МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ПЗВО «МІЖНАРОДНИЙ КЛАСИЧНИЙ УНІВЕРСИТЕТ
імені ПИЛИПА ОРЛИКА»
Економіко-технологічний факультет
Кафедра інженерних технологій

Кваліфікаційна робота
на здобуття освітнього ступеня магістра
за освітньою програмою «Комп'ютерна інженерія»
зі спеціальності 123 «Комп'ютерна інженерія»

на тему: **«ПРОГРАМНИЙ МОДУЛЬ КОНТРОЛЮ ЦІЛІСНОСТІ ІНФОРМАЦІЇ НА ОСНОВІ КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕНЬ»**

Виконав:
здобувач ІІ курсу, групи КІ -20-24
Бугаєнко Владислав Андрійович

Керівник:
к.т.н., доцент кафедри інженерних технологій
Гайша Олександр Олександрович

Миколаїв – 2024

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Програмний модуль контролю цілісності інформації на основі криптографічних перетворень» складається із вступу, основної частини, що містить 3 розділи, висновку, списку використаних джерел. Загальний обсяг роботи – 122 сторінки. Робота містить 55 рисунків, 4 таблиці. Список використаних джерел включає 31 джерело.

Ключові слова: контроль цілісності інформції, коди автентифікації повідомлень, коди виявлення маніпуляцій з даними, імітозахист, алгоритми безпечного хешування, криптографічний захист інформації.

Об’єктами дослідження являються процедура формування криптографічної хеш-функції.

Предметом дослідження є криптографічні методи, алгоритмічні та математичні підходи щодо формування хеш-функції.

Мета роботи – програмна реалізація модуля контролю цілісності інформації на основі криптографічних перетворень.

Метод дослідження. Проведені дослідження базуються на сучасних методах теорії побудови захищених інформаційних мереж та криптографічних методах контролю цілісності інформації.

Наукова новизна. Удосконалено модуль проведння контролю цілісності інформації, за рахунок використання функції хешування сімейства SHA та REPIMD а також поточної дати, часу, які хешуються разом із ключем, що надало йому можливість захисту від атаки грубою силою та забезпечити не тількі цілісності даних, а і цілісності джерела інформації.

Практична значимість роботи полягає у створенні удосконаленого модуля контролю цілісності інформації на основі криптографічних перетворень мовою

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

ІКСМ – Інфомаційно-комунікаційні системи та мережі.

ЕЦП – Електронно-цифровий підпис

MAC – Message Authentication Code - код автентифікації повідомлень.

MDC – Manipulation Detection Code - код виявлення маніпуляцій з даними

HMAC – hashed Message Authentication Code - хеш-код автентифікації повідомлень.

CMAC – Cipher block chaining message authentication code - код автентифікації повідомлення ланцюгуванням шифроблоків

SHA – Secure Hash Algorithm - Алгоритм безпечного хешування

ВСТУП

В умовах глибокого проникнення інформаційних технологій у всі області життєдіяльності людини надійність ідентифікації інформації та контролю її цілісності стає важливою проблемою, причому як науково-технічною, так і соціальною. Науково-технічна проблема включає в себе створення математичних підходів, алгоритмів, програмного та апаратного забезпечення для вирішення цієї проблеми. Соціальний аспект проблеми пов'язаний з необхідністю створення загальнодоступної, зручної та захищеної системи надійної ідентифікації даних, адекватної ступеня розвитку інформаційних технологій.

По наявним оцінкам, більше 90% компаній зіткнулися з внутрішніми вторгненнями, більше половини стикаються з ними постійно, а втрати компаній тільки в США приближуються до 1 трлн. долл. США [1]. Велика частина втрат пов'язана з діями працівників самих цих компаній, так як існуючі методи контролю цілісності даних контролюються самими власниками інформації. При цьому може статися викрадення цінних даних, у тому числі тих, що становлять таємницю. У деяких випадках не менш тяжкі наслідки може мати і модифікація (спотворення, фальсифікація) даних. Причини усвідомленого корпоративного або особистого порушення цілісності інформації - видалення, викривлення, підміни - можуть бути: помста, користь, страх, примушення, вандалізм, цікавість, самоствердження, кар'єрні ідеї, конкуренція, диверсія, саботаж та інші.

Суттєво, що частину порушень цілісності інформації не знає часто і сам власник інформації. У деяких випадках він не може виявити порушень цілісності збереженої інформації, особливо у випадках, наприклад, цільової розбіжності, введеної особами, що мають санкціонований доступ до інформації.

На підвищення довіри до представлених документів, тобто на підтвердження їх цілісності, направлена діяльність таких спеціально створених організацій, як Управління правами інтелектуальної власності (IPRM),

Управління інтелектуальною власністю (IPM), управління цифровими правами (DRM), управління правами (RM), електронне управління авторським правом (ECM). Для цього служить безліч досліджень, розробок і патентів, таких, наприклад, як патент США № 6931545.

Вибір криптографічних методів контролю цілісності інформації обґрунтований роллю "криптографічних протоколів як найбільш перспективного засобу захисту в загальній задачі збереження конфіденційності, цілісності та достовірності інформаційних потоків" [2].

Мета роботи – програмна реалізація модуля контролю цілісності інформації на основі криптографічних перетворень.

Виходячи з мети, завданням даної дипломної роботи є:

- дослідити відомі криптографічні методи захисту інформації та практичні підходи використання контролю цілісності інформації;
- розробити алгоритм проведення контролю цілісності інформації з використанням криптографічних перетворень
- розробити програмну реалізацію модуля контролю цілісності інформації на базі алгоритму HMAC та додаткових параметрів
- провести оцінку ефективності та доцільності використання розробленого програмного модуль з контролю цілісності інформації на основі алгоритму HMAC та додаткових параметрів

Об’єктами дослідження являються процедура формування криптографічної хеш-функції.

Предметом дослідження є криптографічні методи, алгоритмічні та математичні підходи щодо формування хеш-функції.

Проведені дослідження базуються на сучасних методах теорії побудови захищених інформаційних мереж та криптографічних методах контролю цілісності інформації.

Наукова новизна. Удосконалено модуль проведння контролю цілісності інформації, за рахунок використання функції хешування сімейства SHA та REPIMD а також поточної дати, часу, які хешуються разом із ключем, що

надало йому можливість захисту від атаки грубою силою та забезпечити не тількі цілісності даних, а і цілісності джерела інформації.

Практична значимість роботи полягає у створенні удосконаленого модуля контролю цілісності інформації на основі криптографічних перетворень мовою програмування С ++ в середовищі розробки Borland C ++ Builder 6 та бібліотеки CryptoAPI.

Апробація отриманих результатів:

- Огіренко М.С. Дослідження сучасних криптографічних методів контролю цілісності інформації/ М.С. Огіренко // ПОЛІТ-2018: XVIII міжнародна наукова конференції студентів та молодих учених, 3-6 квітня 2018 р.: тези доп. – К., 2018. – С. 10.

- Огіренко М.С. Сучасні криптографічних методів контролю цілісності інформації / Ільєнко А.В., Ільєнко С.С., Огіренко М.С. // Nauka i inowacja – 2019: XV miesdzynarodowej naukowi-praktycznej konferencji, 7-15 października 2019 r.: abstracts. – Przemysl (Polska), 2019. – V.6. – P. 53-57.

РОЗДІЛ 1
СУЧАСНІ ОСНОВИ ОРГАНІЗАЦІЇ ЗАХИСТУ В СУЧАСНИХ КОМПЮТЕРНИХ СИСТЕМАХ ТА МЕРЕЖАХ

1.1. Нормативно-правові основи захисту інформації

Інформаційні ресурси держави або суспільства в цілому, а також окремих організацій і окремих осіб мають певну цінність, мають відповідне матеріальне вираження і вимагають захист від різних внутрішніх впливів, які можуть призвести до зниження цінності інформаційних ресурсів. Впливи, що знижують цінність інформаційних ресурсів, називаються несприятливими. Потенційне несприятливий вплив називається загрозою.

Захист інформації, що обробляється в АС, полягає в створенні і підтримці в дієздатному стані системи заходів, як технічних (інженерних, програмних і апаратних), так і нетехнічних (правових, організаційних), які запобігають або ускладнюють можливість реалізації загроз, а також зменшити потенційні втрати. Іншими словами, захист інформації спрямовано на забезпечення безпеки оброблюваної інформації і АС в цілому, тобто стану, яке зберігає зазначені властивості інформації і АС, який її обробляє. Система цих заходів, яка забезпечує інформаційну безпеку в АС, називається комплексною системою захисту інформації.

Поняття комплексної системи захисту інформації будемо розуміти як сукупність організаційних і технічних заходів, апаратного та програмного забезпечення, що забезпечує захист інформації в ІКСМ: на автономних робочих станціях (автоматизована система класу 1) і в комп'ютерних мережах (автоматизовані класи 2 і 3). [4]. Напрямки розвитку захищених корпоративних і локальних інформаційно-комунікаційних мереж та мереж сприяють постійної зміни методів і засобів комплексного захисту інформаційної безпеки шляхом постійного вдосконалення організації захисту інформаційних ресурсів за допомогою програмно-технічних засобів.

Комплексна система захисту інформації включає заходи і засоби реалізації методів, механізмів захисту інформації від:

А) витік інформації по технічним каналам (бокове електромагнітне випромінювання, акустоелектричні канали і т. Д.);

Б) несанкціоновані дії і несанкціонований доступ до інформації (підключення до апаратних засобів та ліній зв'язку, маскування для зареєстрованих користувачів, використання вбудованих пристроїв або програм і т. Д);

В) особливий вплив на інформацію (формування полів і сигналів з метою порушення цілісності інформації або руйнування системи безпеки).

Для кожної конкретної інформаційної та комунікаційної системи склад, структура та вимоги системи безпеки визначаються властивостями оброблюваної інформації, класом автоматизованої системи та її умовами роботи. Кінцева мета всіх реалізованих заходів інформаційної безпеки - забезпечити безпеку інформації під час її обробки в ІКСМ. Правова і нормативна підтримка, як важливий крок у створенні комплексної системи інформаційної безпеки для сучасних ІКСМ, є важливим компонентом забезпечення ефективної та надійної захисту інформації та інформаційних ресурсів. Створюючи комплексну систему захисту інформації, як сукупність організаційних та інженерних заходів, програмне та апаратне забезпечення повинні керуватися рядом нормативних актів. Основними нормативними документами з організації та побудови комплексної системи захисту інформації в ІКСМ є:

- Закон України "Про інформацію";

- Закон України "Про захист інформації в автоматизованих системах";

- НД ТЗІ 1.1-002-99: Загальні положення про захист інформації в комп'ютерних системах від Ради національної безпеки і оборони (прийнятий наказом Держстату України від 28.04.1999 № 22);

- НД ТЗІ 1.1-003-99: Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу (вступила в силу з розпорядженням Держстату СБУ від 28.04.1999 № 22);

- НД ТЗІ 1.4-001-2000: Типове положення для служби інформаційної безпеки в автоматизованій системі (введено в дію наказом Держстату України від 4 грудня 2000 № 53);

- НД ТЗІ 2.5-005-99: Класифікація автоматизованих систем і стандартні функціональні профілі захисту оброблюваної інформації від несанкціонованого доступу (введено в дію Державним замовленням СБУ від 28 квітня 1999 № 22);

- НД ТЗІ 2.5-004-99: «Критерії оцінки захисту інформації в комп'ютерних системах від несанкціонованого доступу» (введено в дію Постановою Держстату України від 28 квітня 1999 № 22);

- НД ТЗІ 3.7-001-99: Керівництво щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі (введено в дію наказом ГГТУ СБУ від 28.04.1999 р № 22).

НСД слід розуміти як доступ до інформації з використанням засобів, включених в КС, що порушує встановлений КС. Несанкціонований доступ може бути здійснений або за допомогою стандартних інструментів, тобто набору програмно-апаратних засобів, включеного в КС розробником під час розробки, або системним адміністратором в процесі роботи, які включені в затверджену конфігурацію КС.

Основні способи НСД включають в себе:

- прямий доступ до об'єктів з метою отримання певного типу доступу;

- створення програмних і апаратних засобів доступу до об'єктів в обхід функцій безпеки;

- модифікація засобів захисту, що дозволяє реалізувати НСД;

- впровадження в КС програмних або апаратних механізмів, які порушують структуру і функції КС і дозволяють впроваджувати НСД.

Під захистом НСД слід розуміти діяльність, спрямовану на забезпечення відповідності вимога, шляхом створення і підтримки в справному стані системи заходів інформаційної безпеки.

Для забезпечення безпеки інформації під час її обробки в АС створюється КСЗІ, процес управління яким повинен припускатися протягом усього життєвого циклу АС. На етапі розробки мета управління процесом КСЗІ полягає у створенні захисного обладнання, яке могло б ефективно протидіяти можливій загрозі та забезпечити дотримання політики безпеки в майбутньому при обробці інформації. На етапі функціонування АС метою управління системою КСЗІ є оцінка ефективності встановленої КСЗІ та розробка додаткових (уточнюючих) вимог щодо доопрацювання КСЗІ з метою забезпечення її адекватності при зміні початкових умов (характеристики ОС, оброблювану інформацію, фізичне середовище, персонал, призначення АС, політики безпеки тощо).

Основою для побудови комплексної системи інформаційної безпеки сучасного ІКСМ, відповідно до нормативних документів, є забезпечення нормативно-методичної бази для відбору і реалізації вимог щодо захисту інформації та інформаційних ресурсів в ІКСМ. Порядок підбору вимог інформаційної безпеки в ІКСМ визначається відповідно до НД ТЗІ 2.5-005-99 «Класифікація автоматизованих систем і стандартні функціональні профілі захисту оброблюваної інформації від несанкціонованого доступу». Основою надійного й ефективного захисту є вибір стандартного функціонального профілю безпеки.

Стандартні функціональні профілі вибираються на основі існуючих вимог для захисту інформації та інформаційних ресурсів від певних загроз і відомих в даний час функціональних служб, які можуть протидіяти цим загрозам і забезпечувати їх виконання. Основними вимогами до інформаційної безпеки в нашому випадку буде захист цілісності і доступності інформації і інформаційних ресурсів. Порядок вибору вимог щодо інформаційної безпеки в ІКСМ визначається відповідно до НД ТЗІ 2.5-005-99 "Класифікація

автоматизованих систем та стандартних функціональних профілів захисту оброблюваної інформації від несанкціонованого доступу". Основою надійного та ефективного захисту є вибір стандартного функціонального профілю безпеки. Стандартні функціональні профілі вибираються на основі існуючих вимог щодо захисту інформації та інформаційних ресурсів від певних загроз та відомих на даний момент функціональних служб, що дозволяють протидіяти цим загрозам та забезпечують їх виконання.

Основними вимогами до інформаційної безпеки в нашому випадку буде захист цілісності та доступності інформації та інформаційних ресурсів. Отже, функціональний профіль безпеки для сучасних розгалужених ІКС приймає такий вигляд: 3. ЦД.1- 3. ЦД.4. Вибраний профіль безпеки дозволяє захистити об'єкти від несанкціонованих змін інформації, що міститься в них під час їх передачі через незахищене середовище, та включає обов'язкові процедури ідентифікації та автентифікації. Цей набір послуг, представлений функціональним профілем захисту, забезпечує захист від навмисних та ненавмисних помилок користувача та інших випадкових помилок, а також від зміни інформації в разі підключення сторонніх користувачів. Він також передбачає виявлення випадкових або навмисних порушень цілісності та доступності не лише окремих повідомлень, а й потоків повідомлень у цілому. Тобто, метою створення захищеної ІКСМ з урахуванням регуляторної підтримки є виявлення та протидія загрозам безпеці інформації, яка обробляється та передається до них, з метою запобігання порушенням цілісності та доступності інформації.

1.2. Сучасні напрями та методи захисту інформаційних ресурсів

Оскільки комп'ютери стають більш зрозумілими та економічними, щодня з’являється безліч нових програм. Багато з цих програм включають як зберігання інформації, так і одночасне використання декількох осіб. Багато

прикладів систем, які потребують інформаційної безпеки, зустрічаються щодня: банки даних кредитних бюро; правоохоронні інформаційні системи; бюро обслуговування; он-лайн медична інформаційна система; і державні системи обробки соціальних даних. Ці приклади охоплюють широкий спектр потреб в організації та особистій конфіденційності.

Потенційні порушення безпеки можливо розмістити у трьох категоріях:

1) Несанкціоноване звільнення інформації: несанкціонована особа здатна читати та використовувати інформацію, що зберігається на комп'ютері. Зловмисник лише спостерігає зразки використання інформації, вона також включає несанкціоноване використання фірмової програми.

2) Несанкціонована модифікація інформації: несанкціонована особа здатна змінювати збережену інформацію - форма диверсії. Подібне порушення не вимагає від зловмисника бачити, як він змінив інформацію.

3) Несанкціонована відмова у використанні: зловмисник може перешкодити авторизованому користувачеві надсилати чи змінювати інформацію, навіть якщо зловмисник може не мати змоги надсилати чи змінювати інформацію. Причини "збоїв системи", відмови алгоритму планування - приклади відмов у використанні. Це ще одна форма диверсій.

Приклади технік безпеки, які іноді застосовуються до комп'ютерних систем, є наступними:

1) маркування файлів зі списками авторизованих користувачів;

2) перевірка особи потенційного користувача, вимагаючи пароль;

3) екранування комп'ютера для запобігання перехоплення та подальшої інтерпретації електромагнітного випромінювання;

4) шифрування інформації, що надсилається по телефонних лініях;

5) блокування кімнати, що містить комп'ютер;

6) контроль того, кому дозволено вносити зміни до комп'ютерної системи (як її апаратного, так і програмного забезпечення) ;

7) використовуючи надлишкові схеми або запрограмовані перехресні перевірки, які підтримують безпеку внаслідок відмов обладнання або програмного забезпечення;

8) підтвердження того, що апаратне та програмне забезпечення реально реалізовано за призначенням.

В даний час комп'ютерні злочини надзвичайно різноманітні. Це несанкціонований доступ до інформації, що зберігається в комп'ютері, введення в програмне забезпечення логічних бомб, розробка і поширення комп'ютерних вірусів, розкрадання комп'ютерної інформації, недбалість у розробці, виготовленні та експлуатації програмно-обчислювальних комплексів, підробка комп'ютерної інформації.

Всі заходи протидії комп'ютерним злочинам, що безпосередньо забезпечують безпеку інформації, можна підрозділити на такі напрями:

- правові;
- організаційно-адміністративні;
- інженерно-технічні.

До правових заходів (рис. 1.1) варто віднести розробку норм, що встановлюють відповідальність за комп'ютерні злочини, захист авторських прав програмістів, удосконалювання кримінального і цивільного законодавства. До них відносяться також питання суспільного контролю за розроблювачами комп'ютерних систем і прийняття відповідних міжнародних договорів про обмеження, якщо вони впливають або можуть уплинути на військові, економічні і соціальні аспекти країн. Тільки в останні роки з'явилися роботи з проблем правової боротьби з комп'ютерними злочинами. А зовсім недавно і вітчизняне законодавство стало на шлях боротьби з комп'ютерною злочинністю.

До організаційно-адміністративних заходів (рис. 1.2.) відносяться: охорона комп'ютерних систем, підбір персоналу, виключення випадків ведення особливо важливих робіт тільки однією людиною, наявність плану відновлення

Рис. 1.1 Правове забезпечення безпеки

працездатності центру після виходу його з ладу, обслуговування обчислювального центру сторонньою організацією або особами, незацікавленими в приховуванні фактів порушення роботи центру, універсальність засобів захисту від усіх користувачів (включаючи вище керівництво), покладання відповідальності на осіб, що повинні забезпечити безпеку центру, вибір місця розташування центру і т.п.

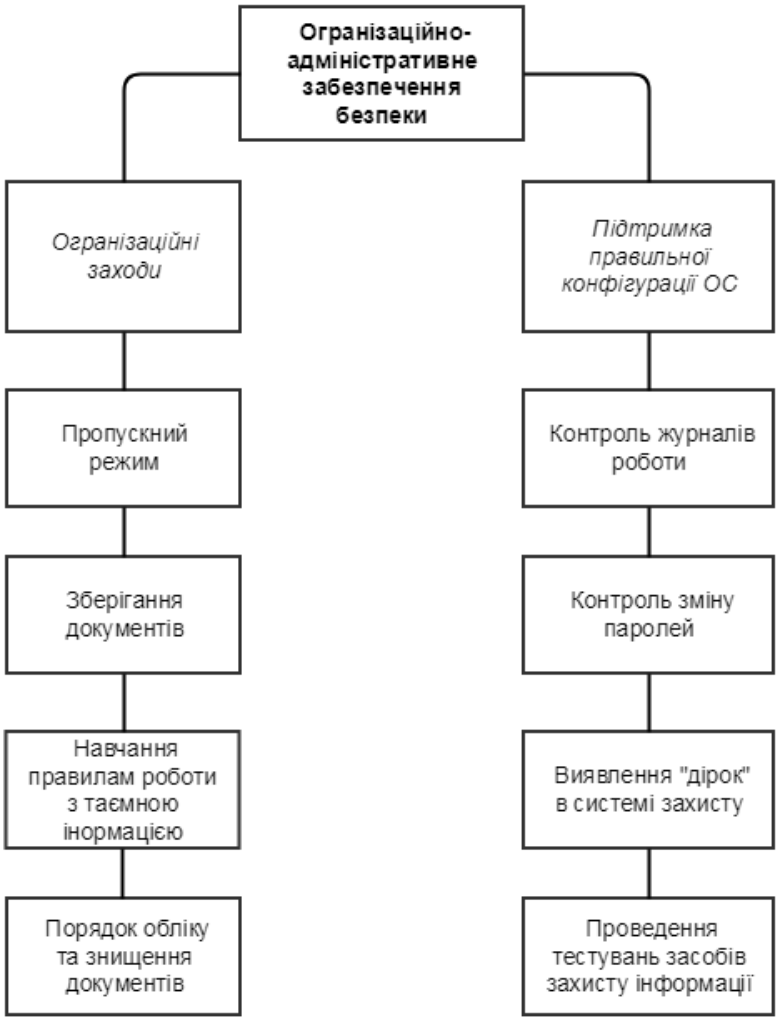


Рис. 1.2 Організаційно-адміністративне забезпечення безпеки

Комплекс методів захисту ресурсів (організаційно-адміністративне):

- розробити внутрішню документацію, в якій вказані правила роботи з комп'ютерним обладнанням та конфіденційною інформацією;
- проводити інструктажі та періодичні перевірки персоналу;
- ініціювати підписання додаткових угод до трудових договорів, в яких визначені обов'язки щодо розголошення або неправомірне використання інформації, пов'язаної з роботою;
- розмежувати обов'язки, щоб уникнути ситуацій, в яких один співробітник має в своєму розпорядженні найбільш важливі файли даних;
- організувати роботу з додатками загального робочого процесу і забезпечити зберігання важливих файлів на мережевих дисках;

- інтегрувати програмні продукти, які захищають дані від копіювання або знищення будь-яким користувачем, включаючи вище керівництво компанії;
- розробіть плани відновлення системи у разі збоїв по будь-якої причини;

До інженерно-технічних заходів (рис. 1.3.) можна віднести захист від несанкціонованого доступу до комп'ютерної системи, резервування важливих комп'ютерних систем, забезпечення захисту від розкрадань і диверсій, резервне електроживлення, розробку і реалізацію спеціальних програмних і апаратних комплексів безпеки тощо.

Рис. 1.3 Інженерно-технічне забезпечення безпеки

Фізичні засоби містять у собі різні інженерні засоби, що перешкоджають фізичному проникненню зловмисників на об'єкти захисту, що захищають персонал (особисті засоби безпеки), матеріальні засоби і фінанси, інформацію від протиправних дій.

До апаратних засобів відносяться прилади, пристрої, пристосування та інші технічні рішення, які використовуються в інтересах забезпечення безпеки. У практиці діяльності будь-якої організації знаходить широке застосування різна апаратура: від телефонного апарату до розроблених автоматизованих інформаційних систем, що забезпечують її виробничу діяльність. Основна задача апаратних засобів – стійка безпека комерційної діяльності.

Програмні засоби - це спеціальні програми, програмні комплекси і системи захисту інформації в інформаційних системах різного призначення і засобах обробки даних.

Комплекс методів (об'єднує апаратні і програмні засоби):

- регулярне резервне копіювання і віддалене зберігання найбільш важливих файлів даних в комп'ютерній системі;
- дублювання і резервне копіювання всіх мережевих підсистем, які важливі для безпеки даних;
- можливість перерозподілу мережевих ресурсів в разі несправності окремих елементів;
- можливість використання систем резервного живлення;
- забезпечення безпеки від пожежі або пошкодження водою;
- установка сучасних продуктів, що захищають бази даних і іншу інформацію від несанкціонованого доступу.

Ідентифікація та аутентифікація використовуються для запобігання несанкціонованому доступу до інформації. Аутентифікація та ідентифікація призначені для надання або відмови в доступі до даних. Справжність встановлюється трьома способами: програмою, апаратом або людиною. Крім людини, що є об'єктом аутентифікації, він може поширюватися на обладнання

(комп'ютер, монітор і носії) або дані. Установка пароля - найпростіший спосіб захисту.

Криптографічні засоби - це спеціальні математичні та алгоритмічні засоби захисту інформації, переданої по мережах зв'язку, збереженої та обробленої на комп'ютерах з використанням методів шифрування.

Самим надійним захистом від несанкціонованого доступу до повідомлень, що передаються якимось чином, і програмних продуктів персонального комп'ютера є застосування різних методів шифрування (криптографічних методів захисту інформаційних ресурсів).

Інформаційна безпека використовує криптографію для перетворення інформації, що використовується в форму, яка робить її непридатною для використання будь-ким, крім авторизованого користувача; цей процес називається шифруванням. Інформація, яка була зашифрована (надана непридатною для використання), може бути перетворена назад в її первісну придатну для використання форму авторизованим користувачем, який має криптографічним ключем, за допомогою процесу дешифрування. Криптографія використовується в інформаційній безпеці для захисту інформації від несанкціонованого або випадкового розкриття, коли інформація знаходиться в процесі передачі (в електронному або фізичному вигляді) і коли інформація знаходиться в сховищі.

Криптологія – наука про математичні методи захисту повідомлень шляхом їх перетворення; це галузь знань, де вивчаються тайнопис (криптографія) і методи її розкривання (криптоаналіз).

Основні напрямки використання криптографічних методів - це передача конфіденційної інформації через канали зв'язку (наприклад, електронна пошта), встановлення дійсності переданих повідомлень, збереження інформації (документів, баз даних) на носіях у зашифрованому виді.

Сучасна криптографія вивчає і розвиває такі напрямки:

- симетричні криптосистеми (із секретним ключем);
- несиметричні криптосистеми (з відкритим ключем);

- системи електронного підпису;
- системи керування ключами.

Сучасні криптографічні системи забезпечують високу стійкість зашифрованих даних за рахунок підтримки режиму таємності криптографічного ключа.

Допомагаючи зберегти зміст повідомлення в таємниці, криптографію можна використовувати для забезпечення:

- аутентифікації;
- цілісності;
- незаперечності.

При аутентифікації одержувачеві повідомлення потрібно переконатися, що воно виходить від конкретного відправника. Зловмисник не може надіслати фальшиве повідомлення від будь-якого імені.

При визначенні цілісності одержувач повідомлення в змозі перевірити, чи були внесені які-небудь зміни в отримане повідомлення під час його передачі. Зловмисникові не дозволено замінювати дійсне повідомлення на фальшиве.

Незаперечність необхідна для того, щоб відправник повідомлення не зміг згодом заперечувати, що він не є автором цього повідомлення.

В даний час аутентифікація, що здійснюється користувачем, забезпечується за допомогою:

- смарт-карт;
- засобів біометрії;
- клавіатури комп'ютера;
- криптографії з унікальними ключами для кожного користувача.

Цілісність інформації забезпечується за допомогою криптографічних контрольних сум і механізмів керування доступом і привілеями. У якості криптографічної контрольної суми для виявлення навмисної або випадкової модифікації даних використовується код аутентифікації повідомлення – MAC (Message Autentification Code).

Для виявлення несанкціонованих змін у переданих повідомленнях можна застосувати:

- електронно-цифровий підпис (ЕЦП), заснований на криптографії з відкритим і закритим ключами;
- програми виявлення вірусів;
- призначення відповідних прав користувачам для керування доступом;
- точне виконання прийнятого механізму привілеїв.

Незаперечність повідомлення підтверджується електронно-цифровим підписом.

1.3. Імітозахист інформації. Контроль цілісності потоку повідомлень

В теорії захисту інформації розглядається захист від двох класів впливів - випадкових і навмисних. Захист інформації, що передається по каналах зв'язку, від випадкових перешкод здійснюється за допомогою її завадостійкого кодування. При такому кодуванні в інформацію вноситься надмірність (додається контрольна сума, обчислена за певним алгоритмом), і на приймальному кінці з використанням цієї надмірності проводиться виявлення і/або виправлення помилок, внесених в повідомлення при його передачі.

В даний час розроблено велику кількість методів завадостійкого кодування, найбільш популярними серед яких є такі:

- кодування з контролем довічного повідомлення на парність (числа одиниць в повідомленні) або на непарність;
- кодування із забезпеченням постійної ваги довічного повідомлення (постійного числа одиниць в повідомленні);
- корреляционное (парафазного) кодування («1» і «О» кодуються парою символів, наприклад, «10» і «01» відповідно);
- кодування на основі лінійних рівнянь в полях Галуа і т.д.

Однак за допомогою завадостійкого кодування важко забезпечити захист від навмисних впливів на повідомлення, так як алгоритми кодування є відкритими і відомі зловмиснику. У цьому випадку він може модифікувати повідомлення і потім знову обчислити контрольну суму, а потім передати змінений повідомлення одержувачу. Він також може нав'язувати неправдиву інформацію, створюючи власні повідомлення, кодуючи їх перешкодостійким кодом і передаючи їх в канал зв'язку.

Захист каналу шифрованого зв'язку від нав'язування неправдивої інформації носить назву імітозахисту.

Для забезпечення імітозахисту необхідно, щоб зловмисник не мав можливості створювати правильні повідомлення (тобто ті, які на приймальному кінці каналу будуть сприйняті як правильні). Більш формально - ймовірність випадково обраного повідомлення пройти перевірку на справжність не повинна перевищувати деяку задану величину.

Це можливо шляхом внесення надмірності в повідомлення подібно до того, як це робиться в разі захисту від випадкових перешкод.

Структура найпростішого перешкодозахисного кріптоканала представлена на рис. 1.4.

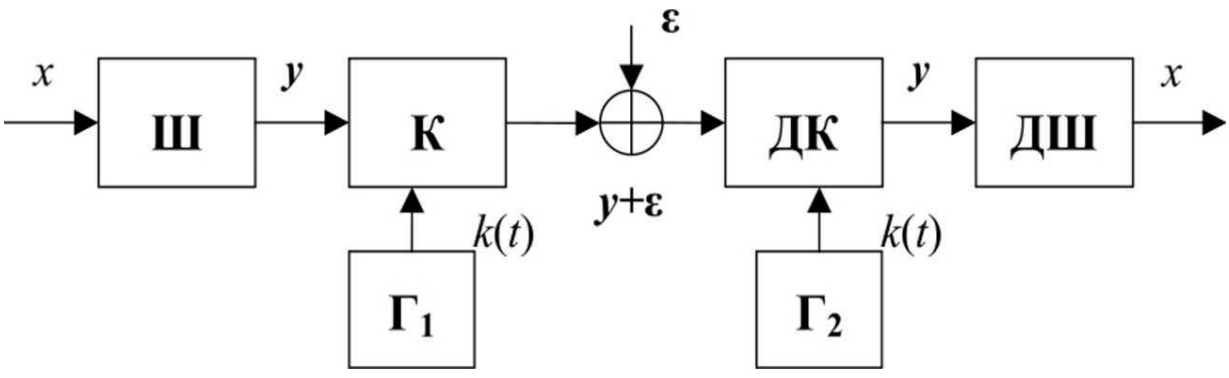


Рис. 1.4 Найпростіший перешкодозахищений кріптоканал

У цій схемі:

- Г1 і Г2 - синхронні генератори ключів k(t);
- Ш - криптографічний шифратор;

- К - перешкодостійкий кодер;
- ДК - перешкодостійкий декодер;
- ДШ - криптографічний дешифратор;
- х - відкритий текст;
- у - шифрування повідомлення;
- Є - помилкова інформація і (або) перешкода.

В даній схемі ключ шифрування змінюється від повідомлення до повідомлення. Це не дає можливості зловмиснику розшифровувати інформацію, оскільки схема роботи генератора ключів (і, отже, послідовність ключів) є секретним параметром.

Не зважаючи на це, захист, що забезпечується за такою схемою, не є повною, оскільки вона дозволяє зловмисникові підміняти повідомлення. В цьому випадку у приймаючої сторони не буде можливості визначити, не користуючись додатковою інформацією (наприклад, надмірністю повідомлення), є чи повідомлення помилковим або істинним. Дану проблему можна вирішити, запровадивши в повідомлення додаткову надмірність перед шифруванням (рис. 1.5.).

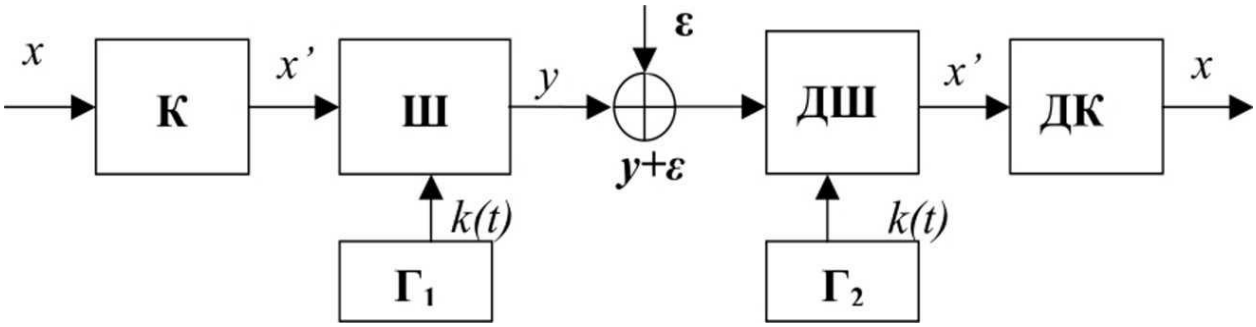


Рис. 1.5 Захист від підміни повідомлень

Тепер, перед шифруванням, повідомлення кодується перешкодостійким кодом, а одержувач після дешифрування перевіряє цілісність повідомлення за допомогою декодера. Якщо зловмисникові невідомий алгоритм шифрування (або завадостійкого кодування), він не зможе підмінити повідомлення, тому що

ймовірність того, що розшифроване неправдиве повідомлення буде правильним кодовим словом завадостійкого коду, мала, і одержувач зможе легко виявити підміну.

При цьому слід мати на увазі, що якщо алгоритми шифрування і завадостійкого кодування не є комутативними по відношенню до спотворень кодових послідовностей за рахунок перешкод в каналі зв'язку, то для завадостійкості даної схеми необхідно введення ще одного перешкодостійкого кодера після криптографічного шифратора, і завадостійкого декодера перед криптографічним дешифратором .

У розглянутих вище схемах невирішеною залишається проблема узгодження роботи генераторів ключів Г1 і Г2. Якщо одне або кілька повідомлень будуть пропущені, то генератори на передавальної і приймальні сторонах виявляться в різних станах, і прийом повідомлень стане неможливим. Необхідно передбачити засіб для синхронізації роботи генераторів ключів. Зауважимо, що передача постійної синхронизируючої інформації для цієї мети не підходить, так як зловмисник може визначити її структуру та в подальшому перешкодити процесу синхронізації.

Пропонована схема (рис. 1.6.) дає можливість забезпечити синхронізацію генераторів ключів, не передаючи ніякої додаткової інформації, а також виявляти випадкові помилки, що вносяться до повідомлення у процесі їх передачі.

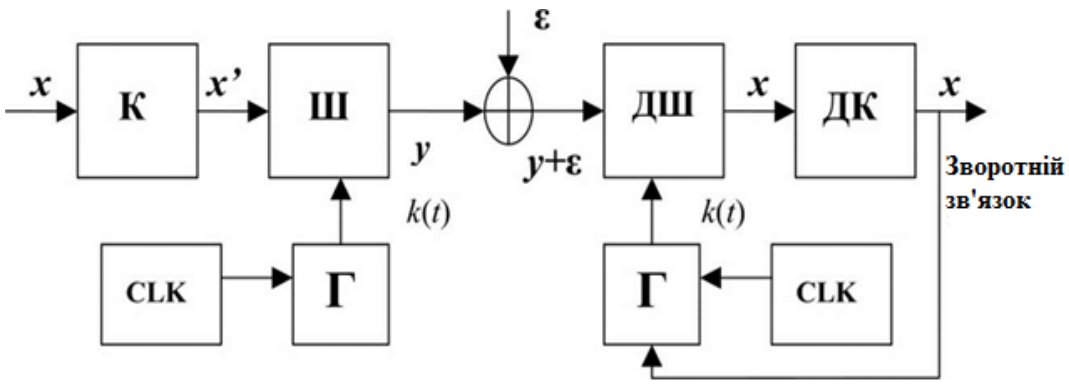


Рис. 1.6 Синхронізація генераторів ключів

На приймальній і передавальній стороні на вхід генераторів ключів подаються сигнали годинника CLK, і генерація ключів проводиться в залежності від їх показань. Оскільки годинник працює автономно, можливо неузгодженість їх показань на деяку, обмежену зверху величину, яка визначається конструкцією годин. Якщо позначити максимальну величину різниці показань годин через Δ, а величину дискретності показань годин через δ, то можливо неузгодженість генераторів ключів максимально на $l = \left|\Delta/\delta\right|$ позицій.

На приймальній стороні після отримання чергового повідомлення виробляється його розшифрування з ключами, які визначаються станами генератора, віддаленими від поточного не більше, ніж на l позицій і проводиться декодування перешкодостійкого коду. У разі, якщо повідомлення буде розшифровано з невірним ключем, ймовірність того, що результат виявиться правильним кодовим словом, мала, можна буде визначити точне значення ключа і величину поправки до показань годин.

Якщо в процесі передачі в повідомлення було внесено спотворення, то це призведе до того, що після розшифрування ми не отримаємо правильного кодового слова ні при якому значенні ключа із зазначеного вище діапазону. Дана ситуація буде свідчити про наявність помилок в повідомленні, воно буде відкинуто, а спроба синхронізації буде здійснена при прийомі наступного повідомлення.

Зазвичай процедура імітозахисту будується на основі певної криптографічного системи: до повідомлення додається відрізок інформації фіксованої довжини, обчислений за певним правилом на основі відкритих даних і ключа, званий імітовставкою, або кодом аутентифікації повідомлення (MAC - Message Authentication Code).

У відкриті дані, які використовуються для вироблення імітовставки може бути включена, крім власне тексту повідомлення, і службова інформація, така як дата і час відправлення повідомлення, реєстраційний номер повідомлення та

т.п. В цьому випадку можна забезпечити також захист від повторної передачі раніше переданого правильного повідомлення.

Забезпечення імітозахисту за схемою, яка наведена на рис. 1.7. При виробленні імітовставки використовуються 16 циклів шифрування в режимі простої заміни. Вироблення імітовставки проводиться з використанням тієї ж ключової інформації, що застосовується для шифрування вихідного тексту. Імітовставка виробляється з урахуванням блоків відкритої інформації (службова інформація, сінхропосилка і т.п.), яка може не зашифровувати. При цьому вся вхідна відкрита інформація розбивається на 64-бітові блоки, які перед виробленням імітовставки діляться на 32-бітові блоки N1, N2. (Якщо останній 64-бітовий блок неповний, то він доповнюється нулями).

На приймальному кінці імітовставка виробляється аналогічно - після дешифрування інформації, і розраховане значення порівнюється з отриманим по каналу зв'язку. У разі збігу повідомлення вважається дійсним, в іншому випадку - хибним.

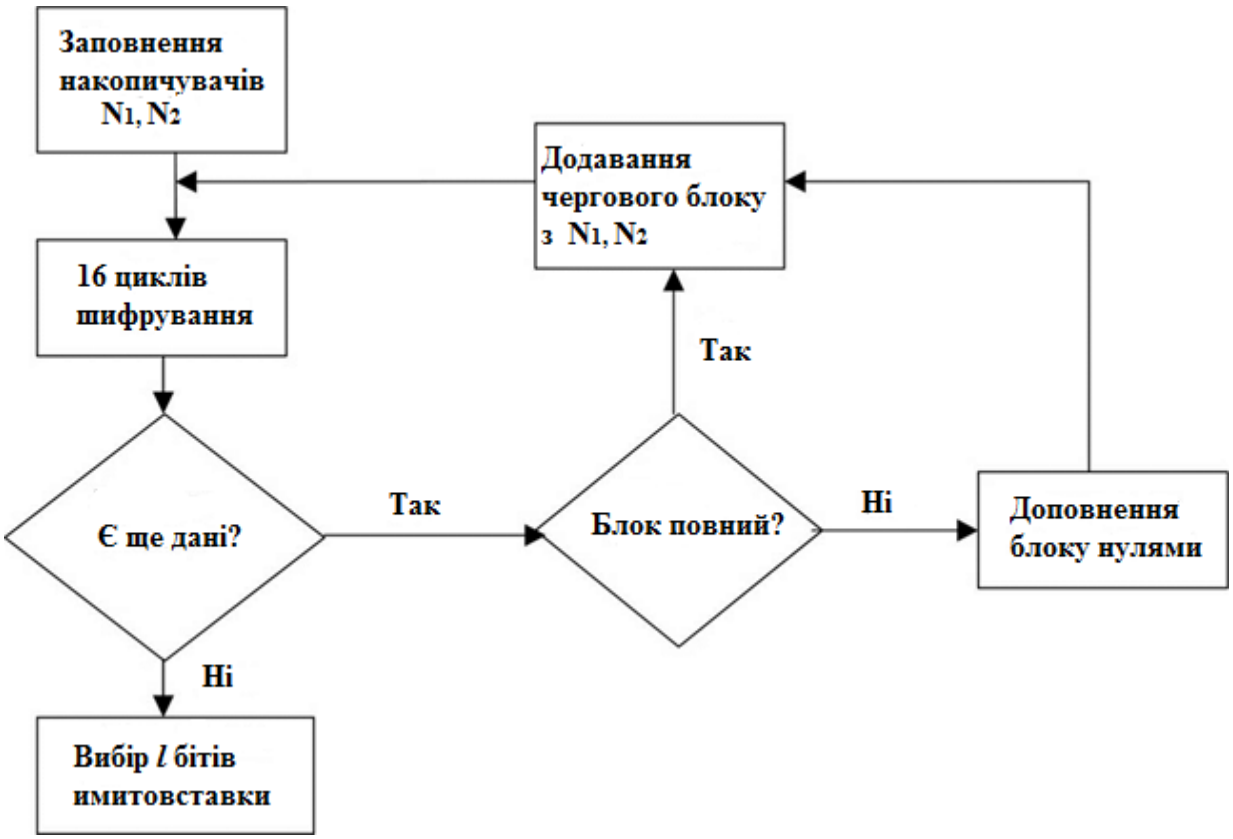


Рис. 1.7 Режим вироблення імітовставки

Даний режим повинен використовуватися для забезпечення імітозахисту зв'язку в умовах, коли виключена можливість введення неправдивої інформації з боку законних користувачів шифрованого зв'язку.

Висновки до 1 розділу

В даному розділі дипломної роботи були описані поняття комплексної системи захисту інформації, розглянуто нормативно-правові основи захисту інформації. Було визначено функціональний профілю захищеності для сучасних розгалужених ІКСМ: 3.ЦД.1- 3.ЦД.4. Вибраний профіль захищеності дозволяє забезпечити захист об'єктів від несанкціонованої модифікації інформації, що міститься в них, під час їх передачі через незахищене середовище та включає в себе обов’язкове проведення процедур ідентифікації і аутентифікації. Цей набір послуг представлений функціональним профілем захищеності забезпечує захист від навмисних та ненавмисних помилок користувача та інших випадкових помилок, а також від модифікації інформації у разі підключенні несанкціонованих користувачів.

Було розглянуто три напрями захисту інформаційних ресурсів - правові; організаційно-адміністративні; інженерно-технічні. Контроль цілісності інформацій можливо забезпечити, якщо комплексно підійти до питання захисту, та використовувати методи захисту інформації та контролю цілісності на кожному із перелічених напрямків. У даній дипломній роботі розглядається контроль цілісності інформації на інженерно-технічному напрямку, а саме – контроль цілісності інформації на основі криптографічних методів. З цієї причини найбільша увага приділялась цьому напряму.

Кожне повідомлення передається по каналу зв’язку, тому в першу чергу було розглянуто методи, за допомогою яких можливо здійснити контроль цілісності потоку повідомлень. Одним із таких методів є завадостійке кодування, яке здійснює захист інформації, що передається по каналах зв'язку,

від випадкових перешкод. Однак за допомогою завадостійкого кодування важко забезпечити захист від навмисних впливів на повідомлення, так як алгоритми кодування є відкритими і відомі зловмиснику. Тому, з точки зору інформаційної безпеки, доречніше використовувати імітозахист – це захист каналу шифрованого зв'язку від нав'язування неправдивої інформації. У данній главі цей метод був розглянутий більш детально, визначено яким чином можливо зробити за допомогою шифрування перешкодозахищений кріптоканал, яким чином треба забезпечити захист каналу зв’язку від підміни повідомлень, та було здійснено вироблення режиму імітозахисту на основі використання імітовставки.

РОЗДІЛ 2
КРИПТОГРАФІЧНІ МЕТОДИ КОНТРОЛЮ ЦІЛІСНОСТІ

2.1. Сучасні методи контролю цілісності на базі криптографічних перетворень

При обміні конфіденційною інформацією одержувач повинен мати впевненість у тому, що повідомлення прийшло цілим від передбачуваного відправника і не було змінено випадково чи іншим чином. Існує два типи загроз цілісності даних: пасивна та активна.

Пасивні загрози. Цей тип загроз існує через випадкові зміни в даних. Ці помилки даних можуть виникати через шум в каналі зв'язку. Крім того, дані можуть бути пошкоджені, поки файл зберігається на диску.

Коди з виправленням помилок і прості контрольні суми, такі як циклічні перевірки надмірності (CRC), використовуються для виявлення втрати цілісності даних. У цих методах дайджест даних обчислюється математично і додається до даних.

Активні загрози. У цьому типі загроз зловмисник може маніпулювати даними зі злим умислом. На найпростішому рівні, якщо дані не дайджест, вони можуть бути змінені без виявлення. Система може використовувати методи додавання CRC до даних для виявлення будь-якої активної модифікації. На більш високому рівні загрози зловмисник може змінити дані і спробувати отримати новий дайджест для змінених даних з існуючого дайджесту. Це можливо, якщо дайджест обчислюється з використанням простих механізмів, таких як CRC.

Цифрові підписи або коди аутентифікації повідомлень є криптографічними механізмами, які можуть використовуватися для виявлення як випадкових змін, які можуть статися через збій обладнання або проблем передачі, так і навмисних змін, які можуть бути виконані зловмисником. Хоча некріптографічні механізми можуть використовуватися для виявлення випадкових змін, вони не є надійними для виявлення навмисних змін.

Механізм безпеки, такий як хеш-функції, використовується для боротьби з погрозами активної модифікації. Хеш-функції надзвичайно корисні і з'являються практично у всіх додатках інформаційної безпеки.

Хеш-функція - це математична функція, яка перетворює числове вхідний значення в інше стислий числове значення. Вхідні дані для хеш-функції мають довільну довжину, але вихідні дані завжди мають фіксовану довжину. Значення, що повертаються хеш-функцією, називаються дайджестом повідомлення або просто хеш-значеннями.

Типові особливості хеш-функцій:

- вихід фіксованої довжини (значення хеша);

o Хеш-функція охоплює дані довільної довжини до фіксованої довжини. Цей процес часто називають хешем даних.

o Загалом, хеш набагато менше, ніж вхідні дані, тому хеш-функції іноді називають функціями стиснення. Оскільки хеш представляє собою меншу уявлення великих даних, його також називають дайджестом.

o Хеш-функція з n-бітовим виходом називається n-бітної хеш-функцією. Популярні хеш-функції генерують значення від 160 до 512 біт.

- ефективність;

o Зазвичай для будь-якої хеш-функції h з входом x обчислення h (x) є швидкої операцією.

o Обчислювальні хеш-функції набагато швидше, ніж симетричне шифрування.

Щоб бути ефективним криптографічним інструментом, бажано, щоб хеш-функція мала такі властивості:

- **першовзорова стійкість;**

Ця властивість означає, що в обчислювальному відношенні має бути важко перевернути хеш-функцію. Іншими словами, якщо хеш-функція h створила хеш-значення z, то буде важко знайти будь-який вхідний значення x, яке хешує до z. Ця властивість захищає від зловмисника, який має тільки хеш-значення і намагається знайти вхідні дані.

- **друга першовзорова стійкість;**

Ця властивість означає, що задані вхідні дані і їх хеш, має бути важко знайти інший вихідний файл з тим же хешем. Іншими словами, якщо хеш-функція h для входу x створює хеш-значення h (x), то повинно бути важко знайти будь-яке інше вхідне значення y таке, що h (y) = h (x). Ця властивість хеш-функції захищає від зловмисника, у якого є вхідний значення і його хеш, і він хоче замінити інше значення допустимим значенням замість початкового вхідного значення.

- **колізійна стійкість.**

Це властивість означає, що має бути важко знайти два різних входа будь-якої довжини, які призводять до одного і того ж хешу. Це властивість також називається хеш-функцією без зіткнень. Іншими словами, для хеш-функції h важко знайти будь-які два різних входу x і y, для яких h (x) = h (y). Оскільки хеш-функція є функцією стиснення з фіксованою довжиною хеш-функції, неможливо, щоб хеш-функція не мала колізій. Це властивість відсутності зіткнень тільки підтверджує, що ці зіткнення важко знайти. Ця властивість дуже заважає зловмисникові знайти два вхідних значення з однаковим хешем. Крім того, якщо хеш-функція стійка до зіткнень, вона є другою стійкою до зображень.

В основі хешування лежить математична функція, яка працює з двома блоками даних фіксованого розміру для створення хеш-коду (Рис. 2.1). Ця хеш-функція є частиною алгоритму хешування.

Розмір кожного блоку даних варіюється в залежності від алгоритму. Зазвичай розміри блоків сягають від 128 до 512 біт.

Рис. 2.1 Формування хешу

Алгоритм хешування включає в себе раунди вищевказаної хеш-функції, такі як блоковий шифр. Кожен раунд вимагає введення фіксованого розміру, зазвичай це комбінація самого останнього блоку повідомлення і результату останнього раунду. Цей процес повторюється стільки раз, скільки потрібно для хешування всього повідомлення (Рис 2.2).

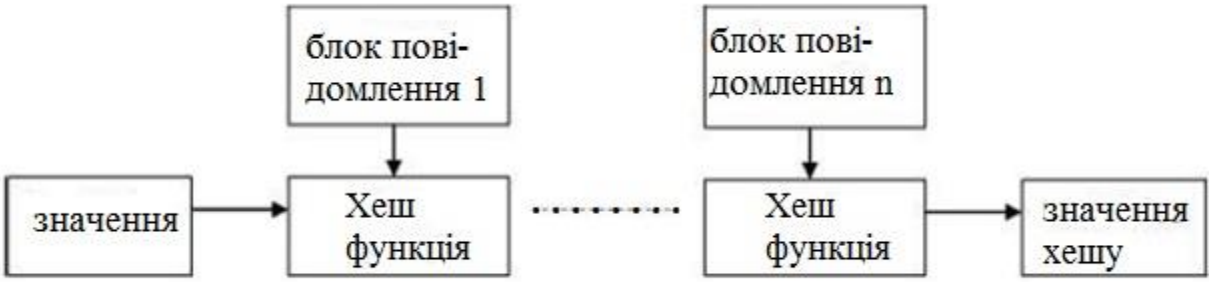


Рис. 2.2 Алгоритм хешування

Оскільки значення хеш-функції першого блоку повідомлень стає входом для другої операції хешування, вихід якої змінює результат третьої операції, і так далі. Цей ефект відомий як лавинний ефект перемішування.

Лавинний ефект призводить до істотно розрізняються значенням хеш-функції для двох повідомлень, які відрізняються навіть одним бітом даних.

Треба розуміти різницю між хеш-функцією і алгоритмом. Хеш-функція генерує хеш-код, працюючи з двома блоками довічних даних фіксованої довжини. Алгоритм хешування - це процес використання хеш-функції, що

визначає, як повідомлення буде розбито і як результати попередніх блоків повідомлень об'єднуються воєдино.

Розглянемо деякі популярні хеш-функції:

- **Дайджест повідомлення (MD):**

  - MD5 був найпопулярнішою і широко використовуваною хеш-функцією протягом декількох років.

  - Сімейство MD складається з хеш-функцій MD2, MD4, MD5 і MD6. Він був прийнятий як Інтернет-стандарт RFC 1321. Це 128-бітна хеш-функція.

  - MD2 призначений для систем з обмеженою пам'яттю, таких як смарт-карти. MD4 розроблено Rivest, аналогічно MD2, але розроблено спеціально для швидкої обробки в програмному забезпеченні. MD5 також розроблений компанією Rivest після того, як в MD4 було повідомлено про можливі слабкостях; ця схема схожа на MD4, але повільніше, тому що більше маніпуляцій з вихідними даними.

  - Дайджести MD5 широко використовуються в світі програмного забезпечення для забезпечення цілісності переданого файлу. Наприклад, файлові сервери часто надають попередньо обчислену контрольну суму MD5 для файлів, щоб користувач міг порівняти з нею контрольну суму завантаженого файлу.

  - У 2004 році зіткнення були виявлені в MD5. Повідомлялося, що аналітична атака була успішною тільки через годину з використанням комп'ютерного кластера. Ця атака зіткнення привела до злому MD5 і, отже, більше не рекомендується для використання.

- **Безпечна хеш-функція (SHA):**

  - Сімейство SHA складається з чотирьох алгоритмів SHA; SHA-0, SHA-1, SHA-2 і SHA-3. Хоча з однієї сім'ї, є структурно різні.

  - Первісна версія - SHA-0, 160-бітна хеш-функція, була опублікована Національним інститутом стандартів і технологій (NIST) в 1993 році. Вона мала кілька слабких місць і не стала дуже популярною. Пізніше, в 1995 році, SHA-1 був розроблений для виправлення передбачуваних недоліків SHA-0.

- SHA-1 є найбільш широко використовуваним з існуючих хеш-функцій SHA. Він використовується в декількох широко використовуваних програмах і протоколах, включаючи протокол Secure Socket Layer (SSL).
- У 2005 році був знайдений метод для виявлення колізій для SHA-1 протягом практичного періоду часу, що робить довгострокову можливість використання SHA-1 сумнівною.
- Сімейство SHA-2 має ще чотири варіанти SHA: SHA-224, SHA-256, SHA-384 і SHA-512 в залежності від кількості біт в їх хеш-значення. Про хеш-функції SHA-2 поки не повідомлялося про успішні атаки.
- Хоча SHA-2 - сильна хеш-функція. Хоча суттєво відрізняється, його базовий дизайн все ще слід за дизайном SHA-1. Отже, NIST закликав до створення нових конкурентоспроможних конструкцій хеш-функцій.
- У жовтні 2012 року NIST вибрав алгоритм Keccak в якості нового стандарту SHA-3. Keccak пропонує безліч переваг, таких як ефективна продуктивність і хороша стійкість до атак.

- **RIPEMD:**

- RIPEND - це абревіатура для дайджесту повідомлення оцінки примітивів RACE. Цей набір хеш-функцій був розроблений відкритим дослідним спільнотою і широко відомий як сімейство європейських хеш-функцій.
- У комплект входять RIPEND, RIPEMD-128 і RIPEMD-160. Також існують 256 і 320-бітові версії цього алгоритму.
- Оригінальний RIPEMD (128 біт) заснований на принципах проектування, використовуваних в MD4 і забезпечує сумнівну безпеку. 128-розрядної версії RIPEMD прийшла в якості швидкої заміни для усунення вразливостей в оригінальній версії RIPEMD.
- RIPEMD-160 є покращеною версією і найбільш широко використовуваної версією в сімействі. 256 і 320-бітові версії знижують ймовірність випадкового зіткнення, але не мають більш високого рівня безпеки в порівнянні з RIPEMD-128 і RIPEMD-160 відповідно.

- **Whirlpool:**

o Це 512-бітна хеш-функція. Його взято з модифікованої версії Advanced Encryption Standard (AES). Одним з дизайнерів був Вінсент Раймо, один з творців AES.

o Три версії Whirlpool були випущені; а саме, WHIRLPOOL-0, WHIRLPOOL-T і WHIRLPOOL.

o Whirlpool працює з повідомленнями довжиною менше 2256 біт і видає дайджест повідомлення з 512 біт. Конструкція цієї хеш-функції дуже відрізняється від MD5 і SHA-1, що робить її несприйнятливою до типам атак, які були успішними для цих хешів.

- **Tiger:**

o Tiger - розроблені Россом Андерсоном і Елі Біхамом, Tiger розроблений для забезпечення безпеки, ефективної роботи на 64-бітних процесорах і простої заміни MD4, MD5, SHA і SHA-1 в інших додатках.

o Tiger/192 виробляє 192-бітний вихід і сумісний з 64-бітними архітектурою Tiger/128 і Tiger/160 створюють хеш довжиною 128 і 160 біт відповідно, щоб забезпечити сумісність з іншими функціями хеширования, згаданими вище.

- **eD2k**:

o eD2k названий для мережі EDonkey2000 (eD2K), хеш eD2k є кореневим хешем списку хеш-файлів MD4 даного файлу.

o Кореневої хеш використовується в тимчасових мережах передачі файлів, де файл розбивається на шматки; кожен блок має свій власний хеш MD4, пов'язаний з ним, і сервер підтримує файл, який містить список хеш всіх блоків. Кореневої хеш - це хеш файлу списку хешів.

- **SM3:**

o SM3 - це 256-бітна хеш-функція, яка працює з 512-бітними вхідними блоками.

o Будучи частиною китайського національного стандарту, SM3 випущений Державним управлінням криптографії Китаю як GM/T0004-2012:

алгоритм криптографічного хешування SM3 (2012) і GB/T32905-2016: методи захисту інформації - алгоритм криптографічного хешування SM3 (2016)).

Існує два безпосередніх застосування хеш-функції на основі її криптографічних властивостей [25]:

- забезпечують захист для зберігання паролів;
- перевірка цілісності даних.

Замість того, щоб зберігати пароль у відкритому вигляді, в основному всі процеси входу в систему зберігають хеш-значення паролів в файлі. Файл паролів складається з таблиці пар в формі (код користувача, h (P)) (Рис 2.3).

Зловмисник може бачити тільки хеші паролів, навіть якщо він отримав доступ до паролю. Він не може увійти в систему з використанням хеш-коду і не може отримати пароль з хеш-значення, оскільки хеш-функція має властивість опору до зображення.

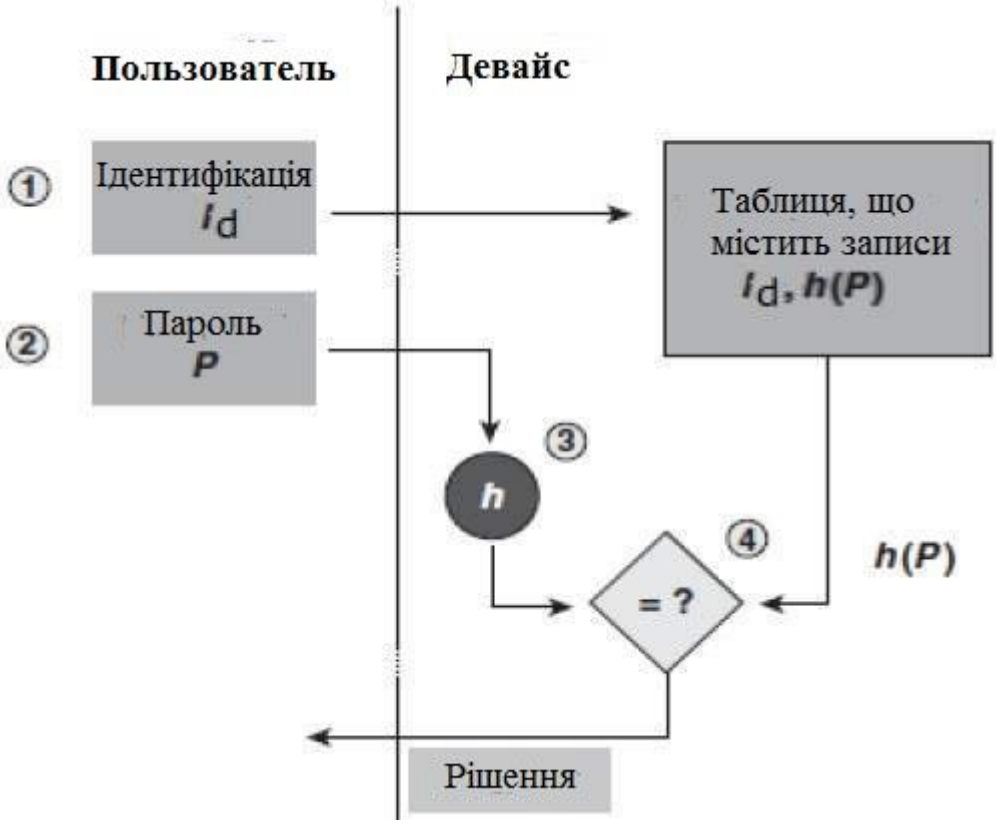


Рис. 2.3 Зберігання хеш-значення паролів

Перевірка цілісності даних є найбільш поширеним застосуванням хеш-функцій. Він використовується для генерації контрольних сум в файлах даних. Ця програма надає користувачеві гарантію правильності даних. Перевірка цілісності допомагає користувачеві знайти будь-які зміни, внесені в вихідний файл (рис 2.4).

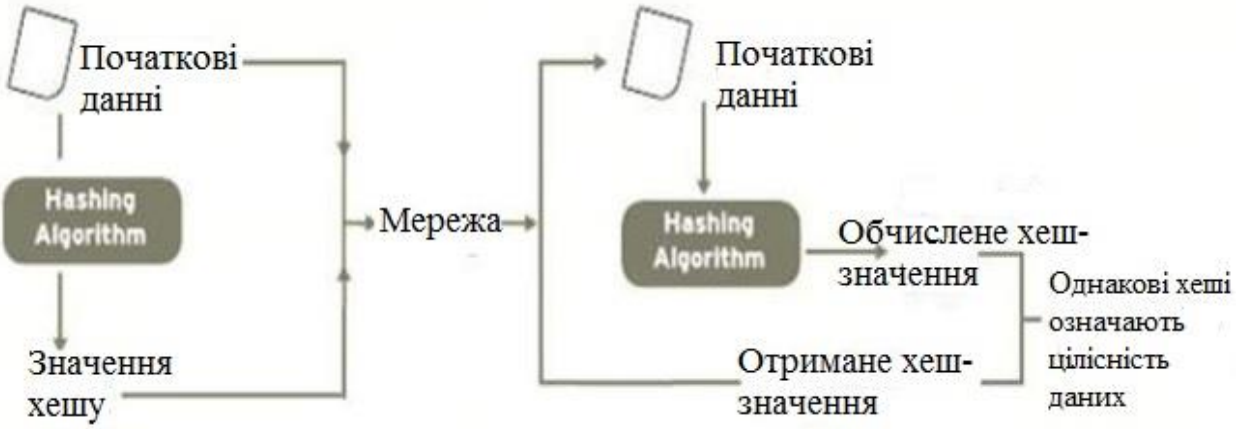


Рис. 2.4 Перевірка цілісності даних

Це, однак, не дає ніяких гарантій щодо оригінальності. Зловмисник, замість того, щоб змінювати дані файлу, може змінити весь файл і обчислити всі разом новий хеш і відправити його одержувачу. Ця програма перевірки цілісності корисно, тільки якщо користувач впевнений в оригінальності файлу.

Ще один засіб контролю цілісності інформації - **цифровий підпис**. Цифровий підпис - це криптографічне значення, яке розраховується на основі даних і секретного ключа, відомого тільки підписувачу.

У реальному світі одержувач повідомлення потребує гарантії того, що повідомлення належить відправнику, і він не повинен мати можливість заперечувати походження цих слів. Ця вимога дуже важливо в бізнес-додатках, так як ймовірність виникнення спору з приводу обміну даними дуже висока.

Як згадувалося раніше, схема цифрового підпису заснована на криптографії з відкритим ключем.

Наступні пункти детально пояснюють весь процес (рис.2.5):

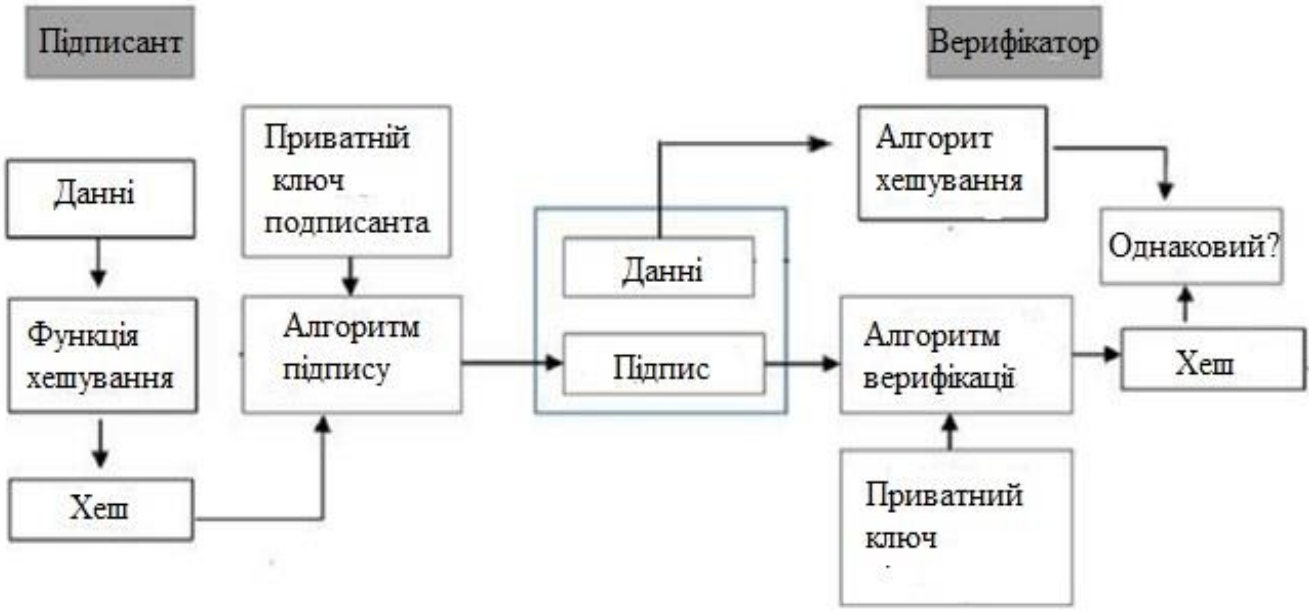


Рис. 2.5 Алгоритм цифрового підпису

- У кожної людини, що приймає цю схему, є пара відкритий-закритий ключ.
- Зазвичай пари ключів, які використовуються для шифрування/ дешифрування і підписи / перевірки, різні. Секретний ключ для підпису, називається ключем підпису, а відкритий ключ - ключем перевірки.
- Людина, яка підписує передає дані в хеш-функцію і генерує хеш-дані.
- Хеш-значення і ключ підпису потім передаються в алгоритм підпису, який створює цифровий підпис для даного хеша. Підпис додається до даних, а потім обидва відправляються верифікатору.
- Верифікатор подає цифровий підпис і ключ перевірки в алгоритм перевірки. Алгоритм перевірки дає деяке значення в якості висновку.
- Верифікатор також виконує ту ж хеш-функцію для отриманих даних, щоб згенерувати хеш-значення.
- Для перевірки порівнюються це значення хеш-функції і вихідні дані алгоритму перевірки. На підставі результату порівняння верифікатор вирішує, чи є цифровий підпис дійсної.

Оскільки цифровий підпис створюється «закритим» ключем підписанта, і ніхто інший не може мати цей ключ; підписувач не може відмовитися від підпису даних в майбутньому.

Слід зазначити, що замість підпису даних безпосередньо за допомогою алгоритму підпису зазвичай створюється хеш даних. Оскільки хеш даних є унікальним поданням даних, досить підписати хеш замість даних. Найбільш важливою причиною використання хешу замість даних безпосередньо для підписання є ефективність схеми.

Крім можливості забезпечити відмову від повідомлення, цифровий підпис також забезпечує аутентифікацію повідомлення і цілісність даних:

- Аутентифікація повідомлення - коли верифікатор перевіряє цифровий підпис, використовуючи відкритий ключ відправника, він упевнений, що підпис був створений тільки відправником, який володіє відповідним секретним закритим ключем, і ніким іншим.

- Цілісність даних. Якщо зловмисник має доступ до даних і змінює їх, перевірка цифрового підпису на стороні одержувача не виконується. Хеш змінених даних і вихідні дані, надані алгоритмом перевірки, не збігатимуться. Отже, одержувач може безпечно відхилити повідомлення, припускаючи, що цілісність даних була порушена.

- Безвідмовність - оскільки передбачається, що тільки підписала, знає ключ підпису, він може створювати унікальну підпис тільки для даних. Таким чином, одержувач може представити дані і цифровий підпис третій стороні як доказ, якщо в майбутньому виникне суперечка.

Додавши шифрування з відкритим ключем до схеми цифрового підпису, ми можемо створити криптосистему, яка може забезпечити чотири основних елементи безпеки, а саме - конфіденційність, аутентифікація, цілісність і відсутність анулювання.

Один із сучасних методів контролю цілісності є стандарт **BelT**. BelT - державний стандарт симетричного шифрування і контролю цілісності Республіки Білорусь [23]. Повна назва стандарту - СТБ 34.101.31-2007

«Інформаційні технології та безпека. Криптографічні алгоритми шифрування і контролю цілісності ». Прийнято в якості попереднього стандарту в 2007 році. Введено в дію в якості остаточного стандарту в 2011 році. Використовується для захисту з'єднань між клієнтом і сервером в мережі Інтернет.

BelT - блоковий шифр з 256-бітовим ключем і 8 циклами криптоперетворень, який оперує з 128-бітними словами. Криптографічні алгоритми стандарту побудовані на основі базових режимів шифрування блоків даних. Всі алгоритми стандарту діляться на 8 груп:

- алгоритми шифрування в режимі простої заміни;
- алгоритми шифрування в режимі зчеплення блоків;
- алгоритми шифрування в режимі гамування зі зворотним зв'язком;
- алгоритми шифрування в режимі лічильника;

Перші чотири групи призначені для забезпечення безпечного обміну повідомленнями. Кожна група включає алгоритм шифрування і алгоритм розшифрування на секретному ключі. Сторони, які мають загальним ключем, можуть організувати обмін повідомленнями шляхом їх шифрування перед відправкою і розшифрування після отримання. У режимах простої заміни та зчеплення блоків шифруються повідомлення, які містять хоча б один блок, а в режимах гамування зі зворотним зв'язком і лічильника - повідомлення довільної довжини.

Алгоритм вироблення імітовставки. Призначений для контролю цілісності повідомлень за допомогою імітовставок - контрольних слів, які визначаються з використанням ключа. Сторони, які мають загальним ключем, можуть організувати контроль цілісності при обміні повідомленнями шляхом додавання до них імітовставок при відправці і перевірки імітовставок при отриманні. Перевірка імітовставок додатково дозволяє стороні-одержувачу переконатися в тому, що сторона-відправник знає ключ, т. б. дозволяє перевірити справжність повідомлень. Цей алгоритм був детально описаний у розділі 1.3 цієї роботи.

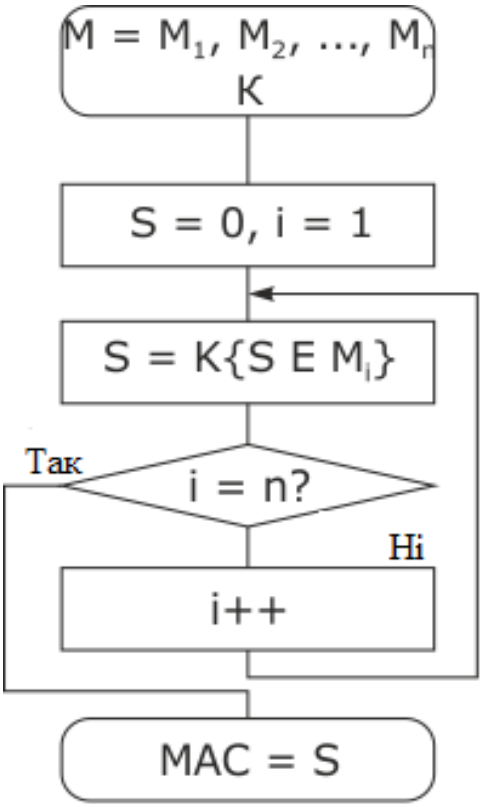
Алгоритми одночасного шифрування та імітозахисту даних. Вихідне повідомлення задається двома частинами: відкритої і критичної. Алгоритми захисту призначені для контролю цілісності обох частин і забезпечення конфіденційності критичній частині. При установці захисту обчислюється імітовставка всього повідомлення і зашифрована його критична частина. При знятті захисту імітовставка перевіряється і, якщо перевірка пройшла успішно, критична частина розшифровується.

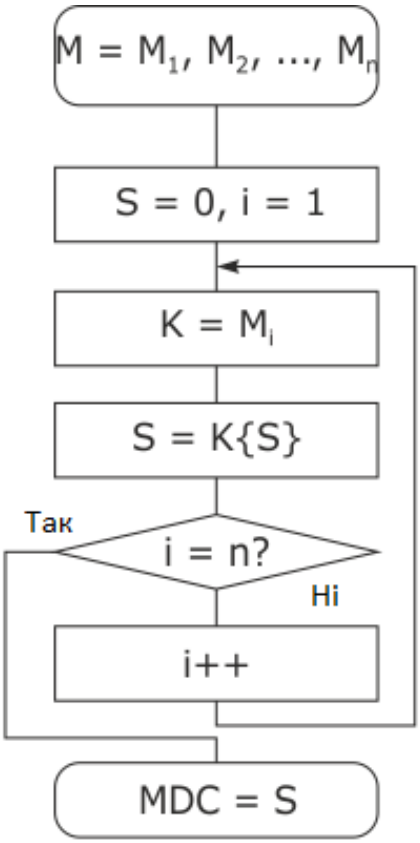
Алгоритми одночасного шифрування та імітозахисту ключа. Довжина повідомлення, що захищається, повинна бути відразу відома, ці алгоритми рекомендується застосовувати для захисту ключів. Ключ, що захищається, супроводжується відкритим заголовком, який містить відкриті атрибути ключа і одночасно є контрольним значенням при перевірці цілісності. Можуть використовуватися фіксовані постійні заголовки, які служать тільки для контролю цілісності. При установці захисту ключ шифрується разом зі своїм заголовком і формується слово, яке є одночасно захищеним ключем і імітовставкою ключа. При знятті захисту виконується зворотне перетворення і розшифрований заголовок порівнюється з контрольним.

Алгоритм хешування. Алгоритм призначений для обчислення хеш-значень - контрольних слів, які визначаються без використання ключа. Сторони можуть організувати контроль цілісності повідомлень шляхом порівняння їх хеш-значень з достовірними контрольними хеш-значеннями. Зміна повідомлення з високою ймовірністю призводить до зміни відповідного хеш-значення і тому хеш-значення можуть використовуватися замість самих повідомлень, наприклад в системах електронного цифрового підпису.

Можна виділити два основних криптографічних підходи до вирішення завдання захисту інформації від несанкціонованих змін даних (рис. 2.6 - 2.7.):

- формування (в режимах CBC або CFB) за допомогою функції шифрування E блочного шіфрокоду аутентифікації повідомлень MAC (Message Authentication Code);
- формування за допомогою незворотної функції стиснення інформації





- Рис. 2.1.6 MAC

Рис. 2.1.7 MDC

(хеш-функції) коду виявлення маніпуляцій з даними MDC (Manipulation Detection Code);

MAC. Для перевірки цілісності до повідомлення засобами відправника додається MAC-підпис, як результат хеш-функції від перетворення зміста повідомлення та криптографічного ключа. Засобами отримувача виконується аналогічне перетворення такою самою хеш-функцією зміста повідомлення та криптографічного ключа. Після цього отриманий MAC-підпис та MAC-підпис, згенерований для перевірки, порівнюються. Якщо обидва MAC-підписи збігаються, це підтверджує цілісність повідомлення та ідентичність відправника. Відмінність MAC-підпису від електронного цифрового підпису полягає в тому, що як криптографічний ключ використовується однаковий набір даних, як відправником, так й одержувачем. У разі використання електронного цифрового підпису, відправник та одержувач для

криптографічного перетворення даних використовують власні особисті ключі, що не збігаються.

Коди виявлення маніпуляції (MDC) визначаються як клас алгоритмів контрольної суми, які можуть виявляти як випадкові, так і шкідливі зміни електронного повідомлення або документа. Хоча результат MDC повинен бути захищений шифруванням, щоб не дати зловмисникові успішно замінити свій власний код виявлення маніпуляції (MDC) разом зі зміненим текстом, алгоритми MDC не вимагають використання секретної інформації, такої як криптографічний ключ. Отже, такі методи дуже корисні для забезпечення можливості здійснення шифрування і аутентифікації повідомлень на різних рівнях протоколів в системі зв'язку без труднощів управління ключами, а також при реалізації схем цифрового підпису.

Більш детально методи MAC та MDC розглянуто у наступних розділах.

У криптографії режим роботи блочного шифру - це алгоритм, який використовує блоковий шифр для забезпечення інформаційної безпеки, такий як конфіденційність або автентичність. Блоковий шифр сам по собі підходить тільки для безпечного криптографічного перетворення (шифрування або дешифрування) однієї групи бітів фіксованої довжини, званої блоком. Режим роботи описує, як багаторазово застосовувати одно блочну операцію шифру для безпечного перетворення об'ємів даних, більших, ніж блок. Деякі сучасні режими роботи ефективно поєднують конфіденційність і автентичність відбитку і відомі як режими шифрування з перевіркою достовірності.

AEAD-режими блочного шифрування (англ. Authenticated Encryption with Associated Data, «аутентифікувати шифрування з приєднаними даними») — клас блочних режимів шифрування, при якому частина повідомлення шифрується, частина залишається відкритою, і все повідомлення повністю аутентифікується [26]. В даний час запропоновано декілька AEAD-режимів шифрування: OCB mode (з версії OCB2), CCM mode, EAX mode, CWC mode, і Galois/Counter Mode. Останній з 2007 року є стандартом NIST.

AEAD - це варіант AE, який дозволяє одержувачеві перевіряти цілісність як зашифрованою, так і незашифрованной інформації в повідомленні. AEAD пов'язує пов'язані дані (AD) з зашифрованим текстом і з контекстом, де він повинен з'являтися, так що спроби «вирізати і вставити» дійсний зашифрований текст в інший контекст виявляються і відхиляються.

Типовий інтерфейс програмування для реалізації AE забезпечує наступні функції [26]:

- шифрування;

Введення: відкритий текст, ключ і, необов'язково, заголовок у вигляді відкритого тексту, який не буде зашифрований, але буде захищений захистом справжності.

Висновок: зашифрований текст і тег аутентифікації (код аутентифікації повідомлення).

- дешифрування.

Введення: зашифрований текст, ключ, тег аутентифікації і, необов'язково, заголовок (якщо використовується під час шифрування).

Висновок: відкритий текст або помилка, якщо тег аутентифікації не відповідає наданому зашифрованого тексту або заголовку.

Частина заголовка призначена для забезпечення автентичності та захисту цілісності для метаданих мережі або сховища, для яких конфіденційність не потрібно, але автентичність бажана.

Підходи до аутентифікованим шифрування:

Encrypt-then-mac (EtM) (рис. 2.8). Відкритий текст спочатку шифрується, потім на основі отриманого зашифрованого тексту створюється MAC. Зашифрований текст і його MAC відправляються разом. Використовується, наприклад, в IPsec. Стандартний метод відповідно до ISO / IEC 19772: 2009.

Encrypt-and-MAC (E&M) (рис. 2.9). MAC створюється на основі відкритого тексту, і відкритий текст шифрується без MAC. MAC відкритого тексту і зашифрований текст відправляються разом. Використовується,

Рис. 2.8 Алгоритм EtM

наприклад, в SSH. Навіть незважаючи на те, що підхід E&M не довів, що він сам по собі є сильно піддається обробці, можна застосувати деякі незначні модифікації SSH, щоб зробити його здатним піддатися обробці, не дивлячись на цей підхід.

Рис. 2.9 Алгоритм E&M

Mac-then-encrypt (MtE) (рис. 2.10). MAC створюється на основі відкритого тексту, потім відкритий текст і MAC разом шифруються для створення зашифрованого тексту на основі обох. Зашифрований текст (що містить зашифрований MAC) відправляється. Використовується, наприклад, в

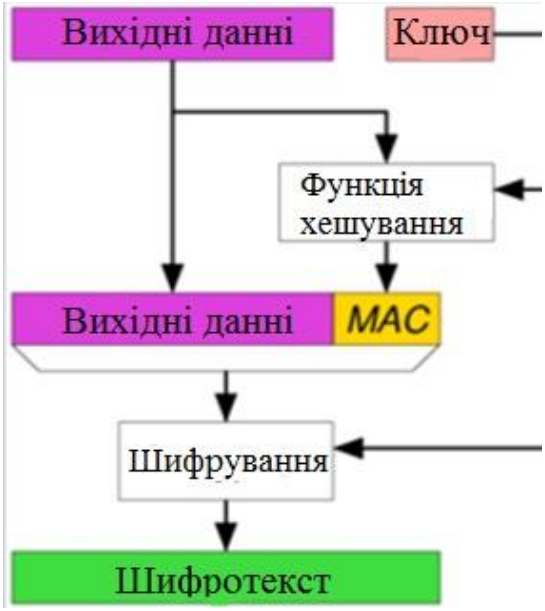


Рис. 2.10 Алгоритм MtE

SSL / TLS. Незважаючи на теоретичну безпеку, більш глибокий аналіз SSL / TLS змоделював захист як MAC-then-pad-then-encrypt, тобто відкритий текст спочатку доповнюється до розміру блоку функції шифрування. Помилки заповнення часто призводять до що виявляється помилок на стороні одержувача, що, в свою чергу, призводить до атак оракула заповнення.

Режим CCM (Лічильник з CBC-MAC) - це режим роботи криптографічних блокових шифрів. Це аутентіфіцированний алгоритм шифрування, розроблений для забезпечення як аутентифікації, так і конфіденційності. Режим CCM визначено тільки для блокових шифрів з довжиною блоку 128 біт.

Як випливає з назви, режим CCM поєднує в собі добре відомий CBC-MAC і добре відомий режим шифрування лічильника. Ці два примітиву застосовуються способом «аутентифицировать-потім-шифрувати», тобто CBC-MAC спочатку обчислюється в повідомленні для отримання тега t; повідомлення і тег потім шифруються в режимі лічильника. Одне розуміння ключа полягає в тому, що один і той же ключ шифрування може використовуватися для обох за умови, що значення лічильників, які

використовуються в шифруванні, не вступають в протиріччя з вектором (попередньої) ініціалізації, використовуваним при аутентифікації.

Режим EAX (encrypt-then-authenticate-then-translate) є режимом роботи для криптографічних блокових шифрів. Це алгоритм аутентифицированного шифрування з асоційованими даними (AEAD), розроблений для одночасної забезпечення як аутентифікації, так і конфіденційності повідомлення (аутентифицироваться шифрування) по двухпрохідному схемою: один прохід для досягнення конфіденційності і один для аутентифікації для кожного блоку.

EAX - це гнучка двопрохідні схема AEAD, що не використовує одноразові дані, без обмежень по використовуваному примітиву блочного шифру і за розміром блоку, а також підтримує повідомлення довільної довжини. Довжина тега аутентифікації може бути довільно збільшена до розміру блоку використовуваного шифру.

У криптографії, режим Галуа / Лічильник (**GCM**) є режим роботи для симетричних ключових криптографічних блокових шифрів широке поширення завдяки своїй ефективності. Пропускна здатність GCM для сучасних високошвидкісних каналів зв'язку може бути досягнута за допомогою недорогих апаратних ресурсів. Операція являє собою алгоритм шифрування з перевіркою достовірності, призначений для забезпечення як справжності (цілісності) даних, так і конфіденційності. GCM визначено для блокових шифрів з розміром блоку 128 біт. Код аутентифікації повідомлення Галуа (GMAC) - це варіант GCM тільки для аутентифікації, який може формувати інкрементний код аутентифікації повідомлення. І GCM, і GMAC можуть приймати вектори ініціалізації довільної довжини.

Різні режими роботи блочного шифру можуть мати істотно відрізняються характеристики продуктивності та ефективності навіть при використанні з одним і тим же блоковим шифром. GCM може в повній мірі використовувати переваги паралельної обробки, а реалізація GCM дозволяє ефективно використовувати конвеєр команд або апаратний конвеєр. Навпаки, режим

роботи ланцюжка блоків шифрування (CBC) призводить до значних затримок конвеєра, які знижують його ефективність і продуктивність.

Параллелізуемий режим з підтримкою цілісності (IAPM) - це режим роботи криптографічних блокових шифрів. Як випливає з назви, він допускає паралельний режим роботи для більш високої пропускної здатності. На момент створення IAPM був одним з перших режимів шифрування, що забезпечують як аутентифікацію, так і конфіденційність за один прохід. В більш ранніх проектах шифрування з перевіркою достовірності було потрібно два проходи: один для шифрування, а другий для обчислення MAC.

Режим OCB (Offset Codebook Mode) - режим аутентифицированного шифрування для криптографічних блокових шифрів. Режим OCB був розроблений для забезпечення аутентифікації і конфіденційності повідомлень. По суті, це схема інтеграції коду аутентифікації повідомлень (MAC) в роботу блочного шифру. Таким чином, режим OCB усуває необхідність використання двох систем: MAC для аутентифікації і шифрування для конфіденційності. Це призводить до зниження обчислювальних витрат в порівнянні з використанням окремих функцій шифрування і аутентифікації.

Існує три версії OCB: OCB1, OCB2 і OCB3. OCB1 був опублікований в 2001 році. OCB2 покращує OCB1, дозволяючи включати в повідомлення пов'язані дані (надаючи AEAD), тобто дані, які не зашифровані, але повинні бути автентифіковано, і новий метод для генерації послідовності зсувів. OCB2 був вперше опублікований в 2003 році і спочатку називався AEM (режим аутентифицированного шифрування або розширений режим шифрування). OCB3, опублікований в 2011 році, знову змінює спосіб обчислення зсувів і вносить незначні поліпшення продуктивності.

Зниження продуктивності OCB мінімально в порівнянні з класичними режимами без аутентифікації, такими як CBC. Для OCB потрібно одна операція блочного шифрування на блок зашифрованого і аутентифицированного повідомлення і одна операція блочного шифру на блок пов'язаних даних. В кінці процесу потрібно ще одна операція блочного шифрування.

Інші сучасні криптографічні методи контролю цілісності інформації:

- **Bitmessage** - це децентралізований, зашифрований, одноранговий, ненадійний протокол зв'язку, який може використовуватися однією людиною для відправки зашифрованих повідомлень іншій людині або декільком передплатникам. Bitmessage був задуманий розробником програмного забезпечення Джонатаном Уорреном, який засновував свій дизайн на децентралізованої цифровий валюті, біткойні. Bitmessage придбав репутацію поза досяжністю від несанкціонованого прослуховування телефонних розмов, проведеного Агентством національної безпеки (NSA), через децентралізованого характеру протоколу, і його шифрування важко зламати.

- **Kerberos** - система шифрування і аутентифікації з секретним ключем, призначена для аутентифікації запитів на мережеві ресурси в призначеному для користувача домені, а не для аутентифікації повідомлень. Kerberos також використовує надійний сторонній підхід; клієнт зв'язується з сервером Kerberos для отримання «облікових даних», щоб він міг звертатися до служб на сервері додатків. Kerberos V4 використовував DES для генерації ключів і шифрування повідомлень; Kerberos V5 використовує DES і інші схеми для генерації ключів. Після того, як клієнт і сервер використовували Kerberos для підтвердження своєї особи, вони також можуть зашифрувати всі свої комунікації, щоб забезпечити конфіденційність і цілісність даних в ході своєї діяльності.

- **Pretty Good Privacy (PGP)** - це програма шифрування, яка забезпечує криптографічний конфіденційність і аутентифікацію для передачі даних. PGP використовується для підпису, шифрування і дешифрування текстів, повідомлень електронної пошти, файлів, каталогів і цілих розділів диска, а також для підвищення безпеки повідомлень електронної пошти. Шифрування PGP використовує послідовну комбінацію хеширования, стиснення даних, криптографії з симетричним ключем і, нарешті, криптографії з відкритим ключем; кожен крок використовує один з декількох підтримуваних алгоритмів. Кожен відкритий ключ пов'язаний з ім'ям користувача або адресою електронної пошти.

- **Секрети моджахедів** - це програма шифрування для Microsoft Windows. Це було публічно запропоновано прихильникам «Аль-Каїди» в якості інструменту захисту конфіденційності їх електронних повідомлень. Автори програмного забезпечення є анонімними. Програма використовує п'ять фіналістіческіх алгоритмів AES, а саме MARS, RC6, Rijndael, Serpent і Twofish. Програмне забезпечення дозволяє користувачам шифрувати і дешифрувати текстові повідомлення і файли за допомогою різних методів шифрування. Це головним чином для того, щоб гарантувати, що будь-які сторони, що перехоплюють повідомлення під час передачі, наприклад, через електронну пошту в Інтернеті або мобільний телефон, не зможуть легко переглянути вміст повідомлення.

- **Signal Protocol** (раніше відомий як Протокол TextSecure) є не-федеративної криптографічний протокол, який може бути використаний для забезпечення шифрування кінця в кінець для відеодзвінків, а, і обмін миттєвими повідомленнями розмов (включаючи групові чати). Використовуючи комбінацію алгоритмів AES, ECC і HMAC, він пропонує такі функції, як конфіденційність, цілісність, аутентифікація, секретність пересилання / майбутнього і відмова від повідомлень. Сигнал особливо цікавий через його походження і широкого використання.

- **SASL** (англ. Simple Authentication and Security Layer - простий рівень аутентифікації і безпеки) - це фреймворк (каркас) для надання аутентифікації і захисту даних в протоколах на основі сполук. Він розділяє механізми аутентифікації від прикладних протоколів, в теорії дозволяючи будь-якому механізму аутентифікації, що підтримує SASL, бути використаним в будь-яких прикладних протоколах, які використовують SASL. Фреймворк також надає захисний шар даних. Для використання SASL протокол включає команду для ідентифікації і аутентифікації користувача на сервері і для опціональною захисту переговорів подальшої інтерактивності протоколу. Якщо це використовується в переговорах, то шар безпеки вставляється між протоколом і

з'єднанням. Вони також можуть забезпечити рівень захисту даних, що пропонує цілісність даних та послуги конфіденційності даних.

- **Telegram** - це облачна служба, яка обмінюється мгновенними повідомленнями та передає голосу за IP (VoIP), з кліентським програмним забезпеченням, доступним для всіх основних операційних системних комп'ютерів та мобільних пристроїв. Telegram дозволяє користувачеві обмінятися повідомленнями, фотографіями, відео та т. Д. Я гарантує скрізне шифрування із застосуванням протоколу MTProto. MTProto використовує 256-бітний AES, 2048-бітний RSA і обменними ключами Диффі-Хеллмана.
- **TrueCrypt** - набір інструментів, написаний криптографічне програмне забезпечення з відкритим вихідним кодом, яке можна використовувати для шифрування файлу, розділу або всього диска. Одна з найбільш цікавих функцій TrueCrypt - це правдоподібне заперечення прихованих томів або прихованих операційних систем. TrueCrypt підтримує окремі шифри AES, Serpent і Twofish. Крім того, є п'ять різних комбінацій каскадних алгоритмів: AES-Twofish, AES-Twofish-Serpent, Serpent-AES, Serpent-Twofish-AES і Twofish-Serpent. У криптографічних хеш - функція, доступна для використання в TrueCrypt є RIPEMD-160, SHA-512, і Whirlpool.
- **KASUMI** - це передбачуваний алгоритм конфіденційності і цілісності як для вмісту повідомлень, так і для даних сигналізації для нових систем мобільного зв'язку. За основу був узятий існуючий алгоритм MISTY1 і оптимізований для використання в стільникового зв'язку. KASUMI використовує 64-бітний розмір блоку і 128-бітний ключ о 8-раундової схемою Фейстеля. У кожному раунді використовується 128-бітний раундовий ключ, що складається з восьми 16-бітних подключей, отриманих з вихідного ключа за фіксованою процедурі генерації подключей.

2.2. Коди автентифікації повідомлень

2.2.1. MAC

Забезпечення цілісності повідомлення - це неможливість зміни повідомлення так, щоб одержувач цього не виявив. Під автентифікацією розуміється підтвердження того, що інформація отримана від законного джерела, і одержувачем є той, хто треба. Один із способів забезпечення цілісності - це обчислення MAC (Message Authentication Code). В даному випадку під MAC розуміється деякий автентифікатор, що є певним способом обчисленим блоком даних, за допомогою якого можна перевірити цілісність повідомлення. В деякій мірі симетричне шифрування всього повідомлення може виконувати функцію аутентифікації цього повідомлення. Але в такому випадку повідомлення повинно містити достатню надмірність, яка дозволяла б перевірити, що повідомлення не було змінено. Надмірність може бути у вигляді певним чином відформатованого повідомлення, тексту на конкретній мові і т.п. Якщо повідомлення допускає довільну послідовність бітів (наприклад, зашифрований ключ сесії), то симетричне шифрування всього повідомлення не може забезпечувати його цілісність, так як при дешифрування в будь-якому випадку вийде послідовність бітів, правильність якої перевірити не можна. Тому набагато частіше використовується криптографічно-створений невеликий блок даних фіксованого розміру, так званий автентифікатор або імітовставка, за допомогою якого перевіряється цілісність повідомлення. Цей блок даних може створюватися за допомогою секретного ключа, який поділяють відправник і одержувач. MAC обчислюється в той момент, коли відомо, що повідомлення коректно. Після цього MAC приєднується до повідомлення і передається разом з ним одержувачу. Одержувач обчислює MAC, використовуючи той же самий секретний ключ, і порівнює обчислене значення з отриманим. Якщо ці значення збігаються, то з великою часткою ймовірності можна вважати, що при пересиланні зміни повідомлення не відбулося.

Розглянемо властивості, якими повинна володіти функція MAC. Якщо довжина ключа, використовуваного при обчисленні MAC, дорівнює k, то за умови сильної функції MAC супротивникові потрібно виконати 2^k спроб для перебору всіх ключів. Якщо довжина значення, створюваного MAC, дорівнює n, то всього існує 2^n різних значень MAC.

Припустимо, що конфіденційності повідомлення немає, тобто опонент має доступ до відкритого повідомлення і відповідного йому значення MAC. Визначимо зусилля, необхідні опонентові для знаходження ключа MAC. Припустимо, що k> n, тобто довжина ключа більше довжини MAC. Тоді, знаючи M_1 і $MAC_1 = C_K (M_1)$, опонент може обчислити $MAC_1 = CK_i (M_1)$ для всіх можливих ключів K_i. При цьому, як мінімум, для одного з ключів буде отримано збіг $MAC_i = MAC_1$. Опонент вирахує 2^k значень MAC, тоді як при довжині MAC n бітів існує всього 2^n значень MAC. Ми припустили, що k> n, тобто $2^k > 2^n$. Таким чином, правильне значення MAC буде отримано для кількох значень ключів. В середньому збіг матиме місце для $2^k / 2^n = 2^{(k-n)}$ ключів. Тому для обчислення єдиного ключа опоненту потрібно знати кілька пар повідомлень і відповідний йому MAC.

Таким чином, простий перебір всіх ключів вимагає не менше, а більше зусиль, ніж пошук ключа симетричного шифрування тієї ж довжини.

Функція обчислення MAC повинна мати наступні властивості:

- Повинно бути важко, знаючи M і C_K (M), знайти повідомлення M ', таке, що $C_K (M) = C_K (M')$.
- Значення C_K (M) повинні бути рівномірно розподіленими в тому сенсі, що для будь-яких повідомлень M і M' ймовірність того, що $C_K (M) = C_K (M')$, повинна дорівнювати 2^{-n}, де n - довжина значення MAC.

Для обчислення MAC може використовуватися алгоритм симетричного шифрування (наприклад, DES) в режимі CBC і нульовий ініціалізації вектор. У цьому випадку повідомлення представляється у вигляді послідовності блоків, довжина яких дорівнює довжині блоку алгоритму шифрування. При

необхідності останній блок доповнюється праворуч нулями, щоб вийшов блок потрібної довжини. Обчислення MAC відбувається за формулою [2.1]:

$$MAC_1 = E_k [P_1]$$
$$MAC_2 = E_k [P_2 \oplus MAC_1]$$
$$\dots \qquad (2.1)$$
$$MAC_N = E_k [P_N \oplus MAC_{N-1}]$$
$$MAC = MAC_N$$

Іншим способом забезпечення цілісності є використання хеш-функції. Хеш-код приєднується до повідомлення в той момент, коли відомо, що повідомлення коректно. Одержувач перевіряє цілісність повідомлення обчисленням хеш-коду отриманого повідомлення і порівнянням його з отриманим хеш-кодом, який повинен бути переданий безпечним способом. Одним з таких безпечних способів може бути шифрування хеш-коду закритим ключем відправника, тобто створення підпису. Можливо також шифрування отриманого хеш-коду алгоритмом симетричного шифрування, якщо відправник і одержувач мають загальний ключ симетричного шифрування.

Формування коду MAC з використанням функції шифрування блочного шифру офіційно закріплено у ГОСТ 28147 2009 - класичний приклад коду імітовставки для контролю цілісності, якій був розглянутий у розділі 1.

Мета противника алгоритму MAC полягає в наступному:

Атакувати MAC: без попереднього знання ключа k, обчислити нову пару текст-MAC $(x, h_k(x))$ для деякого тексту $x \neq x_i$, заданої однією або декількома парами $(x_i, h_k(x_i))$.

Опір обчислень тут має має визначатись, чи передаються тексти x_i, для яких є відповідні MAC, супротивнику, або може бути вільно обраний противником. Аналогічно ситуаціям із схемами підписання, таким чином існують такі сценарії атаки для MAC, для супротивників, що мають більші переваги:

1) відома текстова атака. Доступна одна або кілька пар текстових MAC $(x_i, h_k(x_i))$;

2) атака обраного тексту. Для x_i доступні одна або кілька пар текстових MAC (x_i , $h_k(x_i)$) обраний противником;

3) адаптивна атака обраного тексту. Противник може обрати x_i, як описано вище, що дозволяє послідовні вибори базуватися на результатах попередніх запитів.

Як сертифікаційний контрольно-пропускний пункт, MAC повинні протистояти адаптивній атаці обраного тексту незалежно від того, чи може така атака монтуватися в конкретному середовищі. Деякі практичні програми можуть обмежувати кількість взаємодій, дозволених протягом певного періоду часу, або можуть бути розроблені таким чином, щоб обчислити MAC лише для входів, створених у самій програмі; інші можуть дозволити доступ до необмеженої кількості пар текстових MAC або дозволити перевірку MAC необмеженої кількості повідомлень та прийняти будь-які з правильним MAC для подальшої обробки.

Коли підробка MAC можлива (маючи на увазі, що алгоритм MAC технічно зазнав поразки), тяжкість практичних наслідків може відрізнятися залежно від ступеня контролю супротивника над значенням x, за яке може бути підроблений MAC. Цей ступінь диференціюється наступною класифікацією підробок:

1. Вибіркова підробка - атаки, за допомогою яких супротивник може створити нову пару текст-MAC для тексту на свій вибір (або, можливо, частково під його контролем). Зауважте, що тут вибране значення - це текст, для якого підроблений MAC, тоді як в атаці обраного тексту вибране значення - це текст текстової пари MAC, який використовується в аналітичних цілях (наприклад, для підробки MAC на окремий текст).

2. Екзистенціальна підробка - атаки, за допомогою яких супротивник може створити нову пару текст-MAC, але не контролюючи значення цього тексту.

Ключове відновлення ключа MAC саме по собі є найбільш згубною атакою і тривіально дозволяє вибіркове підроблення. Підробка MAC дозволяє

противнику мати підроблений текст, прийнятий як автентичний. Наслідки можуть бути серйозними навіть у екзистенційному випадку. Класичний приклад - це заміна грошової суми, яка, як відомо, є невеликою на число, випадковим чином розподілене між 0 і 232 - 1. З цієї причини повідомлення, цілісність чи достовірність яких потрібно перевірити, часто обмежуються заздалегідь визначеною структурою або висока ступінь надмірності надмірності, намагаючись уникнути значущих атак.

Код аутентифікації повідомлень може формуватися в режимах CBC або CFB, що забезпечують залежність останнього блоку шифротекста від всіх блоків відкритого тексту. У разі використання перетворення Ek для вироблення контрольного коду вимоги до нього дещо відрізняються від традиційних, використовуваних при шифруванні:

- по-перше, не потрібно властивість оборотності;
- по-друге, його криптостійкість може бути знижена за рахунок зменшення до 16 раундів.

Дійсно, в разі вироблення коду MAC перетворення завжди виконується в одну сторону, при цьому в розпорядженні противника є тільки контрольний код, який залежить від всіх блоків відкритого тексту, в той час як при зашифруванні у нього є набір блоків шифротекста, отриманих з використанням одного секретного ключа.

Розглянемо приклад. Є дві людини - Аліса (А) і Боб (Б), які обмінюються повідомленням М. Щоб гарантувати цілісність повідомлення, справжність початкового повідомлення і те, що творець повідомлення - Аліса, а не хтось інший, потрібно зробити код встановлення автентичності повідомлення (Message Authentication Code - MAC). MAC включає секретність між Алісою і Бобом, наприклад секретний ключ, яким Єва не володіє. Рис 2.11 ілюструє ідею.

Аліса використовує хеш-функцію, щоб створити MAC по об'єднанню ключа і повідомлення - h (K | M). Вона передає повідомлення і MAC Бобу по ненадійному каналу. Боб відокремлює повідомлення від MAC і потім робить

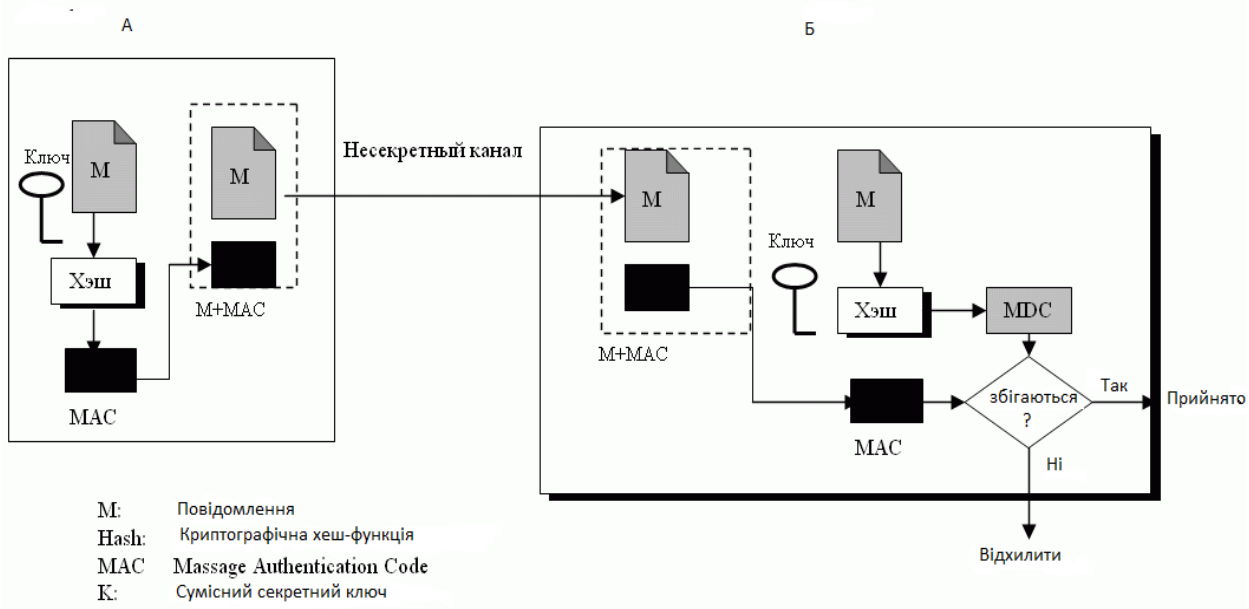


Рис. 2.11 Код автентифікації повідомленя

новий MAC з об'єднання повідомлення і ключа засекречування. Потім Боб порівнює недавно створений MAC з отриманим. Якщо обидва ці MAC збігаються, то повідомлення справжнє і не було змінено противником.

Немає необхідності використовувати два канали. І повідомлення, і MAC можна передати по одному, хоча б і самому ненадійному каналу. Єва може бачити повідомлення, але вона не може створити нове повідомлення, щоб підробити вихідне, тому що Єва не володіє ключем засекречування між Алісою і Бобом. Вона не здатна створити такий же MAC, як це зробила Аліса.

2.2.2 CBC-MAC

CBC-MAC - це алгоритм MAC, заснований на режимі Cipher Block Chaining (CBC) блочного шифру. У режимі CBC попередній зашифрований текст передається в блок відкритого тексту перед застосуванням блочного шифру (рис. 2.12). Значення MAC виходить з останнього блоку зашифрованого тексту. CBC-MAC є одним з найстаріших і популярних алгоритмів MAC.

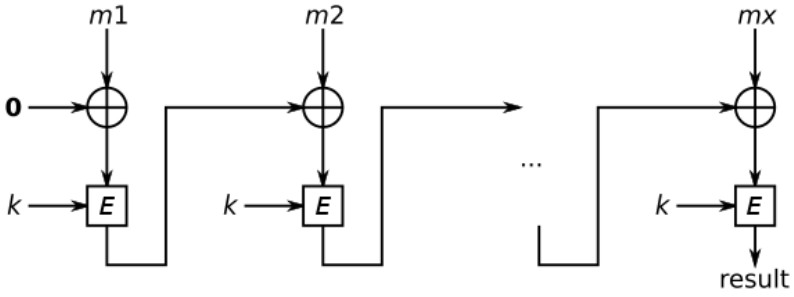


Рис. 2.12 Код автентифікації повідомленя

Це метод перетворення блокового шифру в функцію обчислення MAC. Ключ K при цьому використовується як ключ шифрування. Основна ідея алгоритму CBC-MAC полягає в тому, щоб зашифрувати повідомлення m за допомогою блочного шифру, використовуючи режим CBC, і потім відкинути всі блоки шифрованого тексту крім останнього. Щоб повідомити, що складається з блоків $P_1,\ldots,P_k$, значення MAC обчислюється таким чином[2.2]:

$$H_0 := IV, \qquad (2.2)$$

$$H_i := E_K(P_i \oplus H_{i-1}),$$

$$MAC := H_k.$$

Іноді результатом функції CBC-MAC вважають половину останнього блоку, але це вже залежить від конкретної ситуації.

Класичне визначення алгоритму CBC-MAC вимагає, щоб значення вектора ініціалізації було фіксованим і дорівнювало нулю. Саме таке визначення часто зустрічається в підручниках. Взагалі кажучи, функцію CBC-MAC можна використовувати і з будь-якими іншими типами вектора ініціалізації, які обговорювалися під час розгляду режиму CBC, але очевидних переваг це не дає. Ніколи не використовуйте один і той же ключ для шифрування і аутентифікації (хіба що ви твердо впевнені в необхідності цього кроку). Особливо небезпечно використовувати шифрування в режимі CBC і аутентифікацію CBC-MAC з одним і тим же ключем. У цьому випадку значення MAC буде просто одно останньому блоку шифрованого тексту.

Алгоритм CBC-MAC не дуже простий в застосуванні. Проте в загальному випадку він вважається безпечним, якщо відповідний блоковий шифр також

безпечний. Існує ряд атак на основі колізій, які скорочують рівень безпеки CBC-MAC до половини розміру блоку [12]. Ось простий приклад однієї з таких атак.

Нехай M - це функція CBC-MAC. Якщо відомо, що M (a) = M (b), значить, M (a ‖ c) = M (b ‖ c). Це пояснюється специфікою алгоритму CBC-MAC. Проілюструємо дану властивість на простому прикладі, коли складається з одного блоку тексту. Ми знаємо, що [2.3]:

$$M(a \| c) = EK(c \oplus M(a)), \quad (2.3)$$

$$M(b \| c) = EK(c \oplus M(b)).$$

Оскільки M (a) = M (b), праві частини цих виразів рівні.

Така атака виконується в два етапи. На першому етапі зловмисник збирає значення MAC для великої кількості повідомлень, поки не наткнеться на колізію. В цьому випадку у нього з'являться повідомлення a і b, для яких M (a) = M (b). Якщо тепер зловмиснику вдасться аутентифицировать у одержувача повідомлення a ‖ c, він може замінити його повідомленням b ‖ c, причому значення MAC залишиться колишнім. Одержувач перевірить значення MAC і прийме підроблене повідомлення.

Ця атака не є тривіальною, оскільки вона не спрацьовує для ідеальної функції обчислення MAC. Знайти колізію нескладно і для ідеальної функції. Проте, навіть коли у вас з'являться повідомлення a і b, для яких M (a) = M (b), ви не зможете застосувати їх, щоб підробити значення MAC для нового повідомлення, як при використанні алгоритму CBC-MAC.

Існує кілька теоретичних результатів, які показують, що при використанні конкретної моделі доказів алгоритм CBCMAC забезпечує 64-бітовий рівень безпеки, якщо розмір блоку дорівнює 128 біт . На жаль, цього недостатньо для забезпечення рівня безпеки, встановленого для наших систем. Ми б змогли використати CBC-MAC, якби мали блоковий шифр з 256-бітовим блоком.

Використовувати алгоритм CBC-MAC слід вкрай обережно. Функцію CBC-MAC можна застосовувати до самого повідомлення - це дозволяє

зловмисникові здійснювати досить прості атаки. Необхідно вчинити трохи інакше.

1. Створюємо рядок s шляхом конкатенації значень l і m, де l - це довжина повідомлення m, представленого в форматі фіксованої довжини.

2. Доповнюємо рядок s, щоб її довжина стала кратної довжині блоку.

3. Застосовуємо функцію CBC-MAC до доповненої рядку s.

4. Результатом застосування функції CBC-MAC вважаємо останній блок шифрованого тексту або частина цього блоку. Не використовуємо в якості MAC проміжні значення.

Алгоритм CBC-MAC має одну суттєву перевагу: він використовує той же тип обчислень, що і режими роботи блокових шифрів. До вмісту повідомлення застосовуються тільки дві функції - шифрування і обчислення MAC, а значить, є тільки дві критичні точки відносно швидкості роботи. Використання в цих точках одних і тих же елементарних функцій дозволить підвищити ефективність реалізацій, особливо апаратних.

2.2.3 Код HMAC

Існує варіант побудови коду MAC на основі використання секретного ключа і функції хешування, при якому хешування піддається як результат об'єднання секретного ключа і вихідного повідомлення. Тому, як і в класичному випадку, у противника, який знає ключ, відсутня можливість обчислити контрольний код.

Для підвищення безпеки подібного алгоритму отримання MAC, створена схема вкладеного (nested) MAC, в якій хешування виконується двічі (рис. 2.5). У стандарті NIST FIPS 198 вкладений MAC названий HMAC (hashed MAC).

У документі RFC 2104 представлений наступний список цілей, переслідуваних при розробці алгоритму HMAC:

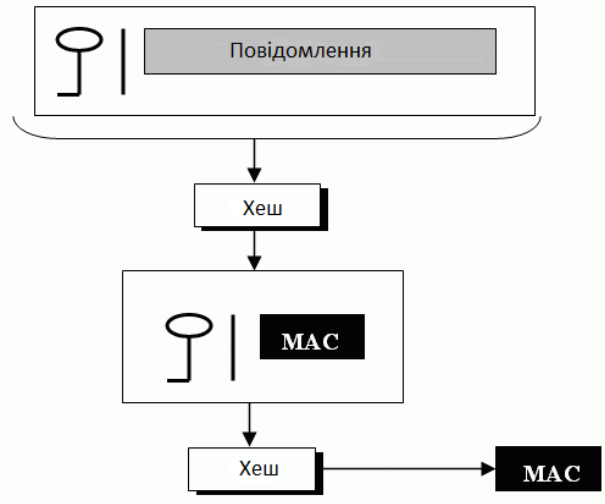


Рис. 1.11 Вкладений MAC

- Можливість використання без модифікацій вже наявних функцій хешування, зокрема, функцій хешування, для яких існують перевірені на практиці, відкриті і широко доступні програмні реалізації.
- Можливість легкої заміни вбудованої функції хешування більш швидкісними або більш захищеними функціями хеширования, якщо такі будуть потрібні і будуть знайдені.
- Збереження швидкості роботи алгоритму, близькою до швидкості роботи відповідної функції хешування, безміна значного погіршення показників швидкості.
- Можливість застосування ключів і простота звернення до них.
- Простота аналізу стійкості механізму аутентифікації при розумних припущеннях щодо використовуваної функції хешування. Остання з перерахованих вище цілей проекту фактично забезпечує основна перевага HMAC в порівнянні з іншими схемами, заснованими на використанні хешування. HMAC забезпечує гарантовану захищеність за умови, що вбудована функція хешування володіє необхідною криптографічного стійкістю.

Алгоритм формування HMAC має наступний вигляд (рис 2.14):

1. Повідомлення ділиться на N блоків, по b біт в кожному.

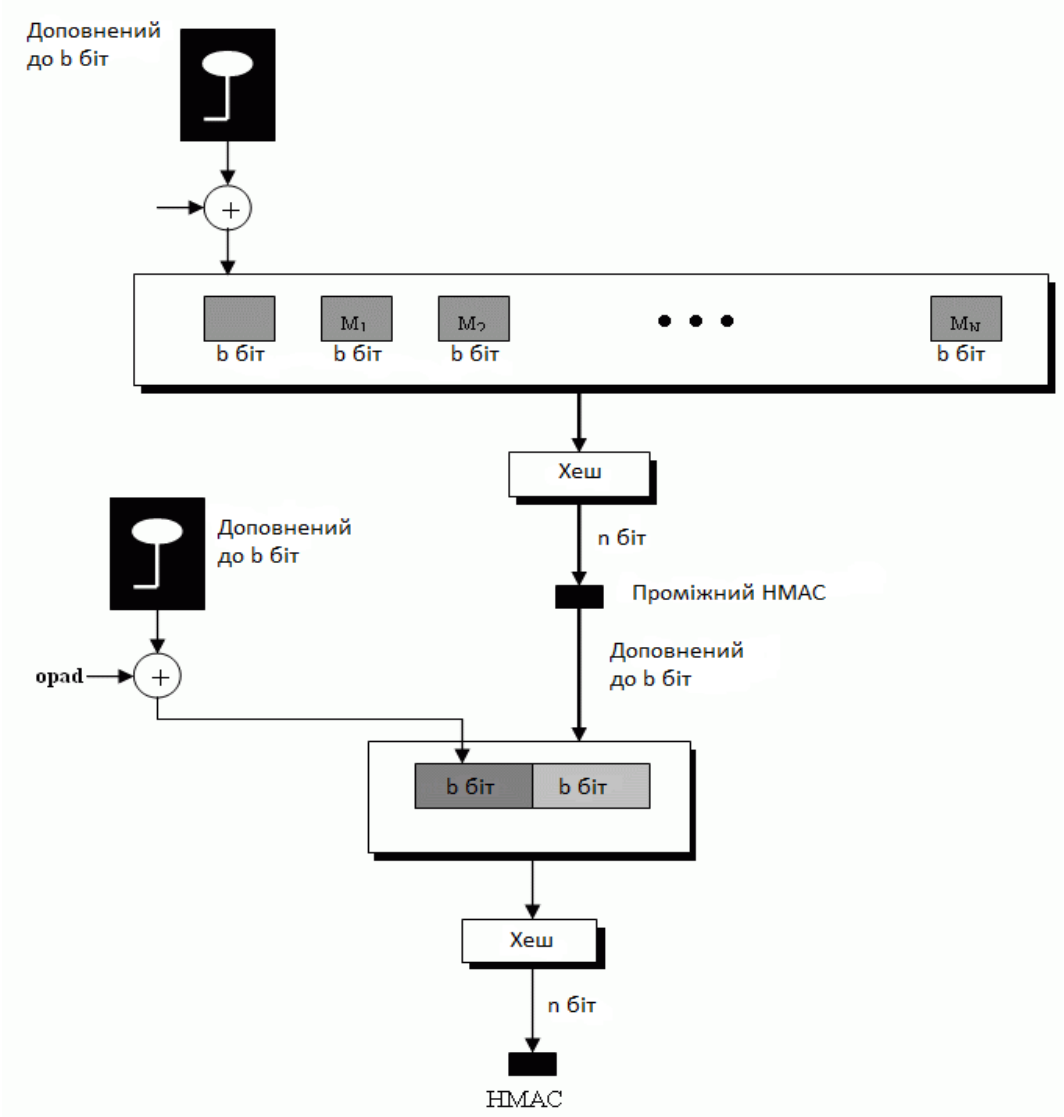


Рис. 2.14 Деталі HMAC

2. Секретний ключ доповнюється зліва нулями до отримання b-розрядної ключа. Рекомендується, щоб секретний ключ до операції доповнення мав довжину, більшу n, де n - розрядність HMAC.

3. Виконується операція XOR результат кроку 2 з константою ipad (input pud). Величина ipad суть b / 8 повторень послідовності 36h.

4. Отриманий блок додається зліва до N-блоковому повідомленням.

5. Результат кроку 4 хешіруется і створюється n-розрядний хеш-образ, званий Intermediate HMAC.

6. Результат кроку 5 доповнюється зліва нулями до отримання b-розрядної блоку.

7. Кроки 2 і 3 повторюються з константою opad (output pud). Величина opad суть b / 8 повторень послідовності 5Ch.

8. Результат кроку 7 додається зліва до блоку, отриманого на кроці 6.

9. Результат кроку 8 хешіруется з використанням тієї ж хеш-функції і створюється фінальний HMAC.

Криптографічний стійкість HMAC залежить від розміру використовуваного секретного ключа. Найпоширеніша атака на HMAC - це груба сила для розкриття секретного ключа. HMAC істотно менше схильні до колізій, ніж тільки їх основні алгоритми хешування.

Структура функції HMAC дозволяє уникнути атак з відновленням ключа і атак, які можуть бути проведені зловмисником в автономному режимі, без взаємодії з системою. Незважаючи на це, HMAC все ще обмежений n/2 двійкового рівнем безпеки. Це пояснюється наявністю атак, побудованих на парадоксі завдання про дні народження, які використовують внутрішні колізії ітеративних функцій хешування. Алгоритм HMAC гарантує, що для виконання таких атак буде потрібно $2^{n/2}$ взаємодій з атакується системою. Це виконати набагато складніше, ніж виконати $2^{n/2}$ обчислень на власному комп'ютері.

2.2.4 Код CMAC

У стандарті NIST FIPS 113 визначена схема формування коду автентифікації повідомлень, названа CMAC (або CBCMAC). OMAC1 еквівалентний CMAC. Офіційно існує два алгоритму OMAC (OMAC1 і OMAC2), які по суті однакові, за винятком невеликої зміни. Алгоритм безкоштовний для будь-якого використання, бо нічого не захищене ніякими патентами. Є блоковим шифром заснованого повідомлення коду аутентифікації алгоритму. Він може використовуватися для забезпечення гарантії достовірності і, отже, цілісності двійкових даних. Цей режим роботи усуває недоліки безпеки CBC-MAC (CBC-MAC безпечний тільки для повідомлень

фіксованої довжини). Ядром алгоритму CMAC є різновид CBC-MAC. Схема формування CMAC показана на рис. 2.15.

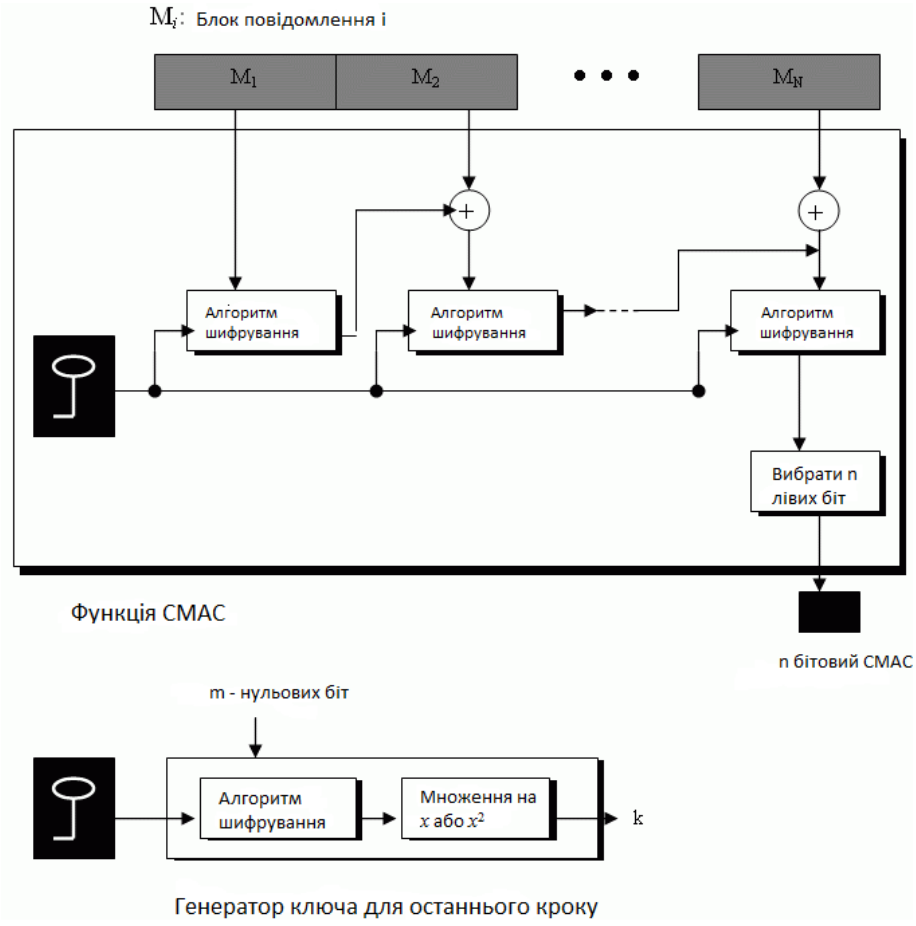


Рис. 2.15 Функція CMAC та генератор ключа для останнього кроку

Оригінальне повідомлення ділиться на N блоків по m розрядів у кожному. Якщо останній блок має довжину, меншу m, він доповнюється одиничним бітом і подальшими нульовими бітами до необхідної довжини. Нехай n - розрядність CMAC. Перший блок повідомлення M_1 шифрується на симетричному ключі K_{AB}, в результаті виходить m-розрядний блок шифротекста.

Цей блок порозрядно підсумовується по модулю 2 з наступним блоком повідомлення M_2, в результаті знову виходить m-розрядний блок шифротекста. Процес повторюється до тих пір, поки не буде зашифрований останній блок повідомлення M_N, після чого n лівих розрядів результату оголошуються кодом CMAC. На додаток до симетричного ключа K_{AB} для отримання CMAC на

останньому кроці шифрування використовується інший ключ k. Він формується в результаті шифрування на ключі K_{AB} блоку, що складається з m нульових біт, після чого результат множиться на x або x^2 в поле GF (2^m), якщо операції доповнення останнього блоку відповідно не проводилася або проводилася.

2.2.5 Код UMAC та VMAC

Код аутентифікації повідомлення на основі універсального хешування, або UMAC, є тип коду аутентифікації повідомлення (MAC), обчислений вибору хеш - функції з класу хеш - функцій відповідно до деяких секретних (випадковим) процесом і застосовуючи його до повідомлення. Отриманий дайджест або відбиток потім шифрується, щоб приховати ідентичність використовуваної хеш-функції. Як і з будь-яким MAC, він може бути використаний одночасно перевіряти як цілісність даних і справжність у вигляді повідомлення.

Швидкість роботи алгоритму UMAC може на цілий порядок перевищувати швидкість роботи HMAC. Крім того, для UMAC існують докази його безпеки. Схожий варіант UMAC, оптимізований для 64-бітних архітектур, надано VMAC.

Швидка «універсальна» функція використовується, для того, щоб хешувати вхідне повідомлення M в коротку рядок. До цієї рядку потім застосовується функція XOR з псевдовипадковим значенням, в результаті чого ми отримуємо тег UMAC:

$$Tag= H_{K3} (M) \oplus F_{K2} (N_{once}), \quad (2.4)$$

де K1 і K2 - секретні випадкові ключі, які мають одержувач і відправник.

Звідси видно, що безпека UMAC залежить від того, яким випадковим способом відправник і одержувач вибрали таємну хеш-функцію і псевдослучайную послідовність. При цьому значення Nonce змінюється кожен такт. Через використання Nonce, приймач і передавач повинні знати час

відправки повідомлення і принцип створення значення Nonce. Замість цього можна використовувати в якості Nonce будь-яке інше неповторяющееся значення, наприклад порядковий номер повідомлення. Одноразовий номер не повинен зберігатися в секреті, але необхідно подбати про те, щоб протягом терміну служби ключа різні одноразові номера використовувалися з кожним повідомленням.

VMAC використовує функцію, звану VHASH, в якості ключової хеш-функції H і псевдослучайной функції F, реалізація за замовчуванням якої використовує блоковий шифр AES. VMAC допускає довжину міток будь-якого 64-бітного кратного до розміру блоку використовуваного блочного шифру. При використанні AES це означає, що VMAC може створювати 64- або 128-бітові теги, а UMAC 32-бітові.

MAC-адрес Вегмана-Картера і аналіз VMAC показують, що якщо «створити екземпляр» VMAC з дійсно випадковими ключами і педами, то ймовірність того, що зловмисник (навіть обчислювально необмежений) видасть правильний тег для повідомлень свого вибору, менше ніж $1/2^{60}$ або $1/2^{120}$, коли мітки мають довжину 64 або 128 біт відповідно. Коли зловмисник здійснює N спроб підробки, ймовірність правильного вибору одного або декількох тегів лінійно зростає до величини, яка менша $N/2^{60}$ або $N/2^{120}$. В прикладної реалізації VMAC, що використовує AES для виробництва ключів і планшетів, ці ймовірності підробки збільшуються на невелику величину, пов'язані з безпекою AES. Поки AES безпечний, цей невеликий додатковий термін незначний для будь-якої практичної атаки.

2.2.6. Інші режими MAC

Message authentication algorithm (**MAA**, алгоритм аутентифікації повідомлень) - алгоритм перевірки цілісності повідомлення. Був розроблений D. W. Davies і D. O. Clayden і опублікований в 1983 році. Алгоритм забезпечує

цілісність повідомлення, але не забезпечує конфіденційність. Стійкість алгоритму ґрунтується на секретності ключа. Алгоритм є частиною ISO 8731-2. Був одним з перших криптографічних функцій для обчислення MAC.

Data Authentication Algorithm (**DAA**, алгоритм аутентифікації даних) є колишнім державним стандартом США для отримання криптографічних кодів аутентифікації повідомлень .Алгорітм не рахується безпечним за сучасними стандартами. Відповідно до стандарту код, створений DAA, називається кодом аутентифікації даних (DAC). Ланцюжок алгоритму шифрує дані, при цьому останній блок шифрування усікається і використовується в якості ЦАП. DAA еквівалентний ISO / IEC 9797-1 MAC-алгоритму 1 або CBC-MAC з DES в якості базового шифру, усіченого до 24-56 біт (включно).

Алгоритм **PMAC** (паралельний MAC) може бути виконаний з використанням безлічі потоків одночасно, на відміну від інших алгоритмів MAC, описаних вище (які потребують послідовної обробки блоків даних). PMAC по функціональності аналогічний алгоритму OMAC.

PMAC використовує два секретних ключа шифрування. Перший секретний ключ використовується в P-функціях. Всі функції P отримують також наступні номери додаткового лічильника. Вихідні біти P-функцій додаються XOR в блоки даних. Результат зашифрований псевдослучайной функцією F, яка використовує другий секретний ключ. Функція P повинна бути нескладною і працювати набагато швидше, ніж функція F.

Вихідні біти з усіх функцій F і вихідні біти з останнього блоку даних (яка не зашифрований функцією F) додаються разом в XOR, а потім результат шифрується з використанням алгоритму функції F з другим секретним ключем шифрування.

Як завжди, можна довести, що PMAC безпечний, якщо секретний ключ час від часу змінюється. Новий ключ повинен бути створений після відправки кількості повідомлень, яке приблизно дорівнює квадрату числа всіх можливих значень блоків даних. PMAC дозволяє легко і швидко оновлювати коди аутентифікації в разі, якщо один з блоків повідомлень був замінений новим.

Алгоритм **NMAC** (Nested MAC) аналогічний алгоритму CBC MAC, описаного раніше. У ній використовується трохи інша псевдослучайная функція F. Функція F повертає числа, які є правильними значеннями секретних ключів (тобто не значеннями блоків даних).

Як і в разі CBC MAC, після шифрування останнього блоку даних виконується одне додаткове шифрування результату з використанням другого секретного ключа шифрування. Оскільки попередній результат шифрування останнього блоку даних складається з того ж кількості бітів, що і секретного ключа, необхідно додати додаткову послідовність бітів (панель виправлень), щоб гарантувати, що результат має той же розмір, що і блоки даних. NMAC зазвичай використовується в системах, де довжина блоків даних набагато більше, ніж розмір секретних ключів.

Без останнього кроку алгоритму (тобто без шифрування з використанням другого ключа) зловмисник зможе додати будь-яку кількість блоків до перехопленому повідомленням з правильно розрахованим кодом аутентифікації. Потім він може розрахувати новий код аутентифікації і прикріпити його до зміненого повідомленням. В якості вхідних даних для першої нової доданої функції F зловмисник буде використовувати вихідний код аутентифікації вихідного повідомлення.

Для забезпечення безпеки NMAC необхідно час від часу міняти секретний ключ. Можна довести, що після відправки кількості повідомлень, приблизно рівного квадрату числа всіх можливих значень секретних ключів, ключ більше не є безпечним.

Badger - це код аутентифікації повідомлень (MAC), заснований на ідеї універсального хешування. Оскільки Badger - це функція MAC, заснована на підході універсальної хеш функції, умови, необхідні для безпеки Badger, такі ж, як і для інших універсальних хеш-функцій, таких як UMAC. Badger MAC обробляє повідомлення довжиною до 2^{64} -1 біти і повертає тег аутентифікації довжини u ·32 біти, де $1 \leq u \leq 5$. Відповідно до вимог безпеки, користувач може вибрати значення u - це кількість паралельних хеш-дерев в Badger. Можна

вибрати великі значення u, але ці значення не впливають на безпеку MAC. Алгоритм використовує 128-бітний ключ і обмежену довжину повідомлення повинні бути оброблені під цей ключ. Налаштування ключа повинні виконуватися тільки один раз для кожного ключа, щоб запустити алгоритм Badger під заданим ключем, оскільки результуючий внутрішній стан MAC можна зберегти для використання з будь-яким іншим повідомленням, яке буде оброблено пізніше.

2.3. Коди виявлення маніпуляцій з даними

MDC є результат дії хеш-функції. Інакше кажучи, MDC - це хеш-образ повідомлення M, до якого застосували хеш-функцію, тобто s = h (M).

Як і в разі методу MAC, функція хешування отримує на вхід повідомлення X довільної довжини, а на виході видає хеш-код h = H (X) фіксованої довжини. На відміну від значення MAC функція хешування не вимагає використання секретного ключа. Існує кілька варіантів реалізації методу MDC. На рис. 2.16 представлена найбільш поширена схема реалізації методу MDC.

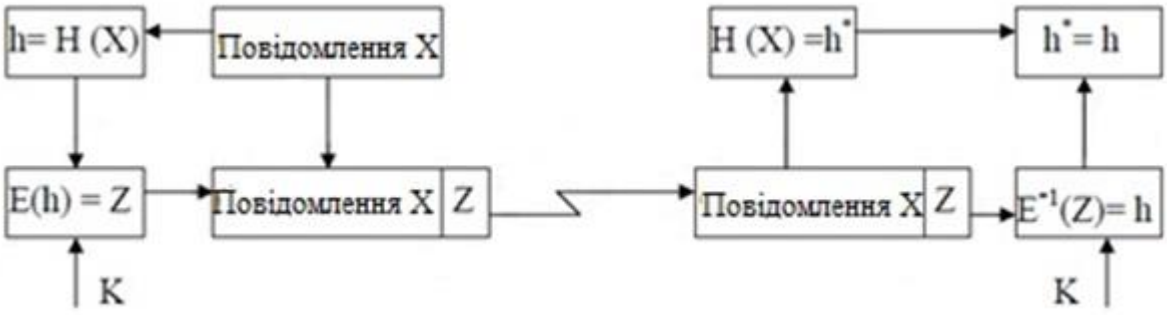


Рис. 2.16 Перший варіант реалізації методу MDC

У цьому методі користувач A обчислює хеш-код повідомлення X, зашифровує значення хеш-коду з використанням симетричного алгоритму, з'єднує зашифроване значення хеш-коду до повідомлення і відправляє

отриманий блок користувачеві В. Користувач В розшифровує отриманий хеш-код, обчислює хеш-код повідомлення X, звіряє результат цього обчислення з розшифрованих значенням отриманого хеш-коду і по результату порівняння робить висновок про справжність отриманого повідомлення.

Схема другого варіанту використання методу MDC приведена на рис. 2.17.

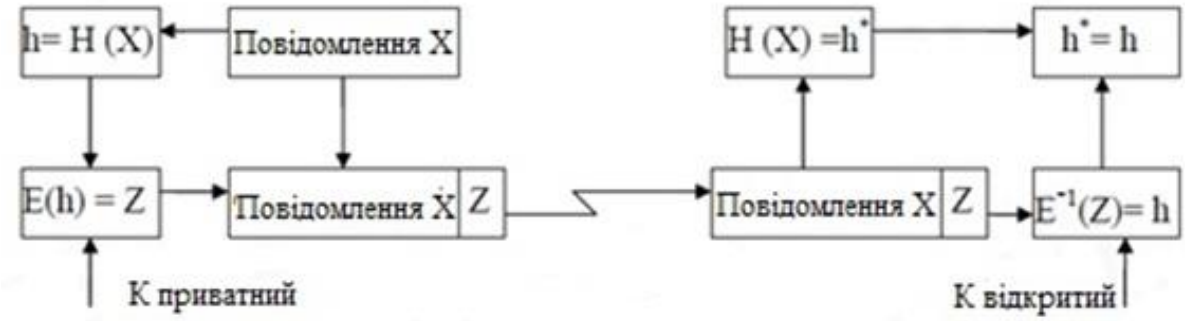


Рис. 2.17 Другий варіант реалізації методу MDC

В даному варіанті одержувачу не потрібно знання секретного ключа, що позбавляє від необхідності доставки секретного ключа сполучених сторонам.

Третій варіант методу MDC передбачає використання функції хешування без шифрування хеш-коду. Схема третього варіанту реалізації методу MDC представлена на рис.2.18.

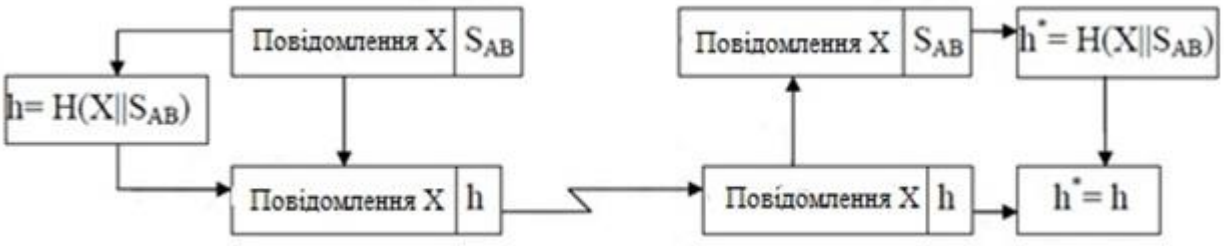


Рис. 2.18 Третій варіант реалізації методу MDC

Основна вимога до хеш-функції: не повинно існувати способу визначення масиву даних M, що має задане значення хеш-образу h (M), відмінного від перебору по всій множині можливих значень M. Найбільш простий спосіб

побудови хеш-функції заснований на використанні обчислювальної незворотності щодо ключа k функції шифрування Ek будь-якого блочного шифру. Навіть при відомих блоках відкритого M і закритого c = Ek (M) текстів ключ k не може бути визначений інакше як перебором по безлічі всіх можливих значень. Отже, схема формування хеш-образу повідомлення M, що володіє гарантованою стійкістю, рівній стійкості використовуваного шифру, може бути наступною:

А) масив даннях M розбивається на блоки фіксованого розміру, рівного розміру ключа $|k|$ блочного шифру, тобто:

$$M = M_1M_2M_3\ldots M_t\ , \qquad (2.5)$$

$$|M_1| = |M_2| = |M_3| = \ldots = |M_{m-1}| = |k|\ ,\ 0 < |M_t| \le |k|;$$

Б) якщо останній блок M_t неповний, він доповнюється будь-яким чином до потрібного розміру $|k|$ *;*

В) хеш-образ повідомлення обчислюється таким чином:

$$s = h(M) = E_{Mt}(\ldots E_{M1}(E_{M2}(E_{M1}(s_0))))\ , \qquad (2.6)$$

де s_0 - сінхропосилка, зазвичай вибирають $s_0 = 0$.

Завдання підбору масиву даних

$$M' = M'_1\ M'_2\ M'_3\ \ldots\ M'_t \qquad (2.7)$$

під заданий контрольний код s еквівалентна системі рівнянь, яку необхідно вирішити для визначення ключа для заданих блоків відкритого та закритого (в режимі простої заміни) повідомлень. Проте в даній ситуації немає необхідності вирішувати всю систему:

$$E_{M'1}(s_0) = s_1\ ; \qquad (2.8)$$

$$E_{M'2}(s_1) = s_2\ ;$$

$$E_{M'3}(s_2) = s_3;$$

$$\ldots$$

$$E_{M't}(s_{M'-1}) = s,$$

Досить вирішити рівняння:

$$E_{M'i}(s_{i-1}) = s_i \qquad (2.9)$$

відносно M'_i, інші блоки масиву M можуть бути довільними. Але і це завдання в разі використання надійної функції E_k обчислювально нерозв'язна.

На жаль, наведена схема формування MDC не враховує наявності так званих побічних ключів шифру. Якщо для $k' \neq k$ справедливо

$$E_{k'}(M_i) = E_k(M_i), \quad (2.10)$$

де M_i - деякий блок відкритого тексту, то такий код k' і є побічним ключем, тобто ключем, що дає при зашифрованими блоку M_i точно такий же результат, що і істинний ключ k.

Виявлення противником побічного ключа при дешифрування повідомлення не є особливим успіхом, так як з імовірністю, близькою до 1, на цьому знайденому побічну ключі він не зможе правильно розшифрувати інші блоки закритого тексту, враховуючи, що для різних блоків побічні ключі в загальному випадку також різні. У разі вироблення коду MDC ситуація прямо протилежна: виявлення побічного ключа означає, що противник знайшов такий помилковий блок даних, використання якого не змінює контрольного коду.

Для зменшення ймовірності нав'язування помилкових даних в результаті знаходження побічних ключів, при перетворенні застосовуються не самі блоки вихідного повідомлення, а результат їх розширення за певним алгоритмом. Під розширенням розуміється процедура отримання блоку даних більшого розміру з блоку даних меншого розміру. Наприклад, для криптоалгоритма, в якому розмір ключа дорівнює 256 біт, можлива наступна схема розширення 128-бітового блоку в 256-бітовий:

вихідний блок:

$$M_i = (b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9\ b_{10}\ b_{11}\ b_{12}\ b_{13}\ b_{14}\ b_{15}\ b_{16}); \quad (2.11)$$

розширений блок:

$$E_{xt}(M_{i)} = (b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8\ b_9\ b_{10}\ b_{11}\ b_{12}\ b_{13}\ b_{14}\ b_{15}\ b_{16}\ b_1\ b_4\ b_7\ b_{10}\ b_{13}\ b_{16}\ b_3\ b_6\ b_9\ b_{12}\ b_{15}\ b_2\ b_5\ b_8\ b_{11}\ b_{14}), \quad (2.12)$$

Де b_i – байти блоку даних.

Цілі супротивників проти MDC:

Мета противника, який бажає "атакувати" MDC, полягає в наступному:

1. для атаки на OWHF: задавши хеш-значення y, знайдіть зображення x таким, що y = h (x); або задавши одну таку пару (x, h (x)), знайдіть другу перевагу x таку, що h (x) = h (x).

2. атакувати CRHF: знайти будь-які два входи x, x, такі, що h (x) = h (x). CRHF повинен бути спроектований таким чином, щоб протистояти стандартним атакам дня народження.

Розглянемо приклад. Є дві людини - Аліса (А) і Боб (Б), які обмінюються повідомленням M. Якщо Аліса повинна передати повідомлення Бобу і хоче бути впевнена, що повідомлення не буде змінено під час передачі, вона може створити дайджест, MDC - повідомлення і послати повідомлення і MDC Бобу. Боб може створити новий MDC з повідомлення і порівняти отриманий MDC і новий MDC. Якщо вони однакові, значить, повідомлення не було змінено. Рис. 2.18 ілюструє ідею.

Рис. 2.18 показує, що повідомлення може бути передано через ненадійний канал. Єва може читати або навіть змінювати повідомлення. MDC, проте, повинен бути переданий через безпечний канал, - такий канал називають безпечний (safe), він не дозволяє змін.

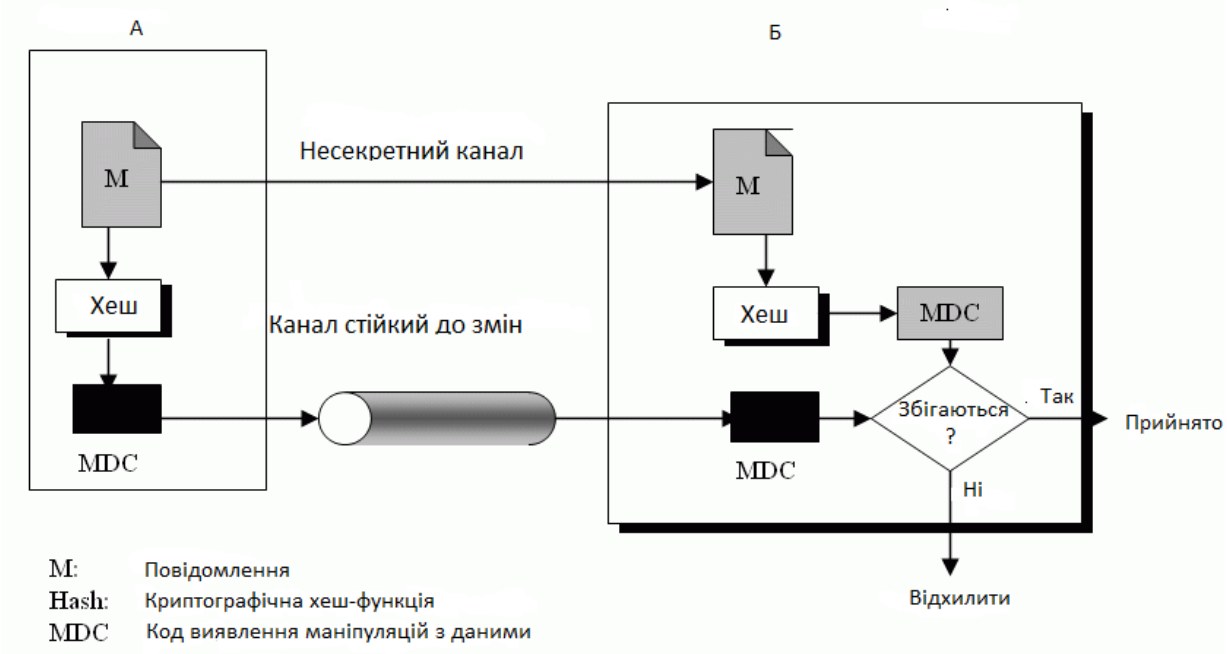


Рис. 2.18 Код виявлення маніпуляцій з даними

Якщо повідомлення і MDC передаються через ненадійний канал, Єва може перехопити повідомлення, змінити його, створити новий MDC з повідомлення і передати їх Бобу. Боб ніколи не дізнається, що повідомлення прийшло від Єви.

2.4. Порівняльний аналіз методів контролю цілісності інформації

Прокоментуємо відмінності: підхід на основі MAC вимагає для обчислення контрольної комбінації секретного ключа, для другого це не потрібно. Зловмисник не зможе обчислити MAC для довільного сфабрикованого їм повідомлення, але зможе вирахувати MDC, так як для цього не потрібно ніяких секретних даних, тому MAC може передаватися від джерела до приймача з відкритого каналу, тоді як для передачі MDC потрібно захищений канал.

Здавалося б, переваги першого підходу настільки очевидні, що другий підхід не зможе знайти собі застосування. Однак це не так - використання MAC вимагає, щоб попередньо між учасниками інформаційного обміну були розподілені ключі. Якщо ж цього не відбулося, для його реалізації необхідний спеціальний канал, що забезпечує секретність і справжність переданої інформації, по якому паралельно з передачею даних будуть передані ключі. Для передачі ж MDC потрібен канал, що забезпечує тільки справжність переданих даних, вимога секретності відсутня, і це робить даний метод переважним при одноразової передачі даних: основна інформація передається по звичайному незахищеному каналу, а MDC повідомляється відправником одержувачу по каналу, який може прослуховуватися, але не може бути використаний для нав'язування помилкових даних (учасники обміну повинні добре знати один одного).

Крім того, підхід на основі вироблення MDC простіший і зручніший, якщо створення і використання інформації розділені в часі, але не в просторі,

тобто для контролю цілісності інформації, що зберігається, а не переданої інформації - наприклад, для контролю незмінності програм і даних в комп'ютерних системах. При цьому контрольна комбінація (MDC) повинна зберігатися в системі таким чином, щоб виключити можливість її модифікації зловмисником.

Табл. 2. 1.

Порівняльні характеристики підходів до вирішення завдання контролю цілісності повідомлень

Параметр порівняння	MAC	MDC
Перетворення, що використовується	Блоковий шифр	Хеш-функція
Секретна інформація, що використовується	Секретний ключ	Не використовується
Можливість для третьої сторони обчислити контрольний код	Зловмисник не може обчислити, якщо йому не відомий секретний ключ	Зловмисник може обчислити для довільного блоку даних
Зберігання та передача контрольної комбінації	Контрольна комбінація може зберігатися і передаватися разом з повідомленням, що захищається	Контрольна комбінація повинна зберігатися і передаватися окремо від масиву даних, що захищається
Параметр порівняння	MAC	MDC
Додаткові умови	Вимагає попереднього розподілу ключів між абонентами	необхідний автентичний канал для передачі контрольного коду
Області, в яких підхід має перевагу	Захист від несанкціонованих змін повідомлень при їх передачі	Разова передача повідомлень, контроль незмінності файлів даних і програм

2.5. Алгоритмічні та математичні підходи щодо формування хеш функції MD5 та SHA-1

2.5.1. Хеш-функция MD5

Розглянемо алгоритм отримання дайджесту повідомлення MD5 (RFC+1321), розроблений Роном Ривестом з MIT. Алгоритм отримує на вході повідомлення довільної довжини і створює в якості виходу дайджест повідомлення довжиною 128 біт. Алгоритм складається з наступних кроків:

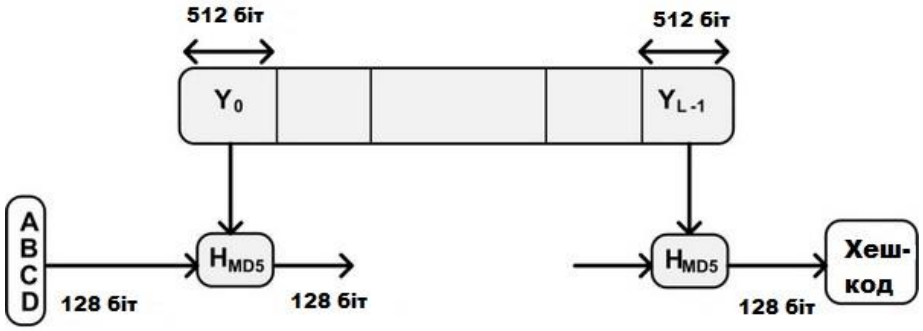


Рис. 2.19 Логіка виконання MD5

Крок 1: додавання відсутніх бітів

Повідомлення доповнюється таким чином, щоб його довжина дорівнювала 448 по модулю 512 (довжина = 448 (mod 512)). Це означає що довжина доданого повідомлення на 64 біта менше ніж число, кратне 512. Додавання проводиться завжди, навіть якщо повідомлення має потрібну довжину. Наприклад, якщо довжина повідомлення 448 бітів, воно доповнюється 512 битами до 960 бітів. Таким чином, число додаються бітів знаходиться в діапазоні від 1 до 512.

Додавання складається з одиниці, за якою слідує необхідна кількість нулів.

Крок 2: додавання довжини

64-бітове представлення довжини вихідного (до додавання) повідомлення у бітах приєднується до результату першого кроку. Якщо первісна довжина більше, ніж 2^{64}, то використовуються тільки останні 64 біта. Таким чином, поле містить довжину вихідного повідомлення по модулю 2^{64}.

В результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам (рис.2.20). Це розширене повідомлення представляється як післідовність 512-бітних блоков Y0, Y1,. . ., YL-1, при цьому загальна довжина розширеного повідомлення дорівнює L*512 бітам. Таким чином, довжина отриманого розширеного повідомлення дорівнює шістнадцяти 32-бітовим словам.

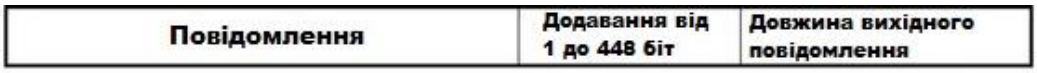


Рис. 2.20 Структура розширеного повідомлення

Крок 3: ініціалізація MD-буфера

Використовується 128-бітний буфер для зберігання проміжних і остаточних результатів хеш-функції. Буфером є чотири 32-бітних регістра (A, B, C, D). Ці регістри инициализируются наступними шестнадцатерічнимі числами:

A = 01234567

B = 89ABCDEF

C = FEDCBA98

D = 76543210

Крок 4: обробка послідовності 512-бітних (16-слівних) блоків

Основою алгоритму є модуль, що складається з чотирьох циклічних обробок, позначений як H_{MD5}. Чотири цикла мають схожу структуру, але кожен цикл використовує свою елементарну логічну функцію, що позначається f_F, f_G, f_H і f_I відповідно (рис. 2.9).

Кожен цикл приймає в якості входу поточний 512-бітний блок Y_q, який обробляється в даний момент, і 128-бітове значення буфера ABCD, яке є проміжним значенням дайджесту, і змінює вміст цього буфера. На кожному циклі також використовується четверта частина 64-елементної таблиці T[1]...T[64], побудованої на основі функції sin.

$$T[i] = [2^{32} * \text{abs}(\sin(i))], \quad (2.13)$$

де [ ] в правій частині означає цілу частину числа, і задано в радіанах. Так як abs(sin(i)) є числом між 0 і 1, кожен елемент T[*i*] є цілим, яке може бути представлено 32 бітами. Таблиця забезпечує "випадковий" набір 32-бітових значень, які повинні ліквідувати будь-яку регулярність у вхідних даних.

Для отримання MD_{q+1} вихід чотирьох циклів складається по модулю 232 з MD_q (рис. 2.21). Додавання виконується незалежно для кожного з чотирьох слів у буфері.

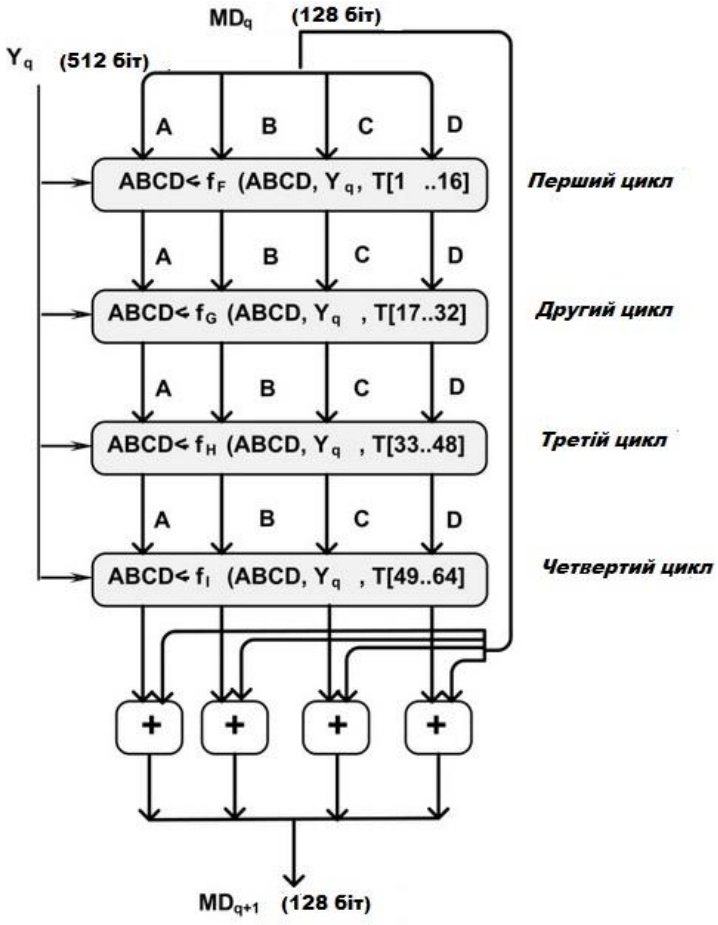


Рис. 2.21 Обробка чергового 512-бітного блоку

Крок 5: вихід

Після обробки всіх L 512-бітних блоків виходом L-ой стадії є 128-бітний дайджест повідомлення.

Розглянемо більш детально логіку кожного з чотирьох циклів виконання одного 512-бітного блоку. Кожен цикл складається з 16 кроків, що оперують з буфером ABCD. Кожен крок представлений на рис. 2.22

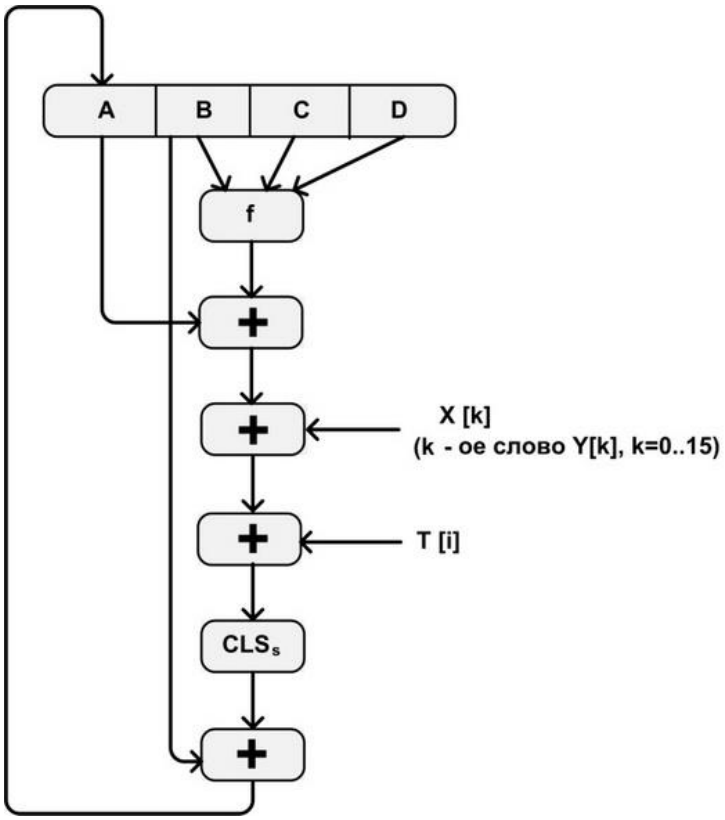


Рис. 2.22 Логіка виконання окремого кроку

$$A \leftarrow B + CLS_S (A + f (B, C, D) + X[k] + T[i]), \quad (2.14)$$

де A, B, C, D - чотири слова буфера; після виконання кожного окремого кроку відбувається циклічний зсув вліво на одне слово.

f - одна з елементарних функцій f_F, f_G, f_H, f_I.

CLS_s- циклічний зсув вліво на s бітів 32-бітного аргументу.

X [*k*] - M[q*16 + *k*] - k-е 32-бітове слово в q-му 512 блоці повідомлення.

T [*i*] - *i*-е 32-бітове слово в таблиці T.

+ - додавання по модулю 232 [20].

У кожному з чотирьох циклів алгоритму використовується одна з чотирьох так званих елементарних логічних функцій. Кожна елементарна функція отримує три 32-бітних слова на вході і на виході створює одне 32-бітове слово. Кожна функція є набором побітових логічних операцій, тобто n-ий біт виходу є функцією від n-ого біта трьох входів. Елементарні функції такі:

F_F = (B&C) V (*not*B&D) f_G = (B&D) V (C&*not*D) f_H = B⊕C⊕Df_I =

= C⊕(B&*not*D) (2.15)

Масив з 32-бітових слів X [0..15] містить значення поточного 512-бітного вхідного блоку, який обробляється в даний момент. Кожен цикл виконується 16 разів, а так як кожен блок вхідного повідомлення обробляється в чотирьох циклах, то кожен блок вхідного повідомлення обробляється за описаною вище схемі 64 рази. Якщо уявити вхідний 512-бітний блок у вигляді шістнадцяти 32-бітний слів, то кожне вхідне 32-бітове слово використовується чотири рази, по одному разу в кожному циклі, і кожен елемент таблиці T, що складається з 64 32-бітових слів, використовується тільки один раз. Після кожного кроку циклу відбувається циклічний зсув вліво чотирьох слів A, B, C і D. На кожному кроці змінюється тільки одне з чотирьох слів буфера ABCD. Отже, кожне слово буфера змінюється 16 разів, і потім 17-й раз в кінці для отримання остаточного виходу даного блоку.

Можна підсумувати алгоритм MD5 наступним чином:
MD_0 = IV $MD_{q+1} = MD_q + f_1[Y_q, f_H [Y_q, f_F [Y_q, MD_q]]]]$ MD = MD_{L-1}, (2.16)

де IV - початкове значення буфера ABCD, певне на кроці 3.

Yq - q-ий 512-бітний блок повідомлення.

L - число блоків в повідомленні (включаючи поля доповнення та довжини).

MD - остаточне значення дайджесту повідомлення.

2.5.2. Хеш-функція SHA-1

Безпечний хеш-алгоритм (Secure Hash Algorithm) був розроблений NIST і опублікований в якості федерального інформаційного стандарту (FIPS PUB 180) в 1993 році. У алгоритмів MD5 і SHA-1 багато спільного.

Алгоритм отримує на вході повідомлення довільної довжини і створює в якості виходу дайджест повідомлення довжиною 160 біт (рис.2.23). Алгоритм складається з наступних кроків:

Крок 1: додавання відсутніх бітів

Повідомлення додається таким чином, щоб його довжина була кратна 448 по модулю 512 (довжина = 448 (mod 512)). Додавання здійснюється завжди, навіть якщо повідомлення вже має потрібну довжину. Таким чином, число додаються бітів знаходиться в діапазоні від 1 до 512.

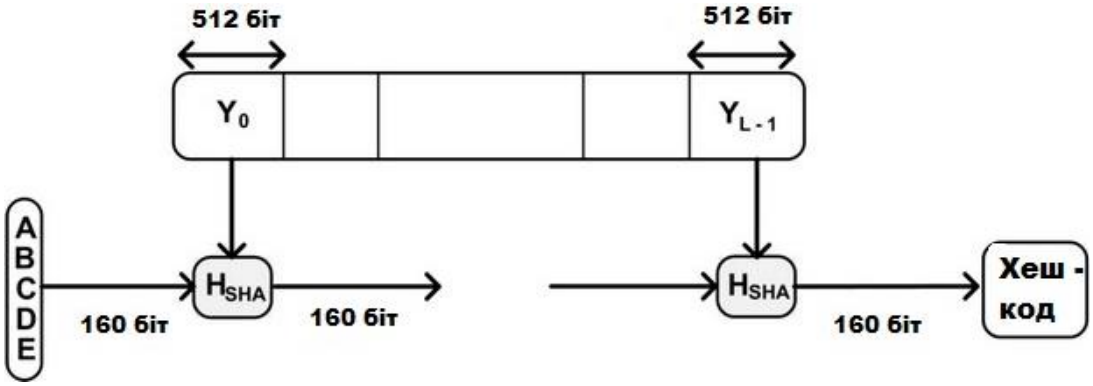


Рис. 2.23 Логіка виконання SHA-1

Додавання складається з одиниці, за якою слідує необхідна кількість нулів.

Крок 2: додавання довжини

До повідомлення додається блок з 64 бітів. Цей блок трактується як беззнаковое 64-бітове ціле і містить довжину вихідного повідомлення до додавання.

Результатом перших двох кроків є повідомлення, довжина якого кратна 512 бітам. Розширене повідомлення може бути представлено як послідовність 512-бітних блоків Y0, Y1,. . . , YL-1, так що загальна довжина розширеного повідомлення є L*512 біт. Таким чином, результат кратний шістнадцяти 32-бітовим словами.

Крок 3: ініціалізація SHA-1 буфера

Використовується 160-бітний буфер для зберігання проміжних і вікончательних результатів хеш-функції. Буфер може бути представлений як п'ять 32-бітових регістрів A, B, C, D і E. Ці регістри инициализируются наступними шестнадцатерічнимі числами:

A = 67452301

B = EFCDAB89

C = 98BADCFE

D = 10325476

E = C3D2E1F0

Обробка повідомлення в 512-бітних (16-слівних) блоках

Основою алгоритму є модуль, що складається з 80 циклічних обробок, позначений як HSHA. Всі 80 циклічних обробок мають однакову структуру (рис. 3.6).

Кожен цикл отримує на вході поточний 512-бітний опрацьований блок Y_q і 160-бітове значення буфера ABCDE, і змінює вміст цього буфера.

У кожному циклі використовується додаткова константа K_t яка приймає тільки чотири різних значення:

$0 \le t \le 19 K_t$ = 5A827999(цілачастиначисла $[2^{30} \times \sqrt{2}]$)$20 \le t \le 39 K_t$ =

= 8F1BBCDC(цілачастиначисла$[2^{30} \times \sqrt{5}]$)$60 \le t \le 79 K_t$ =

= CA69C1D6(цілачастиначисла$[2^{30} \times \sqrt{10}]$) (2.17)

Для отримання SHA_{q+1} вихід 80-го циклу складається із значенням SHA_q. Додавання по модулю 232 виконується незалежно для кожного з п'яти слів в буфері з кожним з відповідних слів в SHA_q [21] (рис. 2.24).

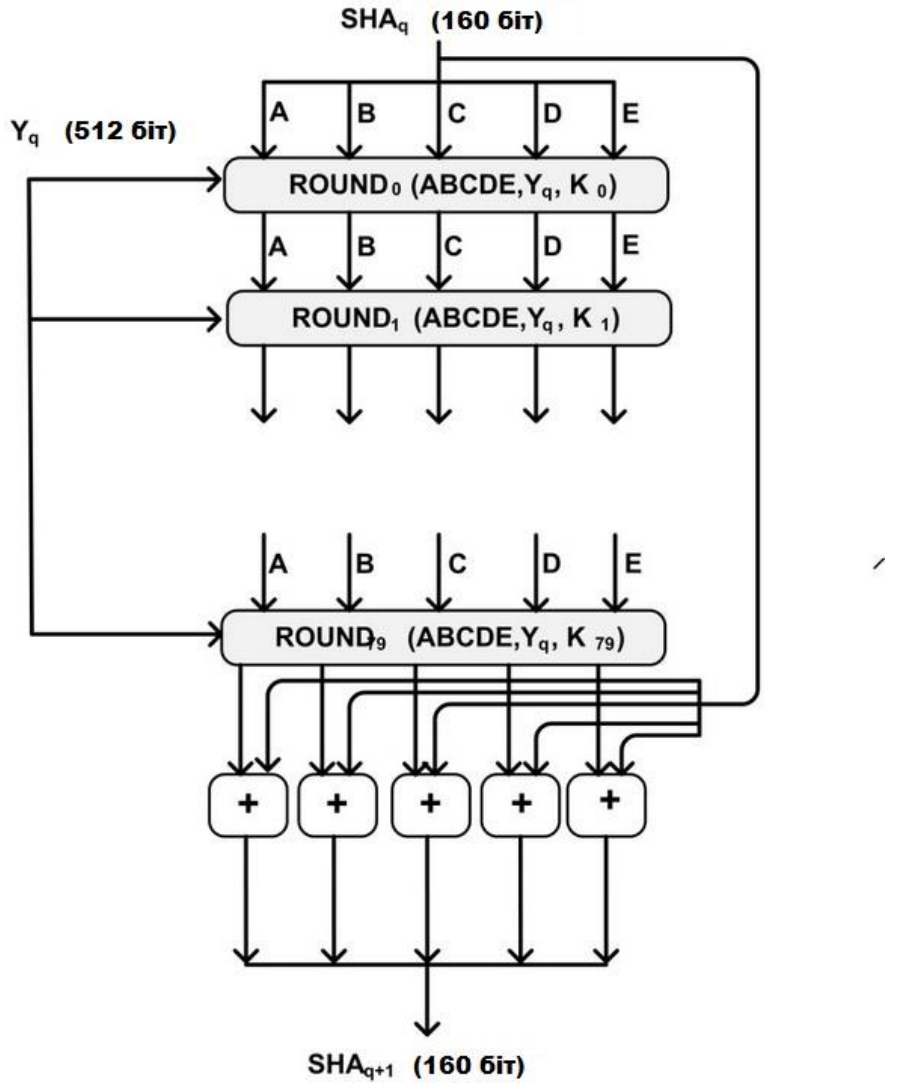


Рис. 2.24 Обробка чергового 512-бітного блоку

Для отримання SHA_{q+1} вихід 80-го циклу складається із значенням SHA_q. Додавання по модулю 232 виконується незалежно для кожного з п'яти слів в буфері з кожним з відповідних слів в SHA_q [21].

Крок 5: вихід

Після обробки всіх 512-бітних блоків виходом L-ой стадії є 160-бітний дайджест повідомлення.

Розглянемо більш детально логіку в кожному з 80 циклів обробки одного 512-бітного блоку. Кожен цикл можна уявити у вигляді:

$$A,B,C,D,E \leftarrow (CLS_5(A) + f_t(B,C,D) + E + W_t + K_t),\ A, CLS_{30}(B), C,\ D \qquad (2.18)$$

де A, B, C, D, E - п'ять слів з буфера.

t - номер циклу, $0 \le t \le 779$.

f_t - елементарна логічна функція.

CLS_s - циклічний лівий зсув 32-бітного аргументу на s бітів.

W_t - 32-бітове слово, отримане з поточного вхідного 512-бітного блоку.

K_t - додаткова константа.

+ - додавання по модулю 232 [21].

Кожна елементарна функція отримує на вході три 32-бітних слова і створює на виході одне 32-бітове слово. Елементарна функція виконує набір побітових логічних операцій, тобто n-ий біт виходу є функцією від n-их бітів трьох входів. Функції наступні:

Табл. 2.2

Принцип роботи елементарних функцій

Номер циклу	f_t (B, C, D)
$(0 \le t \le 19)$	(B&C) V (not B&D)
$(20 \le t \le 39)$	B ⊕ C ⊕ D
$(40 \le t \le 59)$	(B&C) V (B&D) V (C&D)
$(60 \le t \le 79)$	B ⊕ C ⊕ D

Насправді використовуються тільки три різні функції.

Для $0 \le t \le 19$ функція є умовною: if B then C else D

Для $20 \le t \le 39$ і $60 \le t \le 79$ функція створює біт парності.

Для $40 \le t \le 59$ функція є дійсною, якщо два або три аргументи істинні.

32-бітові слова W_t виходять з чергового 512-бітного блоку повідомлення наступним чином:

Перші 16 значень W_t беруться безпосередньо з 16 слів поточного блоку. Решта значеннь визначаються наступним чином:

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3} \qquad (2.17)$$

У перших 16 циклах вхід складається з 32-бітного слова даного блоку. Для решти 64 циклів вхід складається з XOR декількох слів з блоку повідомлення (рис.2.25).

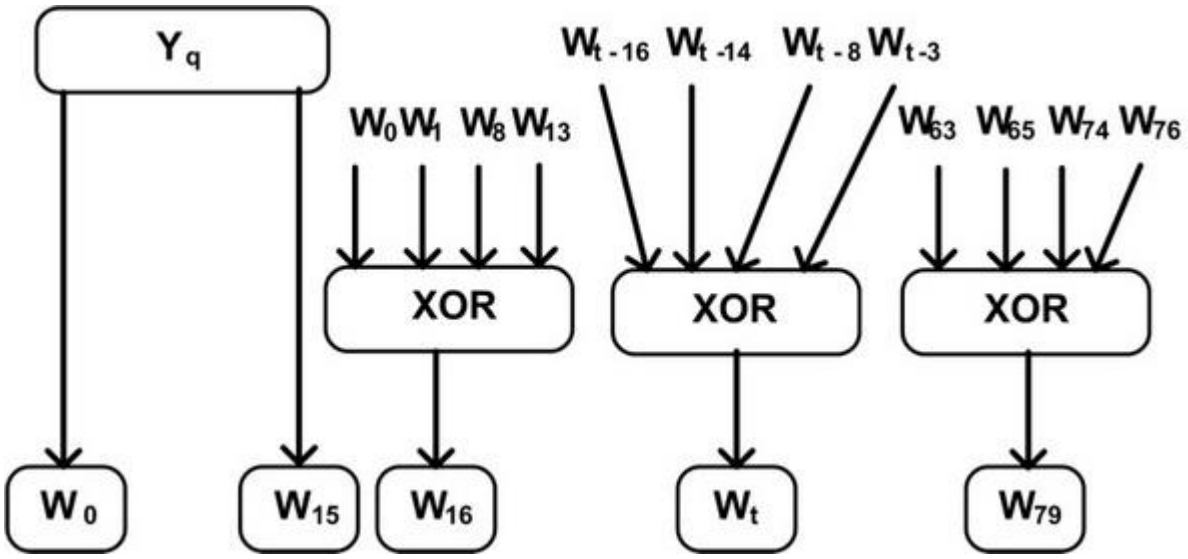


Рис. 2.25. Отримання вхідних значень кожного циклу з чергового блоку

Алгоритм SHA-1 можна підсумувати таким чином:

SHA0 = IV

$SHA_{q+1} = \Sigma 32\ (SHA_q, ABCDE_q)$

SHA = SHAL-1, (2.18)

де

IV - початкове значення буфера ABCDE.

$ABCDE_q$ - результат обробки q-того блоку повідомлення.

L - число блоків в повідомленні, включаючи поля додавання і довжини.

Σ32 - сума по модулю 2^{32}, виконувана окремо для кожного слова буфера.

SHA - значення дайджесту повідомлення [21].

2.6. Порівняльний аналіз алгоритмів безпечного хешування

Алгоритм безпечного хешування (SHA) - стандарт, який був розроблений національним Інститутом Стандартів і Технології (NIST - National Institute of Standards and Technology) і виданий як Федеральний Стандарт Обробки Інформації (FIP 180). Він згадується в літературі як Стандарт Безпечного хешування (SHS - Secure Hash Standard). Стандарт головним чином базується на MD5. У 1995 р він був переглянутий під назвою FIP 180-1, який включає SHA-1. Пізніше він знову був переглянутий під назвою FIP 180-2, який визначає чотири нових версії: SHA-224, SHA-256, SHA-384 і SHA-512.

Табл. 2. 3

Порівняльний аналіз алгоритмів безпечного хешування

Характеристики	MD5	SHA-0	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512	RIPEMD-128	RIPEMD-160
1	2	3	4	5	6	7	8	9	10
Розмір вихідного хешу (біт)	128	160	160	224	256	384	384	128	160
Размір блоку (біт)		512	512	512	512	1024	1024	512	512
Число раундов	64	80	80	64	64	80	80	4	5
Максимальний розмір вхідного повідомлення (біт)	Необмежений	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$	$2^{128} - 1$	Необмежений	Необмежений

Продовження таблиці 2.4

1	2	3	4	5	6	7	8	9	10
Операції	And, Xor, Rot, Add (mod^{32}), Or	And, Xor, Rot, Add (mod^{32}), Or	And, Xor, Rot, Add (mod^{2}), Or	And, Xor, Rot, Add (mod^{32}), Or	And, Xor, Rot, Add (mod^{32}), Or	And, Xor, Rot, Add (mod^{64}), Or, Shr	And, Xor, Rot, Add (mod^{64}), Or, Shr	And, Xor, Rot, Add (mod^{64}), Or,	And, Xor, Rot, Add (mod^{64}), Or,
Знайдені колізії	Існують	Існують	2^{52} операцій	Немає	Немає	Немає	Немає	Немає	Немає

Зробимо порівняльній аналіз безпосередно MD5 и SHA-1 алгоритмів, логика виконання яких була описана у підрозділі 2.6.

Безпека: найбільш очевидне і найбільш важлива відмінність полягає в тому, що дайджест SHA-1 на 32 біта довше, ніж дайджест MD5. Якщо припустити, що обидва алгоритму не містять будь-яких структурованих даних, які вразливі для аналітичних атак, то SHA-1 є більш стійким алгоритмом. Як і всі сімействі SHA - алгоритмів.

Швидкість: SHA-1 містить більше кроків (80 замість 64) і виконується на 160-бітному буфері в порівнянні з 128-бітовим буфером MD5. Таким чином, SHA-1 повинен виконуватися приблизно на 25% повільніше, ніж MD5 на тій же апаратурі.

Простота і компактність: обидва алгоритму прості і в описі, і в реалізації, не вимагають великих програм або підстановлювальних таблиць. Проте, SHA-1 застосовує однокрокову структуру в порівнянні з чотирма структурами, використовуваними в MD5. Більш того, обробка слів в буфері однакова для всіх

кроків SHA-1, в той час як в MD5 структура слів специфічна для кожного кроку.

Алгоритм хешування SHA названий безпечним, тому що він спроектований таким чином, щоб було 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 викличе зміна дайджесту, і повідомлення не пройде перевірку на цілісність.

Висновки до 2 розділу

У другому розділі були розглянуті сучасні методи контролю цілісності на базі криптографічних перетворень а також алгоритми хешування. Визначено, что виділяють два основних криптографічних підхода до вирішення завдання захисту інформації від несанкціонованих змін даних, які передбачають формування:

- коду автентифікації повідомлень MAC (Message Authentication Code);
- коду виявлення маніпуляцій з даними MDC (Manipulation Detection Code).

Було детально розглянуто MAC та MDC коди на схемах та прикладах, розглянуто яким чином зловмисник може зробити підміну повідомлення, та чи можливо це взагалі.

Розглянуто схему алгоритма отримання MAC, яка ще має назву вкладеного (nested) MAC, або HMAC, в якій за для підвищення безпеки хешування виконується двічі. Та код HMAC, який передбачає послідовне використання операцій хешування та шифрування з секретного ключем.

Розглянуто логіку виконання двох алгоритмів хешування: MD5 та SHA-1. Був зроблений порівняльний аналіз розглянутих алгоритмів, та таблиця

існуючих алгоритмів безпечного хешування. Виходячи з наявних даних та порывняльного аналізу, можно зробити висновок, що безпечніше використовувати алгоритми хешування SHA-2. Дайджест SHA-2 довше, ніж дайджест MD5 чи SHA-1, та при відомих методах пошуку колізій у алгоритмі SHA-2 частота виникнення колізій близька до теоретичного мінімуму, у той час, як для алгоритмів MD5, SHA-0, та SHA-1 частота виникнення колізій набагато вища.

У ході порівняльного аналіза методів контролю цілісності інформацію, був зроблений висновок, що підхід на основі MAC вимагає для обчислення контрольної комбінації секретного ключа, для MDC це не потрібно. Зловмисник не зможе обчислити MAC для довільного сфабрикованого їм повідомлення, у той час, зможе вирахувати MDC, так як для цього не потрібно ніяких секретних даних. Але використання MAC вимагає, щоб попередньо між учасниками інформаційного обміну були розподілені ключі. А для передачі ж MDC потрібен канал, що забезпечує тільки справжність переданих даних, вимога секретності відсутня, і це робить даний метод переважним при одноразової передачі даних.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ КОНТРОЛЮ ЦІЛІСНОСТІ ІНФОРМАЦІЇ НА ОСНОВІ КРИПТОГРАФІЧНИХ ПЕРЕТВОРЕНЬ

3.1. Опис середовища реалізації

Програма написана на мові програмування C ++ в середовищі розробки Borland C ++ Builder 6. Також у своїй роботі я використовувала бібліотеку CryptoAPI.

Мова C ++ з'явився в результаті розвитку мови C, створеного в компанії Bell Labs. Подібно C, мова C ++ отримала загальне визнання і широке

поширення. Широка поширеність C ++ сприяла конкуренції між постачальниками компіляторів і середовищ розробки, що зіграло позитивну роль. Однак, оскільки постачальники прагнули до збільшення свого технологічного впливу, вони часто вносили в мову програмування власні унікальні особливості. Хоча такі доробки і були корисні, вони приводили до несумісності програм, написаних для одного компілятора, з іншими компіляторами.

Американський національний інститут стандартів (American National Standards Institute - ANSI) займається координацією і визначенням промислових стандартів в різних областях. Таким чином, ANSI став найбільш придатною організацією, яка могла взятися за проблему стандартизації мов C і C ++.

У XXI столітті відповідність стандарту ANSI стало одним з найбільш важливих властивостей мови. Цей стандарт дозволяє створювати і компілювати програми в різних системах розробки, для різних операційних систем і наборів процесорних інструкцій.

Мова програмування C ++ залишається найбільш поширеною. Вона застосовується для розробки різних додатків - від складних багаторівневих бізнес-систем до високопродуктивних програм візуалізації даних і систем реального часу.

Мова C ++ є ядром середовища C ++ Builder, яке забезпечує дуже високу ступінь підтримки цієї стандартизованої мови програмування.

Програмний продукт компанії Borland C ++ Builder - один з провідних середовищ розробки для створення Internet-додатків, «настільних» і розподілених додатків, а також додатків, заснованих на моделі клієнт / сервер. C ++ Builder поєднує простоту середовища швидкої розробки додатків, або RAD-середовища (Rapid Application Development - RAD), з потужністю і продуктивністю мови C ++, сумісного зі стандартом ANSI.

Основна частина роботи зі створення додатків виконується в інтегрованому середовищі розробки (Integrated Development Environment - IDE) C ++ Builder.

Замість окремого інструментарію, що оперує візуальними елементами управління, в C ++ Builder інтегрована так звана Палітра компонент, розділена картотечними вкладками на кілька функціональних груп. Функціональні можливості компонент можна досить просто модифікувати, а також розробляти компоненти, що володіють абсолютно новою оригінальною поведінкою.

Система містить Бібліотеку з понад 100 повторно використовуваних візуальних компонент, які перетягуються мишею на форму і відразу стають елементами управління прототипу програми. Крім відомих елементів управління Windows (кнопки, лінійки прокрутки, поля редагування, прості і комбіновані списки і т.д.) Бібліотека містить нові компоненти підтримки діалогів, обслуговування баз даних і багато інших.

Після розміщення компонентів на формі інспектор об'єктів допомагає встановлювати їх властивості і прописувати подіям коди обробки. Проект будується поступово, на тлі вироблених змін у властивостях, подіях і функціях використовуваних елементів. Добре продумано поділ і редагування програмного модуля за двома його частинами: інтерфейсної і власне кодової.

C ++ Builder підтримує основні принципи об'єктно-орієнтованого програмування - інкапсуляцію, поліморфізм і множинне спадкування, а також нововведені специфікації і ключові слова в стандарті мови.

На сьогоднішній день на ринку є досить багатий набір засобів комп'ютерної криптографії, орієнтованих як на використання штатних криптографічних засобів ОС Windows, так і криптографічних пакетів сторонніх виробників, що дозволяють виконувати криптографічні перетворення над даними будь-якої структури і складності.

Набір програмних інтерфейсів Windows для роботи з криптографією має назву CryptoAPI. Концепція CryptoAPI має на увазі приховування від програміста всіх тонкощів процесу шифрування даних.

Для використання функцій CryptoAPI в додатках, написаних на C / C ++, необхідно використовувати заголовки WinCrypt.h і підключити до додатка бібліотеку імпорту Crypt32.lib.

Одним з ключових понять CryptoAPI є криптографічний провайдер (CSP, Cryptographic Service Provider), який реалізує базовий набір криптографічних функцій, що відповідають тому чи іншому криптографічним алгоритмом (або сімейства алгоритмів). Криптопровайдери реалізовані у вигляді окремих DLL, так що сторонні розробники можуть самостійно включати до складу CryptoAPI реалізації своїх алгоритмів (хоча для поширення криптопровайдера потрібно підписати його цифровим підписом в Microsoft). Одночасно в операційній системі можна встановити кілька CSP.

Вибір конкретного алгоритму задається параметрами функції і залежить від використовуваного CSP, а універсальний набір функцій визначено для кожного типу криптографічних операцій.

У Crypto API для маніпуляції з хешем використовується спеціальний хеш-об'єкт. Взаємодія з цим об'єктом здійснюється за допомогою наступних трьох функцій:

- CryptCreateHash;
- CryptHashData;
- CryptGetHashParam.

Для первинної ініціалізації хеш-об'єкта застосовують функцію CryptCreateHash. Ця функція має наступний опис:

BOOL CryptCreateHash (HCRYPTPROV hProv, ALG_ID AlgId, HCRYPTKEY hKey, DWORD dwFlags, HCRYPTHASH * phHash);

В якості першого параметра даної функції передається ініціалізований контекст криптопровайдера. Другим параметром вказується алгоритм отримання значення хеша. Параметр hKey необхідний лише в разі застосування спеціалізованих алгоритмів типу MAC і HMAC.

Параметр dwFlags зарезервований під можливе майбутнє використання і повинен бути завжди дорівнює 0. Через параметр phHash функція повертає хендл створеного їй хеш-об'єкта. Після того, як хеш-об'єкт стане непотрібним, потрібно звільнити хеш-об'єкт за допомогою виклику функції CryptDestroyHash.

Після ініціалізації хеш-об'єкта можна почати передачу даних хеш-функції за допомогою виклику CryptHashData. Ця функція має наступний опис:

BOOL CryptHashData (HCRYPTHASH hHash, BYTE * pbData, DWORD dwDataLen, DWORD dwFlags);

В якості першого параметра даної функції передається раніше ініціалізований хендл хеш-об'єкта. Другим параметром передається порція даних для хеш-функції. Параметр dwDataLen є довжину переданих даних. Параметр dwFlags зазвичай дорівнює нулю.

Після повної передачі всього масиву вхідних даних функції CryptHashData виникає необхідність в отриманні значення хеш-функції. Дане завдання вирішується із застосуванням функції CryptGetHashParam. Ця функція має наступний опис:

BOOL CryptGetHashParam (HCRYPTHASH hHash, DWORD dwParam, *pbData, DWORD * pdwDataLen, DWORD dwFlags);

В якості першого параметра даної функції передається раніше ініціалізований хендл хеш-об'єкта. Другий параметр, dwParam, функції визначає тип запитуваної значення. Для отримання хеш-значення необхідно передати другим аргументом значення HP_HASHVAL. Параметри pdData і pdwDataLen відповідають за блок пам'яті, який використовується під значення, що повертається. Параметр dwFlags зарезервований для майбутнього використання і має дорівнювати нулю [18, 19].

Для перевірки правильності хеш-значення потрібно отримати хеш-значення даних і звірити його з перевіряємим хеш-значенням.

3.2. Технічні можливості та характеристики програмної реалізації

Цілісність даних і цілісність джерела даних є основним фактором, який розвиває або вимірює довіру між комунікаторами при будь-якому вигляді зв'язку. Криптографічні хеш-функції широко використовуються в якості

рішень, і для забезпечення цілісності даних використовується велика кількість схем на їх основі. Більшість ціх схем біло описано в розділі 2. У розробленому програмному модулі реалізований контроль цілісності інформації на основі MDC алгоритму з використанням MD5, SHA-0, SHA-1, SHA-256, SHA-384, SHA-512 хешів, та контроль цілісності і джерела даних, з використанням кодів аутентифікації повідомлень.

Контроль цілісності повідомлення заснований на можливості обчислити за допомогою MDC алгоритму хеш повідомлення, і виконати перевірку отриманого хешу на відповідність вказаному тексту. Так само, в розробленому програмному модулі користувач може скористатися вдосконаленим алгорітом, на основі використання дати і часу, секретного ключа, та функції хешування, при якому хешування піддається як результат об'єднання секретного ключа, дати і часу створення хешу, и віхідного повідомлення.

Схеми хеширования в поєднанні з кодами аутентифікації повідомлень (MAC) використовуються для досягнення як цілісності даних, так і цілісності джерела даних і називаються заснованими на хеші MAC (HMAC). На відміну від звичайного HMAC, алгоритм, запропонований в цій роботі, хешування повідомлення виконується з використанням інформації, специфічної для користувача, тобто інформації про дату, час і ключі, і, отже, схема не залежить тільки від самого повідомлення. Основна увага приділяється як цілісності даних, так і забезпечення цілісності джерела даних, що грає так само велику роль у захисті інформації.

Математична модель HMAC:

Далі $N = \{1, 2, \dots\}$ — безліч натуральних чисел, $N_0 = N \cup \{0\} = \{0, 1, 2, \dots\}$, F_2 — кінцеве поле GF(2), V_n - безліч рядків довжини $n \in N_0$ з елементами з F_2. Далі значення параметра t належить множині {256, 512}, а значення довжини ключа n задовольняє умові $256 \le n \le 512$. Під блоком далі розуміється бітова рядок довжини 512.

Функція $HMAC_t$ реалізує відображення:

$$HMAC_t : V_n \times V_{[0,2^{512}-1]} \rightarrow V_t. \qquad (3.1)$$

Першим аргументом є ключ K, другим - дані M. При необхідності ключ доповнюється нулями до рядка довжини 512 біт: $K^* = K \,||\, (0 \ldots 0) \in V_{512}$. Остаточно:

$HMAC_t(K, M) = GH_t(K^* \oplus opad \,||\, GH_t(K^* \oplus ipad \,||\, M))$, де (3.2)

ipad та opad - зумовлені константи, які дорівнюють:

$$ipad = \underbrace{(01011100)|| \ldots ||(01011100)}_{64} \in V_{512}. \quad (3.3)$$

$$opad = \underbrace{(01011100)|| \ldots ||(01011100)}_{64} \in V_{512}. \quad (3.4)$$

Алгоритми $HMAC_t$ визначені відповідно до документа [5]. Також HMAC описаний в документах [6, 7, 8].

Математична модель запропонованого алгоритму HMAC, з використанням дати та часу:

Як і у стандарті, першим аргументом є ключ K, другим аргументом є дата та час створення хешу, об означимо його як D, третім - дані M. При необхідності ключ та дата створення доповнюється нулями до рядка довжини 512 біт:

$K^* = K \,||\, (0 \ldots 0) \in V_{512}$. $D^* = D \,||\, (0 \ldots 0) \in V_{512}$. (3.5)

Отримуємо данну математичну модель запропонованого алгоритму:

$HMAC_t(K, M, D) = GH_t(K^* \oplus D^* \oplus opad \,||\, GH_t(K^* \oplus D^* \oplus ipad \,||\, M))$,

$$ipad = \underbrace{(00110110)|| \ldots ||(00110110)}_{64} \in V_{512}. \quad (3.6)$$

$$opad = \underbrace{(01011100)|| \ldots ||(01011100)}_{64} \in V_{512}. \quad (3.7)$$

Етапи формування запропонованого алгоритму на основі HMAC має наступний вигляд:

1. Повідомлення ділиться на N блоків, по b біт в кожному.

2. Секретний ключ доповнюється зліва нулями до отримання b-розрядної ключа.

3. Фіксується дата та час, записуються у змінну, та доповнюється зліва нулями до отримання b-розрядного значення.

4. Виконується операція XOR результат кроку 2 та 3 з константою ipad (input pud). Величина ipad дорівнює b/8 повторень послідовності 36h.

5. Отриманий блок додається зліва до N-блоковому повідомленням.

6. Результат кроку 5 хешіруется і створюється n-розрядний хеш-образ, званий Intermediate HMAC.

7. Результат кроку 6 доповнюється зліва нулями до отримання b-розрядної блоку.

8. Кроки 2, 3, 4 повторюються з константою opad (output pud). Величина opad дорівнює b / 8 повторень послідовності 5Ch.

9. Результат кроку 8 додається зліва до блоку, отриманого на кроці 7.

10. Результат кроку 9 хешіруется з використанням тієї ж хеш-функції і створюється фінальний HMAC з використанням дати та часу.

3.3. Структурна схема алгоритму

Алгоритм HMAC (рис 3.1 – 3.3):

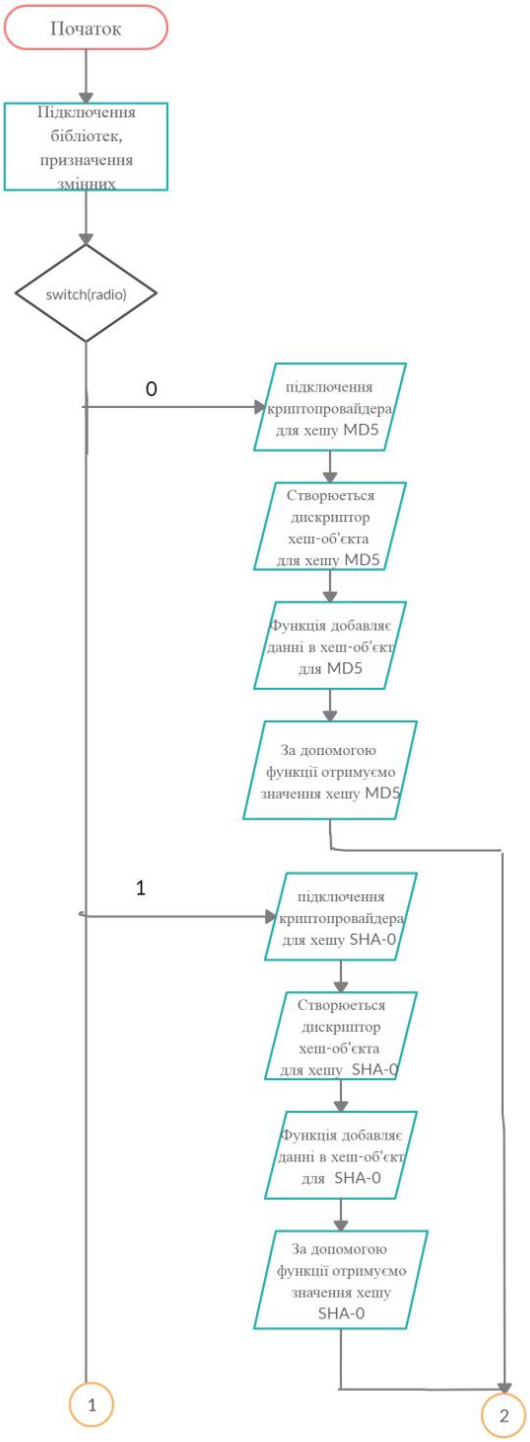


Рис. 3.2 Блок-схема № 1 MDC

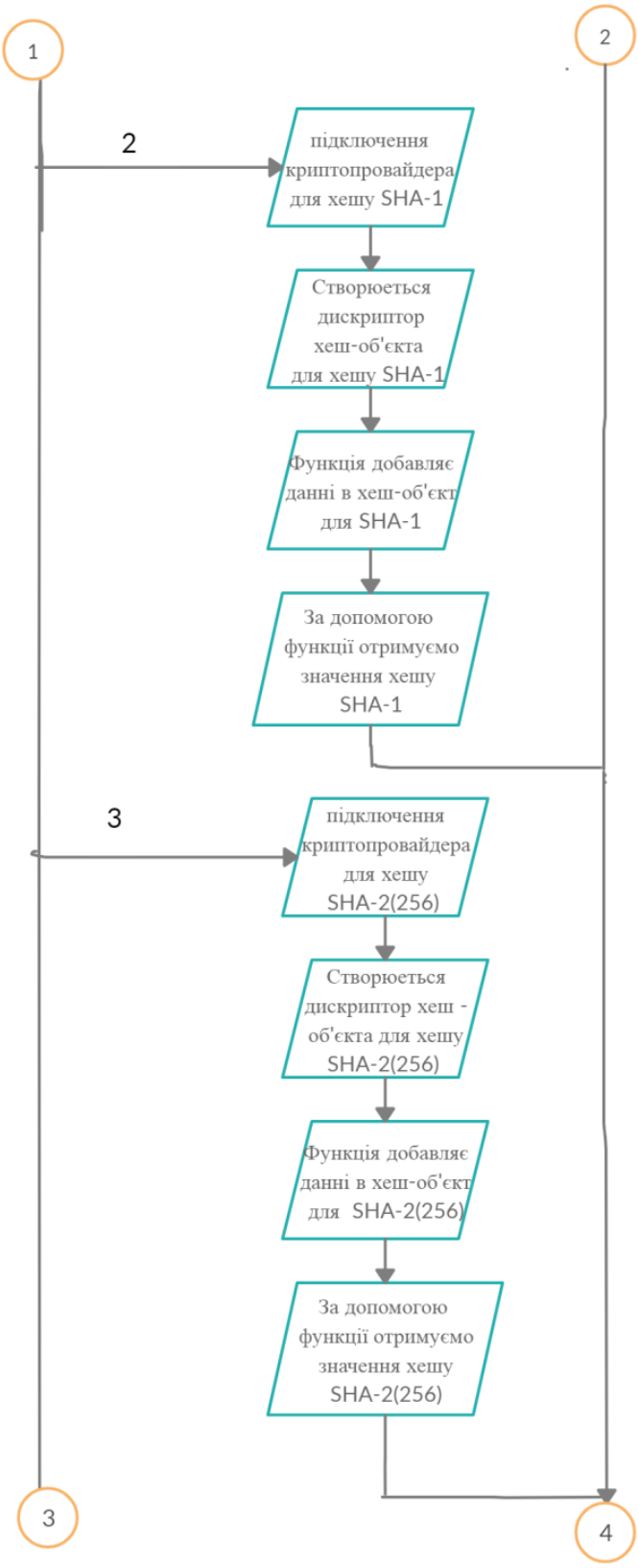


Рис. 3.1 Блок-схема № 2 MDC

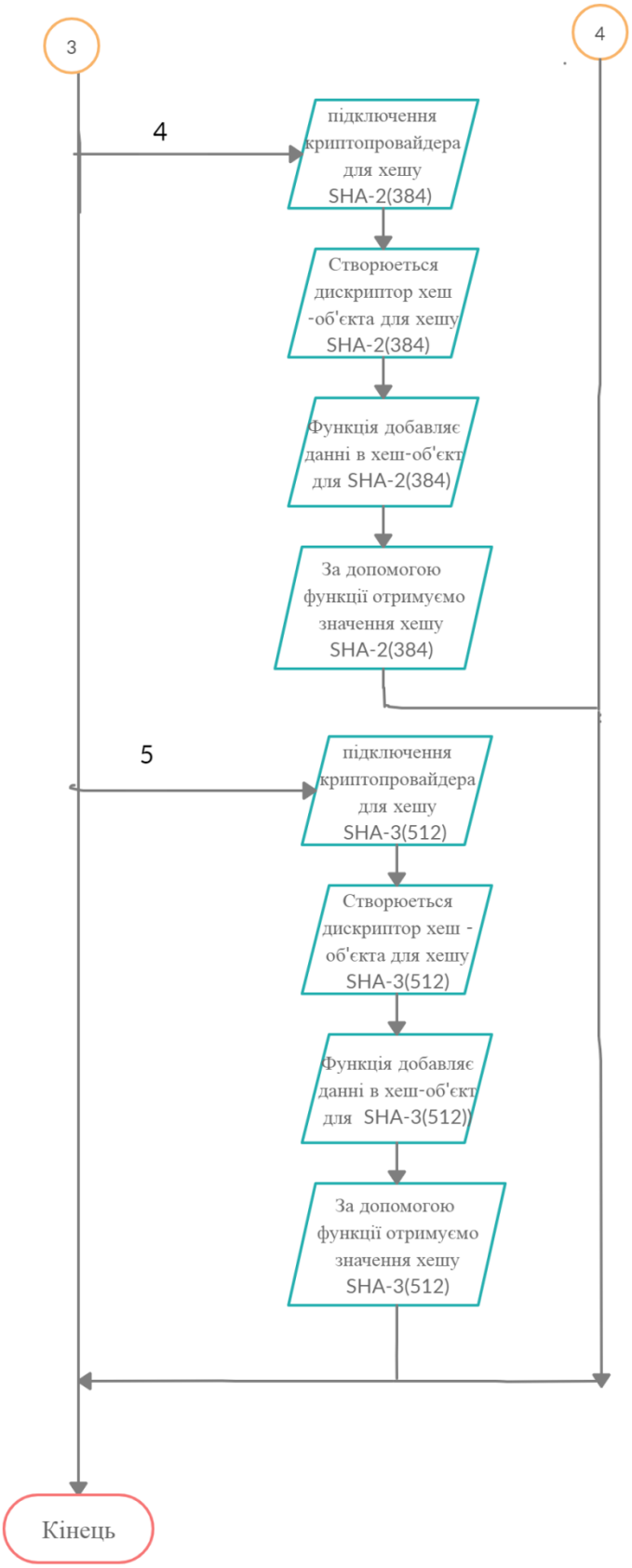


Рис. 3.3 Блок-схема № 3 MDC

Алгоритм HMAC (рис. 3.4 – 3.7):

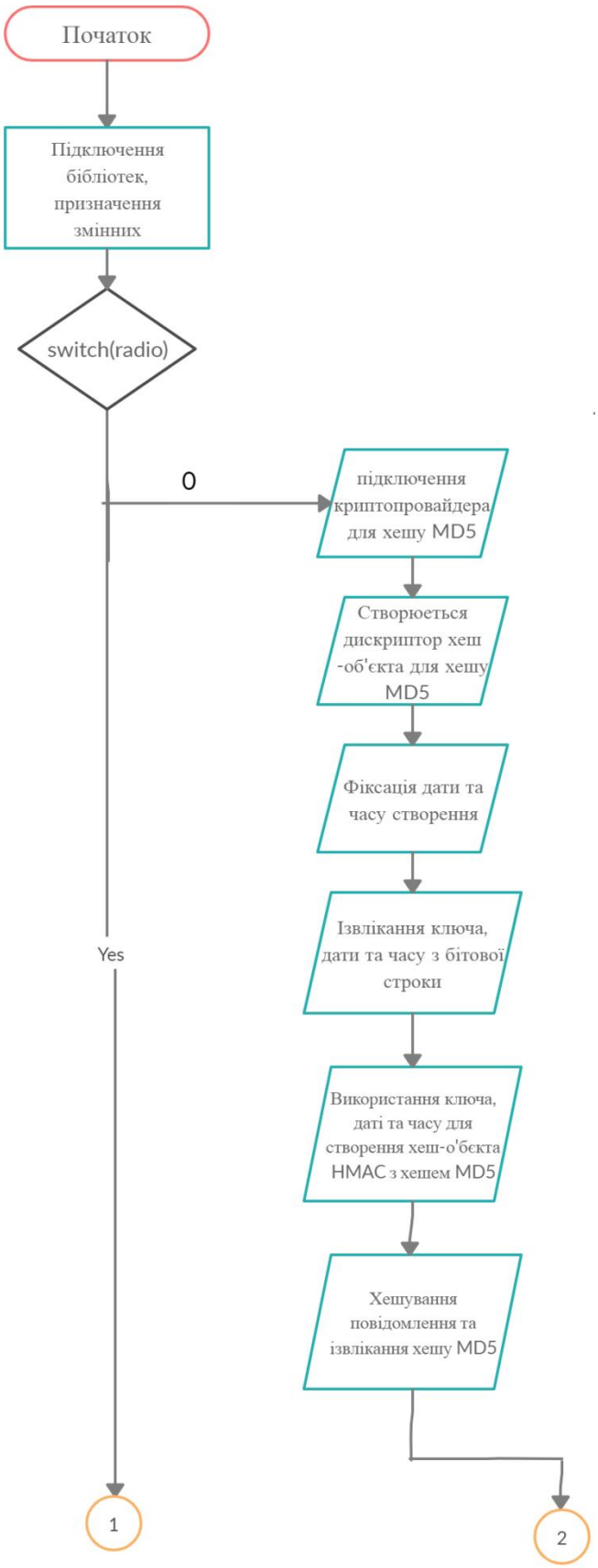


Рис. 3.4 Блок-схема № 1 HMAC

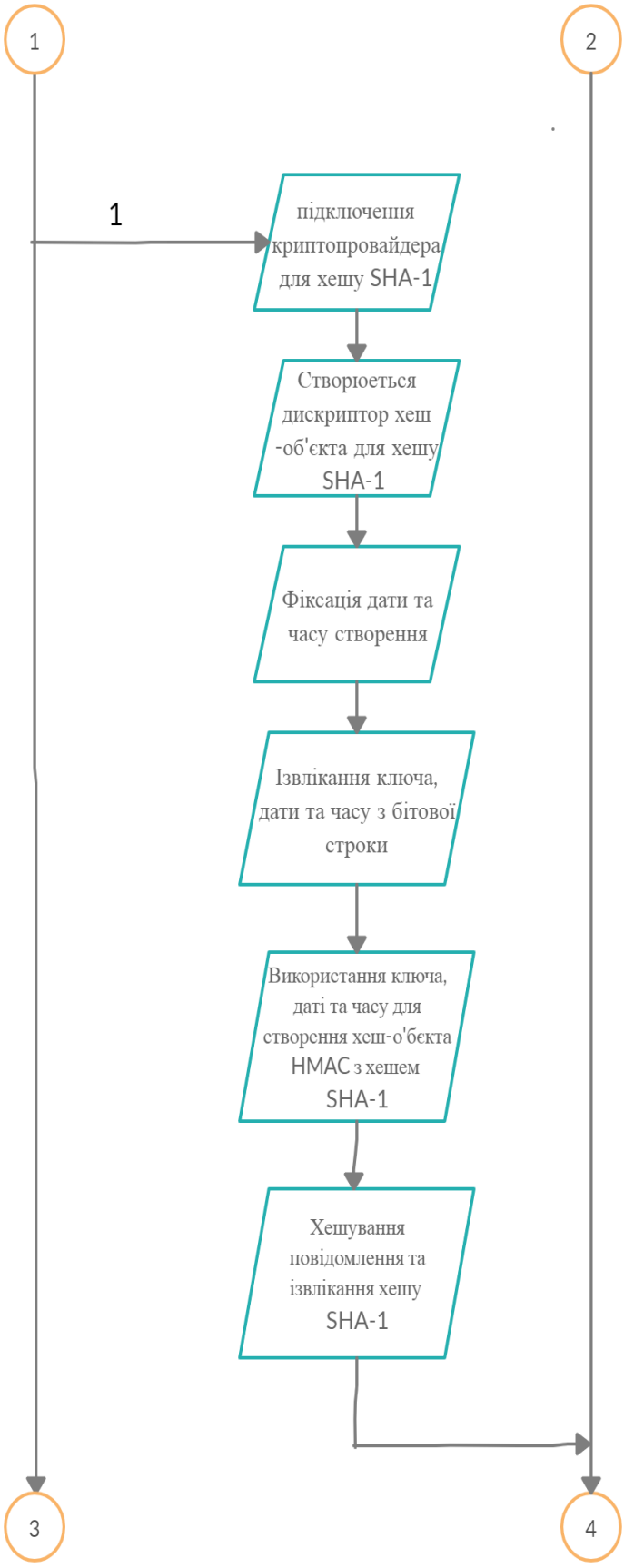


Рис. 3.5 Блок-схема № 2 HMAC

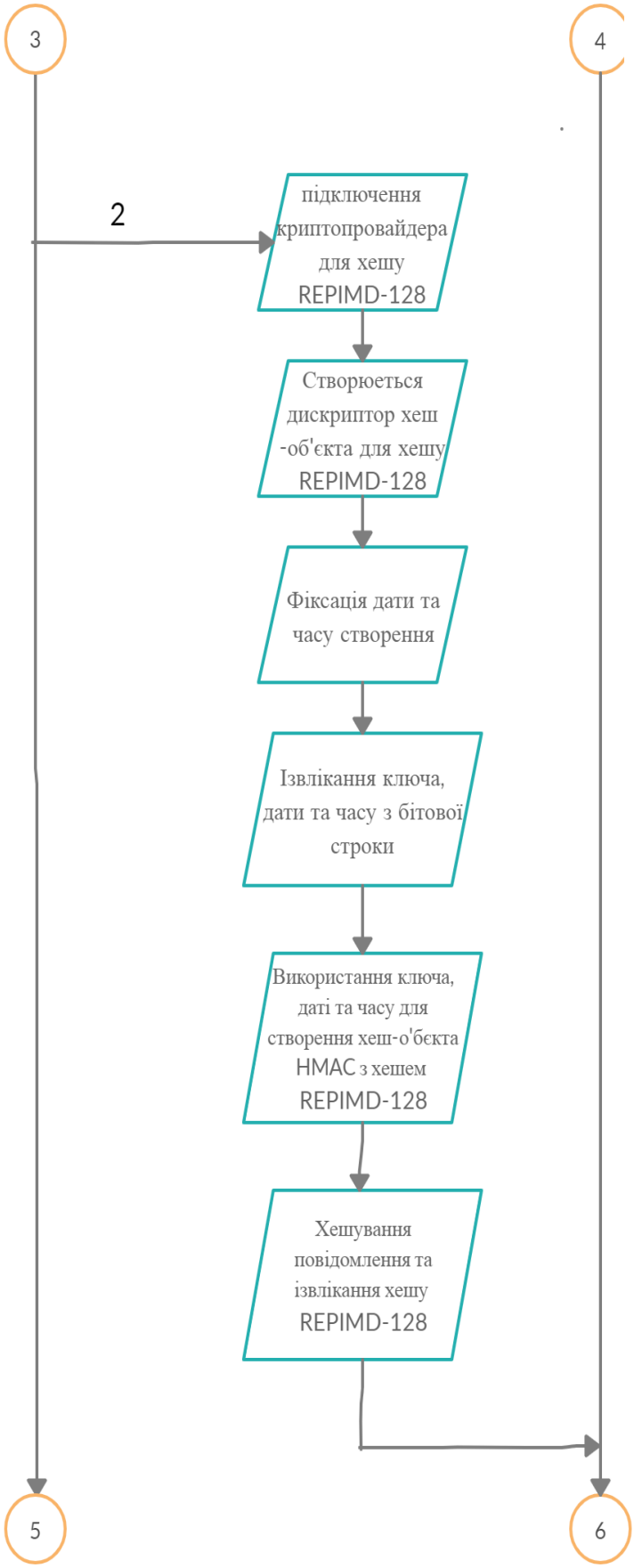


Рис. 3.6 Блок-схема № 3 HMAC

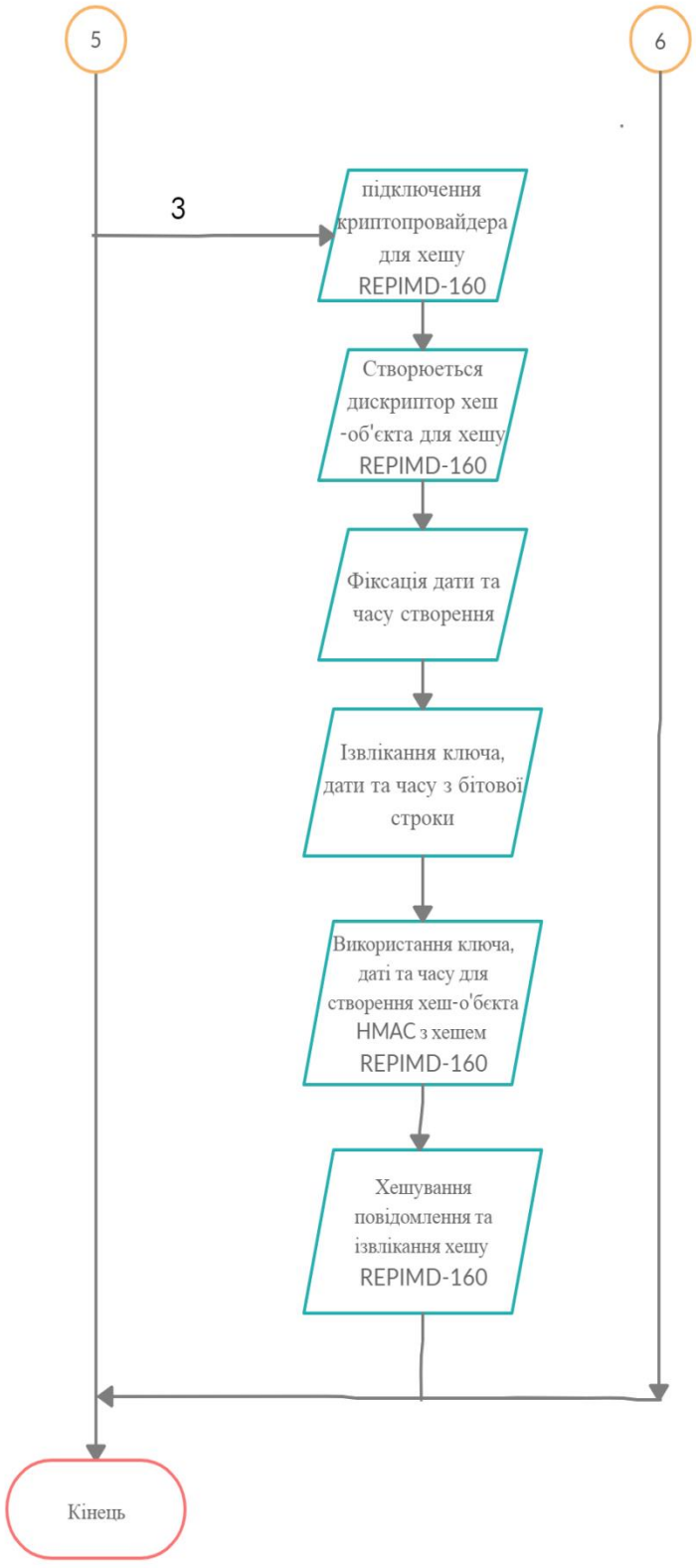


Рис. 3.7 Блок-схема № 4 HMAC

3.4. Інтерфейс користувача та функціональні можливості розробленого програмного продукту

Інтерфейс користувача розробленого програмного продукту представлений на рис. 3.8, рис 3.9. Інтерфейс складається з двох вкладок – MDC та HMAC. На кожній вкладці є три блоки: створення хешу початкового повідомлення, створення хешу перевірочного повідомлення, та перевірка на цілісність.

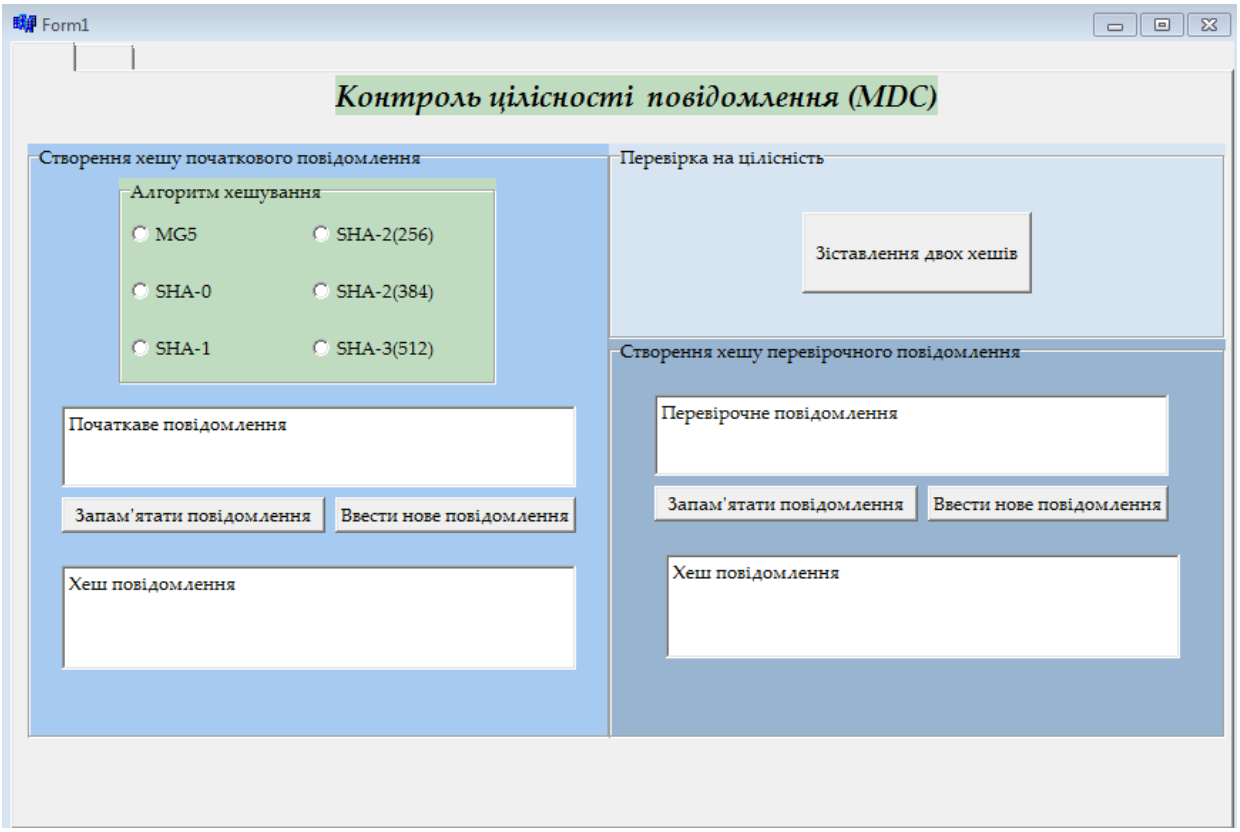


Рис. 3.8 Інтерфейс користувача розробленого програмного продукту, вкладка MDC

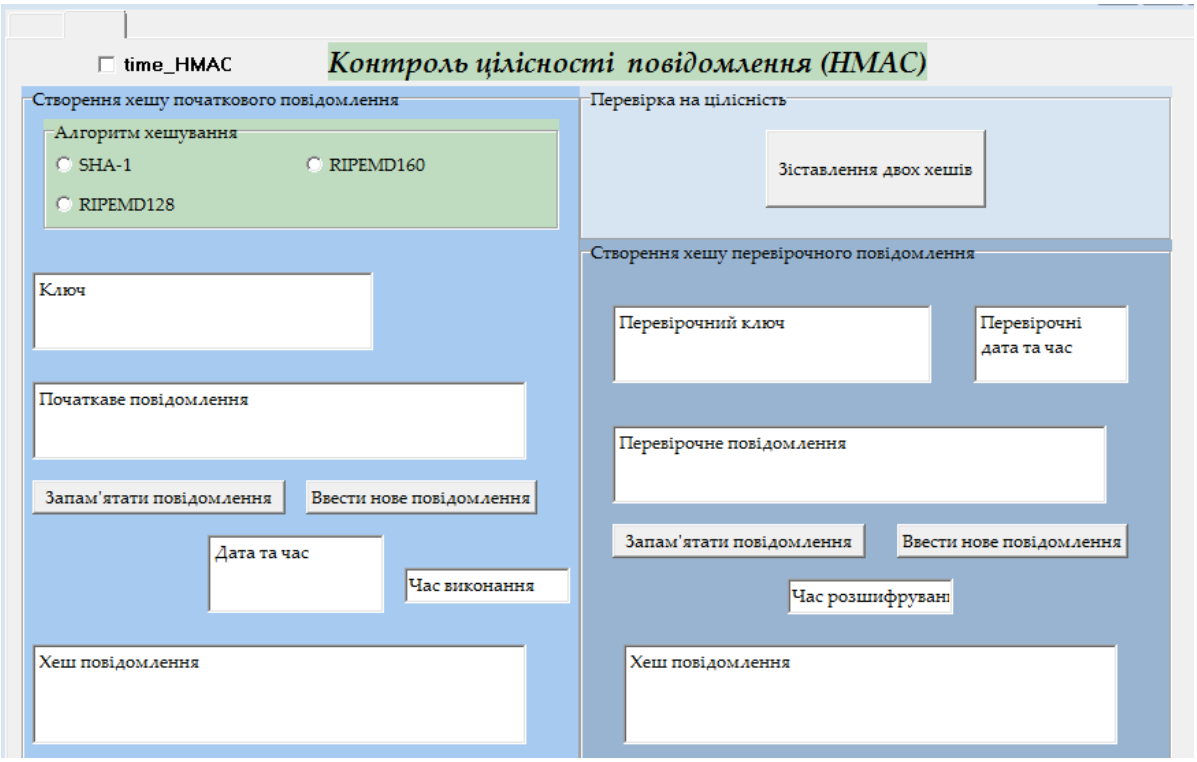


Рис. 3.9 Інтерфейс користувача розробленого програмного продукту, вкладка HMAC

У першому блоці користувачу потрібно вибрати алгоритм хешування. У вкладці MDC на вибір дається 6 алгоритмів - MD5, SHA-0, SHA-1, SHA-256, SHA-384, SHA-512. Порівняльна характеристика цих хешів представлена у розділі 2.6. У вкладці HMAC є 3 алгоритми, які RFC 2104 рекомендує використовувати. Це алгоритми SHA-1, RIPEMD128, RIPEMD160.

Вкладка MDC. Після вибору алгоритму хешування користувачу у полі «Початкове повідомлення» потрібно ввести будь-які символи. Це буде повідомлення, цілісність якого буде контролюватись за допомогою криптографічних перетворень. При натисканні кнопки «Запам’ятати повідомлення» поле з повідомленням блокується від редагування, у нижньому полі з’являється хеш введеного повідомлення за алгоритмом хешування, який був обраний, та в другому блоці у полі «Перевірочне повідомлення» дублюється текст з поля, яке містить початкове повідомлення.

Якщо користувачу потрібно змінити початкове повідомлення, він натискає кнопку «Ввести нове повідомлення», та робить необхідні зміні у полі.

Відповідно змінюється й хеш повідомлення, та автоматично записується нове повідомлення у поле «Перевірочне повідомлення» другого блоку.

Другий блок потрібен для того, щоб можна було чи змінити початкове повідомлення, чи залишити таким як є. Це повідомлення і буде порівнюватись з початковим повідомленням. Точніше, дайджест цього повідомлення. Взаємодія користувача з кнопками аналогічна першому блоку.

Третій блок «Перевірка на цілісність» містить усього одну кнопку – «Зіставлення двох хешів». При натисканні на цю кнопку відбувається зіставлення хешу початкового повідомлення та хешу перевірочного повідомлення. Якщо повідомлення зміненно не було, то хеші будуть однакові, та програма покаже повідомлення, що зміни не відбулися, та перевірка пройдена успішно. Та навпаки, якщо змінено хоч один символ, (наприклад, навіть якщо змінити у повідомленні букву «а» українського алфавіту на таку ж букву англійського алфавіту) зміниться й хеш повідомлення, та при перевірці на цілісність програма покаже повідомлення, що інформація була змінена, відповідно, перевірка не цілісність не успішна.

Вкладка HMAC. У верхньому лівому куті є бокс – “time_HMAC”. Якщо користувач не ставить галочку у цьому полі, він використовує звичайний HMAC, поле «Дата та час» та «Перевірочні дата та час» будут дорівнювати 0. Якщо у полі є галочка, то у алгоритмі HMAC при хешуванні буде використовуватись дата та час. Наступним кроком є вибір алгоритму хешування. Після вибору алгоритму хешування, потрібно увести ключ у відповідне поле, який буде використовуватись у алгоритмі. Він повинен бути у секреті, та відомий тільки двом сторонам, що обмінюються повідомленями. Долі користувачу потрібно у полі «Початкове повідомлення» ввести будь-які символи. Це буде повідомлення, цілісність якого буде контролюватись, за використанням ключа та дати і часу створення хешу. Поле «Хеш повідомлення» та «Дата та час» заблоковані для редагування та запису нової інформації. При натисканні кнопки «Запам’ятати повідомлення» поле з повідомленням блокується від редагування, у нижньому полі з’являється хеш

введеного повідомлення за алгоритмом хешування, який був обраний, з використанням подвійного хешування з ключем, датою та часом створення. Також заповнюється поле «Дата та час» відповідно часом та датою створення хешу, які використовувались у проведенні хешування разом із ключем. У той час у другому блоці у полі «Перевірочне повідомлення» дублюється текст з поля, яке містить початкове повідомлення з першого блоку відповідного поля, у полі «Перевірочний ключ» доблюється інформація з поля «Ключ» першого блоку, та поле «Перевірочні дата та час» дублюється з поля «Дата та час» першого блоку. Важливо, що у другому блоці поля «Перевірочний ключ» та «Перевірочні дата та час» не заблоковані для редагування. Тобто, користувач може отримати хеш повідомлення, яке раніше було створено у первому блоці, або виправити любі данні у полях, і отримати другий перевірочний HMAC код. Так другий користувач має змогу перевірити отриманий HMAC код, ввевші значення у відповідні поля.

Якщо користувачу потрібно змінити поле «Перевірочне повідомлення» або «Початкове повідомленні», він натискає кнопку «Ввести нове повідомлення», та робить необхідні зміні у полі. Відповідно змінюється й хеш повідомлення. Якзо зміни робиться у полі «Початкове повідомленні», автоматично записується нове повідомлення у поле «Перевірочне повідомлення» другого блоку.

Другий блок потрібен для того, щоб можна було чи змінити початкове повідомлення та інші поля, чи залишити таким як є. Це повідомлення і буде порівнюватись з початковим повідомленням. Точніше, HMAC код цього повідомлення. Взаємодія користувача з кнопками аналогічна першому блоку.

Третій блок «Перевірка на цілісність» містить усього одну кнопку – «Зіставлення двох хешів», але це найголовніший компонент розробленого програмного продукту у цій вкладці. При натисканні на цю кнопку відбувається зіставлення хешу початкового повідомлення та хешу перевірочного повідомлення. Якщо повідомлення або ключ чи дата зміненно не було, то хеші

будуть однакові, та програма покаже повідомлення, що зміни не відбулися, та перевірка пройдена успішно, та навпаки, як і в першій вкладці.

3.4.1. Приклад використання програмного забезпечення

Вкладка MDC. Перший крок використання розробленого програмного забезпечення це вибір алгоритму хешування. Оберем, наприклад алгоритм SHA-384.

В поле з початковим повідомленням введемо такий текст: «Огіренко Марина, 242М група, НАУ», та натиснемо клавішу «Запам’ятати повідомлення». З’явився хеш цього повідомлення, та дублювання тексту повідомлення у поле з перевірочним повідомленням (рис. 3.10).

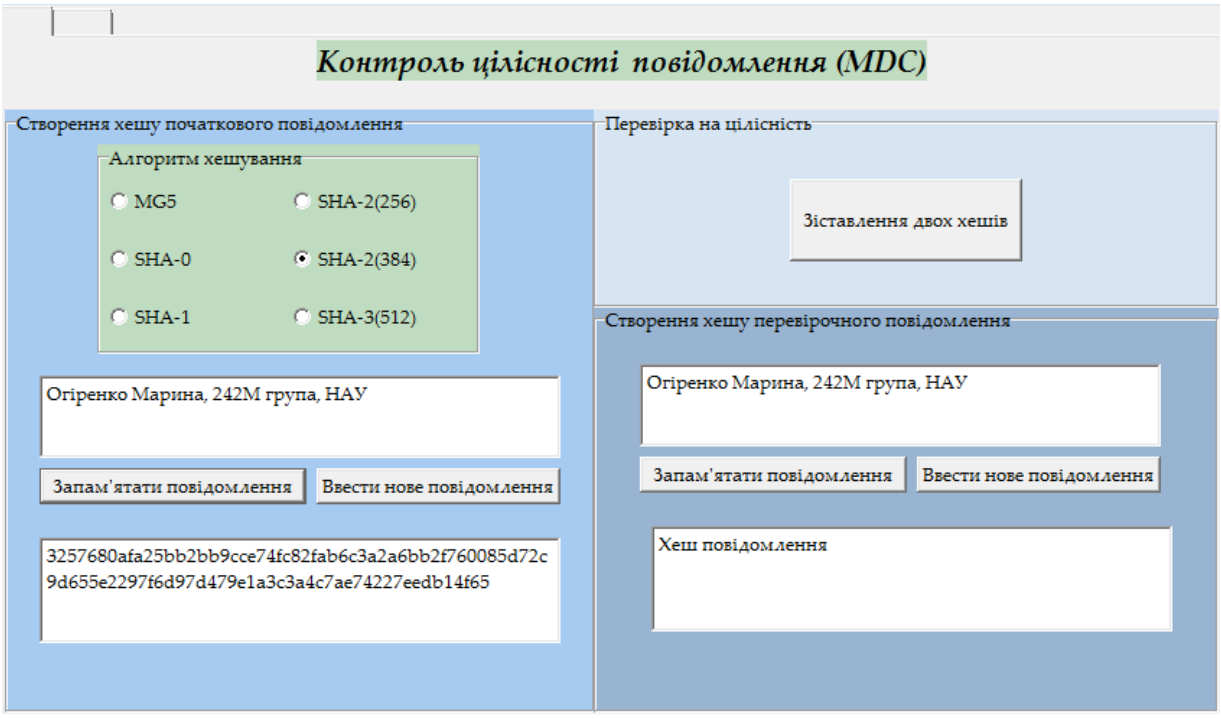


Рис. 3.10 Тестування програми

Не будемо змінювати перевірочне повідомлення, розрахуємо його хеш, та відразу перевіримо повідомлення на цілісність, нажавши кнопку у третьому блоці.

Отримуєм зворотній відгук програми про те, що повідомлення змінено не було (рис 3.11). Отже, цілісність не пошкоджена, що й треба було довести.

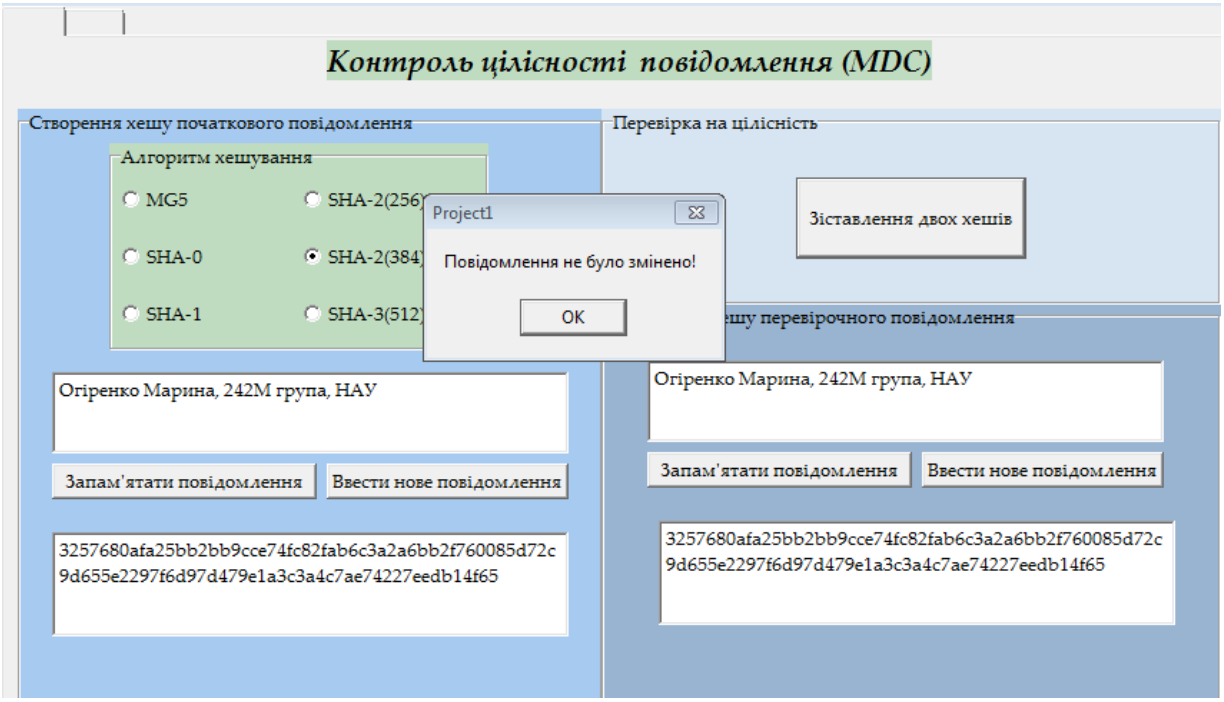


Рис. 3.11 Тестування програми: повідомлення не змінено

Тепер спробуємо змінити повідомлення. У другому блоці натискаємо на кнопку «Ввести нове повідомлення» та вводимо те ж саме повідомлення, тільки у слові «НАУ» змінюємо букву «А» з українського алфавіту на букву з англійського алфавіту. Натискаємо кнопку «Запам’ятати повідомлення! У другому блоці, та отримуємо хеш зміненого повідомлення (рис. 3.12).

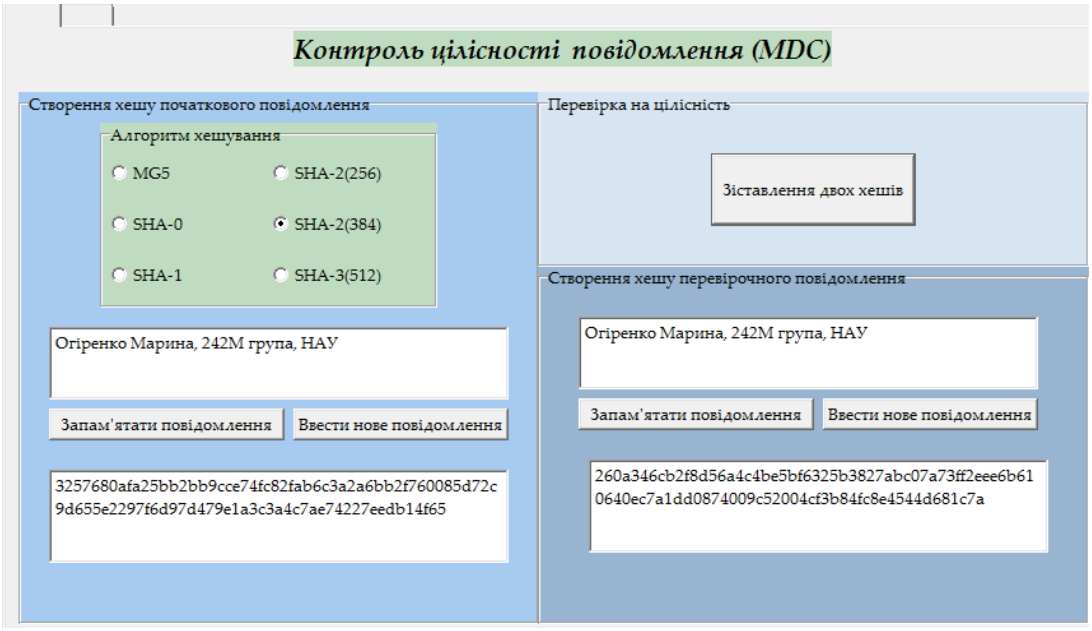


Рис. 3.12 Тестування програми: повідомлення змінено

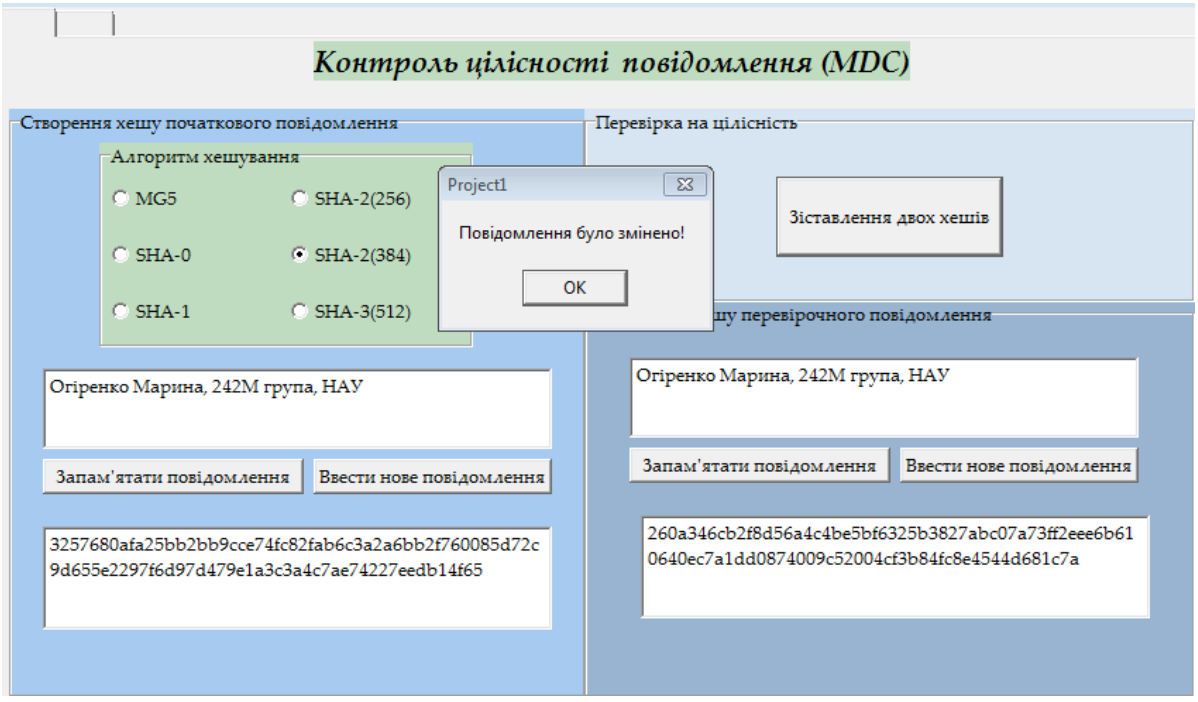


Рис. 3.13 Тестування програми: повідомлення змінено

Як можно побачіти, візуально повідомлення однакове, та, можна сказати, що початкове повідомлення не підвергалося змінам, та його цілісність не порушилась. Але хеш коди початкового та перевірочного повідомлень координально відрізняються один від одного. Тому можна зробити висновок, що повідомлення було змінено. Що й підтверджує повідомлення, яке виникає при натисканні на кнопку у блоці «Перевірка на цілісність» (рис. 3.13).

Вкладка HMAC. Перший крок використання розробленого програмного забезпечення це вибір – використовувати дату та час чи ні. Спочатку використаємо звичайний HMAC. Наступний крок - вибір алгоритму хешування. Оберем, наприклад алгоритм SHA1.

В поле з початковим повідомленням введемо такий текст: «Огіренко Марина, 242М група, НАУ», ключ: «secret NAU OM», та натиснемо клавішу «Запам’ятати повідомлення». З’явився хеш цього повідомлення, та дублювання тексту повідомлення у поле з перевірочним повідомленням (рис. 3.14).

Не будемо змінювати перевірочне повідомлення, ключ, та дату, розрахуємо хеш, та відразу перевіримо повідомлення на цілісність, нажавши кнопку у третьому блоці (рис. 3.15).

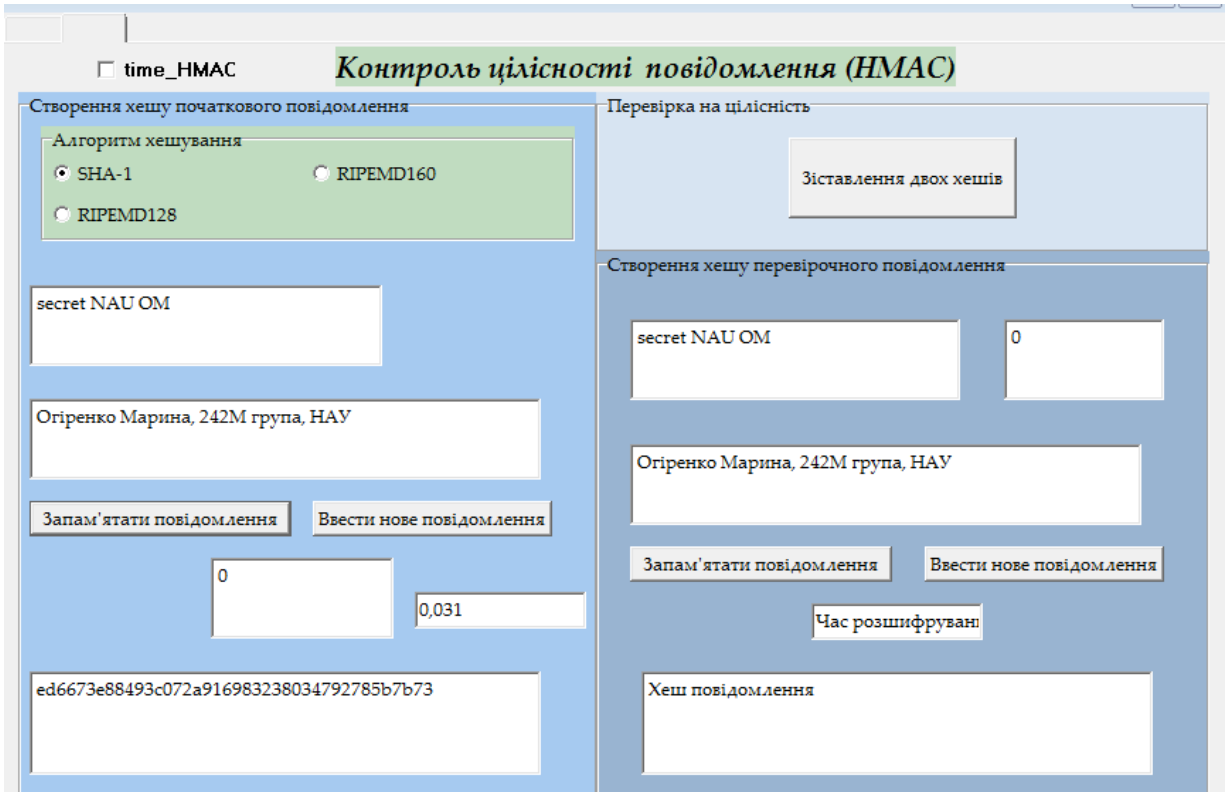


Рис. 3.14 Тестування програми: отримали HMAC

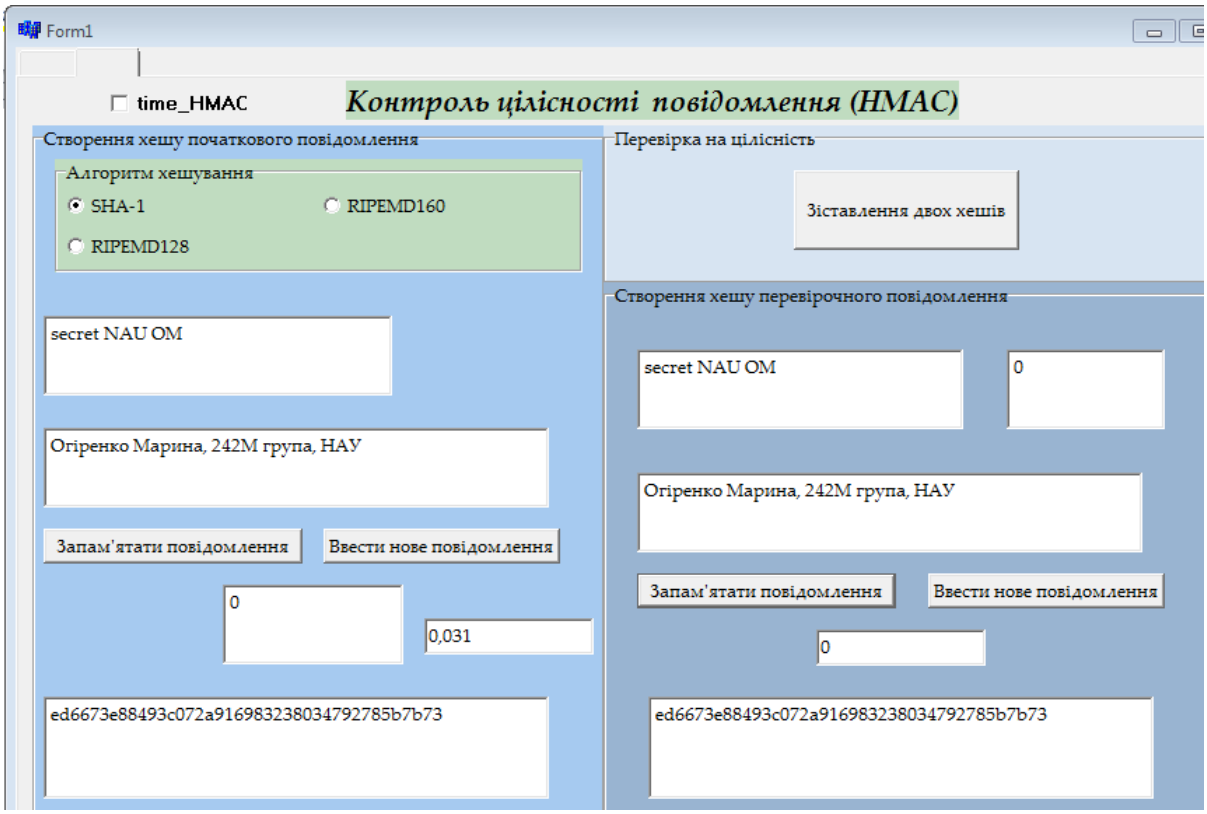


Рис. 3.15 Тестування програми: отримали перевірочний HMAC

Отримуєм зворотній відгук програми про те, що повідомлення змінено не було (рис 3.16). Отже, цілісність не пошкоджена.

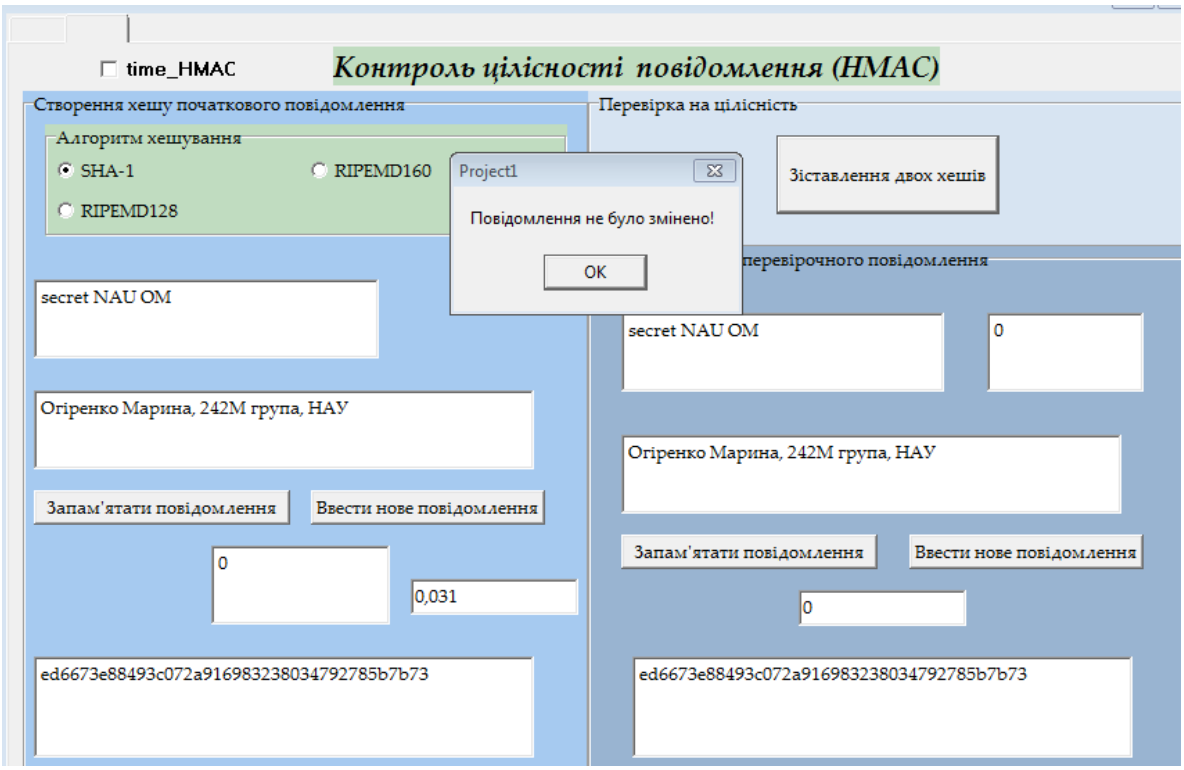


Рис. 3.16 Тестування програми: перевірка отриманих хешів

Тепер спробуємо змінити повідомлення. У другому блоці натискаємо на кнопку «Ввести нове повідомлення» та вводимо те ж саме повідомлення, тільки у слові «Марина» змінюємо букву «а» з українського алфавіту на букву з англійського алфавіту. Натискаємо кнопку «Запам’ятати повідомлення! У другому блоці, та отримуємо хеш зміненого повідомлення (рис. 3.17).

Як можно побачіти, візуально повідомлення однакове, та, можна сказати, що візуально початкове повідомлення не підвергалося змінам, та його цілісність не порушилась. Але хеш коди початкового та перевірочного повідомлень координально відрізняються один від одного. Тому можна зробити висновок, що повідомлення було змінено. Що й підтверджує повідомлення, яке виникає при натисканні на кнопку у блоці «Перевірка на цілісність» (рис. 3.18).

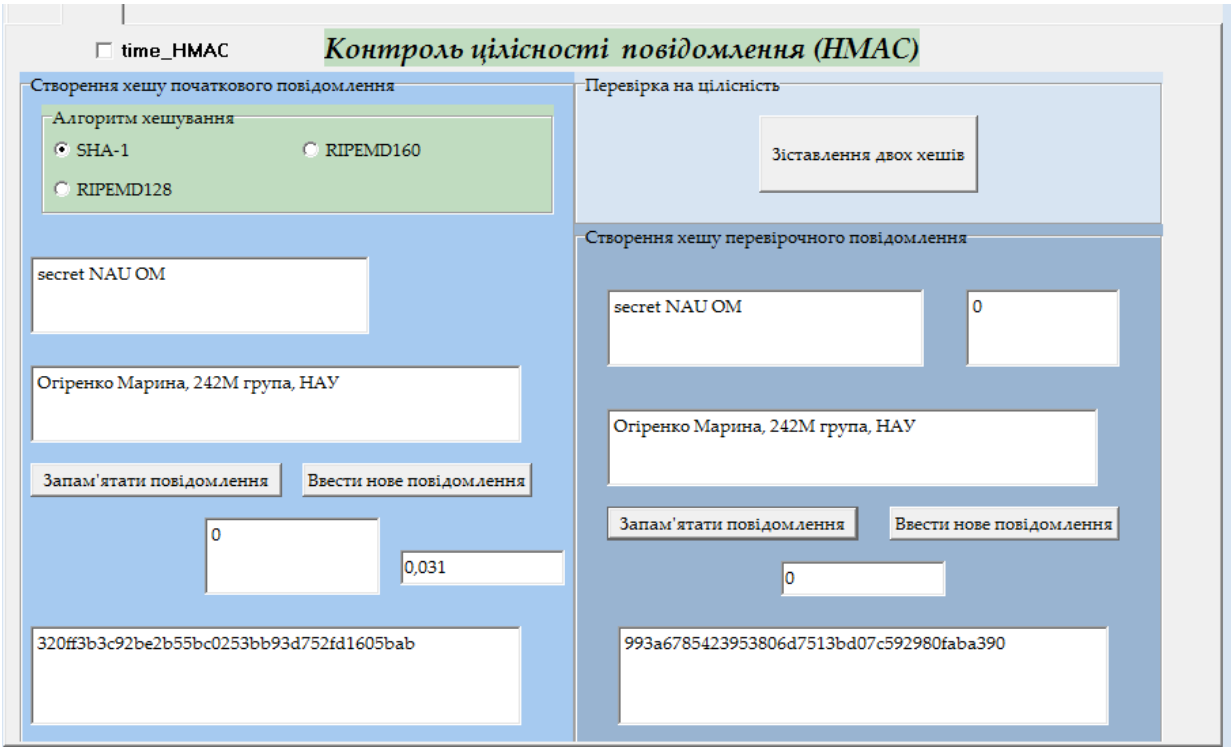


Рис. 3.17 Тестування програми: зміна повідомлення

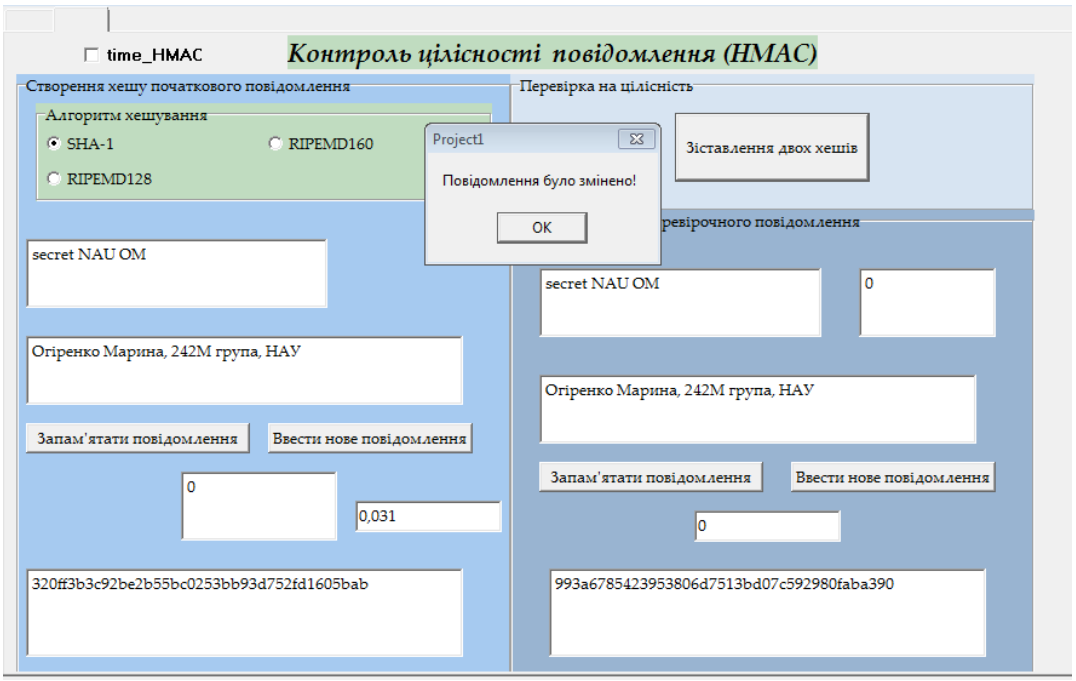


Рис. 3.18 Тестування програми: при зміні букви змінюється хеш повідомлення, отже цілістність було порушено.

Також важливо, щоб два користувачі мали однаковий секретний ключ. Зробимо зміну ключа у другому блоці, та вернем однакове значення вихідного повідомлення у першому а другому блоці, обчислимо хеші та перевірима їх (рис. 3.19).

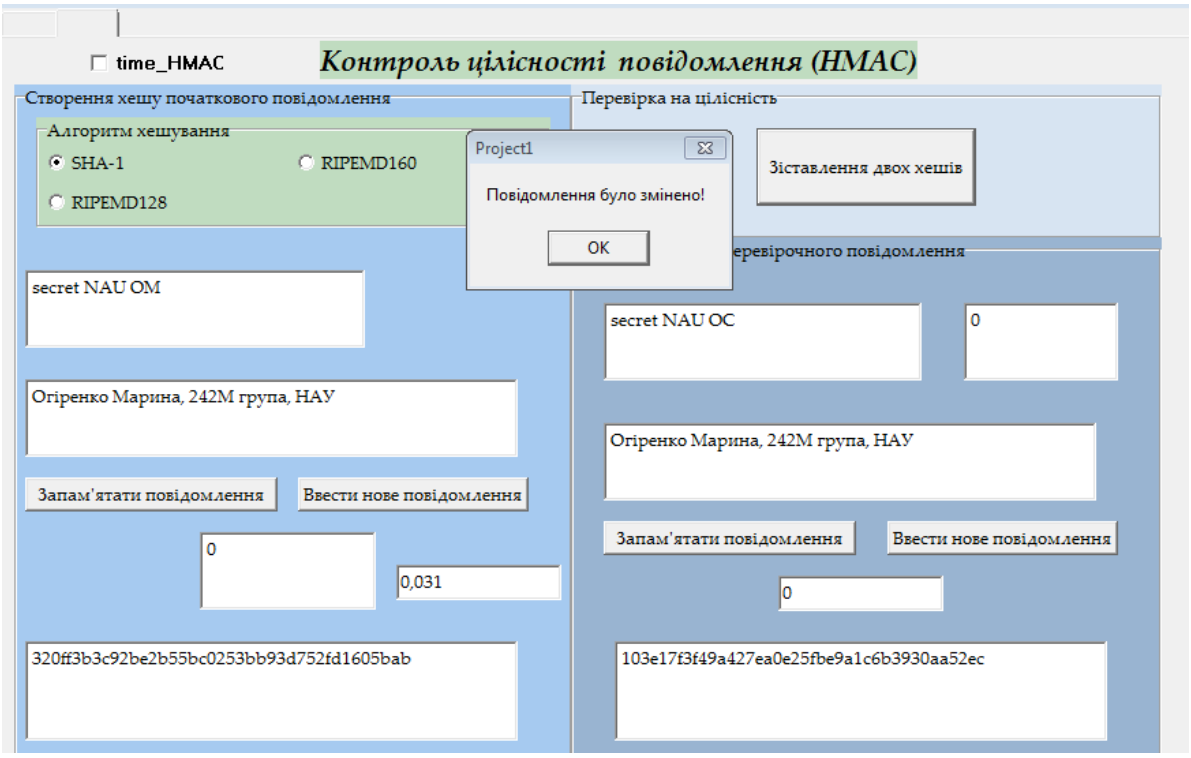


Рис. 3.19 Тестування програми: однакове повідомлення, зміна ключа

Як видно на рис. 3.19 хеші не сбігаются при однаковому повідомленні, але з різними ключами. Це означає, якщо противник захоче перехопити повідомлення на підробити HMAC код, не знаючи ключа це буде трудно зробити, але реально знайти ключ методом перебору. Оскільки у запропонованому алгоритмі використовувалися у процесі шифрування дата та час, навіть повним перебором буде дуже довго знайти потрібне значення.

Використаємо алгоритм “time_HMAC”. Оберемо той самий алгоритм, повідомлення, та ключ (SHA1, “ Огіренко Марина, 242М група, НАУ”, “secret NAU OM”).

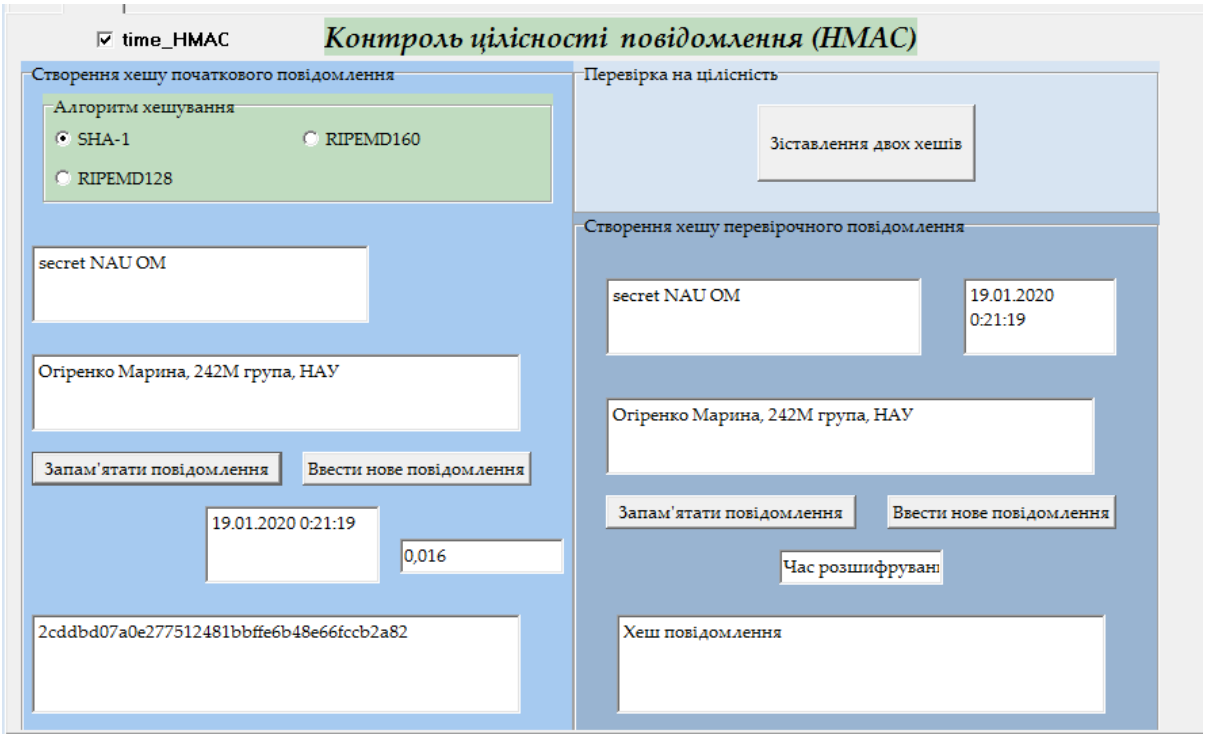


Рис. 3.20 Використання time_HMAC

Як видно на рис. 3.20 у нас з’явилась у полі дата та час виконання хешування, та, якщо порівнювати в рис. 3.14, у якому ми використовували тільки HMAC, змінился хеш повідомлення, при однакових вхідних значеннях.

Для порівняння, використаємо алгоритм REPIMD з такими ж вхідними значеннями (рис. 3.21):

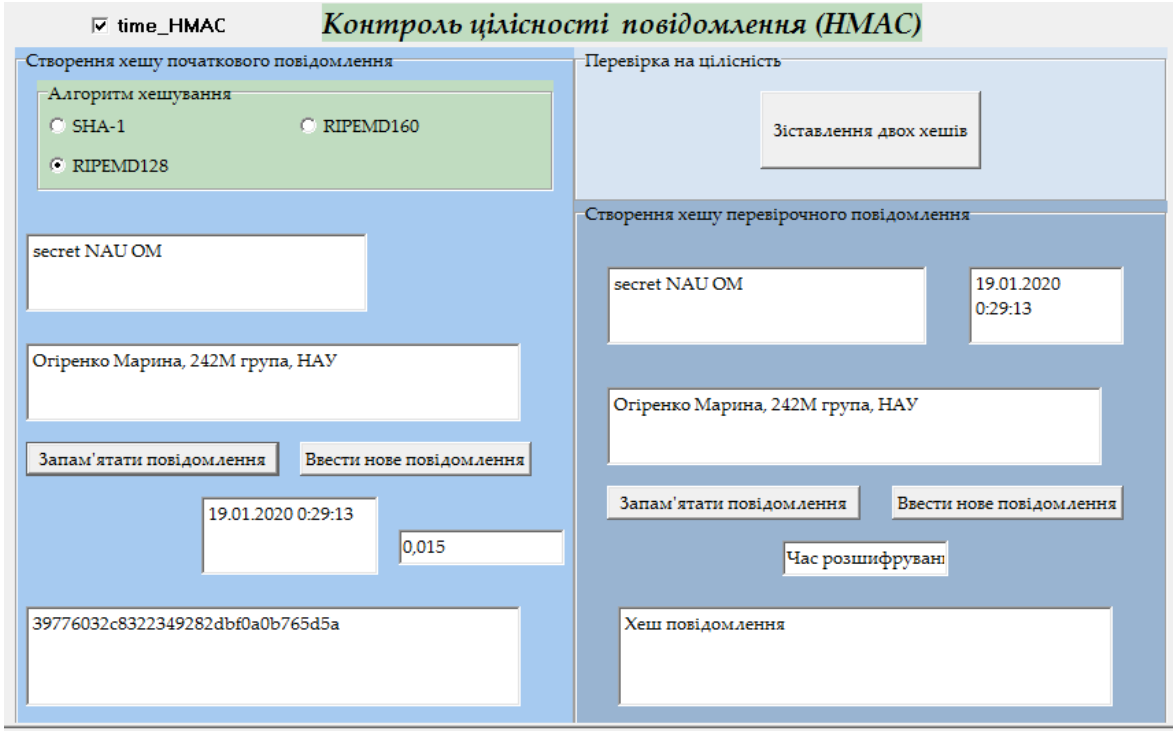


Рис. 3.21 Використання RIPEMD128 та time_HMAC

Змінимо у другому блоці поле «Перевірочні дата та час», та перевіримо два хеши на відповідність (рис. 3.22).

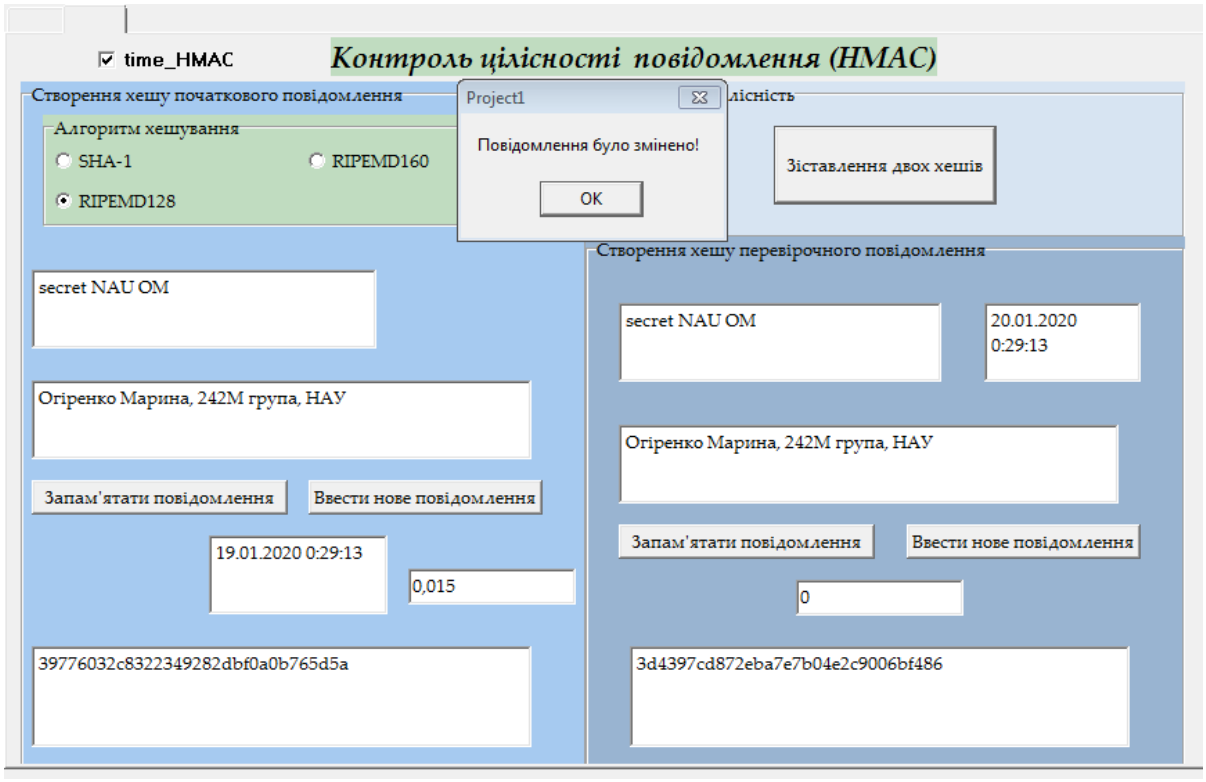


Рис. 3.22 Використання RIPEMD128 та time_HMAC, змінена дата

Як видно з рис. 3.22, 3.19, 3.17, якщо змінити час/ключ шифрування/вхідне повідомлення, зміниться й хеш. Тобто, якщо противник захоче перехопити повідомлення та змінити його, це буде відразу помітно.

3.5. Оцінка ефективності розробленої системи контролю цілісності інформації

У запронованій програмній реалізації використовувався MDC та HMAC коди. Захищеність запропонованих тут кодів аутентифікації повідомлень залежить від криптографічних властивостей хеш-функції H - стійкості до пошуку колізій (для випадку, коли початкове значення випадково і зберігається в секреті, а висновок функції не доступний атакуючому в явному вигляді) і можливостей аутентифікації повідомлень функції стиснення в складі H у випадках застосування до одного блоку (в HMAC ці блоки частково не відомі атакуючому, оскільки вони містять результат внутрішнього розрахунку H і, зокрема, не можуть бути повністю вибрані атакуючим).

Безпека MDC кода у першу чергу залежить від використаної хеш-функції. Також зловмисник може обчислити MDC код без секретних даних, та може легко його сфабрикувати. Оскільки MAC та HMAC використовуюсь секретний ключ, сфабрикувати його набагато складніше. Використання алгоритму дозволяє переконатися в цілісності даних, відсутність яких-небудь змін з моменту створення, передачі або збереження довіреного джерела. Для перевірки такого роду необхідно, щоб дві сторони, які беруть участь в процесі обміну, довіряли один одному і заздалегідь домовилися про використання секретного ключа, який відомий тільки їм. Тим самим гарантується автентичність джерела та шляхів сполучення. Недолік цього підходу - необхідна наявність двох довірених один одному сторін. Використання MAC та HMAC вимагає, щоб попередньо між учасниками інформаційного обміну були розподілені ключі, тож вимагає секретності.

Щоб поліпшити безпеку MAC, був розроблений вкладений HMAC, в якому хешування робиться в два кроки. На першому кроці ключ конкатенуються з повідомленням і хешіруется, щоб створити проміжний дайджест, на другому кроці ключ конкатенуються з проміжним дайджестом, щоб створити кінцевий дайджест. Так як саме секретне значення з повідомленням не пересилається, у порушника немає можливості модифікувати перехоплений повідомлення. До тих пір поки секретне значення залишається секретним, порушник не може нав'язувати помилкові повідомлення, що фактично забезпечує основну перевага HMAC в порівнянні з іншими схемами, заснованими на використанні хешування. HMAC забезпечує гарантовану захищеність за умови, що вбудована функція хешування володіє необхідною криптографічного стійкістю.

Порівняємо час, необхідний для створення та перевірки алгоритму HMAC та data_HMAC, на основі прикладів, які були використані у главі 3.4.1 (поле «Час виконання» у першому на другому блоках). Дані представлені у табл. 3.1.

Табл. 3. 1

Порівняльний аналіз методів контролю цілісності інформації

Метод контролю цілісності	Використані методи хешування	Розмір хешу (біт)	Час формування хеш значення (сек)
Класичний метод HMAC	SHA-1	160	0,31
Удосконалений метод HMAC_time	SHA-1	160	0,16
Удосконалений метод HMAC_time	RIPEMD128	128	0,15

У цій роботі був предложений оновлений алгоритм HMAC з використанням дати та часу хешування, які хешуються разом із ключом. Оскільки найбільш поширена атака проти HMAC є атака грубою силою, щоб розкрити секретний ключ, використання додаткових даних ускладнює здійсненя цієї атаки. Як видно з таблиці 3.1, хешування з використанням дати та часу забеспечує не тількі цілісності даних та цілісності джерела даних, а й пришвидшує час формування перевірочного коду.

Висновки до розділу 3

У даному розділі був розроблений програмний продукт, що забезпечує контроль цілісності інформації на основі використання MDC-кодів. Програма написана на мові програмування C ++ в середовищі розробки Borland C ++ Builder 6. Також у своїй роботі я використовувала бібліотеку CryptoAPI - набір програмних інтерфейсів Windows для роботи з криптографією.

У програмному продукті можливо робити вибір алгоритмів хешування (MD5, SHA-0, SHA-1, SHA-256, SHA-384, SHA-512, RIPEMD), та вибрати код аутентицікації повідомлення (MDC, HMAC), за допомогою якого і буде контролюватись цілісність повідомлення. Користувач може обчислювати код виявлення маніпуляцій з даними за вхідним повідомленням, змінювати текст повідомлення, та безпосередньо контролювати цілісність повідомлення (якщо повідомлення було змінено, то зміниться його хеш-код, та при зіставленні його з хеш-кодом вхідного повідомлення програма видаст повідомлення, що контроль цілісності не пройден, повідомлення було змінено). Також він може використати запропонований алгоритм з використаннім дати та часу, який забеспечує не тількі цілісності даних, а і цілісність джерела даних.

Розроблений програмний продукт добре контролює порушення цілісності повідомлення, використовуючи криптографічні методи, зокрема: коди виявлення маніпуляцій з даними та алгоритми безпечного хешування.

ВИСНОВКИ

У роботі було досліджено основи організації захисту в сучасних комп’ютерних системах та мережах, проведен аналіз нормативно-правових основ захисту інформації, здійснена характеристика сучасних напрямів та методів захисту інформаційних ресурсів, розроблений алгоритм проведення контролю цілісності інформації з використанням криптографічних перетворень.

Кожне повідомлення передається по каналу зв’язку, тому в першу чергу була дана характеристика поняття контролю цілісності, та розглянуто методи, за допомогою яких можливо здійснити контроль цілісності потоку повідомлень.

Далі були розглянуті сучасні методи контролю цілісності на базі криптографічних перетворень а також алгоритми хешування. Визначено, що виділяють два основних криптографічних підходи до вирішення завдання захисту інформації від несанкціонованих змін даних, які передбачають формування: коду автентифікації повідомлень MAC, та коду виявлення маніпуляцій з даними MDC.

Розглянуто логіку виконання двох алгоритмів хешування: MD5 та SHA-1. Був зроблений порівняльний аналіз розглянутих алгоритмів, та таблиця існуючих алгоритмів безпечного хешування.

У ході порівняльного аналіза методів контролю цілісності інформацію, був зроблений висновок, що підхід на основі MAC вимагає для обчислення контрольної комбінації секретного ключа, для MDC це не потрібно. Зловмисник не зможе обчислити MAC для довільного сфабрикованого їм повідомлення, у той час, зможе вирахувати MDC, так як для цього не потрібно ніяких секретних даних. Але використання MAC вимагає, щоб попередньо між учасниками інформаційного обміну були розподілені ключі. А для передачі ж MDC потрібен канал, що забезпечує тільки справжність переданих даних, вимога секретності відсутня, і це робить даний метод переважним при одноразової передачі даних.

Схеми хешування в поєднанні з кодами аутентифікації повідомлень (MAC) використовуються для досягнення як цілісності даних, так і цілісності джерела даних і називаються заснованими на хеші MAC (HMAC). На відміну від звичайного HMAC, алгоритм, запропонований в цій роботі, хешування повідомлення виконується з використанням інформації, специфічної для користувача, тобто інформації про дату, час і ключі, і, отже, схема не залежить тільки від самого повідомлення. Основна увага приділяється як цілісності даних, так і забезпечення цілісності джерела даних, що грає так само велику роль у захисті інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ:

1. Загальні положення з захисту інформації в комп'ютерних системах від НСД: НД ТЗІ 1.1-002-99. [Чинний від 1999.04.28]. К. : ДСТСЗІ СБУ, 1999. № 22. (Нормативний документ системи технічного захисту інформації).
2. Класифікація автоматизованих систем і стандартні функціональні профілі захищеності оброблюваної інформації від несанкціонованого доступу: НД ТЗІ 2.5-005-99. [Чинний від 1999.04.28]. К. : ДСТСЗІ СБУ, 1999. № 22. — (Нормативний документ системи технічного захисту інформації.
3. Критерії оцінки захищеності інформації в комп'ютерних системах від несанкціонованого доступу: НД ТЗІ 2.5-004-99. [Чинний від 1999.04.28]. К. : ДСТСЗІ СБУ, 1999. № 22. (Нормативний документ системи технічного захисту інформації).
4. Методичні вказівки щодо розробки технічного завдання на створення комплексної системи захисту інформації в автоматизованій системі: НД ТЗІ 3.7-001-99. [Чинний від 1999.04.28]. К. : ДСТСЗІ СБУ, 1999. № 22. (Нормативний документ системи технічного захисту інформації).
5. Про захист інформації в автоматизованих системах: Закон України від 05.07.1994 № 81/94-ВР. ВВР. 1994. № 31. С. 287.
6. Типове положення про службу захисту інформації в автоматизованій системі: 1.4-001-2000. [Чинний від 2000.12.04]. К. : ДСТСЗІ СБУ, 2000. № 53. (Нормативний документ системи технічного захисту інформації).
7. Юдін О.К. Інформаційна безпека. Нормативно-правове-забезпечення: підруч. К.: Вид-во Видавництво Національного авіаційного університету «НАУ-друк», 2011. 640 с.

8. The Cryptix Foundation Limited and David Hopwood. Standard Cryptographic Algorithm Naming. Version 1.0.20a - 22 October, 2002. URL: http://www.users.zetnet.co.uk/hopwood/crypto/scan/md.html
9. Gary C. Kessler. An Overview of Cryptography. 1998-2020 r. URL: https://www.garykessler.net/library/crypto.html
10. Handbook of Applied Cryptography, by A. Menezes, P. van Oorschot, and S. Vanstone, CRC Press, 1996.
11. Cryptography tutorial. 2015 by Tutorials Point (I) Pvt. Ltd
12. Authenticated encryption. From Wikipedia, the free encyclopedia. URL: https://en.wikipedia.org/wiki/Authenticated_encryption#Authenticated_encryption_with_associated_data
13. Dobbertin H. «The Status of MD5 After a Recent Attack», RSA Labs' CryptoBytes, Vol. 2 No. 2, Summer 1996.
14. Meyer, S. and Matyas, S.M., Cryptography, New York Wiley, 1982.
15. NIST, FIPS PUB 180-1: Secure Hash Standard, April 1995.
16. Dobbertin H, Bosselaers A. and Preneel B. «RIPEMD-160: A strengthened version of RIPEMD», Fast Software Encryption, LNCS Vol 1039, pp. 71-82.
17. Preneel B.and P. van Oorschot. «Building fast MACs from hash functions», Advances in Cryptology — CRYPTO'95 Proceedings, Lecture Notes in Computer Science, Springer-Verlag Vol.963, 1995, pp. 1-14.
18. .Bellare M, Canetti R. and Krawczyk H.. «Keyed Hash Functions and Message Authentication», Proceedings of Crypto'96, LNCS 1109, pp. 1-15.
19. Mosenia A., Sur-Kolay S., Raghunathan A., Jha N. B. The Internet of Things security: a survey encompassing enabling technologies, protocols, and applications. Computers & Security. 2021. 102494.
20. Sadhu P. K., Pattanayak B. K., Sahoo S. K. Internet of Things: Security and Solutions Survey. Sensors. 2022. 22(19):7433. DOI:10.3390/s22197433.
21. Zou J., Ye X., Choo K. K. R., Xiao L. Integrated Blockchain and Cloud Computing Systems: A Systematic Survey, Solutions, and Challenges. ACM Computing Surveys. 2021. 54(8):160. DOI:10.1145/3466658.

22. Zhu X., Li J., Wang W., Gong X. Digital transformation: A systematic literature review and future research directions. Computers in Industry. 2021. 123:103333.
23. IEEE Digital Reality Initiative. Digital Transformation – White Paper. Piscataway: IEEE, 2019.
24. Proceedings of the IEEE. Special Issue on Edge Computing. 2019. Vol. 107, No. 8.
25. Azad N., Shahin M., Liang P., Babar M. A. DevOps critical success factors – A systematic literature review. Information and Software Technology. 2023. 160:107198.
26. 26. Hassan H. B., Bahsoon R., Kazman R. Survey on serverless computing // arXiv preprint. 2021. arXiv:2106.11773.
27. Dizdarevic J., Carpio F., Jukan A., Masip-Bruin X. A Survey of Communication Protocols for Internet-of-Things and Related Challenges of Fog and Cloud Computing Integration. ACM Computing Surveys. 2019. 51(6):116.
28. Winkler F., Dörнep R., Winner K. A Systematic Literature Review of DevOps Success Factors . Proceedings of the International Conference on Software and Systems Process. ACM, 2023.
29. Ozdogan M., Sen B., Ozkan S. Digital Transformation Maturity Models: A Systematic Literature Review. Journal of Enterprise Information Management. 2021. 34(6):1521-1547.
30. Bogoviz A. V., Ragulina J. V., Alekseev A. N. Industry 5.0: Theory and Practice of Future Management. Cham: Springer Nature, 2021.
31. Fountaine T., McCarthy B., Saleh T. Building the AI-powered Organization. Harvard Business Review. 2019. Vol. 97, No. 4:62–73.