коза виступає в якості того, хто доводить, а семеро козенят - як того, хто перевіряє. Протокол покликаний забезпечувати цілісність. В даному випадку - сімох козенят. У казці описана атака на протокол. Противником виступає вовк, і цей противник активний: спочатку він підслуховує виконання протоколу, потім намагається сам пройти аутентифікацію в якості того, хто доводить (кози) і при цьому накопичує й аналізує отриману інформацію. Протокол виявився нестійким проти активного противника.

Більш серйозне призначення і суть протоколів аутентифікації (або ідентифікації) легко зрозуміти на наступному прикладі. Уявімо собі інформаційну систему, яка працює в комп'ютерній мережі і забезпечує доступ до деяких даних. У адміністратора системи є список всіх її користувачів разом з співставленим кожному з них набором повноважень, на основі

яких здійснюється розмежування доступу до ресурсів системи. Ресурсами можуть бути, наприклад, деякі фрагменти інформації, а також функції, які виконуються системою. Одним користувачам може бути дозволено читати одну частину інформації, іншим - іншу її частину, а третім - ще й вносити в неї зміни. В даному контексті під забезпеченням цілісності розуміється запобігання доступу до системи осіб, які не є її користувачами, а також запобігання доступу користувачів до тих ресурсів, на які у них немає повноважень. Найбільш поширений метод розмежування доступу - парольний захист не забезпечує достатньої стійкості. Тому перейдемо до криптографічній постановці завдання.

Для того, щоб протокол аутентифікації був стійким, досить, щоб він був доказом з нульовим розголошенням. У протоколі є два учасники - Аліса, яка повинна довести свою автентичність, і Боб, який цю автентичність повинен перевірити. У Аліси є два ключі - загальнодоступний відкритий K_o і секретний K_c. Фактично Алісі потрібно довести, що вона знає - K_c, і зробити це таким чином, щоб цей доказ можна було перевірити, знаючи тільки K_o.

Схема аутентифікації Шнорра

Одним з найбільш ефективних практичних протоколів аутентифікації вважається протокол Шнорра.

Нехай

p і q - прості числа такі, що q ділить p-1.

(Шнорр пропонує використовувати p довжини порядку 512 біт і q - довжини близько 140 бітів.)

$g \in Z_p$ таке, що $g^q = 1 \bmod p$, g ≠ 1.

$x \in Z_q$ - секретний ключ

$y = g^x \bmod p$ - відкритий ключ.

Визначення x по y - це завдання дискретного логарифмування, для якої на даний момент не відомі ефективні алгоритми.

Відкриті ключі всіх учасників схеми повинні публікуватися таким чином, щоб виключалася можливість їх підміни (таке сховище ключів називається загальнодоступним сертифікованим довідником). Ця проблема, звана часто проблемою автентичності відкритих ключів, становить окремий предмет досліджень в криптографії.

У протоколі Шнорра кількість раундів дорівнює трьом. Наведемо алгоритм протоколу по крокам:

В якості секретного ключа схеми аутентифікації Аліса вибирає випадкове число

k з множини {1, ..., q - 1}, обчислює відкритий ключ $r = g^k \bmod p$ і посилає r Бобу.

Боб вибирає випадковий запит e з множини {0, ..., 2t - 1}, де t - деякий параметр, і посилає e Алісі.

Аліса обчислює $s = k + xe \bmod q$ і посилає s Бобу.

Боб перевіряє співвідношення $r = g^s y^e \bmod p$ і, якщо воно виконується, приймає доказ, в іншому випадку - відкидає.

Перше з вимог до стійкості протоколів аутентифікації - *коректність*, означає, що противник, знає тільки відкритий ключ y, може пройти аутентифікацію лише з зневажливо малою ймовірністю.

Нескладний аналіз показує, що коректність протоколу Шнорра залежить від обраного значення параметра t. Справді, якщо t невелика, то противник має хороші шанси просто вгадати той запит e, який він отримає від Боба на кроці 2.

Нехай для простоти t = 1. Тоді противник, який не знає секретного ключа x, може діяти таким чином. Підкинувши монету, він вибирає рівноймовірним чином одне зі значень 0 або 1. Позначимо його через e'. Далі противник вибирає довільне s з {0, ..., q-1}, обчислює $r = g^s y^{e'} \bmod p$ посилає r Бобу. Ясно, що запит e, отриманий від Боба на кроці 2, співпаде з e' з ймовірністю 1/2, і саме з такою ймовірністю противник пройде аутентифікацію. Якщо ж

значення t досить велике, то шанси вгадати запит e малі. Шнорр рекомендує $t = 72$. Імовірність простого вгадування буде 2^{-72}, її можна вважати зневажливо малою.

Якщо в схемі Шнорра Аліса є противником, то на кроці 1 замість дій, запропонованих протоколом, вона може вибирати r довільним (але ефективним) чином. Іншими словами, Аліса використовує певний поліноміальний імовірнісний алгоритм, який для кожного конкретного значення r визначає ймовірність його вибору.

Нехай r - деяке значення, яке Аліса передала Бобу на кроці 1. Припустимо, що нам вдалося знайти два запити:

$e_1, e_2 \in \{0,...,2^t-1\}$, $e_1 \neq e_2$

такі, що Аліса може для кожного з них знайти відповідні значення s, для яких перевірка на кроці 4 дасть позитивний результат. Позначимо ці значення s через s_1 і s_2 відповідно. Ми маємо:

$r = g^{s1} y^{e1} \bmod p$ $r = g^{s2} y^{e2} \bmod p$ Звідси:

$(g^{s1} y^{e1} = g^{s2} y^{e2}) \bmod p$, або $(g^{s1-s2} = g^{e2-e1}) \bmod p$

Оскільки $e_1 \neq e_2$, існує $(e_1 - e_2)^{-1} \bmod q$ і, отже,

$(s_1-s_2)(e_2-e_1)^{-1} \bmod q = x$ - дискретний логарифм y.

Таким чином, або запити $e_1 \neq e_2$, такі, що Аліса може відповісти належним чином на обидва з них (при одному і тому ж r) на кроці 3 протоколи, зустрічаються «досить рідко», і це означає, що атака Аліси успішна лише зневажливо малою ймовірністю, або такі значення трапляються «досить часто», і тоді той алгоритм, який застосовує Аліса, можна використовувати для обчислення дискретних логарифмів.

Ця неформально викладена ідея була використана Шнорром для доказу поліноміальної звідності завдання дискретного логарифмування до завдання, що стоїть перед пасивним противником, тобто таким, який намагається пройти аутентифікацію, знаючи лише відкритий ключ. Іншими словами, доведено, що в припущенні труднощі завдання дискретного логарифмування схема аутентифікації Шнорра є стійкою проти пасивного противника, тобто коректною.

Активний противник може провести кілька сеансів виконання протоколу в якості перевіряючого з чесним який доводить (або підслухати такі виконання) і після цього спробувати атакувати схему аутентифікації. Для стійкості проти активного противника досить, щоб протокол аутентифікації був доказом з нульовим розголошенням. Однак властивість нульового розголошення для схеми Шнорра досі нікому довести не вдалося. Більш того, на даний момент відомий єдиний метод доказу властивості нульового розголошення - так званий метод «чорного ящика». У цьому методі машина, яка моделює використовує алгоритм перевіряючого лише в якості оракула, тобто, не аналізуючи сам цей алгоритм, подає йому на вхід будь-які значення за своїм вибором і отримує відповідні вихідні значення.

Доведено, що трьохраундові докази з нульовим розголошенням, в яких остання властивість встановлюється методом «чорного ящика», існують лише в тривіальному випадку, тобто коли перевіряючий може самостійно, без будь-якої допомоги, того хто доводить, перевірити істинність затвердженого. Відносно схеми Шнорра з цього результату випливає, що або існує ефективний алгоритм дискретного логарифмування, або властивість нульового розголошення цього протоколу не може бути доведено методом

«чорного ящика». Питання про існування доказів з нульовим розголошенням, для яких властивість нульового розголошення не може бути доведено методом «чорного ящика», залишається відкритим.

Неважко показати, що схема Шнорра володіє трохи більш слабкою властивістю - властивістю нульового розголошення щодо чесного перевіряючого. У цьому випадку досить побудувати моделюючу машину тільки для чесного перевіряючого, який на кроці 2 і справді вибирає випадковий запит e з множини $\{0, ..., 2t-1\}$. Властивості нульового розголошення щодо чесного перевіряючого може виявитися достатнім, якщо схема аутентифікації використовується, наприклад, для контролю за доступом в приміщенні, що охороняється. В цьому випадку Аліса - це пропуск, виконаний у вигляді інтелектуальної картки, а Боб - комп'ютер охорони. У такій ситуації головне завдання - забезпечити коректність схеми

аутентифікації, а захищатися від нечесного перевіряючого безглуздо. Що ж стосується якості нульового розголошення щодо чесного перевіряючого, то воно видається далеко не зайвим, так як дозволяє забезпечитися від противника, який може спробувати підслуховувати сеанси виконання протоколу з метою виготовлення фальшивого пропуску.

Перш ніж завершити розмову про протоколи аутентифікації, необхідно підкреслити, що ніяке математично строго доведена властивість криптографічного протоколу не може гарантувати його безпеку у всіх випадках життя. Справді, навіть протокол доказу з нульовим розголошенням не захищає від наступної атаки на схему аутентифікації, відомої в криптографічній літературі під назвою «мафіозна загроза». У даному сценарії є чотири учасника: A, B, C і D. Припустимо, що A зайшов в кафе випити чашечку кави і розплачується за допомогою кредитної картки. Для виконання будь-якої банківської операції картка повинна ідентифікувати себе, тобто виконати протокол аутентифікації. Але власник кафе, B, - мафіозі, спільник якого C у той же самий момент знаходиться в магазині ювеліра D і намагається купити діамант, також за допомогою кредитної картки. При цьому C «представляється» як A, а D просить довести це за допомогою протоколу аутентифікації. Пристрій зчитування для карток у B і картка у C - це спеціально виготовлені приймально-передавальні пристрої, які лише пересилають повідомлення між A і D. У результаті A, обмінюючись повідомленнями з B, насправді ідентифікує себе для D. Отже, тільки математичного обґрунтування стійкості того чи іншого криптографічного протоколу недостатньо. Для практичного застосування конкретного протоколу потрібні ще зусилля фахівців в області інформаційної безпеки з аналізу умов його застосування.

Захищені обчислення

До захищених обчислень відносяться різні протоколи, метою яких є спільне обчислення деякого значення таким чином, щоб приховати це значення від всіх або деяких учасників. Розглянемо два протоколи захищених обчислень.

Приховування інформації від оракула

Нехай учасник B - оракул, який вміє обчислювати значення деякої важко обчислюваної функції f. A хоче звернутися до B за обчисленням $f(x)$, але не хоче розкривати значення x.

Параметри протоколу: конструкція (K, E, D), що відповідає вимогам до криптосистем з секретним ключем і з функцією дешифрування, що задовольняє умові:

$D_k(f(E_k(x))) = f(x)$ (*)

Алгоритм протоколу:

A вибирає випадковим чином ключ k, шифрує x на цьому ключі: $E_k(x)$, і посилає його B.

B обчислює значення $f(E_k(x))$ функції від зашифрованого значення і відсилає його A.

A дешифрує отримане значення і отримує за умовою (*) значення функції.

Криптосистема з відкритим ключем тут не потрібна, так як не потрібно пересилати оракулу ні значення ключа, ні алгоритм дешифрування. Основна умова - наявність функцій шифрування і дешифрування, що задовольняють умові (*).

Прикладом такого протоколу може бути протокол обчислення важко обчислюваної функції $f(x)$ - дискретного логарифма x в Z^*_p по основі g, (g – який утворює елемент Z^*_p). У цьому випадку функції шифрування і дешифрування можна вибрати:

$E_k(x) = x\, g^k \bmod p$ і $D_k(y) = (y-k) \bmod (p-1)$, $k \in Z^*$

Завдання про двох мільйонерів

Два мільйонера хочуть з'ясувати у кого більше мільйонів, але так, щоб ніхто не дізнався, скільки їх у іншого.

Параметри протоколу:

Нехай a, b - секретні значення учасників A і B.

У протоколі потрібна криптосистема (K, E, D) з відкритим ключем і число m, таке, що

$a \le m$ і $b \le m$. Нехай k- ключ учасника В.

Алгоритм протоколу:

А вибирає велике просте число x, обчислює $t = E_k(x)$ - a і посилає його В.

В вибирає просте число p, яке задовольняє вказаним нижче умовам і обчислює послідовність чисел $S = (u_1, ..., u_m, p)$ наступним чином:

$$u_i = \begin{cases} z_i, i \le b, \text{ де } z_i = D_k(t+i) \bmod p, \\ z_i = 1, i > b \end{cases}$$

а p вибирається так, щоб для $\forall i \ne j$ виконувалося $|z_i - z_j| > 1$.

Зауважимо, що при цьому $y_a = x$.

А посилає В повідомлення «$a \le b$», якщо $u_a = x \bmod p$ і «$a > b$» в іншому випадку. Цей протокол має очевидні недоліки:

Чим більше m, тим складніше підібрати p, що задовольняє умовам.

Учасник А дізнається результат раніше, ніж В і може відмовитися від виконання 3-го кроку.

Приклади завдань

1. Побудувати приклад інтерактивної системи доказу для твердження S та пояснити, як у протоколі забезпечуються повнота і коректність.
2. Для схеми підкидання «монети по телефону» (типу Блюма–Мікалі) описати можливості активного й пасивного противника та вказати, на якому кроці можливе шахрайство.
3. Сформулювати вимоги до протоколу аутентифікації як доказу з нульовим розголошенням і пояснити, чому «підслуховування» не повинно допомагати зловмиснику.
4. Для схеми аутентифікації Шнорра виписати 3-раундовий обмін і перевірку рівності та пояснити, як параметр виклику впливає на ймовірність успішного вгадування.
5. Для задачі «двох мільйонерів» описати ідею двосторонніх захищених обчислень і вказати, які дані мають залишатися прихованими від кожного з учасників.

Контрольні питання

1. Чим криптографічні протоколи принципово відрізняються від криптосистем (за моделлю довіри, кількістю учасників, інтерактивністю)?
2. Дайте визначення інтерактивної системи доказу (P,V,S) та розкрийте зміст властивостей повнота і коректність.
3. Що таке доказ з нульовим розголошенням і чим відрізняється «нульове розголошення для чесного перевіряючого» від повного нульового розголошення?
4. Які типові загрози для протоколів аутентифікації (підслуховування, повтор, MITM/«мафіозна загроза») і чому математична стійкість протоколу не гарантує безпеки в реальних умовах?
5. У чому суть «захищених обчислень» і які гарантії безпеки зазвичай формулюють (конфіденційність входів, коректність результату, справедливість/узгодженість завершення)?

Література

1. Katz J., Lindell Y. Introduction to Modern Cryptography. 2nd ed. Boca Raton : CRC Press, 2015.
2. Goldreich O. Foundations of Cryptography. Volume 1: Basic Tools. Cambridge : Cambridge University Press, 2001.

3. Goldwasser S., Micali S., Rackoff C. The knowledge complexity of interactive proof systems // SIAM Journal on Computing. 1989. (Первинна версія результатів — 1985).
4. Schnorr C. P. Efficient Identification and Signatures for Smart Cards // Advances in Cryptology — CRYPTO'89 (Lecture Notes in Computer Science). Berlin : Springer, 1990/1991.
5. Yao A. C. Secure computation (garbled circuits) / огляд і формальні доведення протоколу: A Proof of Yao's Protocol for Secure Two-Party Computation. 2004.

Лекція №12 Криптографічні протоколи: спеціальні види електронного підпису; Поділ секрету

Для вирішення завдань аутентифікації і забезпечення цілісності інформації використовується цифровий підпис. Звичайні протоколи електронного підпису, що розробляються для практичного застосування, є неінтерактивними (тобто весь обмін повідомленнями полягає в передачі відправником одержувачу підписаного повідомлення). Це викликано насамперед вимогами до ефективності.

Однак виникають ситуації в яких до електронного підпису пред'являються додаткові вимоги. Розглянемо спеціальні види електронного підпису, в яких використовуються багатораундові протоколи.

Сліпий підпис

Сенс протоколу сліпого підпису полягає в тому, щоб учасник В підписав запропоноване учасником А повідомлення, не отримавши інформації про це повідомлення. У «паперовому» документообігу така схема може бути реалізована за допомогою запечатаного конверту, в якому знаходиться документ, а поверх нього лист копірки. Такий підпис використовується, наприклад, у фінансовій криптографії для забезпечення властивості не відстежуваності електронних грошей.

У цифровому варіанті сліпий підпис являє собою окремий випадок приховання інформації від оракула. Повідомлення m шляхом шифрування упаковується в цифровий конверт P (m), який підписується S (P (m)). При розкритті конверта виходить підпис для m: $P^{-1}(S(P(m)))$.

Наведемо реалізацію сліпого підпису RSA.

Протокол створення сліпого підпису RSA під повідомленням $m \in Z_m$

Ключі учасника В: (n, e) - відкриті і (n, d) - секретні.

А → В $P(m) = mr^e \bmod n$, $r \in Z^*$

В → А $S(P(m)) = P(m)^d \bmod n$

А обчислює $P^{-1}(S(P(m))) = r^{-1} S(P(m)) = m^d \bmod n$

Як правило, необхідно, щоб В підписував тільки повідомлення з деякої множини «прийнятних» повідомлень М, але не знав, яке саме повідомлення він підписав.

Протокол створення сліпого підпису RSA під прийнятним повідомленням $m \in M$. Ключі учасника В: (n, e) - відкриті і (n, d) - секретні.

Нехай $m_1, \ldots, m_k \in M$ - повідомлення, для одного з яких буде отримано цифровий підпис.

1. А → В $(P_1(m_1), \ldots, P_k(m_k))$, де $P_i(m_i)=m_i^e \bmod n$, $r_i \in Z^* \; \forall i=1,\ldots,k$

В → А вибирає номер конверта $j \in \{1, \ldots, k\}$, який підпише.

А → В $\forall (r_i)\ i \neq j$, А, А відкриває всі конверти, крім j-го, В перевіряє, що $\forall i \neq j$

повідомлення прийнятні $m_i \in M$.

B → A $S P_j(m_j)) = P(m_j)^d \bmod n$.

A розкриває підписаний конверт, тобто обчислює підпис для m: $P^{-1}(S(P(m_j))) = r^{-1} S(P(m_j)) = m_j^d \bmod n$

Імовірність того, що A вдасться отримати підпис під неприйнятним повідомленням, не перевищує k^{-1}.

Невидимий підпис

Можна зробити так, щоб, той хто доводить підпис учасника P (prover) не могла бути перевірена перевіряючим V (verifier) без участі P. При цьому вийде так званий невидимий підпис або незаперечний підпис (undeniable signature, термін ввів David Chaum, вперше запропонував таку схему).

Застосування невидимою підписи:

надання фірмою можливості перевірити справжність розповсюджуваного програмного продукту тільки зареєстрованим користувачам;

здійснення без паперового документообігу всередині організації так, щоб в разі витоку документа за межі організації не можна було довести його справжність.

Параметри системи:

Група Z_p, де p - просте число.

Нехай $g \in Z_p$ – який утворює елемент (ділить p -1).

Ключі учасника P:

$x \in Z_p$ - закритий ключ, $y = g^x$ - відкритий ключ. Створення підпису:

$z = S(h) = h^x \bmod p$, де $h \in Z_p$ -хеш-функція повідомлення.

P → V число $w = h^a g^b \bmod p$, де $a,b \in Z_p$ - вибрані P числа.

V → P пару чисел (r,s), де $r = wg^t = h^a g^{b+t} \bmod p$.

де $t \in Z_p$ - обране V число і $s = r^x \bmod p$.

P → V пару чисел (a, b)

і P перевіряє, що $w = h^a g^b \bmod p$.

V → P число t

і V перевіряє, що $r = h^a g^{b+t}$ и $s = z^a y^{b+t} \bmod p$.

Твердження (*надійність протоколу*). Навіть володіючи необмеженою обчислювальною потужністю, P не може з ймовірністю, більшою p, дати вірні відповіді при невірному підпису.

Твердження (*безпека протоколу*). В ході протоколу не відбувається розкриття інформації про закриті ключі P, оскільки V може самостійно побудувати всі повідомлення протоколу з тим же розподілом, але без участі P.

Поділ секрету

Криптографічні схеми поділу секрету були незалежно відкриті Шаміром (A. Shamir) і Блеклі (G. R. Blakley). Основне призначення протоколів або схем поділу секрету - управління ключами. У багатьох практичних додатках доступність інформації залежить від одного-єдиного секретного ключа. Якщо ключ будь-яким чином втрачено (втрачений носій із записаним ключем, зруйнована пам'ять, в якій зберігається ключ, і т.д.), доступ до інформації блокується. Схеми поділу секрету дозволяють вирішити сформульовану проблему. Ідея полягає в поділі секретного ключа на компоненти з подальшим їх розподілом серед легальної спільноти

учасників. Відновлення ключа можливо тільки в тому випадку, якщо деяка легальна коаліція учасників об'єднає свої ключові компоненти.

З точки зору математики завдання поділу секрету є перш за все комбінаторним завданням. Відзначимо, що перша постановка задачі була сформульована у вигляді такої комбінаторної проблеми. Розглянемо ситуацію, в якій одинадцять вчених працюють над секретним проектом. Робочі документи проекту замкнені в сейфі. Умова завдання - тільки шість і більше вчених, об'єднавшись і пред'явивши ключі, можуть відімкнути сейф і отримати доступ до документів.

Постановка завдання зводиться до оцінки найменшого числа замків сейфа і найменшого числа ключів у кожного з учених. Як рішення пропонується наступне міркування. За умовою існує як мінімум один замок, який не може бути відкритим жодним вченим з довільної п'ятірки. Таким чином, кожен з шести вчених, які залишилася, має ключ від цього замка. Причому немає необхідності в більш ніж одному подібному замку для коаліції з п'яти вчених. Таким чином, мінімальне число замків і ключів оцінюється як число поєднань $C^5 = 462$ і $=252$ відповідно. Очевидно, що пропоноване комбінаторне рішення є неприйнятним з практичної точки зору.

Криптографічні схеми поділу секрету пропонують інше рішення проблеми.

Розглянемо її як *груповий криптографічний протокол* з деякою множиною учасників.

Зазвичай в протоколах ми розглядали пару взаємодіючих учасників (A, B). У групових протоколах замість A розглядатимемо групи учасників (A_{i1}, ..., A_{ik}, C), де учасники вибираються з множини можливих учасників U = {A_1, ..., A_l} (їх іноді називають процесорами). При цьому всі A_i і B взаємодіють тільки з виділеним учасником C. Учасник З називається *комбінатором* (combiner) або *дилером* (іноді, лідером) і доданий для досягнення *анонімності*, щоб B не міг дізнатися про склад групи A_i.

На множині U виділяється *структура доступу* Г - множина підмножин U, званих *дозволеними коаліціями*. Множина Г зазвичай монотонно по включенню: якщо $V \subset V'$ і $V' \in$ Г, то $V \in$ Г. Протокол повинен бути працездатний, якщо $\{A_{i1}, \ldots, A_{ik}\} \in$ Г.

Як і в інших типах криптографічних протоколів, в протоколі поділу секрету учасники, взагалі кажучи, не довіряють один одному, і кожен з них може виявитися противником, в тому числі і дилер. Груповий криптографічний протокол повинен відповідати наступним властивостям.

Надійність. Для будь-якої дозволеної коаліції $\{A_{i1}, \ldots, A_{ik}\} \in$ Г протокол ((A_{i1}, ... ,A_{ik}, C), B) задовольняє вимогам надійності та безпеки.

Групова безпека. Для будь-якої невирішеної коаліції $\{A_{i1}, \ldots, A_{ik}\} \notin$ Г ніякі *змовники* $A_{i1}^*, \ldots, A_{ik}^*, C^*$ такі, що будь-який A_i^* і C^* має доступ до секретів всіх A_i і C, не можуть порушити надійність системи.

Надалі будемо розглядати коаліції учасників як підмножини множин номерів учасників $I \in \Gamma \subseteq \{1, \ldots, l\} = L$.

Крім схем поділу секретів групові протоколи найчастіше використовуються для вирішення наступних завдань:

групове обчислення значення деякої функції - може застосовуватися для побудови групових криптосистем, групових підписів, групових ідентифікацій /протоколів конфіденційних обчислень/;

групова генерація псевдовипадкових чисел;

анонімна пошта і віддалене таємне голосування.

Але коло додатків схем поділу, який перевіряється, секрету значно ширше.

Порогова схема поділу секрету

Мета протоколу *поділу секрету* (*secret sharing*) - забезпечити відновлення секретного

значення S∈K тільки дозволеною коаліцією учасників, кожен з яких має значення, званим *часткою* (*share*) або *тінню* (*shadow*) секрету $s_i \in T$.

Передбачається, що схема поділу секрету включає n учасників і керується авторизованим дилером. Основна функція дилера - поділ секрету на n компонентів («тіней») і розподіл їх серед учасників так, що будь-які m (і більше) учасники, зібравшись разом і пред'явивши тіні, можуть відновити секретний ключ. Причому будь-які (m-1) і менш учасників не можуть цього зробити.

Криптографічні схеми поділу секрету відомі також під назвою m-з-n-схем або (m, n) -порогових схем.

Визначення. Множини $\{s_1, ..., s_n\}$, які задовольняють наступним двом умовам, називаються (m, n) -пороговою схемою поділу секрету.

Секрет S легко може бути обчислений за будь-яким m значенням s_i.

Знання будь-яких (m-1) значень s_i не дозволяють визначити секрет S.

Компроміс між крипостійкістю і гнучкістю схеми регулюється вибором параметрів m і n.

Протокол складається з двох фаз.

На *фазі поділу секрету* дилер, знає деякий секрет S, генерує n частку секрету $s_1, ..., s_n$ і посилає s_i процесору A_i по захищеному каналу зв'язку.

На *фазі відновлення секрету* будь-яка підмножина з не менш ніж m процесорів однозначно відновлює секрет, обмінюючись повідомленнями по захищеним каналам зв'язку. А будь-яка підмножина з не більш ніж (m-1) процесорів не може відновити секрет.

Визначення. Схемою поділу секрету з множиною секретів K і множиною частки T називається пара алгоритмів (D, R), таких, що:

D (S) - імовірнісний поліноміальний алгоритм роздачі, який виходячи з секрету $s \in K$ обчислює набір часток учасників $s_i \in T$.

R (I,x_1, … , $x_{|I|}$) - детермінований поліноміальний алгоритм відновлення секрету такий, що:

$D(S) = (s_1, \dots, s_n) \Rightarrow S = R(I, x_1, \dots, x_{|I|})$

Для кожної дозволеної коаліції I алгоритм задає функцію:

$\phi_I(x_1, \dots, x_{|I|}) = R(I, x_1, \dots, x_{|I|})$

для кожної невирішеної коаліції I всі можливі значення S рівноімовірні: $\forall I \notin U$, $\forall x_i \in T$, $P(s=x \mid \forall i \in I, s_i = x_i) = P(s=x)$.

Визначення. Схема поділу секрету називається *досконалою*, якщо довільна коаліція або повністю розкриває секрет, або в результаті не отримує про нього ніякої апостеріорної інформації.

Схема обчислення ключа доступу

Схема обчислення ключа доступу при поділі секрету між двома учасниками.

Розглянемо найпростіший варіант досконалої схеми. Нехай є первинний секретний ключ і два учасника, що утворюють легальну коаліцію для отримання доступу до секретної інформації. Жоден з учасників не знає первинного секретного ключа, і тільки об'єднання тіней, які мають учасники, дозволяє відновити первинний ключ і виконати дешифрування.

Практична реалізація ідеї заснована на властивостях арифметики по модулю 2.

Розглянемо первинний ключ S як послідовність двійкових символів (біт) довжини n.

Тоді тінь першого учасника s_1 обчислюється шляхом побітного підсумовування первинного ключа і шумової (випадкової) послідовності s_2 двійкових символів довжини n. Та ж шумова послідовність s_2 використовується в якості тіні іншого учасника.

Отже, первинний ключ S може бути обчислений підсумовуванням по модулю 2 тіней учасників:

$S = s_1 + s_2$

Оцінка трудомісткості розкриття первинного ключа для кожного з учасників, так само як і для нелегального зловмисника, становить $2n$ спроб дешифрування в гіршому випадку і $2n$ -1 в середньому.

Схема Шаміра

Схема поділу секрету, запропонована Шаміром, побудована за принципом поліноміальної інтерполяції. Нехай заданий багаточлен ступеня (m-1) над кінцевим полем Галуа GF (q) з q елементів (GF (q) = Z_q, де - q просте).

Секретний ключ задається вільним членом a_0, всі інші коефіцієнти багаточлена - випадкові елементи поля. Поле GF (q) відомо всім учасникам. Кожна з n тіней є точкою (x_i, y_i) кривої, що описується багаточленом $P(x)$, $x_i \neq 0$.

Скориставшись інтерполяційною формулою Лагранжа, наведеною нижче, можна відновити вихідний багаточлен (і, отже, секрет a_0,) по будь-яким m точках (тіням).

При цьому ймовірність розкриття секрету в разі довільних (m -1) тіней оцінюється як q-1, тобто в результаті інтерполяції по (m -1) точці секретом може бути будь-який елемент поля з однаковою ймовірністю. Отже, схема Шаміра є досконалою.

На Рис. 12.1 представлений спеціальний випадок; m = 2 (для відновлення секрету необхідно дві тіні).

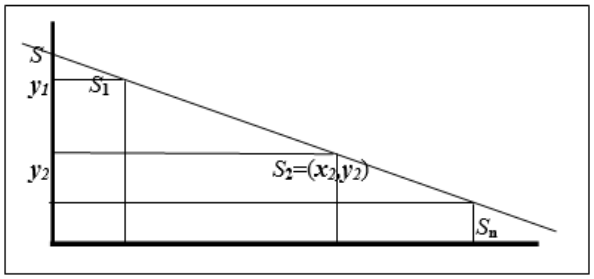


Рис. 12.1. Відновлення секрету по двом тіням

Таким чином, двочлен описує деяку пряму, яка перетинає з віссю у в точці s (секрет). Кожна тінь - точка на прямій. Отже, секрет може бути відновлений за двома довільними тінями, так як для однозначного визначення прямої достатньо двох довільних точок.

У разі завдання однієї тіні в якості шуканого секрету може бути обрана будь-яка точка на осі y, так як через одну точку можна провести безліч різних прямих, що перетинаються з віссю y в довільних точках.

Схема Блеклі

Схема поділу секрету Блеклі має геометричну природу. Секрет є точкою в m- вимірному просторі. Відзначимо, що для випадку m > 2 всі геометричні побудови виконуються над кінцевим полем GF (q). Кожна з n тіней задається як гіперплощина в m- вимірному просторі. Визначення секрету зводиться до знаходження точки перетину m гіперплощин.

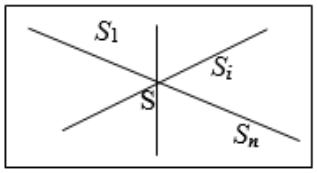


Рис. 12.2 Схема Блеклі

На Рис. 12.2 представлений спеціальний випадок схеми Блеклі. Кожна тінь - пряма, що проходить через цю точку. Для відновлення секрету S (точки на площині) необхідно мати принаймні дві тіні.

Метод «розшаровування» зображення

Наор і Шамір розробили оригінальний варіант схеми поділу секрету. Автори схеми запропонували наступну постановку задачі. Нехай є секретне зображення (в деякому графічному форматі), яке необхідно розподілити серед n учасників.

Для цього зображення «розшаровується» на n складових (тіней) таким чином, що об'єднання будь-яких m з них дозволяє відновити зображення. Зрозуміло, що ні одна з тіней не дає уявлення про вихідне зображення. Більш того, ніякі (m-1) і менш тіней не дозволяють відновити вихідне зображення. Можливий варіант конструктивної реалізації схеми полягає в «розшаровуванні» зображення на чорно-білі клітинки пікселі з подальшою їх обробкою.

Схема є досконалою і проста в реалізації. Додаткова модифікація дозволяє отримувати як тіні не «шумові», а цілком змістовні зображення (наприклад, зображення пейзажу або будівлі і т.п.), що дозволяє приховати сам факт поділу секрету.

Протокол, який перевіряється, поділу секрету

У звичайних схемах поділу секрету розглядається пасивний супротивник, а саме, противником є не більше ніж (m-1) процесорів, які, об'єднавши свої частки, намагаються отримати будь-яку інформацію про значення секрету.

Якщо ж на фазі відновлення секрету діє *активний противник*, можливі наступні ситуації:

дилер намагається роздати процесорам неправильні тіні,

процесори A_i мають на меті зірвати відновлення секрету S, посилаючи чесним учасникам замість частки секрету будь-якої іншої інформації.

Подібна дія призведе до того, що секрет ще не буде правильно відновлений. При цьому неможливо визначити, хто з учасників коаліції винен в цьому. У гіршому випадку учасники просто не виявлять, що відновлений секрет не відповідає вихідному розділеному секрету.

Для захисту в подібній ситуації використовується *протокол, який перевіряється, поділу секрету* або верифікована схема поділу секрету.

Можливе рішення полягає в застосуванні *доказу з нульовим розкриттям*, що дозволяє одному учаснику довести коректність виконаних ним обчислень і знання тіні, отриманої в результаті поділу секрету, іншому учаснику схеми, не розкриваючи при цьому самої тіні. Даний протокол дозволяє вирішувати наступну практичну задачу: кожен з учасників може переконатися в тому, що отримана ним тінь, об'єднана з тінями інших учасників, дозволяє гарантовано відновити розділений секрет.

Протокол, який перевіряється, поділу секрету за схемою Шаміра

Для прикладу розглянемо схему Шаміра і ситуацію, в якій коаліція з m учасників намагається відновити секрет S за умови, що один з учасників - брехун і надає

«помилкову» тінь (не ту, що була отримана в результаті поділу секрету). Фаза поділу секрету починається з того, що дилер публікує секрет S в

«зашифрованому» вигляді (точніше було б сказати - виконує прив'язку до рядка S, за аналогією з прив'язкою до біту).

Дилер вибирає випадковий поліном ступеня m-1 $Q(x) = a_0 + a_1 x + \ldots + a_{m-1}x^{m-1}$, де $a_0 = S$

обчислює $r_i = g^{a_i} \bmod p$ ($i = 1,\ldots, n$) і публікує $r_1,\ldots, r_n$.

За допомогою цієї інформації кожен процесор A_j може перевірити, що значення s_j, отримане ним від дилера, дійсно є часткою секрету S. Для цього він повинен перевірити рівність

$g^{s_j} = r_0 (r_1)^j \ldots (r_{m-1})^{j^{m-1}} \bmod q$

Справді:

$$r r^{\,j} r^{\,j} \ldots r^{\,j} = g^{a_0} g^{a_1 j} g^{a_2 j} \ldots g^{a_n j} = g^{a_0 + a_1 j + a_2 j + \ldots + a_{m-1} j} = g^{Q(j)} \bmod q \; 0 \; 1 \; 2 \; {}_{m-1}$$

На *фазі роздачі секрету* дилер обчислює: $s_j = Q(j), j = 1,\ldots,n$

і посилає це значення (частки або тіні секрету) процесору A_j по захищеному каналу.

Конструкцію протоколу для фази відновлення секрету розглянемо в найбільш простому

випадку, коли дилер чесний. На цій фазі кожен процесор A_j надсилає кожному другому процесора A_j свою частку s_j.

Всякий чесний учасник A_i, отримавши деяке значення s_j від A_j, перевіряє це значення, як описано вище, і відкидає всі частки s_j, які не пройшли перевірку. Оскільки чесних учасників не менше m, A_i отримає принаймні m правильних частки секрету.

Використовуючи алгоритм відновлення секрету зі схеми Шаміра, відновимо значення S. Частки секрету є гіперплощинами в просторі багаточленів над полем Z_q. Для відновлення секрету будується єдина точка перетину m гіперплощин за допомогою інтерполяційного полінома Лагранжа:

$$Q'(x) = \sum_{i \in I} Q(x_i)\pi_i \text{ , где } \pi_i = \prod_{j \in I, j \neq i} \frac{x - x_j}{x_i - x_j}$$

На відміну від звичайних схем поділу секрету стійкість даного протоколу ґрунтується на припущенні про обчислювальну складність завдання дискретного логарифмування. Тому, якщо в звичайних схемах поділу секрету потрібно, щоб будь-яка підмножина учасників, що не становить кворуму, не отримувало жодної інформації про секрет, то в багатьох схемах в яких перевіряється поділ секрету, така підмножина лише

«не може відновити» секрет в тому сенсі, що для його відновлення потрібно вирішити деяку гіпотетично важку обчислювальну задачу. У розглянутому вище прикладі всякий учасник міг би дізнатися секрет S, якби він умів обчислювати дискретні логарифми.

Протоколи конфіденційного обчислення

Припустимо, що за допомогою протоколу поділу секрету розділені два секрета S_1 і S_2 і що обидва ці секрети є числами. Тепер уявімо собі ситуацію, що після цього було потрібно розділити секрет $S = S_1 + S_2$. Це може зробити дилер за допомогою того ж протоколу. А чи можуть процесори виконати те ж саме без участі дилера?

Нехай $Q_1(x) = a_0 + a_1 x + \ldots + a_{m-1}x^{m-1}$

і $Q_2(x) = b_0 + b_1 x + \ldots + b_{m-1}x^{m-1}$

- поліноми, які використовувалися для поділу секретів S_1 і S_2 відповідно.

Нехай $r_i^{(1)} = g^{ai} \bmod q$ і $r_i^{(2)} = g^{bi} \bmod q$, $i = 0,...,m-1$. - значення, що використовуються для перевірки правильності часток секрету.

І нехай $s_j^{(1)} = Q_1(j)$ і $s_j^{(2)} = Q_2(j)$, $j = 1,...,n$ - частки секретів S_1 і S_2, отримані процесором A_j.

Ясно, що $Q(x) = Q_1(x) + Q_2(x)$ -поліном ступеня m-1 і $Q(0) = S$.

Тому кожен процесор A_j може обчислити частку s_j секрету S просто по формулі:

$$s_j = s_j^{(1)} + s_j^{(2)}$$

Ці частки перевіряються за допомогою значень $r_i = r_i^{(1)} r_i^{(2)} \bmod q$.

Виконуючи такого роду обчислення над частками секретів, процесори можуть обчислити будь-яку функцію над кінцевим полем «перевіряємим чином». Типова задача тут така. Потрібно обчислити значення функції f на деякому наборі значень аргументів $y1$, ..., yk. За допомогою схеми, який перевіряється, поділ секрету обчислюється частки, $i^{(1)},..., x^{(k)}$, (i = 1, ... n) цих значень. На початку виконання протоколу частка x_i відома процесору A_i і тільки йому. Протокол повинен забезпечувати обчислення значення:

$$f(x_1^{(1)},..., x_n^{(k)}) = f(y_1, \ldots, y_k)$$

таким чином, щоб для деякого параметра m:

в результаті виконання протоколу будь-яка підмножина з не більш ніж m-1 процесорів не отримувало жодної інформації про значення x_i інших процесорів (крім тієї, яка випливає з відомих їм часток) і значення функції $f(x_i^{(1)},..., x_i^{(k)})$

при будь-яких діях нечесних учасників інші учасники обчислюють правильне значення $f(y_1, ..., y_k)$, якщо тільки кількість чесних учасників не менше m.

Приклади завдань

1. Описати протокол сліпого підпису RSA та показати, як з S(P(m)) отримується коректний підпис під m без розкриття m підписувачу.
2. Для варіанта “підпис лише прийнятних повідомлень” пояснити ідею відкриття всіх конвертів, крім одного, і оцінити ймовірність отримання підпису під неприйнятним повідомленням.
3. Побудувати схему сценарію застосування незаперечного (невидимого) підпису та виписати, які саме дані перевіряються у багаторандовому протоколі верифікації.
4. Для (m,n)-порогової схеми сформулювати структуру доступу \Gamma та на прикладі показати, які коаліції є дозволеними, а які — ні.
5. Для схеми Шаміра з параметрами (m,n) виконати розподіл секрету S над GF(q) і відновити S за допомогою інтерполяції Лагранжа з будь-яких m тіней.

Контрольні питання

1. У чому полягає відмінність “звичайного” неінтерактивного ЕЦП від спеціальних видів підпису, що потребують багаторандової взаємодії?
2. Яка криптографічна ідея лежить в основі сліпого підпису та які властивості він забезпечує у фінансових застосуваннях (анонімність/невідстежуваність)?
3. Що таке незаперечний (undeniable) підпис і чому його неможливо перевірити без участі підписувача (які переваги й ризики це створює)?
4. Дайте визначення (m,n)-порогової схеми поділу секрету та поясніть вимоги: відновлення секрету з m тіней і інформаційна “порожнеча” для будь-яких m-1 тіней.
5. Навіщо потрібен верифікований поділ секрету (VSS) і які атаки він покриває (нечесний дилер, “брехливі” учасники на фазі відновлення)?

Література

1. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton : CRC Press, 1996.
2. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed. New York : John Wiley & Sons, 1996.
3. Chaum D. Blind signatures for untraceable payments // Advances in Cryptology (CRYPTO’82). Boston : Springer, 1983. P. 199–203.
4. Shamir A. How to share a secret // Communications of the ACM. 1979. Vol. 22, No. 11. P. 612–613.
5. Blakley G. R. Safeguarding cryptographic keys // Proceedings of the National Computer Conference. 1979. Vol. 48. P. 313–317.

Лекція №13 Управління ключами. Генерація ключів за стандартом ANSI X9.17. Розподіл, оновлення, накопичення, зберігання та депонування ключів.

Під *ключовою інформацією* розуміється сукупність всіх діючих в ІС ключів. Якщо не забезпечено досить надійне управління ключовою інформацією, то заволодівши нею, зловмисник отримує необмежений доступ до всієї інформації.

Управління ключами являє собою найважчу частину криптографії. Проектувати криптографічні алгоритми та протоколи не просто, проте можна скористатися результатами теоретичних досліджень.

Криптоаналітики часто атакують симетричні і асиметричні криптосистеми, використовуючи недоліки схеми розподілу ключів. Іноді вкрасти ключ коштує набагато дешевше, ніж побудувати суперкомп'ютер або найняти криптоаналітика. Відомо, що деякі комерційні продукти не забезпечують надійного зберігання ключів.

Крім того, якщо для забезпечення конфіденційного обміну інформацією між двома користувачами процес обміну ключами тривіальний, то в ІС, де кількість користувачів становить десятки і сотні, управління ключами - серйозна проблема.

Управління ключами - інформаційний процес, що включає в себе три елементи:

генерацію ключів;

накопичення ключів;

розподіл ключів.

Розглянемо, як вони повинні бути реалізовані для того, щоб забезпечити безпеку ключової інформації в ІС.

Генерація ключів

Безпека криптосистеми зосереджена в ключі. Якщо використовується криптографічно вразливий процес генерації ключів, то криптосистема в цілому вразлива. Криптоаналітику не потрібно займатися криптоаналізом алгоритму криптографічного перетворення, досить проаналізувати алгоритм генерації ключів.

Розмірність ключового простору

Один з основних параметрів стійкості криптосистеми - розмірність ключового простору. Він забезпечує захист від силової атаки, заснованої на повному переборі всіх можливих ключів. Наприклад, DES використовує 56-бітовий ключ. Якщо будь-яка правильно задана послідовність з 56 біт може бути ключем, то існує 2^{56} (10^{16}) можливих ключів. У таблиці 13.1 наведені оцінки розмірності ключового простору для найбільш часто зустрічаючих алфавітів.

У таблиці 13.2 наведено час, необхідний для здійснення силової атаки при продуктивності

один мільйон ключів в секунду. При такій продуктивності можливо розкрити ключі, що складаються з цифр і символів нижнього регістру довжиною до 8 байтів, алфавітно-цифрові ключі до 7 байтів, ключі з печатних символів і ASCII-символів до 6 байтів і ключі з 8-бітових ASCII-символів довжиною до 5 байтів.

Простір ключів	4 байта	5 байтів	6 байтів	7 байтів	8 байтів
Рядкові букви (26)	4.6 105	1.2 107	3.1 108	8.0 109	2.1 1011
Рядкові букви і цифри (36)	1.7 106	6.0 107	2.2 109	7.8 1010	2.8 1012
Алф. і цифр, символи (62)	1.5 107	9.2 108	5.7 1010	3.5 1012	2.2 1014
Печатки, символи (95)	8.1 107	7.7 109	7.4 1011	7.0 1013	6.6 1015
ASCII (128)	2.7 108	3.4 1010	4.4 1012	5.6 1014	7.2 1016
8- бітові ASCII (256)	4.3 109	1.1 1012	2.8 1014	7.2 1016	1.8 1019

Таблиця 13.1. Оцінки потужності ключового простору

Простір ключів	4 байта	5 байтів	6 байтів	7 байтів	8 байтів
Рядкові букви (26)	0.5 сек	12 сек	5 хв	2.2 годин	2.4 дня
Рядкові букви і цифри (36)	1.7 сек	1 хв	36 хв	22 годин	33 дня
Алф. і цифр, символи (62)	15сек	15 хв	16 годин	41 день	6.9 років
Печатки, символи (95)	1.4 хв	2.1 годин	8.5 дня	2.2 року	210 років
ASCII (128)	4.5 хв	9.5 годин	51 день	18 років	2300 років
8- бітові ASCII (256)	1.2 годин	13 днів	8.9 років	2 300 років	580000 років

Таблиця 13.2. Трудоємність силової атаки при виробн. 1 млн. ключів в секунду

Випадкові ключі

Не варто використовувати невипадкові ключі з метою легкості їх запам'ятовування. У серйозних ІС використовуються спеціальні апаратні і програмні методи генерації випадкових ключів.

Добрими ключами є послідовності випадкових бітів, створені деяким автоматичним процесом. Ідеальними генераторами є пристрої на основі "натуральних" випадкових процесів. Наприклад, генерація ключів на основі *білого радіошуму*. Іншим випадковим об'єктом - математичним - є десяткові знаки ірраціональних чисел, наприклад

π або *e*, які обчислюються за допомогою стандартних математичних методів.

Як правило, використовують датчики псевдовипадкових чисел (ПСЧ). Однак ступінь випадковості їх генерації повинна бути досить високою. В ІС з середніми вимогами захищеності цілком прийнятні програмні генератори ключів, які обчислюють ПСЧ як складну функцію від поточного часу і (або) числа, введеного користувачем. Всі можливі ключі при цьому повинні бути рівноімовірні.

Для генерації ключів важливо використовувати хороший генератор випадкових чисел, але набагато важливіше використовувати надійні процедури управління і перевірки ключів. Деякі алгоритми шифрування мають ключі, які мають ряд специфічних властивостей, які полегшують їх розкриття. Якщо ці ключі відомі, необхідно виконувати їх перевірку в процесі генерації. У разі виявлення слабкого ключа генерується новий

ключ. У DES існує 16 «слабких» ключів на 2^{56} можливих, так що ймовірність отримання такого ключа досить мала.

Генерація ключів для асиметричних криптосистем складніше; часто ключі повинні володіти певними математичними властивостями (можливо, вони повинні бути простими числами, квадратичними відрахуваннями і т.п.). Стартові послідовності для генераторів ключів повинні бути випадковими.

Генерація ключів за стандартом ANSI X9.17

Стандарт ANSI X9.17 визначає спосіб генерації ключів (див. Рис.13.1). Спосіб підходить для генерації сеансових ключів або псевдовипадкових чисел. Для генерації ключів використовується DES-шифрування, але воно може бути легко замінено будь-яким іншим

криптоалгоритмом.

Нехай $E_k(x)$ - це повідомлення x, зашифроване на спеціальному ключі k, передбаченому для генерації секретних ключів. V_0 - секретна 64-бітова стартова послідовність. T - мітка часу. Для генерації випадкового ключа R_i необхідно виконати наступні обчислення:

$$R_i = E_k(E_k(T_i) \oplus V_i)$$

Для генерації V_{i+1} необхідно обчислити:

$$V_{i+1} = E_k(E_k(T_i) \oplus R_i)$$

Для перетворення R_i в ключ DES необхідно видалити кожен восьмий біт. Довший ключ виходить шляхом конкатенації кількох коротких ключів. Так для отримання 128- бітового ключа необхідно спочатку згенерувати пару ключів, а потім виконати їх конкатенацію.

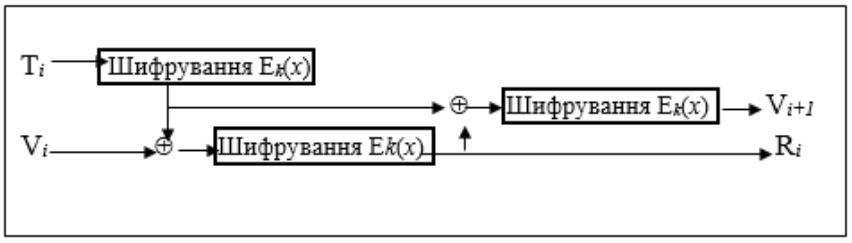


Рис. 13.1. Генерація ключів за стандартом ANSI X9.17

Нерівносильні ключі

Іноді необхідно (наприклад, якщо криптографічний пристрій потрапляє в руки до супротивника), щоб криптоалгоритм забезпечував адекватну криптостійкість тільки з секретними ключами деякого спеціального виду (легальні ключі), а з усіма іншими ключами (нелегальними) для шифрування використовувався б менш криптостійкий алгоритм. Можна зробити так, щоб ймовірність випадкової генерації легального ключа була дуже мала.

Простий спосіб полягає в генерації ключа, що складається з двох частин: безпосередньо ключа і деякої допоміжної фіксованої послідовності біт, зашифрованої на цьому ключі. Перед шифруванням блоку даних розшифровується допоміжна послідовність. Якщо результатом виявляється фіксована послідовність біт, то застосовується криптостійкий алгоритм шифрування, якщо немає, то менш криптостійкий алгоритм. Якщо алгоритм має 128-бітний ключ і 64-бітний розмір блоку, то довжина

повного ключа 192 біта. Таким чином, існує 2^{128} різних легальних ключів, однак імовірність випадкового вибору легального ключа 2^{-64}.

Резервні ключі

Якщо ключ шифрування з якої-небудь причини буде втрачено, буде втрачена і вся зашифрована на ньому інформація. Дублювання ключів збільшує ризик їх компрометації.

Кращим рішенням є використання схеми поділу секрету. Коли Аліса генерує ключ, вона одночасно ділить його на кілька частин і потім посилає ці частини в зашифрованому вигляді різним уповноваженим особам. Жодна з цих частин сама по собі не є ключем, але, зібравши їх разом, можна відновити ключ. Згідно з ідеологією поділу секрету, ключ може бути відновлений навіть при втраті однієї або декількох частин.

Можна просто зберігати різні частини, зашифровані відкритими ключами відповідних довірених осіб, на своєму жорсткому диску. Таким чином, ніхто не бере участі в управлінні ключами, поки це не стане необхідним.

Інша схема резервування використовує інтелектуальні картки для тимчасового вручення ключів. Аліса може записати ключ, яким зашифрований її жорсткий диск, на інтелектуальну картку і передати її Бобу. Боб може використовувати картку для доступу до жорсткого диска Аліси, але так як ключ зберігається на картці, Боб не зможе його розкрити.

Крім того, такий спосіб допускає двосторонній контроль: Боб може упевнитися в тому, що ключ відкриває диск Аліси, а Аліса завжди може перевірити, чи був використаний ключ.

Завдання розподілу ключів

Як Боб, отримавши ключ, дізнається, що ключ переданий саме Алісою, а не будь- ким іншим, хто видає себе за Алісу?

Якщо Аліса посилає свій ключ через довіреного кур'єра, то кур'єру повинен довіряти також і Боб.

Якщо ключ зашифрований ключем шифрування ключів, то Боб повинен бути впевнений, що цей ключ є тільки у Аліси.

Якщо використовується електронний цифровий підпис, Боб при перевірці підпису повинен довіряти відкритому ключу. Він також повинен бути впевнений, що парний секретний ключ Аліси не скомпрометований.

Якщо Центр розподілу ключів (ЦРК) підписує відкритий ключ Аліси, Боб повинен бути впевнений в автентичності відкритого ключа ЦРК.

Асиметрична криптографія, що застосовується разом з електронними цифровими підписами і надійними ЦРК, сильно ускладнює підміну одного ключа іншим. Боб ніколи не може бути абсолютно впевнений в тому, що зловмисник його не контролює, однак Боб може знати напевно, що підміна ключа вимагатиме набагато більше ресурсів, ніж по силам дістати реальному зловмисникові.

Іноді ключі викривляються при передачі. Використання викривленого ключа при дешифруванні не дозволяє отримати відкритий текст, відповідний прийнятому шифртексту. Всі ключі повинні передаватися *з виявленням або виправленням помилок*. Одним з найбільш широко використовуваних методів є шифрування ключем деякої константи і передача перших декількох байтів отриманого шифртекста разом з ключем. Одержувач знає константу і виконує аналогічні дії - шифрує константу на отриманому ключі. Якщо прийнятий шифртекст збігається з результатом шифрування одержувача, то ключ був переданий без помилок. Імовірність помилки знаходиться в діапазоні від 2^{-16} до

2^{-32}.

Розподіл секретних ключів

Розподіл секретних ключів - найвідповідальніший процес в управлінні ключами.

До нього висуваються дві вимоги:

Оперативність і точність розподілу.

Скритність ключів, які розподіляються.

Розподіл ключів між користувачами комп'ютерної мережі реалізується двома різними підходами:

Прямий обмін ключами між користувачами інформаційної системи.

Шляхом створення одного або декількох центрів розподілу ключів (ЦРК).

Ключі шифрування ключів, загальні для пари користувачів, зручно використовувати в мережах з невеликим числом абонентів, однак зі зростанням числа абонентів така схема швидко стає громіздкою. Так як кожна пара абонентів повинна обмінятися ключами, загальне число обмінів ключами в мережі з n абонентів дорівнює n (n-1)/2. У мережі з шістьма користувачами потрібно 15 обмінів ключами. У мережі з 1000 користувачів знадобиться вже близько 500 000 обмінів ключами. У цьому випадку розподіл ключів здійснюється за допомогою центрального сервера.

У разі прямого обміну ключами проблема полягає в тому, щоб надійно засвідчити справжність суб'єктів.

Відомий спосіб організації ЦРК на базі симетричних криптосистем має ряд недоліків:

По-перше, в центрі розподілу відомо, кому і які ключі призначені, і це дозволяє читати всі повідомлення, що циркулюють в ІС. Можливі зловживання значно впливають на захист.

По-друге, значний недолік полягає в тому, що для отримання секретного ключа абонента необхідно звернутися в ЦРК з відповідним запитом, тобто центр повинен обробляти потоки запитів в режимі *on-line*. Очевидно, що навантаження на ЦРК в мережі з великим числом абонентів може бути досить велика.

В обох підходах повинна бути гарантована справжність сеансу зв'язку. Це можна забезпечити двома способами:

Механізм запиту-відповіді, який полягає в наступному. Якщо користувач А бажає бути впевненим, що повідомлення, які він отримує від В, не є помилковими, він включає те що посилається для В повідомлення непередбачуваний елемент (запит). При відповіді користувач В повинен виконати деяку операцію над цим елементом (наприклад, додати 1). Це неможливо здійснити заздалегідь, так як невідомо, яке випадкове число прийде в запиті. Після отримання відповіді з результатами дій, користувач А може бути впевнений, що сеанс є справжнім. Недоліком цього методу є можливість встановлення закономірності (хоча і складної) між запитом і відповіддю.

Механізм позначки часу ("*тимчасової штемпель*"). Він має на увазі фіксацію часу для кожного повідомлення. У цьому випадку кожен користувач ІС може знати, наскільки "старим" є повідомлення, яке прийшло.

В обох випадках слід використовувати шифрування, щоб бути впевненим, що відповідь послана не зловмисником і штемпель позначки часу не змінений.

При використанні відміток часу постає проблема допустимого тимчасового інтервалу затримки для підтвердження автентичності сеансу. Адже повідомлення з "тимчасовим штемпелем" в принципі не може бути передано миттєво. Крім цього, комп'ютерний годинник одержувача і відправника не можуть бути абсолютно синхронізовані. Яке запізнювання "штемпеля" вважати підозрілим? У реальних ІС, наприклад в системах оплати кредитних карток, де використовується саме другий механізм встановлення автентичності та захисту від підробок, використовуваний інтервал становить від однієї до кількох хвилин. Велике число відомих способів крадіжки електронних грошей засноване на "вклинюванні" в цей проміжок з підробленими запитами на зняття грошей.

Розподіл відкритих ключів

Для відкритих ключів проблема розподілу також актуальна. Основна відмінність відкритих ключів від секретних полягає в тому, що їх не треба ховати, але зате їх треба захищати від підробки. Без додаткових заходів безпеки обмін відкритими ключами вразливий з точки зору атаки, відомої під назвою «людина посередині». Зловмисник може підмінити відкриті ключі законних користувачів на свої власні і отримати повну можливість дешифрувати і підміняти всі повідомлення. Атака можлива навіть якщо відкриті ключі зберігаються в базі даних.

Сучасний підхід до вирішення проблеми заснований на властивостях асиметричних криптосистем (асиметрична схема управління ключами) і застосуванні спеціальних сертифікатів: для захисту від підміни відкриті ключі передаються разом з сертифікатами автентичності.

Сертифікат відкритого ключа абонента А створюється учасником протоколу Х і має вигляд:

$C_{XA} = S_X$ ("А"|| час || k_A)

де S_X - цифровий підпис Х, а k_A - відкритий ключ А.

Щоб можна було довіряти сертифікату C_{XA}, потрібні два, що не залежать один від одного, умови:

впевненість в тому, що ключ, за допомогою якого створений цифровий підпис, дійсно належить учаснику Х. Така впевненість може в свою чергу ґрунтуватися на іншому сертифікаті C_{YX}.

довіра по відношенню до учасника Х, тобто впевненість в тому, що Х зі злого наміру або помилково не підпише підроблений ключ.

Є кілька схем використання сертифікатів.

Централізоване управління

Воно засноване на організації спеціальних довірених Центрів Сертифікації (ЦС), які видають сертифікати, що складаються з відкритого ключа абонента і унікальної ідентифікуючої інформації, скріплених цифровим підписом ЦС.

Абоненти обмінюються сертифікатами, отриманими від ЦС при встановленні захищеного режиму обміну, або поміщають їх в загальнодоступні мережеві довідники. Перед використанням відкритого ключа одержувача для шифрування адресованого йому повідомлення відправник може переконатися в його автентичності, перевіривши цифровий підпис сертифіката одержувача за допомогою відомого відкритого ключа центру сертифікації. Аналогічна перевірка відкритого ключа відправника виконується перед перевіркою цифрового підпису прийнятого одержувачем повідомлення.

Централізована схема застосовується при організації захищеного WWW-обміну за допомогою протоколу SSL в продуктах Microsoft і Netscape. При цьому виділяється кілька спеціальних учасників, які називаються *Certification Authority*. Відкритий ключ будь-якого іншого учасника повинен бути підписаний одним з виділених учасників.

Ієрархічна схема застосовується в системі захищеної електронної пошти PEM. Учасники утворюють дерево, причому кожен з них довіряє підписувати ключі всім своїм предкам в дереві і сам підписує ключі своїх безпосередніх нащадків.

Розподілене управління

У деяких випадках спосіб централізованого управління ключами працювати не буде. Можливо, не існує такого ЦС, якому довіряли б обидва абонента. Можливо, абоненти довіряють тільки своїм друзям або взагалі нікому не довіряють.

«Демократична» схема (або розподілене управління ключами), застосовується в програмі PGP. Всі учасники рівноправні, і кожен може підписати ключ будь-якого учасника. Проблема автентичності відкритого ключа вирішується за допомогою поручителів. Поручителі - це абоненти криптомережі, які підписують відкриті ключі своїх друзів, партнерів і т.п.

Наприклад, коли Боб створює свій відкритий ключ, він передає копії ключа своїм друзям. Вони знають Боба, тому кожен з них підписує ключ Боба і видає Бобу копію свого підпису. Тепер, коли Боб пред'являє свій ключ чужій людині, наприклад Алісі, він пред’являє його разом з підписами цих поручителів. Якщо Аліса також знає цих поручителів і довіряє їм, у неї з'являється підстава для довіри Бобу. Для запобігання підміни ключа, перш ніж підписувати, поручитель повинен бути впевнений, що ключ належить саме Бобу.

Вигода цього механізму - відсутність ЦС, якому кожен повинен довіряти. А негативною стороною є відсутність гарантій того, що Аліса, яка отримала відкритий ключ Боба, знає когось із поручителів, і, отже, немає гарантій, що вона повірить в справжність ключа.

Пропонується довіряти ключу учасника A, якщо існує ланцюжок сертифікатів

$C_{X_0 X_1}$, $C_{X_1 X_2}$, … , $C_{X_k A}$

При цьому слід бути впевненим в ключі X_0 і довіряти всім X_i підписувати ключі. Для перестраховки довжина ланцюжка не повинна перевищувати деяке невелике значення, наприклад 3.

Залишається єдина проблема - звідки береться перший відкритий ключ в ланцюжку. Цей ключ повинен бути переданий по захищеному каналу - при особистому контакті, спеціальною поштою, у вигляді підписаного документа з печаткою і т. п. Можна також передати ключ по електронній пошті, а по захищеному каналу - тільки *відбиток ключа*, результат обчислення криптографічно стійкою хеш функції від відкритого ключа.

Для отримання сертифіката в початковий момент часу потрібна організація секретного каналу для передачі в ЦС відкритого ключа та ідентифікуючої інформації.

При заміні відкритого ключа, необхідна для отримання сертифікату, інформація підписується на старому секретному ключі (якщо він не скомпрометований) і передається в ЦС по відкритому каналу. Результат перевірки підпису є критерієм для прийняття рішення про зміну відкритого ключа та видачі нового сертифікату.

Цікаве застосування можна знайти для сертифіката виду

$C_{AA} = S_A(\text{"A"} \,||\text{час}||\, k_A)$

У централізованій схемі він використовується для ключів, що належать виділеним учасникам, що досить безглуздо. У «демократичній» схемі такий сертифікат є сертифікатом відкликання ключа і публікується власником ключа, якщо закритий ключ було розголошено або викрадено. Ідея полягає в тому, що тільки власник ключа або викрадач можуть створити такий сертифікат.

Компрометація ЦС, так само як і в разі ЦРК, призводить до компрометації всієї мережі, однак пропонована асиметрична схема дозволяє зняти навантаження з ЦС, так як кількість запитів на отримання сертифікатів суттєво менше, ніж кількість запитів на встановлення захищеного режиму обміну інформацією на рівні абонентів.

Атаки на ЦС

Оскільки ЦС грає провідну роль в інфраструктурі криптомережі, розглянемо деякі прості атаки на ЦС на прикладі асиметричної схеми управління ключами.

Отримання сертифікату на чуже ім'я. Припустимо, Боб бажає видати себе за Алісу. Якщо Боб вміє підписувати повідомлення від особи Аліси, він може, наприклад, розпорядитися про зняття певної грошової суми з рахунку Аліси в банку. Для цього Боб генерує пару ключів (відкритий і секретний) і посилає відкритий ключ (по секретному каналу) в ЦС з твердженням, що він є Алісою. Боб досягне мети, якщо йому вдасться обдурити ЦС і отримати сертифікат на ім'я Аліси.

Для запобігання подібної атаки необхідно точно ідентифікувати особу запитувача. ЦС може, наприклад, вимагати особистої присутності і/або пред'явлення посвідчення особи. Кожен ЦС може мати свою процедуру аутентифікації, а також методи зберігання копій сертифікатів користувачів. Очевидно, що надійна організація служб ЦС визначає надійність криптомережі в цілому.

Розкриття секретного ключа центру сертифікації. Зловмисник, який розкрив секретний ключ ЦС, може підробляти сертифікати. Для запобігання витоку секретної інформації ЦС повинен зберігати свій секретний ключ, а також підписані на ньому сертифікати в спеціальному наднадійному, ударостійкому, захищеному від електромагнітних випромінювань, електронному приладі з енергонезалежною пам'яттю, так званий пристрій Підпису Сертифікатів (ППС).

Відомо, що силова атака на криптосистему RSA, введений ряд стандартів і широко використовується для шифрування і аутентифікації, зводиться до задачі розкладання великого двоскладного модуля на прості множники. Причому модуль відомий і входить у відкритий ключ ЦС. Дана обставина змушує ЦС використовувати наддовгі ключі (від 1000 біт і більше) і регулярно їх оновлювати.

Періодичність процедури поновлення ключів в ієрархічній структурі ЦС збільшується в напрямку від верхніх рівнів ієрархії до нижніх. Іншими словами, часта зміна ключів в ЦС верхнього рівня може бути непрактична, тому що веде за собою зміну ключів у великого числа користувачів криптомережі.

Підкуп співробітників ЦС. Інша модель атаки розглядає можливість підкупу Боба з боку Аліси. У цій ситуації Боб є співробітником ЦС, і мета підкупу полягає в отриманні Алісою сертифіката на ім'я Фреда. Отримавши сертифікат, Аліса зможе посилати повідомлення від імені Фреда, і всі ці повідомлення, в силу легальності виданого сертифіката, будуть адекватно сприйматися іншими користувачами криптомережі.

Таким чином, для досягнення мети необхідне об'єднання двох або більше зловмисників (коллаборативна атака). З огляду на можливість подібної атаки на ЦС, застосовують спеціальний метод розмежування доступу до ППС, заснований на відомій криптографічній техніці поділу секрету. Так, наприклад, для доступу до ППС може знадобитися участь декількох співробітників ЦС.

Процедура передбачає пред'явлення спеціальних секретних ключів (тіней основного ключа доступу), перевірку їх автентичності та дозвіл доступу в тому випадку, якщо число пред'явлених тіней не менш заданого порогового значення. Необхідно, однак, пам'ятати, що якщо потік запитів на видачу сертифікатів контролюється однією людиною, то описана вище атака може бути успішною.

Підробка старих документів. Інша атака полягає в тому, що Аліса здійснює силову атаку для підбору модуля, що входить до складу відкритого ключа центру сертифікації. В результаті після закінчення певного часу (наприклад 15 років) їй вдається розкласти модуль і відновити старий секретний ключ ЦС. Термін дії цього секретного ключа вже закінчився, проте Аліса може підробити сертифікат 15-річної давності, який посвідчує помилковий відкритий ключ

Боба. В результаті Аліса може підробити будь- який документ з підписом Боба 15-річної давності.

Зазвичай після закінчення певного терміну, наприклад двох років з моменту підпису, електронний документ стає недоступним. Однак існує ряд ситуацій, в яких електронні документи повинні надійно зберігатися тривалий час (довгострокові контракти і т.п.), при цьому повинна бути забезпечена можливість перевірки їх автентичності та цілісності.

Один із способів полягає у використанні крім звичайних ключів спеціального *наддовготривалого ключа*. Такі ключі мають значно більший модуль і зберігаються надійніше, ніж всі інші. Якщо дія наддовготривалого ключа закінчується через 50 років, то всі підписані на ньому документи повинні мати такий же термін дії.

При цьому виникає проблема, пов'язана з необмеженим ростом списку анульованих сертифікатів, оскільки наддовготривалі ключі, так само, як і всі інші, можуть бути скомпрометовані і повинні зберігатися протягом терміну їх дії.

Для усунення зазначеного недоліку пропонується реєструвати наддовготривалі ключі за допомогою стандартної процедури для звичайних ключів, наприклад з терміном дії протягом двох років. Після закінчення дворічного періоду наддовготривалі ключі, якщо вони не були скомпрометовані, повинні бути ресертифікованим на наступні два роки. Тепер скомпрометований наддовготривалий ключ повинен зберігатися в САС протягом двох років, а не п'ятдесят. При цьому, однак, зловмисник може добитися свого, втручаючись в роботу служби єдиного часу з метою зміни періодичності процедури ресертифікації.

Проблема може бути вирішена введенням спеціальної служби *тимчасових міток* (СВМ). Для цього всі довготривалі електронні документи повинні реєструватися із спеціальними і цифровими мітками, що видаються СВМ. Цифрові мітки дозволяють упевнитися в тому, що термін дії ключа (ключів), на якому було підписано довгостроковий документ, не закінчився. Крім того, цифрові мітки дозволяють встановлювати законність довготривалих електронних документів навіть у тому випадку, коли секретний ключ був скомпрометований після того, як документ був підписаний.

Для того, щоб проставити цифрову мітку на підписаному електронному документі, потрібно хешувати цей документ і надіслати отримане значення в СВМ. У відповідь отримуємо завірену підписом СВМ цифрову мітку, що складається з отриманого від неї значення хеш-функції, часу і дати, проставлених СВМ.

Хешування дозволяє не розкривати зміст документа (хеш-функцію неможливо звернути) в процесі взаємодії з СВМ. Очевидно, що передача інформації між користувачем і СВМ здійснюється з урахуванням вимог безпеки та може бути проконтрольована на цілісність і автентичність. Надалі для доказу дати і часу підпису документа Аліса надає сам документ і цифрову мітку.

Перевіряючий обчислює значення хеш-функції представленого документа, порівнює отримане значення з тим, яке зберігається в цифрі і мітці, і потім перевіряє запевнену цифрову мітку підпис СВМ. Доказ буде прийнято при збігу значень хеш- функції і валідності цифрового підпису СВМ.

Як узагальнення сказаного про розподіл ключів слід сказати наступне. Завдання управління ключами зводиться до пошуку такого протоколу розподілу ключів, який забезпечував би:

можливість відмови від центру розподілу ключів;

взаємне підтвердження автентичності учасників сеансу;

підтвердження достовірності сеансу механізмом запиту-відповіді;

використання при обміні ключами мінімального числа повідомлень.

Оновлення ключів

Розподіл ключів при частій їх заміні - складна практична задача. Особливо актуальна ця проблема при розподілі секретних ключів без використання ЦРК. Оригінальне рішення проблеми - використання "блукаючих ключів". Ідея методу досить проста. Після того, як ключ використаний в одному сеансі по деякому правилу, відомому і відправнику, і одержувачу, він змінюється іншим. Якщо правило зміни ключів обережно дотримується і відправником, і

одержувачем, то в кожен момент часу вони мають однаковий ключ. Постійна зміна ключа ускладнює розкриття інформації зловмисником.

Основне завдання в реалізації цього методу - вибір ефективного правила зміни ключів. Найбільш простий шлях - генерація випадкового списку ключів. Зміна ключів здійснюється в порядку списку. Однак очевидно, що список доведеться якимось чином передавати. Необхідно, однак, пам'ятати, що секретність нового ключа визначається секретністю старого. Якщо зловмисникові вдалося розкрити старий ключ, він зможе виконати оновлення ключів самостійно.

Інший варіант - використання математичних алгоритмів, заснованих на так званих перебираючих послідовностях. На множини ключів шляхом однієї і тієї ж операції над елементом виходить інший елемент. Іноді таку схему називають *схемою поновлення ключа*. Для цього необхідна хеш-функція, відома і Алісі, і Бобу. Якщо Аліса і Боб застосують до загального ключу одну і ту ж хеш-функцію, вони отримають однаковий результат. Потім вони можуть вибрати з результату потрібні їм біти і створити новий ключ. Найбільш доступним є використання полів Галуа. За рахунок зведення в ступінь породжуючого елемента можна послідовно переходити від одного числа до іншого. Ці числа приймаються в якості ключів. Ключовою інформацією в даному випадку є вихідний елемент, який перед початком зв'язку повинен бути відомий і відправнику, і одержувачу.

Надійність таких методів повинна бути забезпечена з урахуванням відомості зловмисникові використовуваного правила зміни ключів.

Цікавим і перспективним завданням є реалізація методу "блукаючих ключів" не для двох абонентів, а для досить великої мережі, коли повідомлення пересилаються між усіма учасниками.

Накопичення ключів

Під *накопиченням ключів* розуміється організація їх зберігання, обліку та видалення. Оскільки ключ є найпривабливішим для зловмисника об'єктом, який відкриває йому шлях до конфіденційної інформації, то питанням накопичення ключів слід приділяти особливу увагу.

Секретні ключі ніколи не повинні записуватися в явному вигляді на носії, який може бути зчитаний або скопійований. Кожна інформація про використовувані ключі повинна зберігатися в зашифрованому вигляді. Ключі, які зашифровують ключову інформацію, називаються *майстер-ключами*. Бажано, щоб майстер-ключі кожен користувач знав напам'ять і не зберігав їх взагалі на будь-яких матеріальних носіях, але це не завжди можливо.

Стандарт ANSI X9.17 визначає два типи ключів: *ключі шифрування ключів* і *ключі даних*. На ключах шифрування ключів шифруються ключі даних при передачі по відкритим каналам. На ключах даних шифруються самі повідомлення.

Дуже важливою умовою безпеки інформації є періодичне оновлення ключової інформації в ІС. При цьому перепризначатися повинні як звичайні ключі, так і майстер- ключі. Ключі шифрування ключів повинні розподілятися вручну і змінюватися досить рідко, так як розмір шифрованих повідомлень незначний. Однак необхідно пам'ятати, що при компрометації ключа шифрування ключів будуть скомпрометовані всі зашифровані на ньому ключі даних. Зміна джерел інформації відбувається набагато частіше. В особливо відповідальних ІС оновлення ключової інформації бажано робити щодня.

У досить складній ІС один користувач може працювати з великим об'ємом ключової інформації, і іноді навіть виникає необхідність організації баз даних ключової інформації. Такі бази даних відповідають за прийняття, зберігання, облік і видалення використовуваних ключів.

Іншим рішенням проблеми розподілу є розбиття ключа на кілька різних частин і передача їх по різним каналам. Якщо противник збере всі частини, крім однієї, він все одно не зможе розкрити ключ (за умови, що трудомісткість силової атаки при відновленні відсутньої частини ключа досить велика). Питання оновлення ключової інформації пов'язаний і з третім елементом управління ключами - розподілом ключів.

Зберігання ключів

Можна зберігати ключ в пам'яті користувача. Однак це не дуже надійно. Інше рішення - зберігання ключа у вигляді картки з магнітною смугою, пластикового ключа з вбудованою мікросхемою або інтелектуальної картки. Користувач може ввести свій ключ в систему, вставивши фізичний носій в пристрій, що зчитує, вбудований в автономний шифрувальний/дешифрувальний пристрій або підключений до комп'ютера. Практично будь-який здатний усвідомити, що таке фізичний ключ, яке його значення і як його захистити. Додання криптографічному ключу деякої фізичної форми робить зберігання і захист такого ключа інтуїтивно зрозумілим. Хоч користувач може використовувати ключ, він не знає його і не може його скомпрометувати. Він може використовувати його тільки тим способом і тільки для тих цілей, які визначені вектором контролю.

Розумний спосіб полягає в поділі ключа на дві половини, одна з яких зберігається в терміналі, а друга записана в пам'ять портативного носія. Втрата носія не призводить до компрометації криптографічного ключа. Компрометація також неможлива при несанкціонованому доступі до терміналу. Отже, компрометація носія або терміналу нічого не дає - для компрометації ключа необхідно дістати обидві його частини.

Ключі, які важко запам'ятати, можна зберігати зашифрованими. Наприклад, секретний ключ RSA може бути зашифрований ключем DES і записаний на жорсткий диск. Для відновлення RSA-ключа користувач повинен виконати DES-дешифрування. В ідеалі ключ ніколи не повинен надаватися поза шифрувального/дешифрувального пристрою у відкритому вигляді. Ця мета не завжди досяжна, але до цього потрібно прагнути.

Скомпрометовані ключі

Всі протоколи, методи і алгоритми сучасної криптографії мають сенс тільки в тому випадку, якщо ключ зберігається в секреті. Однак ключ Аліси може бути вкрадений, втрачений або скомпрометований іншим способом. Якщо скомпрометований ключ використовувався в *симетричній криптосистемі*, Алісі доведеться замінити ключ і сподіватися, що збиток мінімальний.

Якщо цей секретний ключ *асиметричній криптосистемі*, то все ускладнюється, так як парний відкритий ключ може зберігатися на багатьох серверах в мережі. Якщо зловмисник отримав доступ до секретного ключа Аліси, він може видати себе за неї і, як наслідок, читати адресовані їй повідомлення і підписуватися за неї.

Якщо ключ розподіляє ЦРК, то Аліса зобов'язана повідомити йому про факт компрометації. Якщо ЦРК не використовується, то їй слід сповістити всім абонентам які можуть отримувати від неї повідомлення. Відповідна служба криптомережі повинна оприлюднити той факт, що будь-яке повідомлення, отримане після компрометації ключа, є підозрілим і що ніхто не повинен посилати повідомлення Алісі, використовуючи відповідний відкритий ключ.

При використанні централізованої схеми розподілу відкритих ключів скомпрометовані ключі поміщаються в спеціальний *список анульованих сертифікатів* (САС). Туди ж поміщаються і відкриті ключі з вичерпаним терміном дії. САС служить механізмом, що дозволяє виключити можливість використання скомпрометованих ключів. САС формується спеціальною службою ЦС і містить інформацію про сертифікати, виданих даними ЦС. Відзначимо, що список містить не самі скасовані сертифікати, а тільки інформацію, що дозволяє встановити, що сертифікат був скасований. Способи поширення САС можуть бути різними: від безпосередньої розсилки користувачам криптомережі до організації в мережі спеціальних вузлів.

Таким чином, перед перевіркою підпису отриманого повідомлення і після верифікації сертифіката (перевірки підпису сертифіката на відкритому ключі ЦС) необхідно встановити, чи входить даний сертифікат в САС. Якщо факт входження встановлено, прийняте повідомлення відкидається.

Добре, якщо Аліса знає, коли був скомпрометований її ключ. Механізм тимчасових міток дозволяє абонентам визначити, які повідомлення законні, а якісь підозрілі. Додаткові ускладнення виникають, якщо Аліса не знає точно, коли її ключ був скомпрометований. Наприклад, Аліса може побажати відмовитися від контракту, так як він був підписаний замість

неї людиною, яка вкрала у неї ключ. Якщо така відмова можлива, то хто завгодно зможе відмовитися від контракту, стверджуючи, що його ключ був скомпрометований до підписання контракту. Для вирішення подібних питань необхідно залучення незалежного арбітра. Цей приклад показує, як небезпечно мати один єдиний ключ (точніше, одну єдину пару ключів). Краще, щоб у Аліси були різні ключі (пари ключів) для різних додатків точно так само, як для різних замків потрібні різні фізичні ключі. Інші рішення проблеми включають біометричну аутентифікацію, обмеження на використання ключа, тимчасову затримку і додатковий підпис.

Час життя ключів

Жоден ключ шифрування не можна використовувати нескінченно. Час його дії повинно минати автоматично, подібно паспортам і ліцензіям. Причини цього в наступному.

Чим довше ви користуєтеся ключ, тим більше:

ймовірність його компрометації. Якщо ключ використовується протягом року, то ймовірність його компрометації набагато вище, ніж якби його використовували тільки один день;

втрати при компрометації ключа. Якщо ключ використовується для шифрування одного документа, то втрата ключа означає компрометацію тільки цього документа. Якщо той же самий ключ використовується для шифрування великих обсягів інформації, то його компрометація призводить до значних втрат;

спокуса докласти необхідних зусиль для його розкриття. Розкриття ключа, використовуваного протягом доби, дозволить прочитати всі повідомлення, передані протягом доби. Розкриття ключа, використовуваного протягом року, дозволить зловмиснику отримати доступ до секретної інформації, переданої протягом року:

У ряді випадків трудомісткість криптоаналізу визначається кількістю шифртекста, отриманих в результаті шифрування на одному ключі.

Для будь-якого криптографічного додатка необхідна стратегія, що визначає допустимий час життя ключа. У різних ключів можуть бути різні часи життя.

Час життя ключів повинно визначатися в залежності від значимості і кількості даних, зашифрованих протягом заданого періоду:

для криптографічного телефону має сенс використовувати ключ тільки протягом телефонної розмови, а для нової розмови використовувати новий ключ.

чим більше швидкість передачі даних по каналу зв'язку, тим, можливо, частіше доведеться міняти ключ.

Необхідно *збалансувати ризики*, пов'язані із заміною ключа. Ключі *шифрування ключів* так часто змінювати не потрібно. При цьому шифр-текста для криптоаналітика утворюється небагато, а відповідний відкритий текст (випадкові ключі) неможливо вгадати. У деяких додатках ключі шифрування ключів замінюються лише раз на місяць або навіть раз на рік. Однак, якщо ключ шифрування ключів скомпрометований, потенційні втрати катастрофічні. Звичайно ж, втрата цього ключа означає втрату всіх ключів, які були на ньому зашифровані. Ключ шифрування ключів повинен зберігатися в безпечному місці.

При шифруванні *даних тривалого зберігання* ключі міняти часто не можна. Щоденне дешифрування і повторне шифрування на новому ключі може привести до негативних наслідків - накопичення криптоаналітиком робочого матеріалу для подальшої атаки. Рішенням може послужити шифрування даних на деякому унікальному ключі з подальшим його шифруванням на ключі шифрування ключів.

Час життя секретних ключів *асиметричної криптографії* залежить від додатків. Секретні ключі для цифрових підписів можуть використовуватися роками (навіть протягом людського життя). У багатьох криптомережах термін дії секретних ключів обмежений двома роками. Старий ключ проте повинен зберігатися в секреті на випадок підтвердження підпису, зробленого під час дії старого ключа. Але для підписання нових документів повинен використовуватися новий ключ. Такий підхід дозволяє зменшити кількість матеріалу, яке

криптоаналітик зможе використовувати для розкриття ключа.

Знищення ключів

Беручи до уваги, що ключі повинні регулярно змінюватися, старі ключі необхідно знищувати. З їх допомогою можна прочитати старі повідомлення, навіть якщо самі вони ніколи більше не використовуються.

Ключі повинні знищуватися надійно. Якщо ключ записаний в EEPROM-пам'ять, то для його знищення потрібно багаторазово перезаписати пам'ять. Якщо ключ записаний в пам'ять EPROM або PROM, то він повинен бути знищений разом з мікросхемою пам'яті.

Якщо ключ зберігається на жорсткому диску комп'ютера, дані відповідної ділянки накопичувача повинні бути багаторазово перезаписані. Серйозна проблема полягає в тому, що в комп'ютері ключі можуть бути розмножені і збережені в різних областях жорсткого диска. Будь-яка ОС, що реалізує будь-яку схему управління пам'яттю, постійно вивантажує програми з оперативної пам'яті на жорсткий диск і завантажує їх назад.

Для знищення ключів необхідно використовувати спеціальну програму, яка спочатку перевіряє наявність копій ключа на фізичному рівні, навіть в невикористовуваних блоках, а потім стирає їх.

Використання ключів

Виконання криптографічних програм під керуванням багатозадачних ОС пов'язано з певним ризиком. Неможливо сказати, коли операційна система зупинить працюючу програму, запише всі проміжні дані на жорсткий диск і передасть ресурси іншої задачі. Повертаючи управління перерваної програми і зчитуючи при цьому всю необхідну інформацію з жорсткого диска, операційна система не очищує пам'ять накопичувача.

Очевидно, що, якщо виконувалася програма шифрування, то крім самої програми на жорсткий диск будуть записані деякі дані, в тому числі і секретний ключ. Секретний ключ буде зберігатися на диску до тих пір, поки не буде виконано запис в цю ж область пам'яті.

У пріоритетному, багатозадачному середовищі для зниження ризику можна встановити високий пріоритет виконання завдання. Але навіть в цьому випадку надійність системи в цілому буде невисокою.

Апаратні реалізації надійніше. Багато з пристроїв шифрування розроблені так, що будь-яке втручання призводить до знищення ключа. Деякі пристрої, наприклад телефонні шифратори, використовують сеансові ключі. Сеансовим називається ключ, який використовується тільки для одного сеансу зв'язку - єдиної телефонної розмови - і потім знищується. Немає сенсу зберігати ключ після того, як він був використаний.

У деяких додатках необхідний контроль процесу використання сеансового ключа. *Сеансові ключі* можуть бути дозволені до використання тільки на певній машині або тільки в певний час.

Одна зі схем управління подібними обмеженнями передбачає додавання до ключа спеціального *вектора контролю*. Значення хеш-функції вектора контролю підсумовується по модулю два з головним ключем. Результат використовується як ключ для шифрування сеансового ключа. Отриманий шифртекст потім зберігається разом з вектором контролю. Для відновлення сеансового ключа потрібно обчислити значення хеш-функції вектора контролю і підсумувати результат з головним ключем. Отриманий ключ використовується для дешифрування і відновлення сеансового ключа.

Перевага цієї схеми полягає в тому, що довжина вектора контролю може бути довільною і що він завжди зберігається у відкритому вигляді разом із зашифрованим ключем. Такий спосіб не висуває вимог щодо захищеності апаратури і передбачає відсутність безпосереднього доступу користувачів до ключів.

Депонування ключів

У квітні 1993 року уряд США анонсувало новий метод криптографічного захисту інформації, що забезпечує високий рівень безпеки при передачі по відкритим каналам зв'язку, з

одного боку, і відповідає вимогам національної безпеки, з іншого.

Метод заснований на застосуванні шифрувальної/дешифрувальної мікросхеми Clipper, розробленої за спеціальною технологією TEMPEST, що перешкоджає зчитування інформації за допомогою зовнішнього впливу і процедури депонування ключа, визначальної дисципліни розкриття унікального ключа мікросхеми.

Генерація та запис ключа виконуються до вбудовування мікросхеми в кінцевий пристрій. Не існує способу, що дозволяє безпосередньо прочитати ключ як під час, так і після закінчення технологічного процесу виробництва і програмування мікросхеми.

Ключ розбивається на два компоненти, кожен з яких шифрується і потім передається на зберігання довіреним агентам депозитної служби, які представляють собою урядові організації, що забезпечують надійне зберігання ключових компонентів протягом терміну їх дії. Агенти видають ключові компоненти тільки тоді, коли відповідний запит підтверджений вирішенню федерального суду. Отримані компоненти дозволяють відновити унікальний ключ і виконати дешифрування.

Технологія поділу ключа заснована на математиці поділу секрету. Нехай є первинний секретний ключ і два учасники, що утворюють легальну коаліцію для отримання доступу до секретної інформації. Останнє означає, що жоден з учасників не знає первинного секретного ключа і тільки об'єднання вторинних ключів, які мають учасники, дозволяє відновити первинний ключ і виконати дешифрування.

Практична реалізація ідеї. Нехай первинний ключ k - це послідовність бітів довжини n. Тоді вторинний ключ першого учасника

$$k_1 = k \oplus \xi$$

де ξ - шумова послідовність двійкових символів довжини n. Та ж шумова послідовність використовується в якості вторинного ключа іншого учасника $k_2 = \xi$. Отже, первинний ключ може бути вирахуваний підсумовуванням за модулем 2 вторинних ключів учасників:

$$k = k_1 \oplus k_2$$

Схема має абсолютну секретність - оцінка трудомісткості розкриття первинного ключа кожним з учасників становить $2n$ спроб дешифрування, в гіршому випадку $2n$-1 - в середньому. Дана схема є окремим випадком класичного методу поділу секрету.

Інший довгостроковий проект уряду США, що діє на підставі закону про комп'ютерну безпеку від 1987 р. пов'язаний з розробкою мікросхеми Capstone. Підтримка проекту здійснюється національним інститутом стандартів і технології (NIST) і агентством національної безпеки (NSA).

У мікросхемі Capstone крім стандартних компонентів мікро-схеми Clipper реалізований алгоритм обміну відкритими ключами KEA (Key Exchange Algorithm), цифровий підпис DSA, хеш-функція SHA, алгоритм швидкого зведення в ступінь за модулем і генератор псевдовипадкових чисел на основі джерела чистого шуму. Всі криптоалгоритми мікросхеми Capstone мають 80-бітний секретний ключ. Основна область застосування мікросхеми Capstone - безпека електронних угод і платежів, а також ряд інших сфер в рамках національної інформаційної інфраструктури.

Спосіб застосування. Для захисту телефонних переговорів кожен з абонентів повинен мати спеціальний криптографічний пристрій, що містить Clipper, Capstone або будь-яку іншу аналогічну мікросхему. У пристрої має бути реалізований протокол, що дозволяє абонентам обмінюватися секретним сеансовим ключем, наприклад, за

допомогою відомого методу цифрового конверта. Даний метод застосовується в пристрої захисту телефонних переговорів TSD (Telephone Security Device), розроблений компанією AT & T. Пристрій підключається щільно один до одного між телефонною трубкою і основним блоком і активізується натисканням кнопки.

Приклади завдань

1. Пояснити, чому в реальних атаках “вкрасти ключ” часто дешевше, ніж

криптоаналізувати алгоритм, і навести 2 типові вразливості управління ключами в ІС.
2. Для заданого алфавіту ключів (26/36/62/95/128/256) оцінити потужність простору ключів для довжини 6–8 байтів і зробити висновок щодо стійкості до brute force при 10^6 ключів/с.
3. Відтворити схему генератора ANSI X9.17 та обчислити два послідовні значення R_i і V_{i+1} за формулами $R_i = E_k(E_k(T_i) \oplus V_i), V_{i+1} = E_k(E_k(T_i) \oplus R_i)$.
4. Спроєктувати процедуру розподілу сеансового ключа в мережі на 1000 абонентів і порівняти прямий обмін ключами з використанням ЦРК (за кількістю обмінів, ризиками та навантаженням).
5. Скласти політику життєвого циклу ключів для сервісу (генерація → розподіл → зберігання → оновлення → відкликання/знищення → архів для перевірки підписів) з урахуванням компрометації та CRL/часових міток.

Контрольні питання

1. Які три базові елементи процесу управління ключами виділяють у криптосистемах і чому кожен з них є критичним для безпеки?
2. Чому “випадковість” ключа і “якість” генератора (TRNG/PRNG) важливіші за складність алгоритму шифрування в практичній стійкості системи?
3. У чому полягає принцип генерації в ANSI X9.17 (роль V_0, мітки часу T, спеціального ключа k, подвійного шифрування та XOR)?
4. Які основні ризики та компроміси мають механізми розподілу ключів: прямий обмін, ЦРК, сертифікати ЦС, “web of trust”, та як проявляється атака “людина посередині”?
5. Які практичні правила визначення часу життя ключів (сеансові/даних/KEK/підпису), і чому безпечне знищення ключів є окремою проблемою в ОС з віртуальною пам’яттю?

Література

1. Menezes A. J., van Oorschot P. C., Vanstone S. A. Handbook of Applied Cryptography. Boca Raton : CRC Press, 1996.
2. Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2nd ed. New York : John Wiley & Sons, 1996.
3. Stallings W. Cryptography and Network Security: Principles and Practice. 7th ed. Boston : Pearson, 2017.
4. Krawczyk H., Bellare M., Canetti R. HMAC: Keyed-hashing for message authentication // RFC 2104. 1997.
5. National Institute of Standards and Technology. Recommendation for Key Management: Part 1 — General : NIST Special Publication 800-57. Gaithersburg, MD : NIST, 2020.

Лекція №14 Постквантова криптографія: загроза квантових обчислень та сучасні стандарти NIST PQC.

1. Квантові обчислення як загроза сучасній криптографії
Сучасна криптографія протягом десятиліть спиралася на припущення про обчислювальну складність певних математичних задач. Безпека алгоритмів RSA, Diffie–Hellman або еліптичних кривих базується не на їх абсолютній нерозв'язності, а на практичній неможливості ефективного обчислення на класичних комп'ютерах. Поява квантових обчислень радикально змінює цю парадигму.

Квантовий комп'ютер і принципи його роботи

Квантовий комп'ютер — це обчислювальна система, що використовує закони квантової механіки для обробки інформації. На відміну від класичного біта, який може перебувати лише в одному з двох станів (0 або 1), кубіт здатний перебувати у стані суперпозиції, тобто одночасно представляти комбінацію станів 0 і 1 з певними ймовірностями.

Ще більш фундаментальною властивістю є квантова заплутаність. Заплутані кубіти утворюють спільний квантовий стан, у якому вимірювання одного кубіта миттєво визначає стан іншого, незалежно від відстані між ними. Саме суперпозиція та заплутаність дозволяють квантовим алгоритмам обробляти величезну кількість станів паралельно, що забезпечує експоненційне прискорення для певних класів задач.

Важливо підкреслити: квантовий комп'ютер не є просто "швидшим процесором". Його обчислювальна модель принципово інша, і вона ефективна лише для специфічних задач, серед яких — ключові криптографічні проблеми.

Алгоритми Шора та Гровера як криптографічна загроза

Криптографічну значущість квантових обчислень визначають насамперед два алгоритми.

Алгоритм Шора дозволяє ефективно розкладати великі числа на прості множники та обчислювати дискретні логарифми за поліноміальний час. Для класичних комп'ютерів ці задачі вважаються практично нерозв'язними при достатньо великих параметрах, що і становить основу безпеки багатьох асиметричних криптосистем. Квантовий комп'ютер із достатньою кількістю стабільних кубітів робить ці припущення недійсними.

Алгоритм Гровера, на відміну від алгоритму Шора, не ламає криптосистеми "повністю", але

забезпечує квадратичне прискорення перебору. Це означає, що симетричні алгоритми та хеш-функції стають ефективно слабшими: наприклад, 128-бітний симетричний ключ у квантовій моделі дає рівень безпеки, близький до 64 біт у класичній.

Таким чином, квантові алгоритми впливають на криптографію по-різному: асиметричні схеми стають принципово зламаними, тоді як симетричні — послабленими, але не непридатними.

Вразливі криптосистеми відкритого ключа

Найбільш критичною є ситуація для криптосистем, що використовуються для:

- обміну ключами;
- цифрового підпису;
- побудови інфраструктури відкритих ключів (PKI).

До них належать RSA, DSA, Diffie–Hellman та всі алгоритми на основі еліптичних кривих (ECDSA, ECDH). Їхня безпека зводиться до факторизації або дискретного логарифмування — задач, які алгоритм Шора вирішує ефективно.

Це означає, що з появою масштабного квантового комп’ютера:

- TLS-з’єднання, захищені класичним обміном ключами, можуть бути розкриті;
- цифрові підписи втратять юридичну значущість;
- довгостроково збережені зашифровані дані стануть доступними для дешифрування.

Концепція “Harvest Now, Decrypt Later”

Особливу небезпеку становить стратегія, відома як Harvest Now, Decrypt Later (“збирай зараз — розшифровуй потім”). Її суть полягає в тому, що зашифровані сьогодні дані можуть бути перехоплені та збережені, навіть якщо їх неможливо розшифрувати негайно.

Коли в майбутньому з’являться достатньо потужні квантові комп’ютери, такі архіви можна буде розкрити заднім числом. Це критично для:

- державних та військових даних;
- медичної та персональної інформації;
- комерційних та наукових секретів з тривалим терміном цінності.

Таким чином, квантова загроза є не лише гіпотетичною проблемою майбутнього, а реальним фактором стратегічного планування криптографічного захисту вже сьогодні.

2. Основи та класифікація постквантової криптографії
У відповідь на потенційну втрату криптографічної стійкості класичних алгоритмів у квантову епоху формується новий напрям — постквантова криптографія (Post-Quantum Cryptography, PQC). Йдеться не про використання квантових фізичних каналів або квантових пристроїв, а про класичні криптографічні алгоритми, які можуть виконуватися на звичайних комп’ютерах, але залишаються стійкими навіть за наявності потужного квантового противника.

Поняття постквантової криптографії

Постквантова криптографія — це сукупність криптографічних методів і алгоритмів, безпека яких не ґрунтується на задачах факторизації або дискретного логарифмування і які вважаються стійкими як до класичних, так і до квантових атак. Ключова ідея PQC полягає у використанні математичних задач, для яких не відомі ефективні квантові алгоритми, аналогічні алгоритму Шора.

Важливо наголосити, що PQC не заперечує класичну криптографію, а еволюційно її

продовжує. Вона розглядається як практичний шлях міграції існуючих систем безпеки до нової загрозової моделі, не змінюючи радикально апаратну або програмну інфраструктуру.

Вимоги до постквантових криптографічних алгоритмів

На відміну від суто теоретичних криптосистем, алгоритми PQC мають відповідати широкому спектру практичних вимог. Криптостійкість залишається базовою умовою, але вона оцінюється одразу в двох моделях — класичній та квантовій. Алгоритм повинен зберігати запас безпеки навіть з урахуванням потенційних майбутніх криптоаналітичних проривів.

Не менш важливою є продуктивність. Багато постквантових схем оперують складнішими математичними структурами, що може призводити до зростання часу обчислень. Для практичного впровадження критичними є швидкість генерації ключів, обміну ключами та створення цифрових підписів.

Окрему проблему становлять розміри ключів, підписів і службових даних. На відміну від класичних алгоритмів, де відкриті ключі можуть мати сотні байтів, у PQC вони нерідко сягають кілобайтів або навіть десятків кілобайтів. Це створює додаткове навантаження на мережі, протоколи та вбудовані системи.

Нарешті, алгоритм має бути реалізовним і безпечним на практиці. Це означає стійкість до атак побічними каналами, відсутність складних умовних переходів, що залежать від секретних даних, а також можливість ефективної реалізації на різних апаратних платформах — від серверів до смарткарт і IoT-пристроїв.

Основні математичні напрями постквантової криптографії

Сучасна PQC не є єдиною криптосистемою, а радше набором різних підходів, кожен з яких спирається на власний клас складних задач.

Найбільш розвиненим і практично орієнтованим напрямом є ґраткова криптографія. Вона базується на задачах, пов’язаних із багатовимірними ґратками, таких як пошук найкоротшого або найближчого вектора. Ці задачі вважаються складними як для класичних, так і для квантових алгоритмів, а також добре масштабуються. Саме ґраткові схеми сьогодні розглядаються як основа для заміни класичних алгоритмів обміну ключами та цифрового підпису.

Інший підхід — кодово-орієнтована криптографія, що використовує задачі декодування випадкових лінійних кодів з помилками. Цей напрям має довгу історію та високий рівень довіри, оскільки деякі алгоритми існують уже десятиліттями без успішних атак. Водночас вони зазвичай характеризуються дуже великими відкритими ключами.

Хеш-орієнтована криптографія займає особливе місце, оскільки її безпека спирається майже виключно на криптографічні хеш-функції. Такі схеми найчастіше використовуються для цифрового підпису і відзначаються надзвичайно високою надійністю, хоча й мають обмеження щодо розміру підписів і швидкості.

Окрему, більш вузьку нішу займають алгоритми на основі ізогеній еліптичних кривих. Вони привабливі компактними ключами, але складність математичного апарату та криптоаналітичні прориви останніх років поставили під сумнів їхню практичну перспективність.

Ще одним напрямом є багатовимірні поліноміальні системи, де безпека базується на складності розв’язування систем нелінійних рівнянь над скінченними полями. Ці схеми потенційно дуже швидкі, але історично виявлялися вразливими до структурних атак, що

зумовлює обережне ставлення до них.

Порівняння класичної та постквантової криптографії

Принципова відмінність між класичною та постквантовою криптографією полягає у типі математичних припущень, на яких ґрунтується безпека. Класична криптографія використовує компактні ключі й добре оптимізовані алгоритми, але спирається на задачі, що є вразливими до квантових обчислень. Постквантова криптографія, навпаки, жертвує компактністю та простотою заради довгострокової стійкості.

У результаті PQC розглядається не як повна заміна класичних алгоритмів “одним махом”, а як поступовий перехід, що часто реалізується у вигляді гібридних схем, де класичні та постквантові алгоритми використовуються паралельно.

3. Процес стандартизації постквантових алгоритмів NIST
Усвідомлення квантової загрози для криптографії швидко вийшло за межі академічних дискусій і перетворилося на інженерну та інституційну проблему. Сучасні інформаційні системи мають життєвий цикл у десятки років, а криптографічні стандарти впроваджуються ще довше. Саме тому питання переходу до постквантових алгоритмів потребувало централізованого, відкритого та науково обґрунтованого підходу.

Причини ініціації конкурсу постквантової криптографії

Ініціація процесу стандартизації PQC була зумовлена не раптовим проривом у квантових технологіях, а стратегічним прогнозуванням ризиків. Навіть за відсутності практичного квантового комп’ютера, здатного зламувати RSA або ECC, сама можливість його появи в середньо- та довгостроковій перспективі ставила під загрозу довіру до чинних криптографічних стандартів.

Особливу роль відіграв феномен Harvest Now, Decrypt Later: інформація, що передається або зберігається сьогодні, може бути розкрита в майбутньому. Для державних структур, фінансового сектору та критичної інфраструктури це означає необхідність випереджального захисту, а не реакції постфактум.

Крім того, відсутність стандартів PQC створювала фрагментоване середовище, де різні організації експериментували з несумісними або слабо перевіреними алгоритмами. Це суперечило фундаментальному принципу криптографії: безпека має ґрунтуватися на відкритості та колективній експертизі.

Роль та підхід NIST

Координатором процесу стандартизації став NIST — інституція з багаторічним досвідом у формуванні криптографічних стандартів, включно з AES, SHA та сучасними рекомендаціями для криптографічних протоколів.

Підхід NIST принципово відрізнявся від закритих або корпоративних моделей стандартизації. Було обрано модель відкритого конкурсу, де будь-яка команда з усього світу могла подати власний криптографічний алгоритм. Усі специфікації, результати аналізу та зауваження публікувалися відкрито, що стимулювало незалежну криптоаналітичну перевірку.

Важливо, що NIST не намагався “винайти” нові алгоритми самостійно. Його роль полягала в організації процесу, формулюванні вимог, агрегації експертних оцінок та поступовому звуженні множини кандидатів до практично придатних стандартів.

Етапи конкурсу постквантових алгоритмів

Процес стандартизації PQC був побудований у вигляді кількох послідовних раундів, кожен з яких суттєво зменшував кількість кандидатів.

Перший раунд мав на меті максимальну різноманітність. Було подано десятки алгоритмів, що представляли всі основні напрями постквантової криптографії. На цьому етапі основна увага приділялася коректності специфікацій, відсутності очевидних уразливостей і відповідності базовим вимогам безпеки.

Другий раунд став етапом поглибленого криптоаналізу. Алгоритми активно досліджувалися на предмет структурних слабкостей, ефективності реалізації та поведінки в різних параметричних режимах. Частина кандидатів була відхилена через виявлені атаки або надмірну складність реалізації.

Третій раунд ознаменував перехід від “експериментальних” схем до реалістичних інженерних рішень. Тут почали чітко вимальовуватися фаворити, придатні для масового використання в протоколах обміну ключами та цифрового підпису.

Четвертий етап був зосереджений на фіналізації стандартів і доборі алгоритмів з різними властивостями, щоб уникнути залежності від одного математичного підходу. Таким чином, результатом процесу стала не одна схема, а набір алгоритмів, кожен з яких покриває певний клас задач і сценаріїв використання.

Критерії оцінювання постквантових алгоритмів

Оцінювання кандидатів у PQC виходило далеко за межі абстрактної криптостійкості. Безпека, безумовно, залишалася головним критерієм, але вона розглядалася з урахуванням як класичних, так і квантових моделей атак.

Велику увагу приділяли ефективності: швидкості виконання, вимогам до пам’яті, можливості оптимізації на сучасних процесорах. Для практичного впровадження важливими виявилися також розміри ключів, підписів і допоміжних структур, оскільки вони безпосередньо впливають на мережеві протоколи та вбудовані системи.

Окремим аспектом була простота та безпека реалізації. Алгоритми з надто складними умовними переходами або залежністю від точності обчислень з плаваючою комою вважалися менш бажаними через ризик помилок реалізації та атак побічними каналами.

Значення відкритої криптографічної експертизи

Однією з ключових особливостей процесу PQC стала масштабна відкрита експертиза. Алгоритми аналізувалися не лише авторами та експертами NIST, а й незалежними дослідниками з усього світу. Саме в ході цього процесу були виявлені як теоретичні вразливості, так і практичні проблеми реалізації окремих схем.

Цей підхід підтвердив фундаментальний принцип криптографії: довіра виникає не з секретності, а з перевірюваності. Стандарти PQC стали результатом багаторічної колективної роботи криптографічної спільноти, а не адміністративного рішення.

4. Обрані та фінальні алгоритми стандарту NIST PQC
Після кількох років відкритого конкурсу та інтенсивної криптоаналітичної перевірки процес стандартизації PQC перейшов від різноманіття теоретичних підходів до конкретних, інженерно придатних алгоритмів. Відібрані схеми мають забезпечити базові криптографічні

примітиви — обмін ключами та електронний підпис — у середовищі, стійкому до квантового противника.

Алгоритми обміну ключами (KEM): CRYSTALS-Kyber

CRYSTALS-Kyber став основним кандидатом для побудови механізмів узгодження спільного секрету. Його математична основа пов’язана з ґратковими задачами, зокрема з проблемами типу Learning With Errors (LWE) та її модифікаціями. Ідея полягає в тому, що навіть за наявності квантового комп’ютера відновлення секретного ключа з відкритих параметрів залишається обчислювально недосяжним.

Практичну цінність Kyber визначає поєднання кількох властивостей. Алгоритм демонструє високу швидкість як на програмному, так і на апаратному рівні, має відносно помірні розміри ключів і добре масштабується для різних рівнів безпеки. Саме ці характеристики зробили його придатним для використання в мережевих протоколах, зокрема в TLS та VPN.

Обмеження Kyber пов’язані передусім із загальною природою ґраткових схем: вони складніші за класичні алгоритми з точки зору математичного апарату і потребують уважної реалізації для уникнення побічних каналів.

Алгоритми електронного підпису: Dilithium, Falcon та SPHINCS+

Для цифрового підпису було обрано не одну, а кілька схем, що відображає різні компроміси між продуктивністю, компактністю та рівнем довіри.

CRYSTALS-Dilithium, як і Kyber, базується на ґраткових задачах. Його ключовою перевагою є простота реалізації. Алгоритм не потребує обчислень з плаваючою комою і добре пристосований до захисту від атак побічними каналами. Це робить Dilithium привабливим варіантом для широкого впровадження, включно з вбудованими системами та апаратними модулями безпеки.

Falcon також є ґратковим алгоритмом, але орієнтованим на мінімізацію розміру підпису. Він дозволяє отримати значно компактніші підписи порівняно з Dilithium, що важливо для мереж з обмеженою пропускною здатністю. Водночас Falcon складніший у реалізації, оскільки використовує точні обчислення з плаваючою комою, що підвищує ризик помилок і атак на реалізацію.

SPHINCS+ представляє зовсім інший підхід. Це хеш-орієнтована схема підпису, безпека якої зводиться до стійкості криптографічних хеш-функцій. Її головна перевага — мінімальні криптографічні припущення і, як наслідок, дуже високий рівень довіри. Проте за це доводиться платити великими розмірами підписів і відносно низькою швидкістю.

Порівняльна характеристика фінальних алгоритмів

Загальна логіка вибору фінальних схем полягає у створенні збалансованого набору інструментів, а не універсального алгоритму “на всі випадки”. Kyber орієнтований на масовий обмін ключами, Dilithium — на надійний і відносно простий цифровий підпис, Falcon — на компактність, а SPHINCS+ — на максимальну консервативність з точки зору криптографічних припущень.

Такий підхід зменшує системні ризики: навіть якщо в майбутньому з’являться нові теоретичні атаки на певний клас задач, не вся постквантова інфраструктура опиниться під загрозою одночасно.

Причини відмови від окремих кандидатів

У процесі конкурсу було відхилено чимало перспективних алгоритмів. Найпоказовішим прикладом є схеми на основі ізогеній еліптичних кривих. Вони приваблювали дуже малими розмірами ключів і підписів, але виявилися надто крихкими з криптоаналітичної точки зору. За відносно короткий час було продемонстровано атаки, що радикально знижували їхню безпеку.

В інших випадках причинами відмови ставали надмірна складність реалізації, нестабільна продуктивність або невигідні параметричні компроміси. Це ще раз підкреслює, що в сучасній криптографії стандарт — це не лише математична ідея, а й практична інженерна якість.

5. Практичні аспекти впровадження постквантової криптографії
Попри те, що постквантова криптографія вже має стандартизовані алгоритми, її впровадження не є простим технічним оновленням. Йдеться про глибоку трансформацію криптографічної інфраструктури, яка охоплює протоколи, програмні реалізації, апаратні платформи та організаційні процеси. Саме тому практичні аспекти PQC відіграють не меншу роль, ніж її математичні основи.

Гібридні криптографічні схеми

Домінуючим підходом на етапі переходу стали гібридні криптографічні схеми, у яких класичні алгоритми використовуються паралельно з постквантовими. Типовим прикладом є гібридний обмін ключами, де спільний секрет формується одночасно з використанням, наприклад, класичного Diffie–Hellman та постквантового KEM.

Такий підхід забезпечує захист у двох загрозових моделях. Якщо квантові атаки виявляться практично нереалізовними або відкладеними на десятиліття, система зберігатиме класичну безпеку. Якщо ж квантовий комп’ютер стане реальністю, постквантова складова продовжить забезпечувати конфіденційність. Гібридні схеми дозволяють здійснювати міграцію без різкого розриву з існуючими стандартами та протоколами.

Застосування PQC у TLS, VPN, PKI та цифрових підписах

Найбільш очевидною сферою впровадження PQC є протоколи захищеного зв’язку. У TLS постквантові алгоритми насамперед замінюють або доповнюють механізми узгодження ключів. Це критично важливо, оскільки саме на цьому етапі виникає загроза Harvest Now, Decrypt Later.

У VPN-системах перехід до PQC ускладнюється різноманіттям реалізацій і жорсткими вимогами до продуктивності. Проте саме VPN часто використовується для захисту довготривалих з’єднань, що робить їх привабливою ціллю для відкладеного дешифрування.

Інфраструктура відкритих ключів (PKI) є ще складнішою для модернізації. Сертифікати з постквантовими підписами мають значно більший розмір, що впливає на зберігання, передавання та перевірку ланцюжків довіри. Перехід до PQC у PKI, ймовірно, буде тривалим і потребуватиме поетапного впровадження гібридних сертифікатів.

Проблеми продуктивності та сумісності

Одним з основних практичних викликів PQC є зростання обчислювальних витрат. Хоча сучасні постквантові алгоритми оптимізовані для програмної реалізації, вони все ж часто повільніші за класичні аналоги, особливо на обмежених платформах.

Сумісність із чинними протоколами та стандартами також становить проблему. Багато

мережевих протоколів були розроблені з урахуванням компактних ключів і підписів, і не завжди готові до роботи з багатокілобайтними структурами. Це вимагає змін у форматах повідомлень, таймінгах і механізмах фрагментації даних.

Вплив розміру ключів і підписів

Розміри криптографічних параметрів у PQC мають прямий вплив на системи з обмеженими ресурсами. Збільшення ключів і підписів означає:

- більший обсяг передаваних даних;
- підвищене навантаження на пам’ять;
- довший час встановлення з’єднання.

Для серверних систем ці витрати зазвичай прийнятні, але для IoT-пристроїв, смарткарт або вбудованих контролерів вони можуть бути критичними. Саме тому в практичних реалізаціях часто доводиться обирати компроміс між рівнем безпеки та ресурсними обмеженнями.

Побічні канали та реалізаційна безпека

Навіть математично стійкий алгоритм може бути скомпрометований через помилки реалізації. Для постквантової криптографії це питання особливо актуальне, оскільки багато схем використовують складні обчислення з великими масивами даних.

Атаки побічними каналами — за часом виконання, споживанням енергії або електромагнітним випромінюванням — залишаються реальною загрозою. Тому практичне впровадження PQC вимагає алгоритмів із передбачуваною поведінкою та реалізацій, що не залежать від секретних даних у своїх контрольних потоках.

6. Перспективи розвитку та виклики постквантової криптографії
Завершуючи розгляд постквантової криптографії, доцільно перейти від поточного стану стандартів і практичних реалізацій до довгострокових перспектив. PQC не є статичною технологією: вона розвивається паралельно з квантовими обчисленнями, криптоаналізом і вимогами до інформаційної безпеки в глобальному масштабі.

Реалістичні терміни появи масштабних квантових комп’ютерів

Оцінка строків появи квантових комп’ютерів, здатних ламати сучасні криптосистеми, залишається предметом дискусій. Існуючі квантові пристрої вже демонструють тисячі фізичних кубітів, однак для практичної криптоатаки необхідні мільйони логічних кубітів із корекцією помилок і стабільною роботою.

Більшість експертних прогнозів сходяться на тому, що такі системи не з’являться миттєво, але їх поява в горизонті 10–20 років є цілком реалістичною. З точки зору криптографії це означає, що чекати “фактичного зламу” — запізніла стратегія. Криптографічні рішення мають прийматися з урахуванням повного життєвого циклу даних, а не лише поточних технічних можливостей.

Міграція криптографічної інфраструктури

Одним з найсерйозніших викликів є масштабна міграція існуючої криптографічної інфраструктури. Сучасні інформаційні системи — від веб-сервісів до промислових мереж — глибоко інтегровані з класичними алгоритмами, стандартами та сертифікаційними центрами.

Перехід до постквантової криптографії неминуче буде поетапним. На практиці це означає тривалий період співіснування класичних, гібридних і повністю постквантових рішень.

Особливу складність становлять системи з довгим терміном експлуатації, такі як промислові контролери, вбудовані пристрої та критична інфраструктура, де оновлення програмного забезпечення є обмеженим або дорогим.

Постквантова криптографія та квантова криптографія (QKD)

Важливо чітко розмежовувати постквантову криптографію та квантову криптографію, зокрема квантовий розподіл ключів (QKD). PQC є програмно-орієнтованим підходом, який може бути реалізований на класичних обчислювальних платформах і інтегрований у чинні протоколи.

QKD, натомість, використовує фізичні властивості квантових систем для передавання ключів і теоретично забезпечує інформаційну стійкість. Проте такі системи потребують спеціалізованого обладнання, мають обмеження щодо дальності та швидкості і складні в масштабуванні.

У практичному вимірі ці підходи не є взаємовиключними. PQC розглядається як масове, універсальне рішення, тоді як QKD може застосовуватися в нішевих сценаріях з надвисокими вимогами до безпеки та контрольованою інфраструктурою.

Відкриті наукові проблеми та напрями майбутніх досліджень

Попри досягнутий прогрес, постквантова криптографія залишається активною науковою галуззю. Тривають дослідження щодо зменшення розмірів ключів і підписів, підвищення продуктивності та покращення захисту від атак побічними каналами.

Окрему увагу приділяють довгостроковій надійності криптографічних припущень. Історія криптографії показує, що деякі математичні задачі, які вважалися складними, з часом втрачали свою стійкість. Тому диверсифікація алгоритмів і постійний перегляд стандартів залишаються критично важливими.

У підсумку постквантова криптографія постає не як разове оновлення, а як безперервний процес адаптації до розвитку обчислювальних технологій. Саме ця динаміка визначатиме криптографічну безпеку інформаційних систем у найближчі десятиліття.

Приклади завдань

4. Проаналізувати, які саме криптографічні примітиви у типовому TLS-з’єднанні стають вразливими в разі появи масштабного квантового комп’ютера.
5. Порівняти ґратковий та хеш-орієнтований підходи в постквантовій криптографії з точки зору довгострокової надійності.
6. Обґрунтувати доцільність використання гібридних криптографічних схем на етапі переходу до PQC.
7. Оцінити вплив збільшених розмірів ключів і підписів PQC на продуктивність мережевих протоколів.
8. Навести приклади можливих атак побічними каналами на реалізації постквантових алгоритмів.

Контрольні питання

1. У чому полягає принципова відмінність постквантової криптографії від квантової криптографії?
2. Чому алгоритм Шора є критичною загрозою для асиметричних криптосистем з відкритим ключем?
3. Які критерії використовував NIST для відбору фінальних алгоритмів постквантової криптографії?
4. Чим зумовлена необхідність використання декількох алгоритмів цифрового підпису в

стандарті NIST PQC?

5. У чому полягає стратегічна небезпека підходу Harvest Now, Decrypt Later?

Література

1. Bernstein D. J., Buchmann J., Dahmen E. Post-Quantum Cryptography. Berlin : Springer, 2009.
2. National Institute of Standards and Technology. Post-Quantum Cryptography Standardization. NISTIR 8413. Gaithersburg, 2023.
3. Mosca M. Cybersecurity in an Era with Quantum Computers: Will We Be Ready? // IEEE Security & Privacy. 2018. Vol. 16, No. 5. P. 38–41.
4. Chen L., Jordan S., Liu Y.-K. et al. Report on Post-Quantum Cryptography. NISTIR 8105. Gaithersburg, 2016.
5. Bindel N., Brendel J., Fischlin M., Goncalves B. Hybrid Key Encapsulation Mechanisms and Their Security // Journal of Cryptology. 2021. Vol. 34. P. 1–52.

Лекція №15 Криптографія в сучасних мережах та системах: TLS 1.3, QUIC, VPN, Zero Trust, безпека IoT.

1. Криптографія як основа сучасних мережевих протоколів

У сучасних мережах “безпека” майже ніколи не означає лише заборону доступу на межі організації. Дані постійно рухаються між браузером і сервером, між мікросервісами всередині датацентру, між мобільним застосунком і API, між IoT-пристроєм і хмарою. У цьому середовищі криптографія перестає бути “додатковим шаром” і стає основним механізмом довіри: саме вона визначає, чи можна вважати з’єднання захищеним, а отримані дані — достовірними.

Конфіденційність, цілісність, автентичність як три опори з’єднання

Коли говорять, що мережевий протокол “захищений”, зазвичай мають на увазі три властивості.

Конфіденційність означає, що сторонній спостерігач не може прочитати вміст трафіку. На практиці це реалізується симетричним шифруванням: після узгодження ключа сторони шифрують дані швидкими алгоритмами (наприклад, AES-GCM або ChaCha20-Poly1305). Критично, що конфіденційність стосується не лише “секретів” на кшталт паролів: навіть звичайні запити можуть розкривати приватні дані через контекст, метадані або шаблони поведінки.

Цілісність відповідає за те, щоб дані не могли бути непомітно змінені під час передачі. У мережі противник може не лише “підслуховувати”, а й активно втручатися: підмінювати пакети, вставляти власні повідомлення, повторювати старі фрагменти трафіку. Тому сучасні протоколи використовують автентифіковане шифрування (AEAD), де перевірка цілісності вбудована в сам механізм шифрування: якщо байт змінено — перевірка тегу не пройде, і повідомлення буде відкинуте.

Автентичність означає, що сторона з’єднання є саме тією, за кого себе видає. Тут важливо відрізняти два рівні. Перший — автентичність сервера (клієнт впевнений, що підключився до правильного домену/сервісу). Другий — автентичність клієнта (сервер впевнений, хто саме під’єднався). У вебі історично переважає модель “сервер завжди автентичний через сертифікат, клієнт — через пароль/токен”, але в корпоративних і промислових мережах дедалі частіше потрібна взаємна автентифікація: сертифікати або ключі є у двох сторін.

Ці три властивості не існують окремо: їх поєднання і є тим, що називають криптографічно захищеним каналом. Власне, TLS 1.3 або QUIC — це насамперед стандартизований спосіб створення такого каналу в умовах агресивної мережі.

Від периметра до захищених каналів

Традиційна модель мережевої безпеки довгий час будувалася навколо ідеї периметра: існує “внутрішня” мережа, якій довіряють, і “зовнішня”, якій не довіряють. Захист забезпечується межовими пристроями — фаєрволами, проксі, шлюзами. Ця логіка частково працювала, поки системи були монолітними, користувачі — в офісі, а сервери — у власному датацентрі.

Сучасні інфраструктури руйнують периметр з кількох причин. По-перше, бізнес-системи виходять у хмару, а частина сервісів стає зовнішньою за визначенням. По-друге, з’являються

мікросервіси: внутрішній трафік між компонентами за обсягом часто перевищує зовнішній, і “внутрішнє” вже не означає “безпечне”. По-третє, люди працюють віддалено, а пристрої підключаються з непередбачуваних мереж. Нарешті, IoT додає величезну кількість слабких вузлів, які фізично доступні противнику і часто мають мінімальний захист.

У результаті центр ваги змістився: замість того, щоб намагатися “утримати ворога за периметром”, системи проектують так, ніби мережа завжди недовірена. Тому захищений канал стає базовою одиницею безпеки: довіра переноситься з географії мережі на криптографічні факти — ключі, сертифікати, перевірені властивості протоколу.

Це добре видно на прикладі еволюції TLS. Старі підходи намагалися “накласти” шифрування поверх існуючого транспорту й зберегти сумісність будь-якою ціною. Нові підходи (TLS 1.3, QUIC) максимально зменшують поверхню атаки: прибирають застарілі алгоритми, роблять безпеку “за замовчуванням”, скорочують кількість небезпечних варіантів узгодження параметрів.

Як криптографічні протоколи пов’язані з архітектурою мереж

Криптографічний протокол — це не абстрактний “алгоритм шифрування”, а контракт між компонентами архітектури. Те, як побудована мережа, визначає, де саме і як використовуються ключі, хто кому довіряє і які наслідки має компрометація.

Якщо архітектура має централізований шлюз (наприклад, один API Gateway), то саме він стає точкою завершення TLS і місцем, де концентруються приватні ключі, політики, журнали. Це спрощує керування, але робить шлюз критичною точкою ризику. Якщо ж архітектура сервісна (service-to-service), то TLS потрібно не лише “на вході”, а між усіма сервісами. Тоді на перший план виходить mTLS, автоматична видача/ротація сертифікатів, сервісні меші, контроль ідентичностей.

У моделі Zero Trust логіка ще жорсткіша: мережеве розташування майже не має значення, а доступ дозволяється тільки після криптографічно підтвердженої ідентичності та контекстної перевірки. Тут криптографія вже не просто “шифрує трафік”, а стає частиною механізму авторизації, аудиту і керування довірою.

Для IoT-застосунків взаємозв’язок ще відчутніший. Обмежені ресурси пристроїв диктують вибір алгоритмів і режимів; обмеженість оновлень диктує вимоги до довгоживучих ключів і безпечного бутстрепінгу; фізичний доступ до пристрою робить критичними апаратні корені довіри та захист ключового матеріалу. Тобто архітектура мережі та життєвий цикл пристрою прямо визначають, чи буде криптографія реально працювати, а не лише “бути в документації”.

У підсумку, криптографія в мережах — це поєднання трьох рівнів: математичних примітивів, протокольних конструкцій і архітектурних рішень. Слабкість будь-якого рівня руйнує безпеку в цілому: можна мати сильні алгоритми, але неправильну модель довіри; можна мати правильну модель, але небезпечну реалізацію; можна мати чудову реалізацію, але застарілий протокол.

Далі в розділі 2 логічно перейти до TLS 1.3 і розглянути, як саме сучасний протокол формалізує ці властивості: яким чином відбувається узгодження ключів, чому Forward Secrecy стало стандартом, і що саме було спрощено порівняно з попередніми версіями TLS.

2. Протокол TLS 1.3: сучасна модель захищеного з’єднання

Протокол TLS 1.3 є не просто черговим оновленням попередніх версій TLS, а переглянутою моделлю побудови захищеного мережевого з’єднання. Його розробка стала відповіддю на накопичений досвід атак, помилок реалізації та надмірної складності TLS 1.0–1.2. У TLS 1.3

безпека розглядається як властивість “за замовчуванням”, а не як опціональний набір налаштувань.

Архітектура та принципи роботи TLS 1.3

В основі TLS 1.3 лежить ідея максимально раннього створення захищеного каналу. Якщо в попередніх версіях TLS значна частина початкового обміну відбувалася у відкритому вигляді, то в TLS 1.3 шифрування застосовується вже на ранніх етапах встановлення з’єднання.

Архітектурно протокол чітко розділяє:
- фазу узгодження криптографічних параметрів;
- фазу захищеного обміну прикладними даними.

При цьому кількість варіантів поведінки протоколу була радикально зменшена. Багато застарілих механізмів — окремі режими шифрування, складні схеми повторного узгодження, нестабільні розширення — були вилучені. Це зменшило поверхню атаки і зробило протокол значно більш передбачуваним з точки зору безпеки.

Обмін ключами та криптографічні примітиви

Ключовою відмінністю TLS 1.3 є те, що всі допустимі схеми обміну ключами забезпечують Forward Secrecy. Класичні механізми на основі статичних RSA-ключів повністю вилучені. Узгодження спільного секрету базується на тимчасових ключах Diffie–Hellman або еліптичних кривих.

Після встановлення спільного секрету він використовується для виведення набору сеансових ключів за допомогою криптографічних хеш-функцій і механізмів розгортання ключів. Симетричне шифрування здійснюється виключно в режимах автентифікованого шифрування, що поєднують конфіденційність і контроль цілісності в одному механізмі.

Таким чином, у TLS 1.3 зникає можливість некоректних або небезпечних комбінацій алгоритмів: клієнт і сервер домовляються лише про сучасні, перевірені криптографічні примітиви.

Forward Secrecy та захист від атак

Forward Secrecy є фундаментальною властивістю TLS 1.3. Вона гарантує, що компрометація довготривалих ключів сервера в майбутньому не дозволить розшифрувати трафік, перехоплений у минулому. Це критично важливо в контексті масового збору зашифрованих даних і концепції відкладеного дешифрування.

З точки зору загрозової моделі TLS 1.3 орієнтований як на пасивного противника, що лише спостерігає трафік, так і на активного, здатного підмінювати повідомлення, переривати з’єднання або ініціювати атаки типу “людина посередині”. Протокол побудований так, щоб будь-яке втручання у криптографічно значущі дані призводило до негайного виявлення та розриву з’єднання.

Особливу увагу приділено усуненню відомих класів атак на попередні версії TLS, пов’язаних із downgrade-атаками, повторним узгодженням параметрів або некоректною обробкою помилок.

Скорочення латентності та спрощення протоколу

Окрім підвищення безпеки, TLS 1.3 істотно оптимізує продуктивність мережевих з’єднань.

Класичний процес встановлення TLS-з’єднання був відносно “важким” з точки зору кількості раундів обміну повідомленнями. У TLS 1.3 стандартним став режим з одним мережевим раундом для повного встановлення захищеного каналу.

Для повторних з’єднань можливе ще більше скорочення затримки за рахунок механізмів відновлення сеансу, при яких клієнт може надсилати зашифровані дані практично одразу. Важливо, що ці оптимізації реалізовані без компромісів щодо криптографічної стійкості.

Узагальнюючи, TLS 1.3 демонструє сучасний підхід до мережевої безпеки: менше варіантів — менше помилок, швидше встановлення з’єднання — безпечніші канали за замовчуванням. Саме ці принципи згодом були перенесені й на протоколи нового покоління, зокрема QUIC, який інтегрує криптографію безпосередньо в транспортний рівень.

У розділі 3 доцільно розглянути, як ці ідеї реалізуються в QUIC і чому класична модель “TCP + TLS” поступається місцем новим транспортним протоколам.

3. QUIC та захищені транспортні протоколи нового покоління
Поява QUIC стала наслідком того, що класична модель мережевої взаємодії TCP + TLS почала обмежувати розвиток сучасних мережевих сервісів. Веб-застосунки, потокове відео, мобільні клієнти та хмарні сервіси вимагають не лише криптографічної безпеки, а й мінімальної затримки, стійкості до нестабільних мереж і швидкого відновлення з’єднань. QUIC було спроєктовано як відповідь саме на ці вимоги.

Причини появи QUIC та відмінності від TCP + TLS

TCP історично відповідає за надійність доставки, а TLS — за криптографічний захист. Такий поділ був логічним у часи відносно стабільних мереж, але згодом виявив низку системних проблем. Установлення з’єднання вимагало кількох послідовних етапів: спочатку TCP-handshake, потім TLS-handshake. Кожен з них додавав затримку, що особливо помітно в мобільних або географічно розподілених мережах.

Ще однією проблемою стала жорстка прив’язка TCP-з’єднання до пари IP-адрес і портів. Зміна мережі — наприклад, перехід мобільного пристрою з Wi-Fi на LTE — часто призводила до повного розриву з’єднання і необхідності повторної ініціалізації всіх протокольних рівнів.

QUIC вирішує ці проблеми шляхом переосмислення транспортного рівня. Він побудований поверх UDP, але реалізує власні механізми надійності, керування потоками та відновлення втрат. Криптографія при цьому не є “надбудовою”, а інтегрована безпосередньо в сам протокол.

Інтеграція криптографії в транспортний рівень

Принципова особливість QUIC полягає в тому, що захист є невід’ємною частиною транспорту. На відміну від TCP, де дані передаються у відкритому вигляді до моменту завершення TLS-handshake, у QUIC майже весь обмін після першого пакета вже зашифрований.

Криптографічна логіка QUIC значною мірою базується на ідеях TLS 1.3: використовується сучасний обмін ключами з Forward Secrecy, автентифіковане шифрування і мінімальний набір допустимих алгоритмів. Однак на відміну від класичного TLS, ці механізми “вбудовані” в транспортний протокол і тісно пов’язані з його станами та ідентифікаторами з’єднань.

Це дає змогу швидше встановлювати захищене з’єднання, а також зменшує кількість помилкових або небезпечних варіантів взаємодії між рівнями.

Шифрування метаданих і зменшення атак на мережевий рівень

Однією з важливих переваг QUIC є зменшення обсягу відкритих метаданих. У класичному TCP значна частина інформації — номери послідовностей, підтвердження, керування потоками — передається у відкритому вигляді. Це дозволяє мережевому противнику не лише аналізувати трафік, а й активно втручатися, наприклад, шляхом підміни або ін'єкції пакетів.

У QUIC більшість службової інформації зашифрована. Це ускладнює як пасивний аналіз трафіку, так і активні атаки на транспортному рівні. Протокол також краще протидіє маніпуляціям із затримками та втратами пакетів, оскільки логіка відновлення і повторної передачі тісно пов'язана з криптографічно захищеним контекстом з'єднання.

Водночас такий підхід змінює традиційну роль мережевих пристроїв: фаєрволи та системи аналізу трафіку бачать менше інформації, що вимагає нових підходів до моніторингу та безпеки мереж.

Практичні сценарії використання QUIC

Найбільш відомим прикладом застосування QUIC є HTTP/3, де протокол використовується як транспортна основа для сучасного вебу. У цьому сценарії поєднання швидкого встановлення з'єднання, мультиплексування потоків і криптографічного захисту безпосередньо на транспортному рівні дає відчутний приріст продуктивності та надійності.

QUIC також добре підходить для мобільних застосунків, де з'єднання часто змінюють мережеві умови. Механізми ідентифікації з'єднання дозволяють зберігати логічний сеанс навіть при зміні IP-адреси, не жертвуючи безпекою.

У ширшому контексті QUIC демонструє загальну тенденцію розвитку мережевих протоколів: криптографія перестає бути окремим шаром і стає фундаментальною властивістю транспорту. Цей підхід впливає і на VPN-рішення, і на архітектури Zero Trust, де захищений канал є не винятком, а нормою.

У розділі 4 логічно перейти до VPN та розглянути, як класичні та сучасні тунельні протоколи використовують криптографію для захисту мережевого трафіку в різних архітектурних моделях.

4. VPN та криптографія захищених тунелів

Віртуальні приватні мережі (VPN) тривалий час залишалися основним інструментом захисту мережевого трафіку поза межами локальної інфраструктури. З криптографічної точки зору VPN — це механізм створення захищеного тунелю поверх недовіреної мережі, який дозволяє передавати дані так, ніби вузли перебувають у спільному приватному середовищі.

Класичні моделі VPN та їх криптографічна логіка

Традиційна модель VPN ґрунтується на ідеї логічного розширення внутрішньої мережі. Клієнт, під'єднавшись до VPN-шлюзу, отримує доступ до ресурсів так, ніби він фізично знаходиться "всередині" мережі організації. Криптографія в цій моделі виконує дві ключові функції: захищає канал зв'язку та автентифікує учасників тунелю.

У класичних VPN-схемах зазвичай поєднуються асиметричні алгоритми для початкової автентифікації й обміну ключами та симетричне шифрування для передавання основного трафіку. Після встановлення тунелю весь трафік між клієнтом і шлюзом шифрується незалежно від типу прикладного протоколу, що спрощує використання, але водночас створює значну концентрацію довіри в одній точці.

IPSec, OpenVPN і WireGuard: еволюція підходів

IPSec є одним із найстаріших і найформалізованіших підходів до побудови VPN. Він працює на мережевому рівні й інтегрується безпосередньо в стек протоколів. Його криптографічна модель гнучка, але складна: велика кількість режимів, алгоритмів і параметрів робить конфігурацію потенційно вразливою до помилок. На практиці безпека IPSec значною мірою залежить від коректності налаштувань.

OpenVPN став популярним завдяки простішій моделі та використанню перевірених механізмів TLS. Криптографія тут більш прозора: автентифікація, обмін ключами й захист даних реалізуються за логікою захищеного каналу, схожого на HTTPS. Це спрощує впровадження й аудит, але водночас додає накладні витрати через використання користувацького простору й складного програмного стека.

WireGuard представляє сучасний мінімалістичний підхід. Його дизайн орієнтований на зменшення поверхні атаки: фіксований набір криптографічних примітивів, відсутність складних переговорів параметрів і чітка модель ключів. Криптографія в WireGuard тісно інтегрована з протоколом, що забезпечує високу продуктивність і передбачувану безпеку. Водночас така жорстка структура означає меншу гнучкість порівняно з класичними VPN-рішеннями.

Управління ключами та автентифікація

Незалежно від конкретного протоколу, найбільш критичним аспектом VPN є управління ключами. Саме воно визначає, хто має доступ до мережі, як швидко можна відкликати компрометований обліковий запис і наскільки безпечно відбувається початкове встановлення тунелю.

У традиційних VPN часто використовуються довготривалі ключі або сертифікати, що зберігаються на клієнтських пристроях. Це зручно, але створює ризики: втрата пристрою або компрометація ключа фактично означає втрату контролю над доступом. Сучасні підходи дедалі частіше поєднують криптографічну автентифікацію з додатковими факторами — апаратними токенами, короткоживучими сертифікатами або централізованими системами ідентифікації.

Обмеження традиційних VPN-рішень

Попри свою поширеність, VPN мають концептуальні обмеження, які стають дедалі очевиднішими. Класичний VPN часто надає надто широкий доступ: після підключення користувач фактично опиняється “всередині” мережі, що суперечить принципу мінімальних привілеїв.

Крім того, VPN концентрує трафік і довіру в центральному шлюзі, який стає критичною точкою відмови й привабливою ціллю для атак. У масштабних або динамічних середовищах така модель погано масштабується й ускладнює контроль доступу на рівні окремих сервісів.

Саме ці обмеження зумовили перехід від VPN як універсального рішення до більш гнучких архітектур, де захищений канал поєднується з контекстною автентифікацією та деталізованим контролем доступу. Цей підхід лежить в основі Zero Trust Architecture, до розгляду якої логічно перейти в наступному розділі.

5. Zero Trust Architecture та криптографія
Концепція Zero Trust Architecture (ZTA) є логічним продовженням еволюції мережевої безпеки в умовах, коли класичний периметр втратив практичний сенс. У середовищах із хмарними

сервісами, мікросервісною архітектурою, віддаленою роботою та IoT припущення “всередині — безпечно, ззовні — небезпечно” більше не працює. Zero Trust виходить з протилежної позиції: жоден елемент системи не вважається надійним за замовчуванням, незалежно від його розташування в мережі.

Відмова від периметрової моделі

У традиційній мережевій архітектурі головним об’єктом захисту був периметр. Якщо користувач або пристрій проходив автентифікацію на вході (VPN, корпоративна мережа), то подальший доступ до внутрішніх ресурсів часто вважався допустимим. Це призводило до ситуацій, коли компрометація одного вузла відкривала шлях до всієї мережі.

Zero Trust принципово змінює цю логіку. Доступ більше не залежить від “місця в мережі”, а визначається контекстом ідентичності, станом пристрою, політиками доступу та поточними умовами. Кожен запит до ресурсу розглядається окремо, а довіра не накопичується з часом. У такій моделі мережа сама по собі вважається недовіреним середовищем, а безпека переноситься на рівень протоколів і криптографічних механізмів.

Криптографічна автентифікація та авторизація

У Zero Trust криптографія відіграє центральну роль у встановленні довіри. Автентифікація перестає бути одноразовою подією під час “входу в мережу” і стає постійним процесом перевірки ідентичності. Це означає, що кожна сторона взаємодії — користувач, сервіс або пристрій — повинна криптографічно підтверджувати, ким вона є.

Авторизація при цьому тісно пов’язана з автентифікацією. Рішення про доступ приймається не лише на основі ролі або облікового запису, а з урахуванням контексту: типу ресурсу, часу доступу, характеристик пристрою, попередньої поведінки. Криптографічні механізми забезпечують достовірність цих атрибутів і захищають їх від підробки.

Криптографія в системах ідентифікації та доступу

У Zero Trust системи керування ідентичностями стають ядром безпеки. Саме тут криптографія використовується для захисту облікових даних, токенів доступу та сертифікатів. Замість статичних паролів дедалі ширше застосовуються асиметричні ключі, короткоживучі криптографічні токени та механізми взаємної автентифікації.

Особливо важливим є перехід від “користувач → мережа” до моделі “ідентичність → сервіс”. У мікросервісних середовищах сервіси автентифікують один одного за допомогою криптографічних доказів, а не через мережеві адреси чи сегментацію. Це зменшує наслідки компрометації окремих компонентів і дозволяє будувати гнучкі політики доступу.

Захищені канали як фундамент Zero Trust

У Zero Trust захищений канал перестає бути допоміжним механізмом і стає базовою умовою будь-якої взаємодії. Кожен запит, незалежно від того, відбувається він між користувачем і сервером чи між двома внутрішніми сервісами, має передаватися через криптографічно захищене з’єднання.

Це означає повсюдне використання TLS або його похідних, часто з взаємною автентифікацією. Захищений канал не лише шифрує дані, а й слугує носієм ідентичності: саме в процесі встановлення з’єднання сторони доводять свою справжність і отримують право на взаємодію.

Таким чином, Zero Trust Architecture не існує без криптографії. Вона спирається на неї як на

універсальний механізм довіри, що замінює географічні межі мережі формальними, перевірюваними криптографічними фактами.

Інтернет речей радикально розширює межі мережевих систем, але водночас створює одну з найскладніших загрозових моделей для криптографії. IoT-пристрої масово розгортаються в неконтрольованих середовищах, мають тривалий життєвий цикл і часто виконують критичні функції. За цих умов криптографічний захист перестає бути суто програмною проблемою і тісно переплітається з апаратними, експлуатаційними та організаційними аспектами.

6. Криптографічні виклики безпеки IoT

Особливості IoT-середовищ з криптографічної точки зору

На відміну від серверів або персональних комп'ютерів, IoT-пристрої зазвичай вважаються потенційно скомпрометованими фізично. Зловмисник може мати прямий доступ до пристрою, зчитувати пам'ять, аналізувати електричні сигнали або модифікувати прошивку. Це суттєво змінює модель загроз: криптографія має протистояти не лише мережевим атакам, а й локальному аналізу.

Крім того, IoT-середовище є надзвичайно гетерогенним. В одній системі можуть співіснувати датчики з мінімальними можливостями, шлюзи з повноцінною ОС та хмарні сервіси. Єдиної "універсальної" криптографічної схеми тут не існує — захист завжди є компромісом між безпекою, вартістю та функціональністю.

Обмежені ресурси та вибір криптографічних алгоритмів

Одним з головних викликів є обмежені обчислювальні та енергетичні ресурси. Багато IoT-пристроїв мають слабкі процесори, обмежену оперативну пам'ять і працюють від батарей або енерго-збору. Це накладає серйозні обмеження на вибір криптографічних алгоритмів і режимів їх використання.

У таких умовах пріоритет часто надається алгоритмам із невеликими ключами та передбачуваною обчислювальною складністю. Симетрична криптографія зазвичай значно дешевша за асиметричну, але потребує надійного механізму розповсюдження ключів. Асиметричні алгоритми використовуються обмежено — переважно на етапі початкової ініціалізації або для встановлення довіри між пристроєм і сервером.

Управління ключами та життєвий цикл пристроїв

Найскладнішим аспектом безпеки IoT є життєвий цикл ключів. Пристрій може перебувати в експлуатації 10–20 років, протягом яких змінюються криптографічні рекомендації, загрози та навіть власник системи. Якщо ключі закладені жорстко на етапі виробництва і не можуть бути змінені, компрометація одного елемента може мати катастрофічні наслідки для всієї мережі.

Тому сучасні підходи орієнтовані на безпечний бутстрепінг: пристрій при першому запуску встановлює унікальну криптографічну ідентичність, а ключі можуть оновлюватися або відкликатися протягом усього життєвого циклу. Важливу роль тут відіграють апаратні корені довіри, які дозволяють захищати ключовий матеріал навіть у разі часткової компрометації програмного забезпечення.

Захищені протоколи зв'язку для IoT

На мережевому рівні безпека IoT реалізується через спеціалізовані або адаптовані протоколи зв'язку. Вони мають забезпечувати шифрування, автентифікацію та захист від повторних атак,

але водночас залишатися достатньо легкими для обмежених пристроїв.

Захищені канали в IoT часто будуються поверх протоколів із мінімальними накладними витратами, а криптографічні механізми спрощуються порівняно з “повноцінними” мережевими протоколами. При цьому критично важливо, щоб спрощення не підривали базові властивості безпеки — конфіденційність, цілісність і автентичність.

У підсумку безпека IoT — це не стільки питання вибору “правильного алгоритму”, скільки узгодження криптографії з реальними обмеженнями середовища. Помилка на будь-якому етапі — від генерації ключів до оновлення прошивки — може звести нанівець навіть найсильніші криптографічні примітиви.

Приклади завдань

1. Пояснити, які криптографічні механізми забезпечують конфіденційність і цілісність даних у TLS 1.3 порівняно з попередніми версіями TLS.
2. Проаналізувати, чому інтеграція криптографії безпосередньо в транспортний рівень стала ключовою ідеєю протоколу QUIC.
3. Порівняти класичну VPN-модель доступу з підходом Zero Trust з точки зору контролю привілеїв.
4. Оцінити криптографічні ризики використання довготривалих ключів у VPN-інфраструктурі.
5. Запропонувати підхід до безпечного управління ключами для IoT-пристроїв із тривалим життєвим циклом.

Контрольні питання

1. Яку роль відіграє криптографія в забезпеченні безпеки сучасних мережевих протоколів?
2. У чому полягають принципові відмінності TLS 1.3 від TLS 1.2 з точки зору безпеки та продуктивності?
3. Чому протокол QUIC вважається більш стійким до мережевих атак, ніж класичний стек TCP + TLS?
4. Які криптографічні обмеження традиційних VPN-рішень зумовили появу Zero Trust Architecture?
5. Чому IoT-середовища потребують спеціалізованих криптографічних підходів?

Література

1. Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. Internet Engineering Task Force, 2018.
2. Iyengar J., Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport. RFC 9000. Internet Engineering Task Force, 2021.
3. Kaufman C., Hoffman P., Nir Y., Eronen P. Internet Key Exchange Protocol Version 2 (IKEv2). RFC 7296. Internet Engineering Task Force, 2014.
4. Rose S., Borchert O., Mitchell S., Connelly S. Zero Trust Architecture. NIST Special Publication 800-207. Gaithersburg, 2020.
5. Garcia-Morchon O., Kumar S., Keoh S., Hummen R., Struik R. Security Considerations in the IP-Based Internet of Things. RFC 8576. Internet Engineering Task Force, 2019.