Євсеєв С. П.
Йохов О. Ю.
Король О. Г.

ГЕШУВАННЯ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

Монографія

Харків. Вид. ХНЕУ, 2013

УДК 004.631.54
ББК 32,973
Є25

Рецензенти: докт. техн. наук, професор кафедри безпеки інформаційних технологій Національного авіаційного університету *Хорошко В. О.;* докт. техн. наук, професор, головний науковий співробітник Науково-дослідного центру службово-бойової діяльності Академії внутрішніх військ МВС України *Морозов О. О.*

Рекомендовано до видання рішенням вченої ради Харківського національного економічного університету.

Протокол № 6 від 21.01.2013 р.

Авторський колектив: канд. техн. наук, ст. наук. співробітник Євсеєв С. П. – розділи 1, 3, 5; канд. техн. наук, ст. наук. співробітник Йохов О. Ю. – вступ, п. 4.1, п. 4.2; викладач Король О. Г. – розділ 2, п. 4.3, п. 4.4.

Євсеєв С. П.

Є25 Гешування даних в інформаційних системах : монографія / С. П. Євсеєв, О. Ю. Йохов, О. Г. Король. – Х. : Вид. ХНЕУ, 2013. – 312 с. (Укр. мов.)

Викладено загальні поняття та положення в галузі захисту інформації щодо механізмів забезпечення автентичності даних і цифрового підпису. Досліджено методи формування геш-функцій на основі безключових та ключових геш-функцій і використання універсальних класів. Розглянуто результати міжнародних криптографічних конкурсів у сфері гешування. Розкрито теоретичні основи й методи побудови кодів автентичності повідомлень для забезпечення автентичності та цілісності даних в інформаційних системах.

Рекомендовано для викладачів, наукових та інженерно-технічних співробітників у галузі захисту інформації та інформаційних систем, а також для ад'юнктів, аспірантів і студентів старших курсів відповідних спеціальностей.

ISBN

УДК 004.631.54
ББК 32,973

© Євсеєв С. П.
Йохов О. Ю.
Король О. Г.
2013

Вступ

Роль інформації в сучасному світі неухильно зростає, стрімко підвищується цінність конфіденційної та секретної інформації, що є головним об'єктом загроз національній безпеці України в інформаційній сфері.

Досвід побудови та експлуатації комплексних систем захисту інформації показує, що більше 60 % порушень безпеки припадає на несанкціоновану модифікацію повідомлень, порушення цілісності й автентичності оброблюваних і переданих даних в інформаційних системах.

Одним з найбільш ефективних механізмів забезпечення цілісності й автентичності інформації в комп'ютерних системах і мережах є ключове гешування. У той же час на сьогоднішній день Україна не має Національного стандарту ключового гешування, для забезпечення цілісності й автентичності інформації використовуються алгоритми, які відповідають міжнародним та Російським стандартам.

Математичні основи сучасної теорії автентифікації закладені відомим вченим Густавусом Сіммонсом, що формалізував процес передачі та обробки автентифікаційованих повідомлень і з позицій теоретико-інформаційного підходу вперше ввів поняття безумовної автентифікації. Центральним поняттям безумовної автентифікації є універсальне гешування повідомлень, суть якого полягає в рівномірному розподілі автентифікаторів за оброблюваними повідомленнями для довільного правила кодування, що задається, наприклад, секретним ключем. Абстрактне визначення універсального гешування і найпростіші методи побудови відповідних сімейств функцій гешування вперше введені Картером і Вергманом, які одержали подальший розвиток ідеї універсального гешування інформації на ортогональних масивах, з використанням надлишкових кодів і композиційних схем на їх основі. Перспективним напрямом для забезпечення автентичності даних в інформаційних системах є методи формування універсальних функцій гешування з використанням алгеброгеометричних кодів. Останнім значущим результатом у даному напрямі є розробка декількох алгоритмів автентифікації (UMAC16, UMAC32) з використанням сімейств

універсальних функцій гешування на поліноміальних схемах (UHash16, UHash32), відзначених як переможці європейського криптографічного конкурсу NESSIE серед методів і механізмів автентифікації повідомлень в інформаційних системах. Поряд з високими показниками стійкості дані алгоритми забезпечують формування автентифікаторів з рекордно низькою обчислювальною складністю, що поряд зі специфічними властивостями універсального гешування робить їх найбільш привабливими механізмами забезпечення цілісності й автентичності інформації в комп'ютерних системах і мережах. У той же час алгоритмам UMAC16 і UMAC32 властиві такі недоліки:

використовувана в структурі алгоритму схема універсального гешування (UHash16 або UHash32) не припускає формування геш-коду великої довжини (> 32 біт), що істотно обмежує сферу практичного використання, тим більше з урахуванням широкого поширення обчислювальних систем на 64-розрядних платформах;

криптографічна стійкість схеми автентифікації повідомлень із використанням алгоритмів UMAC16 і UMAC32 досягається за рахунок шифрування геш-коду блочно-симетричним алгоритмом, наприклад, AES (FIPS-197). Отже, компрометація використовуваного алгоритму блоково-симетричного шифрування призведе до зниження цілісності й автентичності оброблюваних і переданих даних;

шифрування геш-коду блоково-симетричним алгоритмом приводить до зміни структурних властивостей схеми автентифікації – результуюча схема не належить сімейству універсальних функцій гешування, і, як наслідок, не може бути використана в схемах безумовної автентифікації.

Таким чином, методи й алгоритми ключового гешування інформації є галуззю широких досліджень в усьому світі, на їх основі формуються найбільш перспективні механізми забезпечення цілісності й автентичності інформації в інформаційних системах. У монографії наводяться математичні моделі та методи універсального гешування інформації з високими показниками криптографічної стійкості, у тому числі ті що володіють властивостями доказово стійких криптографічних систем, які розроблені на основі обчислювальних алгоритмів ключового гешування, завдання бесключового читання яких зводиться до вирішення однієї з відомих теоретико-складних завдань у криптографії.

Розділ 1. Загальні поняття та положення у галузі захисту інформації

Розглянуто основні терміни та визначення послуг і механізмів галузі захисту інформації в інформаційних системах відповідно до міжнародних стандартів ISO7498, ISO/IEC 9797, ISO/IEC 10181. Проведено аналіз сучасних загроз на основні складові комунікаційних систем і технологій.

1.1. Основні терміни та визначення галузі захисту інформації

Діяльність, спрямована на забезпечення захисту конфіденційності, цілісності та доступності важливої для держави, суспільства та особи інформації, в тому числі відкритої, охорона якої передбачається законом, називається *захистом інформації* [6].

ДСТУ 3396.2 – 97 "Захист інформації. Технічний захист інформації. Терміни та визначення", "Положення про порядок здійснення криптографічного захисту інформації в Україні" установлює терміни та визначення понять у сфері технічного захисту інформації (ТЗІ) [42; 97].

Терміни, регламентовані у цьому стандарті, обов'язкові для використання в усіх видах організаційної та нормативної документації, а також для робіт зі стандартизації, і рекомендовані для використання у довідковій та навчально-методичній літературі, що належить до сфери технічного захисту інформації.

Терміни стандарту є обов'язковими для використання підприємствами та установами всіх форм власності і підпорядкування, громадянами – суб'єктами підприємницької діяльності, міністерствами (відомствами), центральними і місцевими органами державної виконавчої влади, військовими частинами всіх військових формувань, представництвами України за кордоном, які володіють, використовують та розпоряджаються інформацією, що становить державну чи іншу передбачену законом таємницю або є конфіденційною інформацією, яка належить державі.

Далі наведені визначення, що встановлені ДСТУ 3396.2 – 97 [42; 97].

Види інформації, які підлягають технічному захисту:

інформація – відомості про об'єкти, процеси та явища (ДСТУ 2226);

інформація з обмеженим доступом – інформація, право доступу до якої обмежено встановленими правовими нормами і (чи) правилами;

таємна інформація – інформація з обмеженим доступом, яка містить відомості, що становлять державну або іншу передбачену законом таємницю;

конфіденційна інформація – інформація з обмеженим доступом, якою володіють, користуються чи розпоряджаються окремі фізичні чи юридичні особи або держава і порядок доступу до якої встановлюється ними.

Загроза для інформації:

витік інформації – неконтрольоване поширення інформації, яке призводить до її несанкціонованого одержання;

порушення цілісності інформації – спотворення інформації, її руйнування або знищення;

блокування інформації – унеможливлення санкціонованого доступу до інформації;

загроза для інформації – витік, можливість блокування чи порушення цілісності інформації. (Примітка. Загрози для інформації можуть виникати під час використання технічних засобів чи технологій, недосконалих щодо захисту інформації);

модель загроз для інформації – формалізований опис методів та засобів здійснення загроз для інформації;

доступ до інформації (ТЗІ) – можливість одержання, оброблення інформації, її блокування та (чи) порушення цілісності;

несанкціонований доступ до інформації (НСД) – доступ до інформації, за якого порушуються встановлений порядок його здійснення та (чи) правові норми;

закладний пристрій (ТЗІ) – потай встановлюваний технічний засіб, який створює загрозу для інформації;

програмна закладка – потай впроваджена програма, яка створює загрозу для інформації, що міститься у комп'ютері;

комп'ютерний вірус – програма, що розмножується та поширюється самочинно. Комп'ютерний вірус може порушувати цілісність інформації, програмне забезпечення та (чи) режим роботи обчислювальної техніки;

спеціальний вплив (ТЗІ) – вплив на технічні засоби, що призводить до здійснення загрози для інформації;

технічна розвідка – несанкціоноване здобування інформації за допомогою технічних засобів та її аналіз;

модель технічної розвідки – формалізований опис методів, засобів та можливостей технічної розвідки;

Технічні канали витоку інформації:

носій інформації (ТЗІ) – матеріальний об'єкт, що містить інформацію з обмеженим доступом;

інформативний сигнал (ТЗІ) – фізичне поле та (чи) хімічна речовина, що містять інформацію з обмеженим доступом;

(технічний) канал витоку інформації – сукупність носія інформації, середовища його поширення та засобу технічної розвідки;

самочинний (технічний) канал витоку інформації – ненавмисний канал витоку інформації. Технічний канал витоку інформації, в якому носії інформації та (чи) середовище їх поширення формуються самочинно;

штучний (технічний) канал витоку інформації – навмисний канал витоку інформації;

побічне електромагнітне випромінювання і навід (ПЕМВН) – електромагнітне випромінювання та навід, що є побічним результатом функціонування технічного засобу і може бути носієм інформації.

Технічний захист інформації:

технічний захист інформації (ТЗІ) – діяльність, спрямована на запобігання порушенню цілісності, блокуванню та (чи) витоку інформації технічними каналами;

технічний засіб із захистом – захищений (технічний) засіб; захищена техніка. Технічний засіб, у якому додатково до основного призначення передбачено функцію захисту інформації від загроз;

засіб технічного захисту інформації – пристрій та (чи) програмний засіб, основне призначення яких – захист інформації від загроз;

приховування інформації – спосіб технічного захисту інформації, який полягає в унеможливленні або суттєвому утрудненні несанкціонованого одержання інформації;

пасивне приховування інформації – приховування інформації послабленням енергетичних характеристик фізичних полів або зниженням концентрації речовин;

активне приховування інформації – приховування інформації створенням таких фізичних полів та речовин, які утруднюють здобування інформації або спричиняють невизначеність її змісту;

дезінформування – спосіб технічного захисту інформації, який полягає у формуванні свідомо хибної інформації для унеможливлення несанкціонованого доступу до істинної інформації.

Організація технічного захисту інформації:

система технічного захисту інформації – сукупність організаційних структур, нормативно-правових документів та матеріально-технічної бази. Основними елементами матеріально-технічної бази системи ТЗІ є технічні засоби із захистом, засоби технічного захисту інформації та засоби контролю за ефективністю технічного захисту інформації;

зона безпеки інформації – простір, за межами якого забезпечена безпека інформації;

рівень (технічного) захисту інформації – сукупність вимог (в тому числі і тих, що нормуються), які визначаються режимом доступу до інформації та загрозами для неї;

ефективність технічного захисту інформації – степінь відповідності вжитих заходів щодо технічного захисту інформації встановленим вимогам.

ДСТУ Р ISO7498-2-99 визначає базову еталонну модель взаємозв'язку відкритих систем (ВВС). Цей стандарт встановлює основи для забезпечення скоординованих наробіток діючих та майбутніх стандартів щодо ВВС. Для розуміння стандарту необхідно знати основні дані щодо захисту інформації. Нижче наведені терміни, що застосовуються в цьому стандарті [19; 115].

Управління доступом – запобігання несанкціонованого використання якого-небудь ресурсу, включаючи запобігання використання ресурсу неповноважним способом.

Список управління доступом – список логічних об'єктів, що мають дозвіл на доступ до ресурсу, разом з переліком їх прав на доступ.

Обліковість – властивість, що забезпечує однозначне відстеження власних дій будь-якого логічного об'єкта.

Активна загроза – загроза навмисної несанкціонованої зміни стану системи.

Інформація автентифікації – інформація, використовувана для встановлення автентичності запитуваної особистості.

Обмін автентифікацією – механізм, призначений для підтвердження справжності якого-небудь логічного об'єкта шляхом обміну інформацією.

Повноваження – надання прав, які гарантують доступ к ресурсам відповідно до прав на доступ.

Доступність – властивість бути доступною й використовуваним по запиту з боку вповноваженого логічного об'єкта.

Функціональна можливість – маркер, використовуваний у якості ідентифікатора якого-небудь ресурсу, оволодіння яким надає право на доступ до даного ресурсу.

Канал – маршрут передачі інформації.

Шифротекст – дані, одержані в результаті використання шифрування. Семантичний зміст отриманих у результаті шифрування даних недоступно.

Відкритий текст – значеннєві дані, семантичний зміст яких є доступним.

Конфіденційність – властивість, що дозволяє не давати права на доступ до інформації або не розкривати її неповноважним особам, логічним об'єктам або процесам.

Посвідчення особи – дані, передані для встановлення заявленої автентичності логічного об'єкта.

Криптоаналіз – аналіз криптографічної системи й/або її входів і виходів з метою одержання конфіденційних змінних і/або чутливих даних, включаючи відкритий текст.

Криптографічне контрольне значення – інформація, одержувана в результаті виконання криптографічного перетворення (див. криптографія) блоку даних.

Криптографія – дисципліна, що охоплює принципи, засоби й методи перетворення даних для приховання їх інформаційного вмісту, запобігання їх модифікації, що не виявляється, та/або їх несанкціонованого використання.

Цілісність даних – здатність даних не піддається зміні або анулюванню в результаті несанкціонованого доступу.

Автентифікація відправника даних – підтвердження того, що відправник отриманих даних відповідає заявленому.

Розшифрування – процес, зворотний процесу шифрування.

Відхилення послуги – запобігання санкціонованого доступу до ресурсів або затримка операцій, критичних у часі.

Цифровий підпис – додаткові дані або криптографічне перетворення якого-небудь блоку даних, що дозволяють одержувачу блоку даних переконатися в автентичності відправника й цілісності блоку даних і захистити його від викривлення за допомогою, наприклад, засобів одержувача.

Міжкінцеве шифрування – шифрування даних усередині або на стороні відправника скінченої системи з відповідним дешифруванням, виконуване тільки усередині або на стороні одержувача скінченої системи.

Стратегія захисту, заснована на ідентифікації – стратегія захисту інформації, заснована на ідентифікаторах і/або атрибутах користувачів, групи користувачів або логічних об'єктів, що діють від імені користувачів і доступних їм ресурсів/логічних об'єктів.

Ключ – послідовність символів, що управляє операціями шифрування й расшифрування.

Адміністративне управління ключем – генерація, збереження, розподіл, видалення, каталогізування й застосування ключів відповідно до стратегії захисту.

Поланкове шифрування – індивідуальне прикладне застосування шифрування до даних па кожній ланці системи обміну даних (див. також міжкінцеве шифрування).

Виявлення маніпуляції – механізм, використовуваний для виявлення можливої модифікації блоку даних (випадкової або навмисної).

Маскування – прагнення якого-небудь логічного об'єкта під інший логічний об'єкт.

Нотаризація – реєстрація даних довіреною третьою особою, яка забезпечує таке підтвердження правильності їх характеристик, таких як зміст, відправник, час і одержувач.

Пасивна загроза – загроза несанкціонованого розкриття інформації без зміни стану системи.

Пароль – конфіденційна інформація автентифікації, що звичайно складається з рядка знаків.

Автентифікація рівноправного логічного об'єкта – підтвердження того, що рівноправний логічний об'єкт у якій-небудь асоціації є заявленим логічним об'єктом.

Фізичний захист – засоби, які використовуються для забезпечення фізичного захисту ресурсів від навмисної або випадкової загрози.

Власність – право окремих осіб контролювати або впливати на збір і зберігання стосовної до них інформації і на визначення тих, ким і для кого може бути розкрита ця інформація.

Самовідмова – самозаперечення одного з логічних об'єктів, що брав участь в обміні даними, повної або часткової своєї участі у цьому обміні.

Управління маршрутизацією – застосування правил у процесі маршрутизації за вибором або виключенням конкретних мереж, ланок даних або ретрансляторів.

Стратегія захисту, заснована на правилах – стратегія захисту, заснована на загальних правилах, що висуваються до всіх користувачів. Ці правила звичайно ґрунтуються на порівнянні чутливості доступних ресурсів і володінні окремими користувачами, групами користувачів або логічними об'єктами, що діють від імені користувачів атрибутами, що відповідають.

Аналіз процедур захисти – незалежний перегляд і аналіз системних записів і активностей з метою перевірки їх адекватності системним керуючим функціям для забезпечення відповідності із прийнятою стратегією захисту й операційними процедурами, виявлення пробілів у захисті й видачі рекомендацій з будь-яких зазначених змін в управлінні, стратегії й процедурах.

Дані трасування захисту – накопичені й готові до використання дані, призначені для деталізації причини аналізу процедур захисту.

Мітка захисту – гранична мітка, що привласнюється якому-небудь ресурсу (у якості якого може виступати блок даних, який називає або позначає атрибути захисту цього ресурсу).

Стратегія захисту – набір критеріїв для забезпечення послуг захисту.

Послуга захисту – послуга, що надається яким-небудь рівнем взаємозалежних відкритих систем, яка забезпечує адекватний захист систем або процедур передачі даних.

Виборчий захист поля – захист конкретних полів усередині повідомлення, що підлягає передачі.

Чутливість – характеристика ресурсу, яка визначає його цінність або важливість і може враховувати його вразливість.

Загроза – потенційна можливість порушення захисту.

Аналіз трафіка – висновок про стан інформації на основі спостереження за потоками трафіка (наявність, відсутність, обсяг, напрям і частота).

Конфіденційність потоку трафіка – послуга конфіденційності, призначена для захисту від аналізу трафіка.

Заповнення трафіка – генерація фіктивних сеансів обміну даними, фіктивних блоків даних і/або фіктивних даних у складі блоків даних.

Довірча функціональність – функціонування, яке сприймається правильним з погляду деякого критерію, наприклад, критерію, що висувається за допомогою стратегії захисту.

Механізми, що використовують блоковий шифр описуються міжнародним стандартом ISO/IEC: ISO/IEC 9797-1:1999(E). ISO/IEC 9797 використовує такі основні стосовні до безпеки терміни, визначені в ISO 7498-2 [19; 115; 118; 119].

Цілісність даних – властивість, що дані не були змінені або знищені неавторизованим методом.

Блок – бітовий рядок довжини *n*.

Ключ блокового шифру – ключ, що контролює роботу блокового шифру.

Первісне перетворення – функція, яка застосовується на початку MAC-алгоритму.

Ключ MAC-алгоритму – ключ, що контролює роботу MAC-алгоритму.

Код автентифікації – повідомлення (MAC): рядок біт, що є виходом MAC-алгоритму [118].

Алгоритм коду автентифікації повідомлення (MAC) – алгоритм обчислення функції, який відображає рядки біт і секретного ключа на рядок біт фіксованої довжини, задовольняючи при цьому двом умовам:

для будь-якого ключа й будь-якого вхідного рядка функція може бути ефективно обчислена;

для будь-якого фіксованого ключа у відсутності яких або даних про ключ обчислювально неможливо обчислити значення функції по будь-якому іншому вхідному рядку, навіть якщо дана множина вхідних рядків і відповідних значень функцій, де значення *і-го* вхідного рядка може бути обране після спостереження значення попередніх *і – 1* значень функцій. MAC-алгоритм також називається криптографічною перевірочною функцією (див. ISO 7498-2) [115].

Обчислювальна здійснюваність залежить від конкретних вимог безпеки користувача й оточення.

Вихідне перетворення – функція, що застосовується після закінчення MAC-алгоритму, перед операцією усікання.

Криптограма – дані, які були перетворені, щоб сховати їх інформаційний зміст.

Розшифрування – зворотна шифруванню операція.

Шифрування – (оборотне) перетворення даних за допомогою криптографічного алгоритму для одержання криптограми, тобто для того, щоб сховати інформаційний зміст даних.

Ключ – послідовність символів, яка керує криптографічним перетворенням (а саме шифруванням, розшифруванням, обчисленням перевірочної криптографічної функції, створенням цифрового підпису або перевіркою цифрового підпису).

Відкритий текст – незашифрована інформація.

n-бітовий блоковий шифр – блоковий шифр із такою властивістю, що блоки відкритого тексту й блоки криптограми мають довжину в n-біт.

Міжнародний стандарт ISO/IEC 9797-2:2002(E) застосовує такі визначення з ISO/IEC 9797-1 [118; 119].

Код автентифікації повідомлення (MAC) – рядок біт, що є виходом MAC-алгоритму.

Стійкі до колізій геш-функції – геш-функції, що задовольняють таким умовам: обчислювально неможливо знайти будь-які два різні вхідні значення, які відображаються в теж саме вихідне значення.

Рядок даних (дані) – рядок біт, що є вхідний для деякої геш-функції.

Геш-код – рядок біт, який є виходом геш-функції.

Геш-функція – функція, яка відображає рядки біт на фіксовані по довжині рядки біт, яка задовольняє такі умови:

для заданого вихідного значення обчислювально неможливо знайти вхідне значення таке, щоб воно відображалося в цьому вихідному значенні;

для заданого вхідного значення обчислювально неможливо знайти друге вхідне значення, яке відображається в тому ж самому вихідному значенні;

початкове значення – значення, що використовується у визначенні початкової точки геш-функції.

Набивання (заповнення) – додавання додаткових біт до рядка даних.

Блок – бітовий рядок довжини L_1, тобто довжина першого вхідного значення циклічної функції.

Циклічна функція – функція φ, яка перетворить два бінарні рядки довжин L_1 і L_2 у бінарний рядок L_2. Вона використовується ітеративно як частина геш-функції, коли вона поєднує рядок даних довжини L_1 з попереднім вихідним значенням довжини L_2.

Слово – рядок 32 біт.

1.2. Сучасні загрози безпеки

Національна безпека України як стан захищеності життєво важливих інтересів фізичних осіб, суспільства й держави від внутрішніх і зовнішніх загроз є необхідною умовою збереження й збільшення духовних і матеріальних цінностей. В умовах стрімкої інформатизації суспільства, широкого застосування засобів обчислювальної техніки й комп'ютерних систем особливу актуальність набувають питання інформаційної безпеки держави, найбільш складними з яких є необхідність захисту коштовної конфіденційної й секретної інформації в державних і приватних підприємствах, в органах та установах державного управління, банківської й інших системах [1; 16; 97].

Соціально-економічні наслідки інформатизації сучасного суспільства призвели до появи ряду проблем інформаційної безпеки, наприклад: необхідність захисту майнових прав громадян, підприємств і держави на інформацію й обчислювальні ресурси відповідно до вимог цивільного, адміністративного й господарського права; необхідність забезпечення ефективності функціонування найважливіших (критичних) автоматизованих та інформаційних систем і технологій, обумовлена вимогами фізичної безпеки людей, екологічного середовища, збереження духовних і матеріальних цінностей та інше; необхідність захисту цивільних прав і свобод, гарантованих чинним законодавством та ін.

Серед найнебезпечніших загроз національної безпеки України в інформаційній сфері відзначаються [16]: незважена державна політика й відсутність необхідної інфраструктури в інформаційній сфері; повільне входження України у світовий інформаційний простір, недостача у міжнародного співтовариства об'єктивного уявлення про Україну; інформаційна експансія з боку інших держав; витік інформації, що становить державну й іншу, передбачену законом, таємницю, а також конфіденційної інформації, яка є власністю держави; введення цензури.

Основними напрямами державної політики Національної безпеки України в інформаційній сфері є [1]: застосування комплексних заходів щодо захисту свого інформаційного простору й входу України у світовий інформаційний простір; виявлення й усунення причин інформаційної дискримінації України; усунення негативних факторів порушення інформаційного простору, інформаційної експансії з боку інших держав; розробка й упровадження необхідних засобів і режимів одержання,

збереження, поширення й використання суспільно значущої інформації, створення розвинутої інфраструктури в інформаційній сфері.

Державна політика національної безпеки визначається виходячи із пріоритетності національних інтересів і загроз Національної безпеки України. Вона здійснюється шляхом реалізації відповідних доктрин, стратегій, концепцій і програм у різних сферах національної безпеки відповідно до чинного законодавства. Основним комплексним заходом щодо захисту національного інформаційного простору України є, на сьогоднішній день, побудова Національної системи конфіденційному зв'язку.

Національна система конфіденційного зв'язку – це сукупність спеціальних систем (мереж) зв'язку подвійного призначення, які за допомогою криптографічних і/або технічних засобів забезпечують обмін конфіденційною інформацією в інтересах органів державної влади й органів місцевого самоврядування, створюють належні умови для їх взаємодії в мирний час і у випадку введення особливого і військового стану. Національна система конфіденційного зв'язку входить до складу Єдиної національної системи зв'язку України [1].

Складовими Національної системи конфіденційного зв'язку є [1; 17]:

спеціальні системи (мережі) зв'язку;

стаціонарні й мобільні компоненти спеціальних систем (мереж) зв'язку;

централізовані системи захисту інформації й оперативно-технічного управління.

Спеціальна система (мережа) зв'язку – система (мережа) зв'язку, призначена для обміну інформацією з обмеженим доступом.

Спеціальна система (мережа) зв'язку подвійного призначення – спеціальна система (мережа) зв'язку, призначена для забезпечення зв'язку в інтересах органів державної влади й органів місцевого самоврядування, з використанням частини її ресурсу для надання послуг іншим споживачам.

Суб'єктами Національної системи конфіденційного зв'язку виступають органи державної влади та органи місцевого самоврядування, юридичні й фізичні особи, які беруть участь у створенні, функціонуванні, розвитку й використанні цієї системи.

Управління Національною системою конфіденційного зв'язку, її функціонування, розвиток, використання й захист інформації забезпе-

чуються спеціально вповноваженим центральним органом виконавчої влади в сфері конфіденційного зв'язку відповідно до законодавства. Централізовані системи захисту інформації й оперативно-технічного управління перебувають у державній власності й не підлягають приватизації. Власниками інших компонентів Національної системи конфіденційного зв'язку можуть бути суб'єкти господарської діяльності незалежно від форми власності.

Захист інформації в структурі Національної системи конфіденційного зв'язку забезпечується: дотриманням суб'єктами правового співвідношення норм, вимог і правил організаційного й технічного характеру щодо захисту оброблюваної інформації; використанням засобів обчислювальної техніки, програмного забезпечення, засобів зв'язку й автоматизованих систем у цілому, засобів захисту інформації, які відповідають установленим вимогам до захисту інформації (мають відповідний сертифікат); перевіркою відповідності засобів обчислювальної техніки, програмного забезпечення, засобів зв'язку й автоматизованих систем у цілому встановленим вимогам до захисту інформації (сертифікація засобів обчислювальної техніки, засобів зв'язку й автоматизованих систем); здійсненням контролю захисту інформації.

Відповідно до основних нормативно-правових актів України розв'язок завдань захисту національного інформаційного простору України покладається на Національну систему конфіденційного зв'язку, концептуальні питання створення, функціонування, розвитку й використання якої регулюються Конституцією України, законами України "Про інформацію", "Про державну таємницю", "Про захист інформації в автоматизованих системах", "Про зв'язок", "Про підприємництво", "Про ліцензування визначених видів господарської діяльності", "Про Національну систему конфіденційного зв'язку".

Відповідно до основних функціональних завдань, використання Національної системи конфіденційного зв'язку повинне забезпечити ефективний захист цінної конфіденційної й секретної інформації в державних та приватних підприємствах, в органах і установах державного управління, банківської й інших системах.

Система забезпечення інформаційної безпеки – організована державою сукупність державних органів, посадових осіб, громадських організації, окремих громадян, об'єднаних цілями та завданнями захисту національних інтересів України в інформаційній сфері, які реалізують

державну політику та здійснюють узгоджену діяльність у межах законодавства України.

Основними функціями системи забезпечення інформаційної діяльності є [1]: створення та забезпечення діяльності державних органів – елементів системи; управління діяльністю системи; міжнародне співробітництво в сфері інформаційної безпеки.

Основними елементами системи забезпечення інформаційної безпеки України є:

громадяни України;

Верховна Рада України;

Комітет Верховної Ради України з питань національної безпеки і оборони;

Комітет Верховної Ради України з питань свободи слова та інформації;

Комітет Верховної Ради України з питань транспорту і зв'язку ;

Президент України;

Національний інститут стратегічних досліджень;

Рада Національної безпеки й оборони України. Безпосередньо РНБО України підпорядковуються: Інститут проблем національної безпеки та Національний інститут проблем міжнародної безпеки;

Апарат Ради національної безпеки і оборони України;

Кабінет Міністрів України. У складі Секретаріату Кабінету Міністрів України діє Управління стратегії розвитку інформаційних ресурсів та технологій;

Урядова комісія з питань інформаційно-аналітичного забезпечення діяльності органів виконавчої влади;

Національна комісія з питань регулювання зв'язку України;

Міністерство освіти та науки України. У складі Міністерства освіти та науки України діє Державний департамент інтелектуальної власності;

Державний комітет інформаційної політики, телебачення та радіомовлення;

Державний департамент з питань зв'язку та інформатизації Міністерства транспорту та зв'язку України;

Державний комітет архівів;

Державний комітет статистики України;

Служба безпеки України;

Державна служба спеціального зв'язку та захисту інформації України;

Міністерство внутрішніх справ України. У складі – відділ боротьби з правопорушеннями у сфері високих технологій Державної служби боротьби з економічною злочинністю;

Національна Рада України з питань телебачення та радіомовлення;

Конституційний Суд України;

Суди загальної юрисдикції;

Генеральна прокуратура України;

Рада Міністрів Автономної Республіки Крим, обласні державні адміністрації, Київська та Севастопольська міські державні адміністрації;

органи місцевого самоврядування;

інші державні органи та організації;

засоби масової інформації;

політичні партії та рухи;

громадські організації та професійні спілки;

неурядові дослідницькі організації;

організації та установи, що здійснюють діяльність в інформаційній сфері.

Виходячи з принципів і положень державної політики забезпечення інформаційної безпеки найбільшу небезпеку становлять загрози в політичній, економічній, оборонній та інших сферах діяльності держави (рис. 1.1).

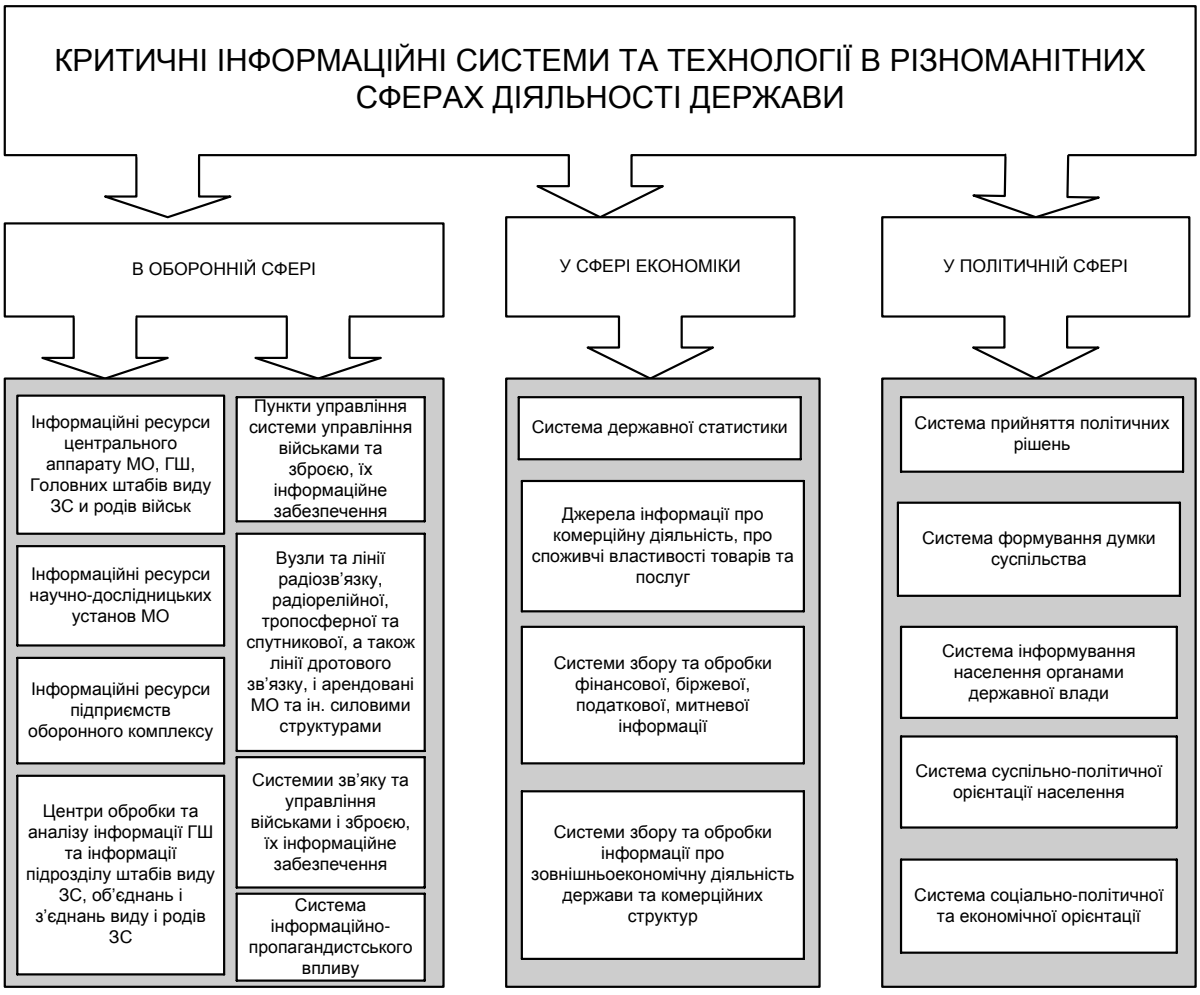


Рис. 1.1. **Критичні інформаційні системи та технології в різних сферах діяльності держави**

У політичній сфері найбільш серйозній небезпеці піддаються [17]: суспільна свідомість і політична орієнтація різних груп населення країни (регіонів), що безперервно формуються під впливом вітчизняних і зарубіжних засобів масової інформації (преса, радіо, телебачення); система прийняття політичних рішень, яка істотно залежить від якості та своєчасності її інформаційного забезпечення; право політичних організацій, партій, об'єднань і рухів на вільне вираз своїх програм, соціально-політичних та економічних орієнтацій через засоби масової інформації; система регулярного інформування населення органами державної влади та управління про політичне та соціально-економічне життя через засоби масової інформації, прес-центри, центри громадських зв'язків тощо; система формування громадської думки, що включає спеціальні інститути, центри та служби виявлення, вивчення та аналізу громадської думки.

У сфері економіки найбільш схильні до впливу загроз інформаційної безпеки система державної статистики, джерела, які породжують інформацію про комерційну діяльність господарських суб'єктів усіх форм власності, про споживчі властивості товарів і послуг, системи збору та обробки фінансової, біржової, податкової, митної інформації та інформації про зовнішньоекономічну діяльність держави і комерційних структур.

В оборонній сфері до найбільш вразливих ланок відносяться: інформаційні ресурси апарату Міністерства оборони, Генерального штабу, Головних штабів видів Збройних сил і родів військ, науково-дослідних установ, які містять відомості й дані про оперативні і стратегічні плани підготовки і ведення бойових дій, про склад і дислокацію військ, про мобілізаційну готовність, тактико-технічні характеристики озброєння і військової техніки; інформаційні ресурси підприємств оборонного комплексу, що містять відомості про науково-технічний і виробничий потенціал, про обсяги поставок і запаси стратегічних видів сировини і матеріалів, про основні напрями розвитку озброєння, військової техніки, їх бойових можливостей і проводяться в інтересах оборони фундаментальних і прикладних НДР; системи зв'язку та управління військами і зброєю, їх інформаційне забезпечення; політико-моральний стан військ у частині, що залежить від інформаційно-пропагандистського впливу; інформаційна інфраструктура, в тому числі центри обробки та аналізу інформації Генерального штабу та інформаційні підрозділи штабів видів Збройних Сил, штаби об'єднань і з'єднань

видів Збройних Сил і родів військ, пункти управління, вузли та лінії радіозв'язку, радіорелейного, тропосферного та супутникового, а також лінії дротового зв'язку, що розгортаються і орендовані Міністерством оборони та іншими силовими структурами.

Під *загрозою* в широкому сенсі звичайно розуміють потенційно можливу подію, дію (вплив), процес або явище, яке може призвести до нанесення шкоди будь-чиїм інтересам. Загрозою інтересам суб'єктів інформаційних відносин називають потенційно можливу подію, процес або явище, яке за допомогою впливу на інформацію або інші компоненти інформаційної системи може прямо або опосередковано призвести до нанесення шкоди [30; 41].

У процесі зберігання й обробки інформація може бути піддана впливу факторів, як випадкових, так і навмисних. Найбільш частими і найбільш небезпечними, з точки зору розміру шкоди, є ненавмисні помилки користувачів, операторів, системних адміністраторів та інших осіб, які обслуговують інформаційні системи, що виникають, помилки обробки і передачі інформації. Згідно зі статистикою, 65 % втрат – наслідок ненавмисних помилок. Очевидно, найрадикальніший спосіб боротьби з ненавмисними помилками – максимальна автоматизація і строгий контроль за правильністю виконаних дій.

Загрози, які походять від навколишнього середовища, відрізняються великою різноманітністю. У першу чергу, це порушення інфраструктури – аварії електроживлення, тимчасова відсутність зв'язку, перебої з водопостачанням, громадянські заворушення і т. п. Особливу небезпеку становлять стихійні лиха та події: пожежі, повені, землетруси, урагани. За даними статистики на ці загрози припадає 13 % втрат, завданих інформаційним системам. На рис. 1.2. наведена, в загальному вигляді, класифікація загроз інформації.

Особливої небезпеки останнім часом набули навмисні загрози, реалізація яких доступна терористичним групам і організаціям. Потенційною мішенню злочинних намірів можуть служити інформаційні системи міністерства оборони та інших силових структур, системи управління атомних, хімічних та інших небезпечних виробництв, обчислювальні системи банків і великих промислових підприємств. Реалізація умисних загроз може призвести до тяжких наслідків для оборони, промисловості, економіки, банківської сфери та інших галузей господарської діяльності, екології, життя і здоров'я населення. У зв'язку з цим, слід зазначити, що можливості реалізації навмисних загроз

останнім часом різко зросли. Це пояснюється стрімким розвитком обчислювальної техніки, в тому числі, так званих суперкомп'ютерів, широким поширенням технологій розподілених обчислень [15].

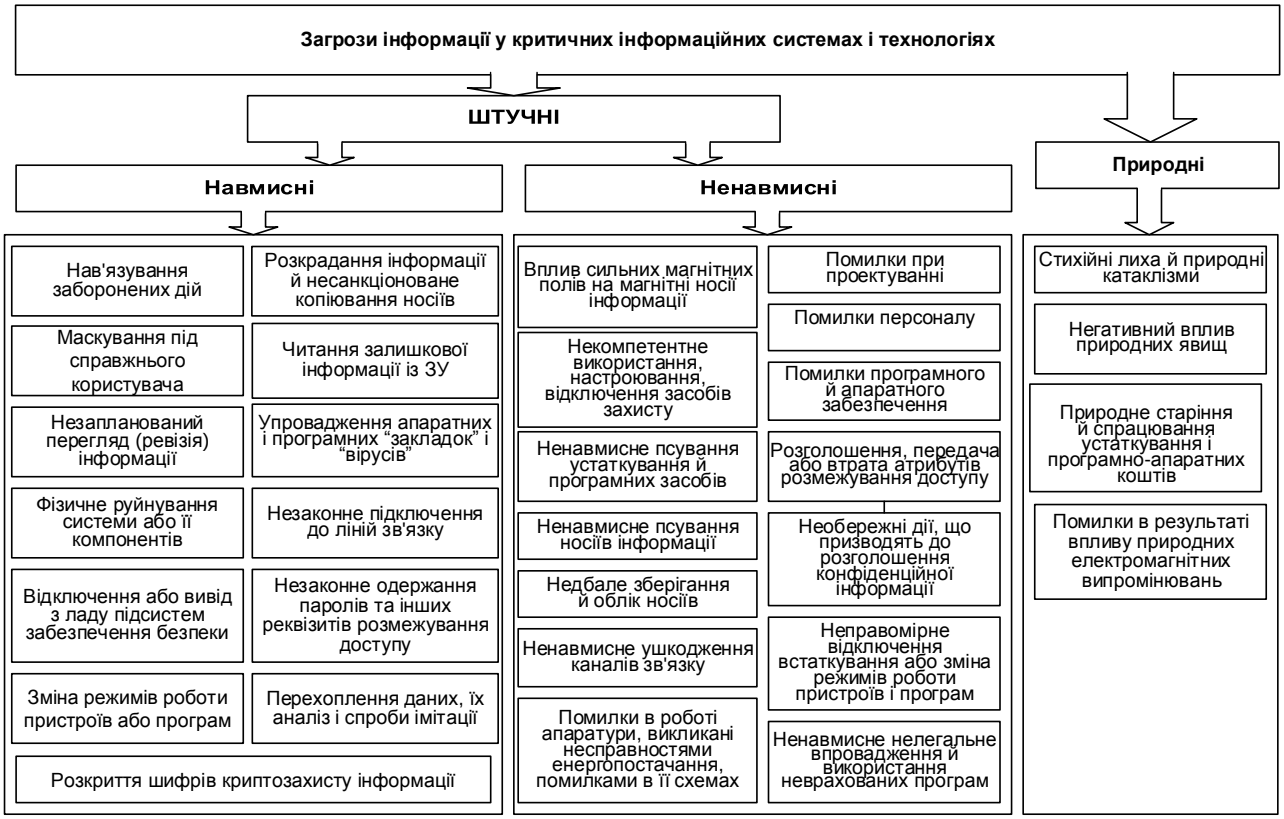


Рис. 1.2. **Загрози інформації в критичних інформаційних системах та технологіях**

1.3. Послуги та механізми захисту інформації (ISO 7498, ISO/IEC 10181)

Стандарт ISO 7498-2 (1989 року) [19; 115] присвячений питанням безпеки комп'ютерних мереж, побудованих на основі моделі взаємодії відкритих мереж (ВВС). Стандартом викладені результати аналізу існуючих погроз безпеки комп'ютерних мереж, вводиться спеціальна термінологія для опису послуг і механізмів безпеки, приводяться рекомендації з контролю й оцінки пропонованих засобів забезпечення безпеки мереж, а також визначається концепція управління безпекою. Даний документ розроблявся в тісному зв'язку з моделлю ВВС, і містить практичні рекомендації, стосовно того, які послуги безпеки можуть бути реалізовані конкретними протоколами на кожному рівні моделі ВВС, який механізм безпеки може бути використаний для реалізації конкретної послуги безпеки.

Розробники документа спирались на те, що існує певний життєвий цикл розробки системи безпеки, який складається з п’яти етапів:

визначення політики безпеки, що містить абстрактний ряд вимог до безпеки системи;

аналіз вимог безпеки, включаючи аналіз ризиків, аналіз урядових, правових і стандартних вимог;

визначення послуг безпеки, необхідних для задоволення пред’явлених вимог;

побудова і впровадження системи безпеки, включаючи вибір механізмів безпеки, що забезпечують конкретні обрані послуги безпеки;

безперервне управління безпекою.

Механізми безпеки є конкретними заходами для реалізації послуг безпеки (рис. 1.3). Взаємозв’язок послуг і механізмів безпеки наведено на рис. 1.4. У контексті моделі ISO/OSI життєвого циклу системи безпеки можна сказати, що загрозою безпеки є будь-що (дія, бездіяльність, засіб та інше), що сприяє порушенню політики безпеки і зламу системи безпеки (наприклад, втраті цілісності або конфіденційності). Послуга безпеки вибирається для забезпечення захисту від ідентифікованої загрози і становить абстрактні поняття, які можуть бути використані для характеристики вимог безпеки, а механізм безпеки є засобом, шляхом використання якого реалізується і застосовується послуга. Важливо відзначити відмінність між послугою безпеки, що визначена для системи, і механізмом безпеки, що забезпечує реалізацію послуги.

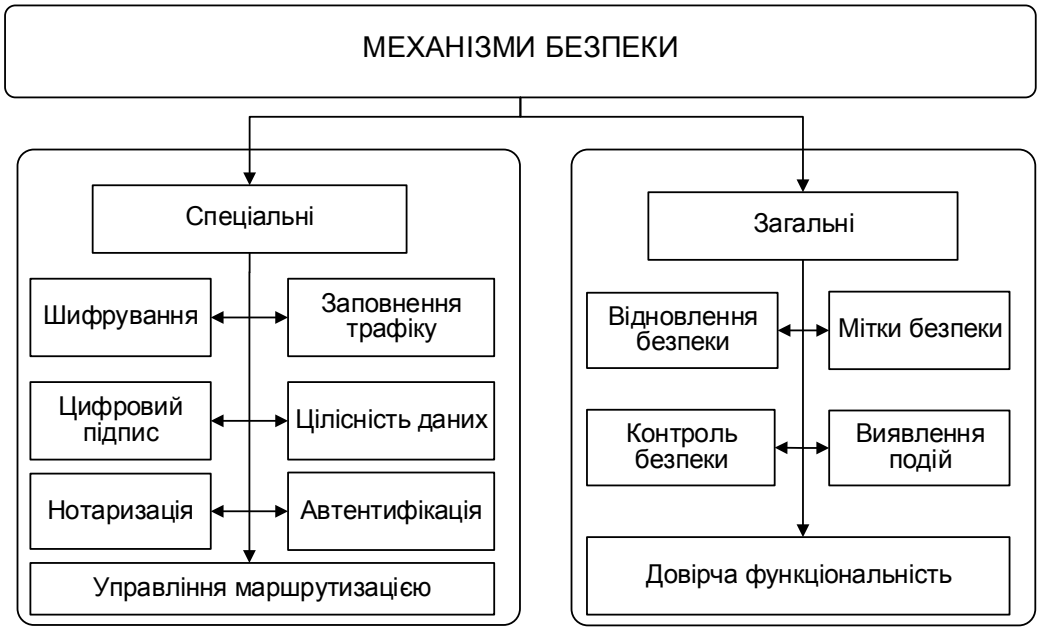


Рис. 1.3. **Загальна класифікація механізмів безпеки**

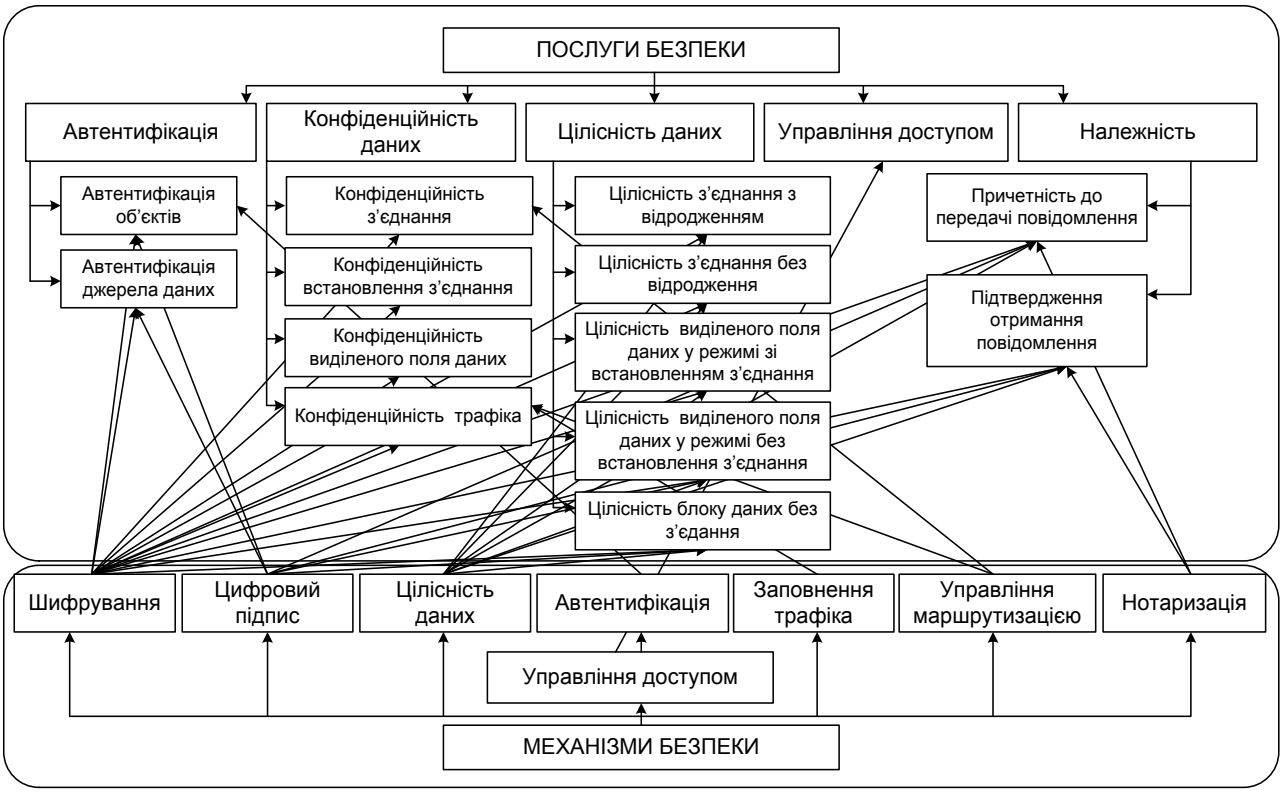


Рис. 1.4. **Взаємозв'язок послуг та механізмів безпеки**

Наприклад, конфіденційність є послугою безпеки, а шифрування – механізмом, що використовується для забезпечення конфіденційності. Шифрування в той же час може використовуватися для реалізації інших послуг, наприклад, послуги автентифікації, а конфіденційність може бути забезпечена шляхом застосування інших механізмів, наприклад, засобами фізичного захисту даних.

У будь-якій системі безпеки правила, що визначають режим безпеки, повинні бути чітко представлені у вигляді політики безпеки. ISO 7498-2 визначає політику безпеки як "множину задокументованих правил забезпечення і впровадження послуг безпеки". Іншим фундаментальним поняттям є поняття домену безпеки, під яким розуміють обмежену групу об'єктів і суб'єктів безпеки, керованих загальним адміністратором безпеки і до яких застосовується єдина політика безпеки. Політика безпеки є активним компонентом захисту, що включає в себе аналіз можливих загроз і вибір дій протидії загрозам.

Концепція архітектурних засобів безпеки ISO включає п'ять основних компонентів [115]:

визначення послуг забезпечення безпеки;

визначення механізмів безпеки;

рівневу модель побудови послуг забезпечення безпеки;

співвіднесення послуг забезпечення безпеки і рівневої моделі;

співвіднесення механізмів забезпечення безпеки і послуг.

Основою визначення переліку послуг забезпечення безпеки є такі принципи побудови рівневої моделі безпеки:

кількість альтернативних способів забезпечення безпеки повинна бути мінімізована. Ця вимога спрямована на мінімізацію вартості як самих послуг, так і додатків на їх основі;

послуги забезпечення безпеки можуть працювати більше ніж на одному рівні моделі при побудові безпечної системи;

функції засобів забезпечення безпеки не повинні дублювати існуючі аналогічні функції засобів комунікації.

Рекомендується не порушувати незалежність рівнів.

Кількість неконтрольованих (довірчих) функцій повинна бути мінімізована. Якщо який-небудь захисний механізм одного рівня базується на використанні послуг більш низького рівня, то не повинно існувати ніяких проміжних рівнів, що забороняють або не гарантують такого зв’язку.

Послуги забезпечення безпеки, реалізовані на кожному рівні, повинні бути визначені таким чином, щоб допускати модульне доповнення до базових комунікаційних послуг.

Стандарт ISO 7498-2 визначає п’ять базових послуг безпеки [115]:

автентифікація (authentication);

управління доступом (access control);

конфіденційність даних (data confidentiality);

цілісність даних (data integrity);

належність (причетність) (non-repudiation).

Автентифікація. Стандарт визначає два типи автентифікації: автентифікацію об’єкта (entity authentication) і автентифікацію джерела даних (origin authentication).

Автентифікація об’єкта комунікацій визначається як "підтвердження того, що об’єкт комунікації при з’єднанні саме той, який є оголошеним". Ця послуга необхідна в системах зі встановленням з’єднань і може застосовуватися періодично під час сеансу. Вона слугує для запобігання таких загроз як маскарад і повтор попереднього сеансу зв’язку.

Автентифікація джерела даних допускає підтвердження того, що "джерело отриманих даних саме те, яке зазначене або оголошене".

Ця послуга істотна для комунікацій без встановлення з'єднання, для яких кожний пакет є незалежним від інших, і єдине, що може бути гарантоване, з погляду автентифікації, – це те, що джерело пакета саме те, яке зазначене у його заголовку. Функція не забезпечує захист від повторної передачі або модифікації даних.

Обидва види автентифікації визначені для мережного, транспортного й прикладного рівнів, на яких реалізуються протоколи з відновленням і без встановлення з'єднань.

Управління доступом. Дана послуга визначена як запобігання неавторизованого використання ресурсів, включаючи запобігання використання ресурсів неприпустимим способом". Тобто дана послуга забезпечує доступ до ресурсів тільки авторизованих користувачів (процесів), а також гарантує тільки зазначені права доступу для авторизованих користувачів і запобігає неавторизований доступ як "внутрішніх", так і "зовнішніх" користувачів. Ця послуга застосовується до різних типів доступу до ресурсів, наприклад, використання комунікаційних ресурсів, читання, запис або видалення інформаційних ресурсів, використання ресурсів обчислювальних систем щодо обробки даних і т. д. Дана послуга використовується для встановлення політики управління/обмеження доступу. Згідно з ISO 7498-2 політика управління доступом може визначатися критеріями ухвалення рішення про доступ або засоби, за допомогою яких регулюється доступ. Залежно від використовуваних критеріїв ухвалення рішення політика управління доступом може бути заснована на визначенні ідентичності явищ і об'єктів (identity-based) або правил (послідовності дій) доступу (rule-based) [93]. У першому випадку управління доступом засноване на використанні автентифікації для перевірки ідентичності суб'єкта доступу (користувача, процесу, проміжної або кінцевої системи) до надання йому доступу до ресурсів. У другому випадку політика управління доступом допускає ухвалення рішення про доступ на підставі послідовності правил, які співвідносять автентифікацію з точністю. Наприклад, правила можуть бути виражені в термінах часу і дати доступу або "надійності", що має даний користувач.

Стандарт визначає два типи політики управління доступом: обумовлена користувачем і обумовлена адміністратором. У більшості операційних систем (ОС) реалізований перший тип управління доступом. Другий варіант часто реалізується в мережах загального користування.

Конфіденційність даних. Конфіденційністю є "властивість, що гарантує, що інформація не може бути доступна або розкрита для неавторизованих (не уповноважених) осіб, об'єктів або процесів". Питанням забезпечення конфіденційності приділяють велику увагу в тих системах, де розкриття інформації можливе в багатьох точках шляху передачі даних.

Для послуги визначені чотири типи:

конфіденційність з'єднання забезпечує конфіденційність всіх даних користувача цього з'єднання;

конфіденційність у режимі без встановлення з'єднання забезпечує конфіденційність всіх даних користувача в окремому блоці даних;

конфіденційність виділеного поля даних призначена для захисту окремих інформаційних полів і вимагає, щоб тільки окремі поля в пакетах були захищені. Даний тип послуги використовується як у мережах із установленням з'єднання, так і в мережах без встановлення з'єднання;

конфіденційність трафіка запобігає одержанню інформації шляхом спостереження (аналіз) трафіка. Це досягається шляхом захисту інформації про джерело – призначення, кількість переданих даних і частота передачі.

Цілісність даних. Захист цілісності даних має дві базові реалізації (для мереж із установленням і без встановлення з'єднання), кожна з яких може застосовуватися для вибраних груп інформаційних полів. Захист цілісності даних у мережах із установленням з'єднання допускає "виявлення будь-якої модифікації, включення, видалення або повторної передачі даних у послідовності (пакетів)".

У мережах без встановлення з'єднання захист цілісності орієнтований на виявлення модифікацій кожного пакета без виконання аналізу великого обсягу інформації, наприклад, за сеанс або цикл передач. Ця послуга не запобігає навмисному видаленню, включенню або повторній передачі пакетів і доповнює автентифікацію джерела даних. Послуга захисту цілісність даних може додатково включати функції відновлення даних у випадках порушення їх цілісності.

Стандарт визначає п'ять типів послуги цілісності даних:

цілісність з'єднання з відновленням забезпечує цілісність даних користувача цього з'єднання й виявляє будь-яку модифікацію, вставку, видалення або повторення даних з їх можливим таким відновленням;

цілісність з’єднання без відновлення аналогічна попередній послузі, але без можливості відновлення даних;

цілісність виділеного поля в режимі зі встановленням з’єднання забезпечує цілісність виділеного поля даних у пакеті користувача, переданих через це з’єднання;

цілісність виділеного поля в режимі без встановлення з’єднання забезпечує цілісність виділеного поля в пакеті даних;

цілісність блоку даних без з’єднання забезпечує цілісність окремого блоку даних (пакета). Орієнтована на виявлення тільки модифікацій і не запобігає навмисному видаленню, включенню або повторній передачі пакетів.

Причетність. У стандарті ISO 7498-2 [115] причетність визначається як запобігання можливості відмови одним з реальних учасників комунікацій від факту його повної або часткової участі в передачі даних.

Визначено дві форми причетності: причетність до посилки повідомлення (доказ джерела) надає одержувачу доказ того, що повідомлення було послано джерелом (на випадок відмови відправника від цього факту) і його цілісність не порушена; підтвердження (доказ) одержання повідомлень надає відправнику докази того, що повідомлення було отримане одержувачем, у випадку спроб останнього відмовитися від цього факту.

Відмінність послуги причетності від автентифікації полягає в тому, що одержувач або відправник даних може довести третій стороні (арбітрові, судді) факт передачі (одержання) даних і невтручання сторонніх у процес передачі даних. Доступність визначена як додаткова послуга забезпечення захищеності мереж. Механізми забезпечення доступності запобігають появам атак, що має своєю метою зробити ресурси або послуги комп’ютерної системи недоступними (або зробити їх "якість" незадовільним) для користувача.

Доступність є характеристикою якості ресурсу або послуги і частково визначається послугою управління доступом. Однак характер атак з метою обмеження доступу користувача й засоби боротьби з ними не ставляться безпосередньо до питань управління доступом. Тому доцільно виділити окрему послугу забезпечення доступності, що повинна реалізовуватися спеціальними механізмами на мережному рівні (наприклад, можливістю використання альтернативного шляху при атаці на доступну смугу основного каналу) або прикладному рівні.

Спеціальні механізми безпеки

Стандарт ISO 7498-2 [115] містить короткий опис механізмів забезпечення безпеки й таблицю можливого співвідношення їх з рівнями моделі ВВС і послугами. Механізми безпеки є конкретними мірами для реалізації послуг безпеки. Стандарт ділить механізми безпеки на два класи, а саме спеціальні механізми забезпечення безпеки, які використовуються для реалізації специфічних послуг і відрізняються для різних послуг, і загальні механізми, які не висуваються до конкретних послуг.

До спеціальних механізмів забезпечення безпеки відносяться такі механізми:

шифрування (encipherment);

механізми цифрового підпису (digital signature mechanisms);

механізми управління доступом (access control mechanisms);

механізми забезпечення захисту цілісності даних (data integrity mechanisms), які включають криптографічні контрольні функції;

механізми автентифікації (authentication exchange mechanisms);

механізми заповнення трафіка (padding traffic mechanisms);

механізми управління маршрутизацією (routing control mechanisms);

механізми нотаризації (notarisation mechanisms).

Розглянемо кожний тип механізмів більш докладно.

Механізми шифрування. Шифрування допускає використання криптографічних перетворень даних для того, щоб зробити їх такими, що не читаються або не піддаються осмисленню. Шифрування застосовується разом зі зворотною функцією – розшифровування.

При шифруванні повідомлення M_i піддається криптографічному перетворенню F_k, у результаті чого формується криптограма:

$$C_i = F_k(M_i),$$

а розшифровування криптограми виробляється за правилом:

$$M_i = F_k^{-1}(C_i) = D_k(C_i),$$

де $k \in \{K\}_{N_{k\Delta}}$ – ключ.

Використовується шифрування із симетричними ("закритими", secret key) або несиметричними ("відкритими", public key) ключами.

При симетричному шифруванні ключі шифрування й дешифрування однакові, тобто $К_ш = К_{расш}$. Симетричне шифрування має сенс тільки тоді, коли ключ не відомий іншим об'єктам, тобто коли він секретний.

При несиметричному шифруванні ключі шифрування й дешифрування різні, тобто $К_ш \neq К_{расш}$. У цьому випадку об'єкт генерує пари ключів, один із яких стає загальнодоступним або відкритим (public key), а другий зберігається об'єктом у таємниці з особистим ключем. При цьому знання відкритого ключа не дає можливості обчислити особистий ключ.

Шифрування використовується для забезпечення послуги конфіденційності, але може також підтримувати інші послуги забезпечення безпеки. Така можливість існує, тому що будь-яка зміна шифрограми призводить до непередбачених змін вихідного тексту. При використанні шифрування можна також реалізовувати механізми забезпечення автентифікації та захисту цілісності даних. Слід відзначити, що при використанні шифрування особливу значущість має завдання генерації, зберігання і поширення криптографічних ключів, що є окремим завданням управління безпекою.

Найбільш відомими алгоритмами шифрування, які є по суті стандартами де факто, є алгоритми DES і RSA. Після невдалих спроб стандартизувати DES і RSA на міжнародному рівні було ухвалене рішення, що механізми шифрування не будуть більше об'єктами стандартизації в ISO/IEC. Замість цього здійснюється тільки реєстрація алгоритмів. Порядок реєстрації визначений у стандарті ISO/IEC 9979 (1991 року). Хоча механізми шифрування і не є об'єктами стандартизації, міжнародний стандарт ISO/IEC 10116 (1997 року, друга редакція) стандартизує режими роботи n-розрядних блокових симетричних алгоритмів шифрування [8].

Механізми цифрового підпису. Цифровий підпис є цифровим еквівалентом підпису (печатки, штампа тощо), наявність якої в повідомленні дозволяє з високою точністю визначити джерело повідомлення (документа) та юридично довести, що з певною імовірністю, тільки він міг створити й підписати цей документ.

Електронний цифровий підпис (ЕЦП) використовує "відкриті" ключі, генерується відправником даних і перевіряється одержувачем. Для шифрування контрольної суми повідомлення, що підписується, можуть

бути використані методи несиметричного шифрування. ЕЦП складається із двох процедур: процедури накладання (генерації, формування) підпису й процедури зняття (перевірки, верифікації, деструкції) підпису [35].

Використання "відкритих" ключів для ЕЦП служить для підтвердження походження повідомлення, але не контролює одержувача повідомлення. ЕЦП використовується для забезпечення послуг автентифікації і захисту цілісності, для яких суб'єкт верифікації підписаних даних заздалегідь невідомий. При певному виборі контрольованого параметра цифровий підпис також може застосовуватися для реалізації послуги підтвердження причетності.

Загальні механізми забезпечення безпеки. У стандарті ISO 7498-2 визначені такі загальні механізми забезпечення безпеки [115]: довірча функціональність (trusted functionality), мітки безпеки (security lables), виявлення подій (event detection), контроль безпеки (security audit trail), відновлення безпеки (security recovery).

Довірча функціональність використовується разом з іншими механізмами безпеки і становить сукупність рекомендацій і способів, які повинні бути реалізовані для забезпечення гарантії правильної і надійної роботи інших механізмів безпеки. Довірча функціональність допускає широке використання нормативної документації при розробці програмних або апаратних засобів, що реалізують механізми безпеки. Розробка цих засобів повинна вестися при дотриманні відповідних організаційних вимог. Програмні та апаратні засоби повинні розроблятися, тестуватися і сертифікуватися на основі єдиних методик. Тут же забезпечуються всі необхідні вимоги і рекомендації до електромагнітних випромінювань, можливостей фізичного втручання, з використання безпечних каналів поширення та ін.

Мітки безпеки. Будь-який ресурс (записані дані, обчислювальні потужності і/або комунікаційні послуги) може мати асоційовані з ними мітки безпеки, які позначають їх рівень таємності. Мітки безпеки можуть бути явно або побічно зв'язані як окремими пакетами даних, так і з послідовностями пакетів. Звичайно мітки безпеки використовуються для реалізації методики управління доступом на основі встановлених правил, а також для управління маршрутизацією.

Передані дані також можуть мати мітки безпеки, які передаються разом з ними безпечним чином. У цьому випадку для забезпечення захисту міток застосовуються криптографічні функції, а мітки безпеки використовуються для забезпечення контролю за цілісністю повідомлень.

Виявлення подій. Механізми виявлення подій у системах захисту інформації служать для виявлення як спроб порушення безпеки, так і для реєстрації легітимної активності користувачів. Виявлення може бути локальним і/або дистанційним і реалізується через сигналізацію про події (event reporting (alarm)), реєстрацію подій (event logging) і відбудовні дії (recovery actions). Реалізація механізмів виявлення спроб порушень безпеки – досить складне завдання, що вимагає залучення методів штучного інтелекту. Тут проблема полягає у визначенні того мінімуму інформації, який би дозволив виявити (або не пропустити) можливі події з втручання в роботу комп'ютерної системи.

Реєстрація контролю безпеки. Під контролем безпеки розуміють незалежний розгляд і аналіз записів безпеки з метою перевірки достатності управління системою, гарантувати відповідність функціонування системи політиці безпеки і рекомендувати необхідні зміни в управлінні, політиці і процесах безпеки. Звичайно розглядають дві процедури: протоколювання й аудит.

Під протоколюванням розуміється збір і нагромадження інформації про події, що відбуваються в інформаційній системі.

Під аудитом розуміється оперативний аналіз накопиченої інформації, проведений постійно або періодично.

Механізми протоколювання й аудиту служать для рішення таких завдань: забезпечення підзвітності користувачів і адміністраторів, що є засобом стримування; забезпечення можливості відновлення послідовності подій, що дозволяє виявити слабкі місця в захисті інформації, виявити винуватця вторгнення, оцінити масштаби заподіяного збитку і повернутися до нормальної роботи; надання інформації для виявлення та аналізу проблем, через підготовку відповідних звітів і рапортів.

Особливістю протоколювання й аудиту є їх залежність від послуг і механізмів безпеки. Так, ідентифікація й автентифікація слугують відправною точкою підзвітності користувачів. Для забезпечення конфіденційності та цілісності реєстраційної інформації застосовують механізми управління доступом. Можливе застосування й криптографічні засоби. Основною проблемою при реалізації механізмів протоколювання й аудиту є формулювання критеріїв відбору записів.

Відновлення безпеки. Механізми відновлення безпеки виконують функцію реакції системи на порушення безпеки. Такими діями можуть бути, наприклад, негайне роз'єднання або припинення роботи, відмова суб'єкта в доступі, тимчасове позбавлення суб'єкта прав, занесення суб'єкта в "чорний список" і т. д.

Розділ 2. Механізми забезпечення цілісності та автентичності інформації

Розглянуто основні механізми забезпечення автентичності і цілісності даних в інформаційних системах. Проведено аналіз механізмів забезпечення автентичності даних та цифрового підпису на основі використання кодів цілісності даних (MDC-кодів) і кодів автентичності даних (MAC-кодів).

Питання забезпечення цілісності й автентичності інформації є досить актуальним сьогодні. Оскільки методи криптоаналізу постійно вдосконалюються, потрібно проводити аналіз існуючих методів забезпечення автентифікації для усунення недоліків [5; 11; 31; 46].

Розповсюджені такі методи забезпечення автентичності повідомлення (рис. 2.1) [11; 15; 27; 39; 44]:

додавання до повідомлення коду автентифікації повідомлення (message authentification code, MAC-код) або зашифрованої контрольної суми;

введення цифрових підписів;

контрольні суми, контроль CRC, гешування та цифровий підпис – базові засоби автентифікації при цифровій передачі даних.

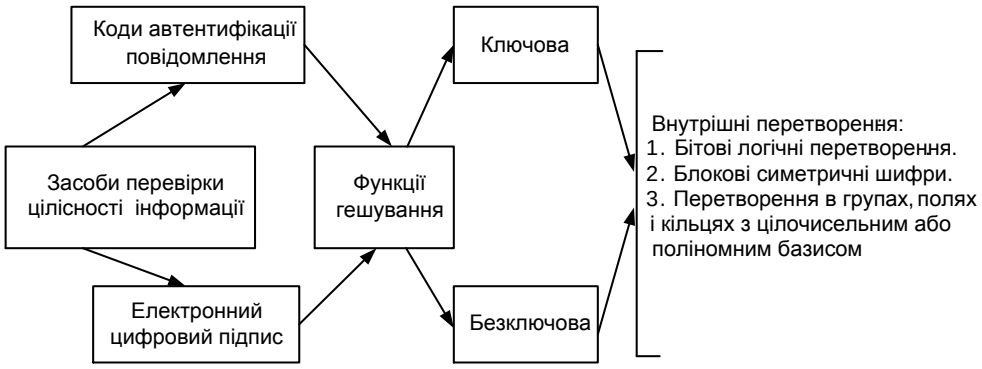


Рис. 2.1. **Класифікація засобів перевірки цілісності інформації**

2.1. Електронний цифровий підпис

Електронний цифровий підпис (ЕЦП) – реквізит електронного документа, призначений для захисту даного електронного документа від підробки, отриманий у результаті криптографічного перетворення

інформації з використанням закритого ключа електронного цифрового підпису, що дозволяє ідентифікувати власника сертифіката ключа підпису, а також установити відсутність перекручування інформації в електронному документі.

Основними стандартами ЕЦП є [14; 50 – 52; 110; 111]:

міжнародний стандарт ISO/IEC 9796, який визначає ЕЦП з відновленням повідомлення (digital signature with message recovery);

міжнародний стандарт ISO/IEC 14888, який визначає ЕЦП з додаванням (digital signature with appendix);

російський стандарт цифрового підпису на еліптичній кривій ГОСТ Р34.10-2001 [9];

американський національний стандарт цифрового підпису (FIPS 186);

американський фінансовий стандарт цифрового підпису з додаванням еліптичною кривої (ANSI X9.62) [52];

стандарт на ЕЦП PKCS #1, який визначає ЕЦП на основі алгоритму RSA [8];

стандарт цифрового підпису з додаванням і відновленням повідомлення IEEE 1363 [105];

стандарт цифрового підпису з додаванням еліптичної кривої IEEE P1363 [106];

міжнародний стандарт ISO/IEC CD 15946-2 стандартизується ЕЦП еліптичною кривою з додаванням;

Державний стандарт України ДСТУ-4145 – 2002 [14].

На основі існуючих стандартів ЕЦП в [8; 9; 14; 50 – 52; 105; 106; 113; 114] запропонована класифікація ЕЦП.

За способом побудови схеми ЕЦП діляться на два класи:

схема ЕЦП із відновленням повідомлення;

схема ЕЦП із додаванням.

За кількістю учасників ЕЦП підрозділяється:

одиночна схема ЕЦП;

групова схема ЕЦП.

У процесі виконання алгоритму формування цифрового підпису в одиночних схемах ЕЦП досить одного учасника, а у групових схемах їх два або більше.

За способом перевірки ЕЦП діляться на два класи:

інтерактивні схеми ЕЦП, що вимагають протокольної взаємодії;

не інтерактивні схеми ЕЦП, не потребуючі протокольної взаємодії.

Існуючі алгоритми ЕЦП можна розділити також за типами використовуваних односпрямованих функцій із секретом:

схеми ЕЦП, засновані на стійкості факторизації великого числа;

схеми ЕЦП, засновані на стійкості дискретного логарифма;

схеми ЕЦП, засновані на стійкості дискретного логарифма в групі точок ЕК.

Кожна з цих схем може бути детермінована або рандомована. Застосування детермінованих схем характеризується тим, що цифровий підпис одним і тим же вхідним рядком даних, приводить до формування однакових цифрових підписів. У рандомованій схемі при генерації підпису використовується деякий випадковий параметр, що приводить до формування різних підписів, навіть для однакових вхідних рядків. У рандомізованих схемах необхідно забезпечити непередбачуваність випадкових чисел [14].

У свою чергу детерміновані схеми поділяють на схеми ЕЦП одноразового застосування і схеми ЕЦП багаторазового застосування. Слід розглянути основні стандарти ЕЦП, що застосовуються у комплексних системах захисту банківської інформації, провести зіставлення основних характеристик ЕЦП (довжину ключів, довжину цифрового підпису, складність (час) обчислення і складність (час) перевірки дійсності цифрового підпису) з метою виявлення їх переваг і недоліків, за умови, що рівень стійкості підпису стосовно будь-яких методів фальсифікації не нижче, ніж 10^{21} (або 30 років безперервної роботи мережі з 1000 суперкомп’ютерів).

У якості "базової" довжини ключів і довжини самого цифрового підпису буде розглядатися довжина в 64 байти.

Цифрові підписи з відновленням повідомлення є об’єктом розгляду двох стандартів ISO/IEC 9796 (1991 року) і ISO/IEC 9796-2 (1997 року). У стадії розробки перебуває четверта частина стандарту ISO/IEC 9796-4. Механізми ЕЦП певні в ISO/IEC 9796 застосовуються тільки до коротких повідомлень, тоді як механізми ISO/IEC 9796-2 застосовуються до повідомлень довільної довжини.

Стандарт ISO/IEC 14888 визначає механізми ЕЦП другого класу. Дані механізми застосовані до повідомлень довільної довжини. При обчисленні цифрових підписів з додаванням особливу роль відіграють однобічні геш-функції. Геш-функції також є об’єктом міжнародної стандартизації. Зокрема основним нормативним документом у даній сфері

є міжнародний стандарт ISO/IEC 10118. Він складається з декількох частин і вводить модель геш-функції (ISO/IEC 10118-1(1994 року), розглядає два методи для побудови геш-функцій на основі блокових шифрів (ISO/IEC 10118-2 (1994 року), визначає три спеціалізованих (dedicated) геш-функції, тобто геш-функції, які розроблені спеціально для обчислення контрольних сум (ISO/IEC 10118-3 (1998 року) [107 – 109].

Одна з них є стандартом NIST "Secure Hash Standard" (SHS). Дві інші RIPEMD-128 і RIPEMD-160 є розробкою Європейські організації зі стандартизації в рамках проекту "RIPE". Нарешті ISO/IEC 10118-4 (1998 року) визначає дві геш-функції на основі модульного зведення у квадрат для використання з механізмами ЕЦП, що базуються на модульній арифметиці.

Відомі методи забезпечення автентичності та цілісності даних засновані на внесенні надмірності (імітовставки, коду автентифікації, цифрового підпису) в оброблювану послідовність. В останні роки ця галузь знань зазнає бурхливого розвитку, запропоновано велику кількість різних криптографічних методів і алгоритмів. Найбільшого поширення набули протоколи, засновані на використанні односторонніх геш-функцій.

Для опису процесів обробки інформації з використанням механізмів ЕЦП варто скористатися такою термінологією.

1. *Алгоритм генерації ЕЦП* – це метод формування ЕЦП.

2. *Алгоритм перевірки (верифікації) ЕЦП* – метод перевірки того, що підпис є автентичним, тобто дійсно створений конкретним об’єктом і не модифікований при передачі.

3. *Схема ЕЦП* (або *механізм ЕЦП*) – сукупність взаємозалежних алгоритмів генерації і верифікації цифрового підпису.

4. *Процес* (*процедура*) *накладання ЕЦП* – сукупність математичного алгоритму генерації ЕЦП і методів представлення (форматування) даних, що підписуються.

5. *Процес* (*процедура*) *зняття ЕЦП* – сукупність алгоритму верифікації ЕЦП і методів відновлення даних.

Для побудови схеми ЕЦП необхідно визначити два алгоритми: алгоритм генерації ЕЦП і алгоритм верифікації ЕЦП. Алгоритм верифікації доступний для всіх потенційних одержувачів підписаних повідомлень, у той час як алгоритм генерації ЕЦП відомий тільки тій особі, яка підписує, що для деякого повідомлення $m \in M$ визначає відповідний підпис $s \in S$. Верифікатор, одержавши пари (m, s) і деяку

відкриту інформацію про особу, що підписує, застосовує відповідний алгоритм верифікації ЕЦП. Даний алгоритм видає двійковий результат: "так", якщо підпис правильний (автентичний) і "ні" – в інакшому випадку.

Існуючі на сьогоднішній день схеми ЕЦП діляться на два класи (рис. 2.2):

схеми ЕЦП із відновленням повідомлення;

схеми ЕЦП із додаванням.

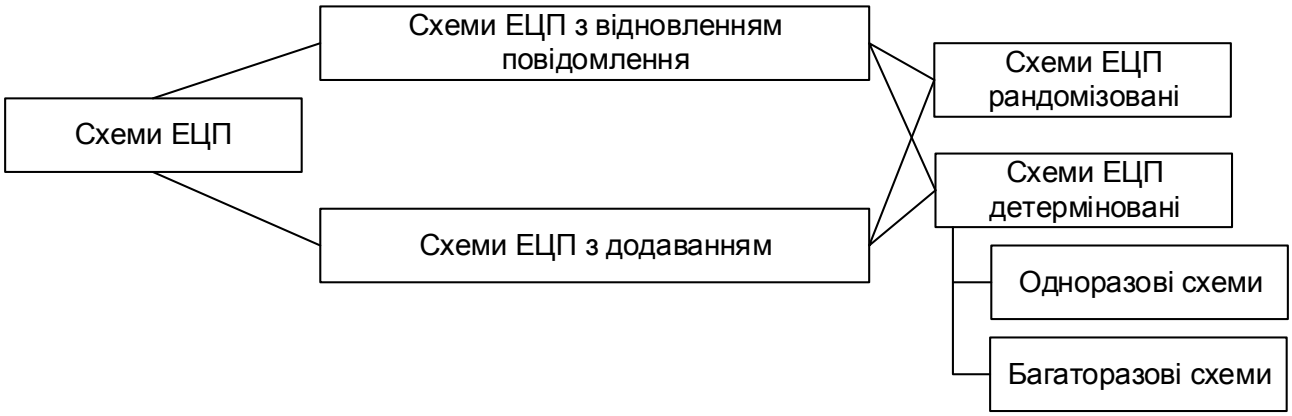


Рис. 2.2. **Класифікація схем ЕЦП**

У *схемах ЕЦП із відновленням повідомлення* всі або частина підписаного повідомлення може бути відновлена безпосередньо з цифрового підпису. Таким чином, на вхід алгоритму верифікації надходить лише цифровий підпис *s*.

У *схемах ЕЦП із додаванням* цифровий підпис приєднується до повідомлення й у такому вигляді відправляється адресату. Для верифікації такого ЕЦП необхідно мати і підпис *s,* і відповідне повідомлення *m*.

Кожна з цих схем може бути *детермінованою* або *рандомізованою*. Застосування детермінованих схем характеризується тим, що цифровий підпис одного і того ж вхідного рядка даних призводить до формування однакових цифрових підписів. У рандомізованій схемі при генерації підпису використовується деякий випадковий параметр (число), що призводить до формування різних підписів для однакових вхідних рядків (при використанні тих самих ключів). У рандомізованих схемах необхідно забезпечити непередбачуваність випадкових чисел.

У свою чергу детерміновані схеми поділяються на схеми ЕЦП одноразового застосування (one-time) і схеми ЕЦП багаторазового застосування (multiple-use).

Цифрові підписи з додаванням

Механізми цифрового підпису з додаванням є об'єктом декількох стандартів. На міжнародному рівні ці механізми визначає стандарт ISO/IEC 14888. Американський національний стандарт FIPS 186 і російський національний стандарт ГОСТ 34.310 стандартизують схеми ЕЦП із додаванням, які є варіантами схеми El Gamal [20]. Стандартизацією схем ЕЦП займаються і дослідницькі організації. Тут виділяється стандарт IEEE P1363, який визначає ЕЦП на основі несиметричної криптографії і промисловий стандарт PKCS № 1 [106].

Стандарт ISO/IEC 14888 складається з трьох частин. Перша частина стандарту ISO/IEC 14888-1 вводить загальну модель схеми ЕЦП із додаванням [110].

Друга частина визначає схеми підпису на основі використання ідентифікаційної інформації [111].

Нарешті, у ISO/IEC 14888-3 описується два типи схем. По-перше, це схеми на основі використання проблеми дискретного логарифма, а саме: схема DSA, схема Pointcheval-Vaudenay і схема ECDSA (схема DSA на еліптичних кривих) [112]. По-друге, це схеми, стійкість яких заснована на проблемі факторизації. У стандарті подана: схема підпису за ISO/IEC 9796-1 з використанням гешування і схема ESIGN.

Схеми підпису на основі використання ідентифікаційної інформації. Схема підпису Гіллоу-Куіскуотера (GQ-схема)

У схемах формування підпису на основі ідентифікаційної інформації відкритий ключ перевірки підпису формується з використанням ідентифікаційної інформації сторони, яка підписує. Таким чином, відпадає необхідність використання сертифікатів відкритих ключів. Особисті (секретні) ключі генерації підпису повинна генерувати довірча третя сторона. У таких схемах ДТС матиме доступ до всіх особистих ключів. У зв'язку з цим схема формування підпису на основі використання ідентифікаційної інформації може використовуватися не для всіх додатків. Найчастіше вона реалізується в закритих доменах безпеки, наприклад, усередині великої компанії або корпоративних закритих мереж, де існує "природна" довірча сторона.

ISO/IEC 14888-2 визначає три схеми підпису, що відносяться до класу рандомізованих схем і є різними варіантами схеми підпису Гіллоу – Куіскуотера (Guillou-Quisquater). В основу *GQ*-схеми підпису покладено

використання ідентифікаційного протоколу GQ. Схема підпису, визначена в ISO/IEC 14888-2, допускає обов'язкове залучення ДТС [111]. ДТС генерує особисті ключі підпису, у той час як ключ верифікації може генеруватися верифікатором.

Генерація ключів для GQ-схеми підпису.

У результаті генерації ключів утворюється відкритий ключ (n, e, I_A) і відповідний особистий ключ *a*.

Довірча третя сторона виконує такі дії:

1. Обрати випадкові різні прості числа p і q, які зберігаються в секреті, і формує модуль $n = p \times q$.

2. Обрати ціле $e \in \{1, 2, \ldots, n-1\}$ таке, що НСД $(e,(p-1)(q-1)) = 1$.

3. Будь-яка сторона, що бажає використовувати GQ-схему підпису, надає ДТС унікальну ідентифікаційну інформацію у вигляді двійкового рядка I.

Ціле число I_A, $1 < I_A < n$ є ідентифікатором сторони A (тобто містить інформацію про ім'я, адресу, номер паспорта, PIN і т. д.). ДТС на основі інформації I_A формує величину:

$$Y = f(I_A),$$

де f – функція введення надмірності (ISO/IEC 9796-1) така, що НСД (Y, n) = 1.

4. ДТС визначає ціле число $a \in Z_n$ таке, що $Ya^e \equiv 1 \pmod n$ відповідно до такого алгоритму.

4.1. Обчислення $Y^{-1} \bmod n$.

4.2. Обчислення $d_1 = e^{-1} \bmod (p-1)$ і $d_2 = e^{-1} \bmod (q-1)$

4.3. Обчислення $a_1 = \left(Y^{-1}\right)^{d1} \bmod p$ і $a_2 = \left(Y^{-1}\right)^{d2} \bmod q$.

4.4. Пошук значення a, що задовольняє рівнянням:

$$a = a_1 \pmod p;$$
$$a = a_2 \pmod q.$$

За відкритий ключ сторони A приймається вектор (n, e, I_A). За особистий ключ сторони A приймається число a.

Процедури накладання і зняття *GQ*-підпису.

Процедура накладання *GQ*-підпису містить у собі виконання 3 кроки. Сторона A підписує двійкове повідомлення m довільної довжини. Для цього сторона A повинна:

1. Вибрати випадкове число k, $1 \le k \le n-1$, і обчислити $r = k^e \bmod n$.

2. Обчислити геш-код $v = h(m||r)$.

Значення геш-коду повинно задовольняти нерівності $0 < v < e - 1$. Якщо v не задовольняє нерівності, то значення v приводять за модулем e.

3. Обчислити $s = r \times a^v \bmod n$.

Процедура зняття *GQ*-підпису. Для того щоб верифікувати цифровий підпис (s, v) за повідомленням m, сторона B повинна одержати повідомлення m, підпис (s, v) і автентичний відкритий ключ сторони A (n, e, I_A). Далі сторона B повинна обчислити:

$$u = s^e I_A^v \bmod n;$$
$$v' = h(m||n).$$

Якщо $v = v'$, підпис вважається дійсним. Справедливість цього твердження випливає з того, що:

$$u \equiv s^e I_A^v \equiv (ka^v)^e I_A^v \equiv k^e (a^e I_A)^v \equiv k^e \equiv r (\bmod n).$$

Отже, $u = r \times i$, звідси, $v = v'$.

Стійкість схеми *GQ*-підпису спирається на складність завдання факторизації складового модуля. Сучасні методи розкладання числа на складові множники і будуть визначати вимоги до параметрів схеми. Останні досягнення в сфері факторизації свідчать про те, що модуль n повинен бути, принаймні, 768-бітним. З метою забезпечення стійкості схеми ЕЦП до атак, що спираються на парадокс дня народження, довжина експоненти e повинна бути не менше 128 бітів. Розміри геш-кодів, що можуть застосовуватися в *GQ*-схемі підпису, рівні 128 або 160 бітам. Таким чином, при 768-бітному модулі n і 128-бітній експоненті e розмір відкритого ключа складе 896+u бітів, де u – кількість бітів, необхідна для представлення ідентифікатора I_A. Довжина особистого ключа a складе 768 бітів.

З погляду продуктивності *GQ*-схема вимагає виконання двох операцій зведення в степінь за модулем і однієї операції модульного множення. При використанні 768-бітного модуля n 128-бітної величини e і 128-бітного геш-коду v генерація підпису потребуватиме виконання приблизно такої ж кількості операцій. При схожих початкових умовах схема підпису RSA вимагає виконання 1152 модульних множень, а схема Фейга – Фіата – Шаміра (Feige – Fiat – Shamir) або схема FFS –

64 модульні множення. Але порівняно з останньою, *GQ*-схема вимагає значно меншого місця для збереження ключів. (У схемі FFS довжина особистого ключа складе 98304 біти, а довжина відкритого ключа – 99072 біти).

Схеми підпису на основі використання властивостей дискретного логарифма

Третя частина стандарту ISO/IEC 14888-3 описує схеми підпису на основі використання дискретного логарифма (DSA і схема Pointcheval-Vaudenay), схеми DSA на еліптичних кривих – ECDSA і схеми підпису на основі використання проблеми факторизації; схеми підпису заISO/IEC 9796-1 з гешуванням і схеми підпису ESIGN [102; 112].

Схема підпису DSA (Digital Signature Algorithm) була прийнята як федеральний стандарт США в 1993 р. як складова частина стандарту цифрового підпису DSS (Digital Signature Standard) FIPS 186. Стандарт FIPS 186 визначає використання геш-функції SHA-1 (FIPS 180-1, ISO/IEC 10118-3) разом з DSA [108; 112].

DSA є розвитком схем цифрових підписів Ель Гамаля й К. Шнорра та ЕЦП із додаванням. Стійкість схеми підпису спирається на складність задачі взяття дискретного алгоритму.

Використання алгоритму.

Для підписування повідомлень необхідна пара ключів – відкритий і закритий. При цьому закритий ключ повинен бути відомий тільки тому, хто підписує повідомлення, а відкритий – будь-якому бажаючому перевірити справжність повідомлення. Також загальнодоступними є параметри самого алгоритму. Для забезпечення такого доступу достатньо того, щоб авторитетна організація (або кілька організацій) підтримувала базу відповідності між реальними реквізитами автора (це може бути як приватна особа, так і організація) і відкритими ключами, а також усіма необхідними параметрами схеми цифрового підпису (геш-функція, що використовується). Ця організація також видає цифрові сертифікати.

Параметри схеми цифрового підпису.

Для побудови системи цифрового підпису бажаючий повинен зробити такі дії:

1. Обрати криптографічну геш-функцію H(x).

2. Обрати велике просте число q, розмірність якого N у бітах збігається з розмірністю в бітах значень геш-функції H(x).

3. Обрати просте числа p, такого, що (p – 1) ділиться на q. Бітова довжина p позначається L ($2^{L-1} \le p \le 2^L$).

4. Обрати число g таке, що його мультиплікативний порядок по модулю p дорівнює q. Для його обчислення можна скористатися $g = h^{(p-1)/q} \bmod p$ формулою $g = h^{(p-1)/q}$, де h – деяке довільне число, таке, що g ≠ 0. У більшості випадків значення h = 2 задовольняє цій вимозі.

Як уже згадувалось, а також у DSS (Digital Signature Standard), першочерговим параметром схеми цифрового підпису є криптографічна геш-функція, що використовується, необхідна для перетворення тексту повідомлення в число. Важливою характеристикою цієї функції є бітова довжина вихідної послідовності, що позначається далі N (160 для функції SHA-1). У першій версії стандарту DSS рекомендована функція SHA-1 і, відповідно, бітова довжина числа, що підписується, 160 біт. Зараз SHA-1 вже не є достатньо безпечним. У стандарті зазначені такі можливі пари значень чисел L і N:

1. L = 1024, N = 160.
2. L = 2048, N = 224.
3. L = 2048, N = 256.
4. L = 3072, N = 256.

Відповідно до цього рекомендовані геш-функції сімейства SHA-2. Урядові організації повинні використовувати один з цих варіантів, але всі інші вільні вибирати. Людина, яка проектує систему може вибрати будь-яку геш-функцію. Тому далі увага не буде загострюватися на використанні конкретної геш-функції. Стійкість криптосистеми на основі DSA не перевершує стійкість геш-функції, що використовується, і стійкість пари (L, N), чия стійкість не більше стійкості кожного з чисел окремо. Раніше рекомендувалася довжина p L = 1024 біта. У даний момент для систем, які повинні бути стійкими до 2010 (2030) р., рекомендується довжина в 2048 (3072) біт.

Відкритий і секретний ключі.

1. Секретний ключ становить число $x \in (0, q)$.

2. Відкритий ключ обчислюється за формулою $y = g^x \bmod p$.

Відкритими параметрами є числа (p, q, g, y). Закритий параметр тільки один – число x. При цьому числа (p, q, g) можуть бути загальними для групи користувачів, а числа x і y є відповідно закритим і відкритим

ключами конкретного користувача. При підписанні повідомлення використовуються секретні числа x і k, причому число k повинне вибиратися випадковим чином (на практиці псевдовипадковим) при підписуванні кожного такого повідомлення.

Оскільки (p, q, g) можуть бути використані для декількох користувачів, на практиці часто ділять користувачів за деякими критеріями на групи з однаковими (p, q, g). Тому ці параметри називають доменними параметрами (Domain Parameters).

Генерація ключів DSA.

У ході генерації ключів кожен об'єкт формує секретний ключ і відповідний особистий ключ.

Будь-який об'єкт A повинен виконати такі операції:

1. Обрати просте число q таке, що $2^{159} < q < 2^{160}$.

2. Обрати параметр t такий, що $0 \le t \le 8$ і обрати просте число p таке, що $2^{511+64t} < p < 2^{512+64t}$ і розкладання $(p - 1)$ має у своєму розкладанні простий співмножник q.

3. Визначити генератор (первісний елемент) Θ циклічної групи порядку q в Z_p^*. Для цього:

3.1. Обрати елемент $g \in Z_p^*$ і обчислити $\Theta = g^{(p-1)/q} \bmod p$.

3.2. Якщо $\Theta = 1$, то повернутися в 3.1.

4. Обрати випадкове ціле число a таке, що $1 \le a \le q - 1$.

5. Обчислити число $y = \Theta^a \bmod p$.

6. Відкритим ключем сторони A є вектор параметрів (p, q, Θ, y), особистим ключем є число *a*.

При формуванні простих чисел першим формується число q, а на його основі генерується число p. Алгоритм генерації простих чисел, що рекомендується викладений у FIPS 186.

Процедура накладання цифрового підпису.

Нехай об'єкт A бажає підписати двійкове повідомлення m довільної довжини. Для цього об'єкт A повинен виконати такі дії:

1. Обрати випадкове секретне ціле число k, $0 < k < q$. Для кожного знову обчисленого підпису обираються різні і непередбачені числа k. Таким чином, DSA – рандомізована схема.

2. Обчислити число $r = (\Theta^k \bmod p) \bmod q$.

3. Обчислити $k^{-1} \bmod q$.

4. За повідомленням m обчислити геш-код h(m). Потім двійковий рядок h(m) перетворити в ціле число.

5. Обчислити $s = k^{-1}\{h(m) + ar\} \bmod q$.

6. Підпис за повідомленням m є парою чисел (r, s).

Процедура зняття цифрового підпису.

Для того щоб верифікувати цифровий підпис (r, s) під повідомленням m, сторона B повинна виконати такі дії.

1. Одержати автентичну копію відкритого ключа об'єкта A (p, q, Θ, y).

2. Перевірити, що $0 < r < q$ та $0 < s < q$. У інакшому випадку підпис вважається недійсним.

3. Обчислити значення геш-коду h(m) і:

$$w = s^{-1} \bmod q;$$

$$u_1 = w \cdot h(m) \bmod q;$$

$$u_2 = r \cdot w \bmod q;$$

$$v = (\Theta^{u_1} y^{u_2} \bmod p) \bmod q.$$

4. Якщо v = r, то підпис приймається як дійсний.

Коректність підпису випливає з такого. Якщо (r, s) є правильним підписом об'єкта A, тоді справедливе порівняння $h(m) \equiv -ar + ks \pmod q$.

Помноживши обидві частини порівняння на w, одержимо $w \times h(m) + arw \equiv k \pmod q$. Але з іншої сторони це порівняння не що інше, як порівняння вигляду $u_1 + au_2 \equiv k \pmod q$.

Зведення первісного елемента Θ у степінь, показником якого виступають обидві частини порівняння, дає такий вираз:

$$(\Theta^{u_1} y^{u_2} \bmod p) \bmod q = (\Theta^k \bmod p) \bmod q.$$

Отже, v = r, що і потрібно було довести.

Стійкість схеми DSA ґрунтується на проблемі дискретного логарифма в групі Z_p^* і циклічній підгрупі порядку q в Z_p^*.

Розмір числа q є фіксованим і становить 160 бітів, у той час як розмір числа p може приймати будь-яке значення, кратне 64 у межах від 512 до 1024 бітів. Використання 512-бітного числа є мінімально допустимою умовою реалізації схеми. Стандарти рекомендують застосовувати схему з довжиною модуля не менше 768 бітів. Стандарт FIPS 186 не дозволяє застосовувати прості числа p довжиною більше ніж 1024 бітів.

Процедура накладання підпису вимагає виконання однієї операції модульного зведення в степінь, який вимагає в середньому близько 240 модульних множень (при довжині модуля 768 бітів), однієї операції обчислення зворотного елемента з 160-бітним модулем, двох 160-бітних модульних множень і одного додавання. Основний внесок у часові витрати вносить операція модульного зведення в степінь. Однак перевагою DSA є можливість здійснення передобчислень. Випадкова величина k не залежить від повідомлення і може бути обрана заздалегідь. Оскільки значення r є функцією від k, воно також може бути обчислене заздалегідь. Аналогічно можна знайти значення ar і k^{-1}. Таким чином, підпис може бути згенерований досить швидко. Усе що потрібно для генерації підпису – це обчислити геш-код h(m) + ar і здійснити модульне множення результату на k^{-1}. Основний обчислювальний внесок при знятті підпису виконується в дві операції: зведення в степінь за модулем p і з 160-бітним показником степеня. У середньому на виконання кожної з цих операцій буде потрібно 240 модульних множень. У цьому DSA поступається RSA, у якій як секретний ключ можливе використання малих показників, що значно спрощує верифікацію. Зовсім не обов'язково, щоб кожен об'єкт мав свої особисті числа p і q. Стандарт DSS дозволяє використання p, q і Θ як системних параметрів, хоча це більш привабливий варіант для зловмисника. На рис. 2.3 наведено схему створення і перевірки підпису за алгоритмом DSA.

Коректність схеми.

Дана схема цифрового підпису коректна тією мірою, що бажаючий перевірити справжність підпису завжди отримає позитивний результат у випадку автентичності. Потрібно показати це:

По-перше, якщо $g = h^{(p-1)/q} \bmod p$, оскільки g > 1 і q просте число, то r повинно мати мультиплікативний порядок q по модулю p.

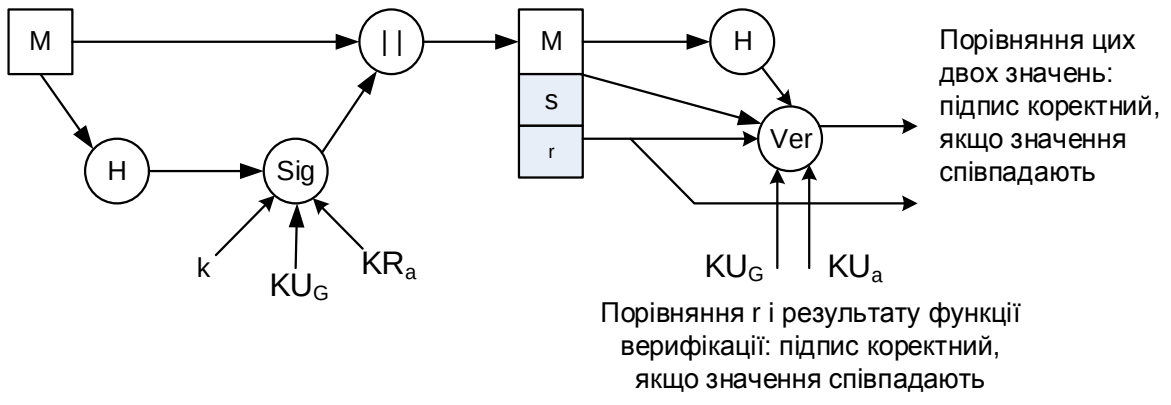


Рис. 2.3. **Створення і перевірка підпису за алгоритмом DSA**

Для підпису повідомлення обчислюється:

$$g^k = g^{h(m)} \cdot w \bmod q^{x-r-w} \bmod q = g = h^{(p-1)/q} \bmod p.$$

Нарешті, коректність схеми DSA випливає з:

$$r = (g^k \bmod p) \bmod q = (g^{u1} y^{u2} \bmod p) = u.$$

Оскільки фактично підписується не саме повідомлення, а його геш, то очевидно, що кілька різних повідомлень можуть мати однаковий підпис. Це пов'язано з наявністю у геш-функцій колізій (існують множини повідомлень такі, що всередині кожної такої множини геш-значення елементів збігаються).

Схема ESIGN

Схема ESIGN (Efficient digital SIGNature) запропонована корпорацією Nippon (Японія) і є схемою, чия стійкість заснована на складності розв'язання задачі факторизації великого цілого числа.

Генерація ключів.

Кожен об'єкт створює відкритий ключ і відповідний закритий ключ. Для цього об'єкт A повинен виконати такі операції.

1. Обрати два великих простих числа p і q таких, що $p \geq q$ і p, q приблизно рівних за довжиною.
2. Обчислити модуль $n = p^2 q$.
3. Обрати позитивне ціле число $k \geq 4$.
4. Відкритим ключем сторони A є пара чисел (n, k), закритим ключем – (p, q).

Процедура накладання підпису.

Алгоритм генерації підпису обчислює ціле число s таке, що $s^k \bmod n$ лежить у визначеному інтервалі, обумовленому повідомленням m. Для того, щоб підписати повідомлення m, що є двійковим рядком довільної довжини, об'єкт A повинен виконати такі дії.

1. Обчислити геш-код v = h(m).
2. Обрати випадкове секретне ціле число x, $0 \leq x \leq pq$.

3. Обчислити:

$$w = \lceil ((v - x^k) \bmod n)/(p \cdot q) \rceil;$$
$$y = w \cdot (kx^{k-1})^{-1} \bmod p;$$
$$s = x + ypq \bmod n.$$

4. Цифровим підписом повідомлення *m* є число *s*.

Процедура верифікації підпису.

У ході верифікації демонструється, що величина $s^k \bmod n$ належить визначеному інтервалу. Для цього об'єкт B повинен виконати такі дії:

одержати автентичний відкритий ключ (n, k) сторони A.

Обчислити:

$$u = s^k \bmod n;$$
$$z = h(m).$$

Прийняти підпис, якщо:

$$z \le u \le z + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil}.$$

У інакшому випадку підпис вважається недійсним.

Коректність підпису випливає з доказу.

Слід зазначити, що:

$$s^k \equiv (x + ypq)^k \equiv \sum_{i=0}^{k} \binom{k}{i} x^{k-i} (ypq)^i \equiv x^k + kypqx^{k-1} (\bmod n).$$

Але $kx^{k-1}y \equiv w \bmod p$ i, таким чином $kx^{k-1}y \equiv w + lp$ для деяких $l \in Z$.

Отже,

$$s^k \equiv x^k + pq(w + lp) \equiv x^k + pqw \equiv x^k + pq\left\lceil \frac{(h(m) - x^k) \bmod n}{pq} \right\rceil \equiv$$

$$\equiv x^k + pq\left(\frac{h(m) - x^k + jn + \varepsilon}{pq} \right) (\bmod n), \text{ де } \varepsilon = (x^k - h(m)) \bmod pq.$$

Звідси $s^k \equiv x^k + h(m) - x^k + \varepsilon \equiv h(m) + \varepsilon \ (\bmod n)$.

Оскільки $0 \le \varepsilon < pq$, то:

$$h(m) \le s^k \bmod n \le h(m) + pq \le h(m) + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil},$$

що і потрібно було довести.

У схемі підпису ESIGN використовується модуль $n = p^2q$, що відрізняється від RSA-модулів. Розроблювачі ESIGN стверджують, що за стійкістю дана схема не поступається схемам RSA і DSA. Однак не відомо, наскільки простіше або складніше розкладання модуля $n = p^2q$ на складові співмножники порівняно зі звичайним модулем $n = pq$.

Специфічний доказ коректності підпису ESIGN дозволяє реалізувати такі атаки.

Для даного правильного підпису s за повідомленням m зловмисник може підробити підпис для іншого повідомлення м', якщо геш-код h(м') приймає значення таке, що:

$$h(m') \le u \le h(m') + 2^{\left\lceil \frac{2}{3} \lg n \right\rceil},$$

де $u = s^k \bmod n$.

Якщо буде знайдене повідомлення м', що володіє такими властивостями, то підпис s буде дійсним і для нього. Така ситуація можлива, якщо h(m) і h(м') збігаються в (lg n)/3 старших значущих бітах. Припускаючи, що h має властивості випадкової функції, зловмиснику для пошуку такого повідомлення необхідно випробувати $2^{(\lg n)/3}$ варіантів.

Інший можливий варіант підробки підпису полягає в перебуванні пари повідомлень m і м' таких, що h(m) і h(м') збігатимуться в старших (lg n)/3 бітах. Відповідно до парадокса дня народження необхідно випробувати $O(2^{(\lg n)/6})$ варіантів. Якщо зловмисник здатний сформувати коректний підпис для повідомлення m, то цей же підпис буде дійсним і для повідомлення м'.

Розглянуті приклади атак і визначають вимоги до величини модуля.

Схема підпису ESIGN з погляду продуктивності перевершує схему підпису RSA. Крім того, схема ESIGN дозволяє виконувати передобчислення, що сприяють збільшенню обчислювальної ефективності

алгоритму. Після вибору випадкового числа *x* сторона А може здійснити такі передобчислення:

$$u_1 = x^k \bmod n;$$
$$u_2 = 1 / (kx^{k-1}) \bmod p.$$

Потім визначити:

$$w = \lceil w\text{-}u_1 / (pq) \rceil;$$
$$s = x + (wu_2 \bmod p) \times pq.$$

Використання передобчислювань дозволяє збільшити швидкість формування підпису до 10 разів. На ефективність обчислень підпису впливає значення параметра k. Розроблювачі рекомендують вибирати його з множини k = {4, 8, 16, 32, 64, 128, 256, 512, 1024}.

Так, для k = 4 і 768-бітний модулі n генерації підпису ESIGN перевершує за швидкістю генерацію підпису RSA від 10 до 100 разів. Процедура верифікації також досить ефективна і порівняна зі схемою RSA з малими відкритими ключами. Даний алгоритм може бути реалізований і на еліптичних кривих.

Цифрові підписи з відновленням повідомлення

Схеми ЕЦП із відновленням повідомлення є об'єктом стандартизації міжнародного стандарту ISO/IEC 9796. Даний міжнародний стандарт складається з трьох частин:

ISO/IEC 9796-1 стандартизує схему ЕЦП на основі RSA-перетворення;

ISO/IEC 9796-2 стандартизує схему ЕЦП аналогічну схемі, визначеній в першій частині, але використовуючи для введення надмірності геш-функцію. У даному варіанті схеми ЕЦП забезпечується часткове відновлення повідомлення і схема може бути використана для підпису повідомлень довільної довжини;

ISO/IEC 9796-3 описує схему, в якій як алгоритм генерації підпису використовується дискретний логарифм (схема ElGamal).

Схема підпису за ISO/IEC 9796-1

Загальна ідея застосування цифрового підпису з відновленням повідомлення полягає в тому, що вихідне коротке повідомлення подовжується, потім у нього вводиться надмірність, після чого воно

підписується. Схема ЕЦП, визначена в ISO/IEC 9796-1, забезпечує необхідні показники стійкості тільки при дотриманні всіх обмежень, визначених у стандарті. Схема ЕЦП за ISO/IEC 9796-1 заснована на використанні криптографії з відкритими ключами з використанням схеми RSA або Rabin. Алгоритм підпису повинен відображати *k*-бітний вхідний рядок у *k*-бітний рядок підпису. Схема не вимагає застосування геш-функциії і може використовуватися для підпису коротких повідомлень, що потім відновлюються з підпису.

При описі схеми ЕЦП використовуються такі позначення:

s – двійковий рядок підпису;

$d \le 8\lfloor (s+3)/16 \rfloor$ – двійковий рядок повідомлення m, яке підписується, обирається так, що:

$z + \lceil d/8 \rceil$ – кількість байтів у доповненому повідомленні;

$r = 8z - d + 1$ – число, більше на одиницю кількості доповнених бітів;

$t = \lceil (s-1)/16 \rceil$ – найменше ціле число таке, що рядок з 2t байтів включатиме принаймні s – 1 біт.

Приклад. Нехай необхідно підписати повідомлення m довжиною 150 бітів. При цьому довжина підпису повинна дорівнювати 1024 бітам. Тоді наведені параметри приймуть такі значення: s = 1024 біти, d = 150 бітів, z = 19 байтів, r = 3 біти, t = 64 байти.

Процес накладання ЕЦП за ISO/IEC 9796-1.

Процес накладання ЕЦП містить у собі виконання п’яти кроків (рис. 2.4).

1. *Доповнення.* Нехай m – повідомлення, які необхідно підписати. Доповнення полягає у формуванні доповненого повідомлення MP (massage padding) шляхом конкатенації ліворуч вихідного повідомлення m з r нульовими бітами:

$$MP = 0^{r-1} \| m, z,$$

де $1 \le r \le 8$ обирається таким, що кількість бітів у MP буде кратним восьми.

Кількість байтів у MP позначатимемо символом:

$$MP = m_z \| m_{z-1} \| \ldots \| m_2 \| m_1,$$

де m_i – байт.

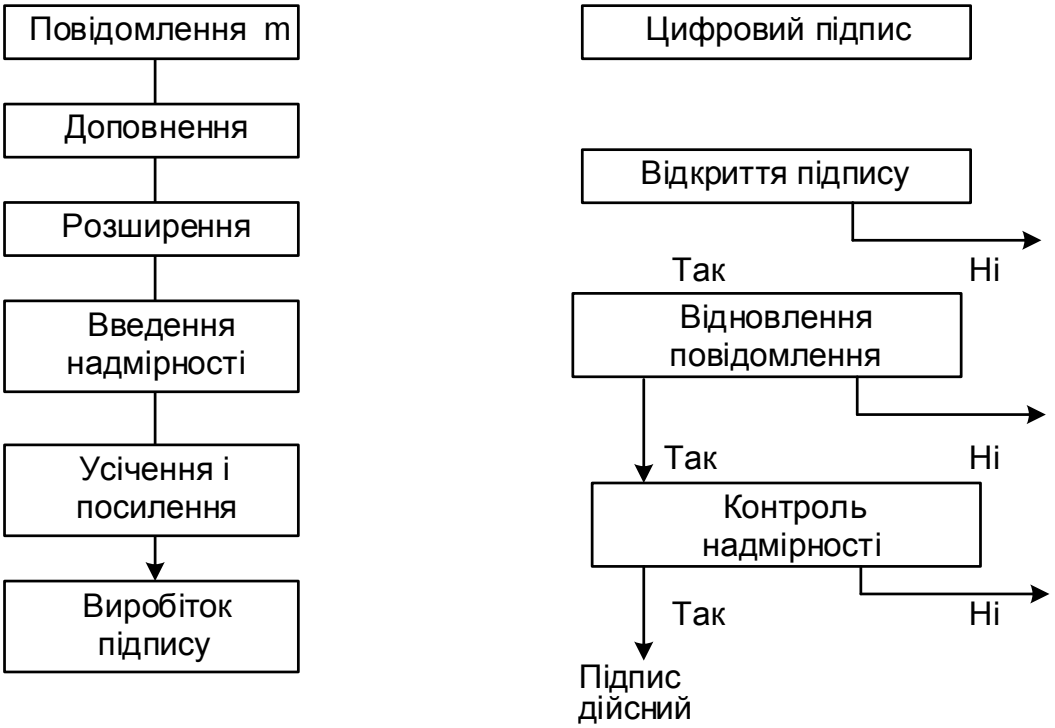
2. *Розширення повідомлення.* З доповненого повідомлення формується розширене повідомлення ME (message extension) шляхом ітеративної конкатенації ліворуч доповненого повідомлення із самим собою.

Повторення конкатенації здійснюється доти, поки не буде сформований рядок:

$$ME = ME_t \| ME_{t\ 1} \| ... \| ME_2 \| ME_1,$$

який містить рівно t байтів. Якщо число t не кратне числу z, то останні приєднані байти будуть частиною множини байтів з MP, що є послідовними праворуч доповненого повідомлення, тобто:

$$ME_i = m_{i \bmod z+1}, \forall i = \overline{0, t-1},$$



Процедура накладення підпису Процедура верифікації підпису

Рис. 2.4. **Процедури накладання і верифікації підпису за ISO/IES 9796-1**

Уведення надмірності. Надмірність вводиться в ME з метою одержання надлишкового повідомлення MR (message redundancy) вигляду $ME = ME2_t \| ME_{2t-1} \| ... \| ME_2 \| ME_1$. Уведення надмірності здійснюється в такий спосіб – надлишкове повідомлення формується

шляхом перемноження t байтів розширеного повідомлення з t надлишковими байтами, з подальшим узгодженням M_{2z}-го байта отриманого рядка. Усі байти надлишкового повідомлення формуються згідно з такими виразами:

$$MR_{2i-1} = ME_i, MR_{2i} = F(M_i) \forall i = \overline{1,t}$$

де F(u) – функція перетворення байта u.

Функція F(u) визначена в такий спосіб – якщо байт $u = u_1 \| u_2$, де u_1 і u_2 – напівбайти, то:

$$F(u) = \pi(u_2) \| \pi(u_1),$$

де π – перестановка вигляду:

$$\pi = \begin{pmatrix} 0 & 12 & 34 & 56 & 78 & 9A & BC & DE & F \\ E & 35 & 89 & 42 & F0 & DB & 67 & AC & 1 \end{pmatrix}.$$

Нарешті, останній байт MR формується шляхом заміни байта M_{2z} на байт вигляду $r \oplus MR_{2z}$. Таке перетворення дозволяє верифікатору підпису відновити довжину повідомлення. Оскільки довжина повідомлення дорівнює $d = 8z - r + 1$, то для визначення значення d досить визначити значення z і r. Ці значення можуть бути відновлені з MR.

4. *Усічення і посилення.* На основі надлишкового повідомлення формується s-бітне проміжне ціле число IR. Формування IR здійснюється в такий спосіб:

а) до s – 1 молодшим значущим бітам надлишкового повідомлення ліворуч додається одиничний біт. Інші старші біти (якщо вони є) просто відкидаються;

б) здійснюється модифікація молодшого значущого байта $u_2 \| u_1$ шляхом заміни його на байт вигляду $u_1 \| 0110$.

Таким чином, IR = 6 (mod 16). На рис. 2.5. наведено процес усічення і посилення.

5. *Генерація підпису.* На цьому кроці формується цифровий підпис шляхом застосування алгоритму генерації ЕЦП до рядка *IR:* $S = SIG_k(IR)$.

Схема ЕЦП ISO/IEC 9796-1 орієнтована на генерацію ЕЦП на основі алгоритмів RSA або Rabin. Необхідно розглянути застосування даних алгоритмів.

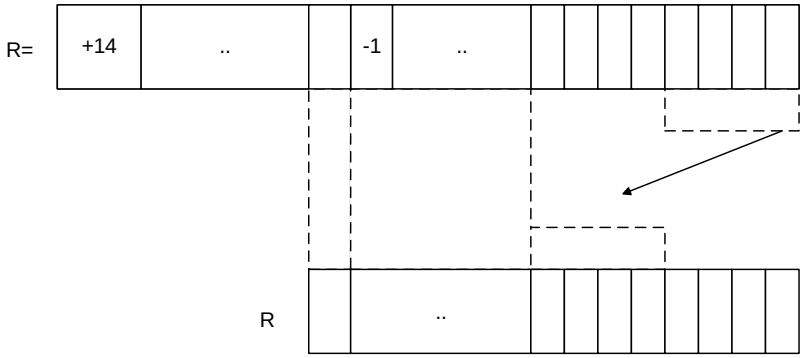


Рис. 2.5. **Формування рядка *IR***

Алгоритм генерації ЕЦП RSA за ISO/IEC 9796-1

Спочатку сторона, яка підписується, формує необхідні ключі відповідно до такого алгоритму.

Сформувати два великих відмінних один від одного простих числа p і q, приблизно однієї довжини. При цьому $p \neq q \bmod 8$.

Обчислити $n = pq$ і $\varphi(n) = (p-1)(q-1)$.

Вибрати випадкове ціле число v, при цьому $1 < v < \varphi(n)$

$(e, p-1) = 1$ і $(e, q-1) = 1$, при e непарному;

$\left(e, \frac{p-1}{2}\right) = 1$ і $\left(e, \frac{q-1}{2}\right) = 1$, при e парному.

Використовуючи розширений алгоритм Евкліда, обчислити число v з рівнянь:

$ve \equiv 1 \bmod \varphi(n)$, при e парному;

$ve \equiv 1 \bmod \frac{\varphi(n)}{2}$, при e непарному.

Відкритим ключем сторони, що підписує, є пара чисел (e,n). Число *e* ще називають перевірочним компонентом або ключем верифікації. Особистим ключем є число d.

Обчислення підпису здійснюється в два етапи. Спочатку з рядка IR формується репрезентативний елемент RR. Формування RR здійснюється в такий спосіб: якщо e непарне, то RR = IR;

якщо e парне і символ Якобі $J\ \frac{IR}{n} = 1$, то RR = IR,

якщо e парне і символ Якобі $J\left(\frac{IR}{n}\right) = -1$, то RR = IR / 2.

Варто визначити, що: $J\left(\frac{a}{n}\right) = (a^{(p-1)/2} \bmod p)(a^{(q-1)/2} \bmod q)$, де $n = p \times q$.

Цифровий підпис s обчислюється як:

$$s = \min(RR^d \bmod n, n - (RR^d \bmod n)).$$

Алгоритм генерації ЕЦП Rabin

Схема Рабіна аналогічна схемі RSA, але як відкритий ключ e використовуються тільки парні числа і найчастіше e = 2.

Процес зняття ЕЦП за ISO/IEC 9796-1.

Зняття цифрового підпису здійснюється в три етапи (див. рис. 3.7).

1) відкриття підпису;

2) відновлення повідомлення;

3) контроль надмірності.

На кожному з цих трьох етапів підпис може бути відкинутий як недійсний, якщо він не пройде відповідної перевірки. У таких випадках обробка підпису припиняється з відповідним повідомленням при цьому зацікавлених сторін.

Необхідно розглянути більш детально кожний з цих етапів.

1. *Відкриття підпису.* На даному етапі здійснюється перетворення цифрового підпису s у рядок IR' – відновлене проміжне ціле:

$$IR' = VER_{k_v}(s).$$

Для схеми RSA k_v = e; для схеми Rabin k_v = 2. Підпис відкидається, якщо рядок IR’ не є *s*-бітним рядком зі старшим значущим бітом, рівним "1" і молодшими значущими бітами, рівними "0110". У випадку правильного ЕЦП справедлива рівність IR = IR’.

2. *Відновлення повідомлення.* У ході відновлення повідомлення з рядка IR’ формується рядок надлишкового повідомлення MR’, що складається з *2t* байтів. Відновлення здійснюється в такий спосіб:

а) нехай X буде рядком з s – 1 молодших значущих бітів рядка IR’;

б) молодший значущий байт рядка X, що представляється як $u_1 \| u_3 \| u_1 \| 0110$, де u_i – напівбайт, заміняється відповідно до виразу $\pi^{-1}(u_4) \| u_1$;

в) відновлене повідомлення MR’ формується шляхом доповнення ліворуч рядка X нульовими бітами (від 0 до 15 бітів), у результаті чого загальна довжина MR’ складе 2t байтів.

Якщо цифровий підпис правильний, то значення MR’ буде збігатися із значенням рядка MR за винятком 16t – s + 1 старших значущих бітів.

Далі здійснюється обчислення значення r і z:

а) здійснюється порівняння парних і непарних байтів відновленого надлишкового повідомлення MR’. Для усіх $i = \overline{1,t}$ перевіряють виконання умови $MR'_{2i} \oplus F(MR'_{2i-1}) = 0$.

Якщо для всіх $i = \overline{1,t}$ ця умова справедлива, то цифровий підпис вважається недійсним;

б) нехай *z* – найменше позитивне ціле число *i*, при якому значення $f = MR'_{2i} \oplus F(MR'_{2i-1}) \neq 0$;

в) нехай r буде найменшим значущим напівбайтом числа f. Підпис вважається недійсним, якщо шістнадцятирічне значення r лежить поза інтервалом $1 \le r \le 8$.

Відновлене доповнене повідомлення MP’ довжиною z байтів формується з MR’ відповідно до виразів:

а) $MR'_{2i} = MR'_{2i-1}$ для всіх $i = \overline{1,z}$;

б) якщо r – 1 старших значущих бітів рядка MP’ не дорівнюють нулю, то цифровий підпис вважається недійсним.

Оригінальне повідомлення M’ є 8z – r + 1 молодших значущих бітів рядка MP’.

3. *Контроль надмірності.* Цифровий підпис верифікується в такий спосіб:

а) з отриманого на попередньому етапі повідомлення M’ формується рядок MR’’ шляхом застосування операцій доповнення, розширення повідомлення і введення надмірності (аналогічно до процесу накладання цифрового підпису);

б) цифровий підпис вважається дійсним тільки тоді, коли s – 1 молодших значущих бітів рядка MR’’ збігаються з s – 1 молодшими значущими бітами рядка MR'.

Розглянута схема цифрового підпису може бути модифікована в схему ЕЦП з додаванням і, отже, буде застосовна для підпису повідом-

лення довільної довжини. У цьому випадку можливе застосування однобічної вільної від колізій геш-функції.

Процедура накладання цифрового підпису полягає в тому, що повідомлення довільної довжини m спочатку гешується, а потім геш-код h(m) є входом у процедуру накладання підпису за ISO/IEC 9796-1.

Цифровий підпис з частковим відновленням повідомлення

Схема ЕЦП із частковим відновленням повідомлення визначена в ISO/IEC 9796-2 і має такі властивості.

По-перше, схема забезпечує часткове відновлення повідомлення і застосована для повідомлень довільної довжини. Тобто якщо повідомлення коротке, то воно може бути цілком відновлене з підпису. Якщо повідомлення занадто довге, тобто коли довжина повідомлення d не погодиться з довжиною підпису s, то частина повідомлення може бути відновлена з підпису, а інша частина повинна бути передана одержувачу будь-яким іншим способом.

По-друге, в раніше розглянутій схемі підпису для введення надмірності необхідно відводити не менше половини доступного простору блоку підпису. У ISO/IEC 9796-2 метод уведення надмірності допускає використання геш-функції, що дозволяє збільшити обсяг даних у блоці підпису. Так, якщо за алгоритм генерації підпису взяти 768-бітне перетворення RSA то, відповідно до процедури формування підпису за ISO/IEC 9796-1, безпосередньо для переданих даних буде доступно не більше 384 бітів. При застосуванні схеми, визначеної в другій частині стандарту, дозволяється використовувати для даних до 600 бітів.

Основна ідея схеми ЕЦП із частковим відновленням повідомлення полягає в тому, що:

1. Повідомлення m гешується з використанням геш-функції, у результаті чого формується геш-код h(m).

2. Якщо повідомлення занадто довге, щоб бути цілком включеним у підпис, то виділяється деяка його частина, що надалі буде відновлюватися з підпису.

3. До відновлюваної частини повідомлення додається прапорний біт (more date bit).

4. Відновлювана частина повідомлення, прапорний біт і геш-код разом з іншими бітами формату (наприклад, ідентифікатор геш-функції)

конкатуються і є вхідними параметрами в алгоритм генерації підпису. Як алгоритм генерації підпису рекомендується використовувати схему RSA.

Схема цифрового підпису Нуберга – Рюппела

Четверта частина стандарту ISO/IEC 9796-4 визначає схему Нуберга – Рюппела (Nyberg – Rueppel, NR-схема), у якій як алгоритм генерації підпису використовується варіант схеми підпису Ель Гамаля. У даний момент запропоновані дві версії NR-схеми підпису: на основі обчислення дискретного логарифма в Z_p^* і на основі обчислення логарифма на еліптичних кривих.

NR-схема є рандомізованою схемою з відновленням повідомлення. Для даної схеми простір, що підписується, $M_s = Z_p^*$, де p – просте число. Простір підписів $S = Z_p \times Z_q$, де q – просте число і q ділить (p – 1). Функція введення надмірності відображає множину повідомлень M у множину M_s. Алгоритм генерації ключів аналогічний алгоритму генерації ключів для схеми DSA, за винятком того, що на довжину чисел p і q не накладається ніяких обмежень.

Генерація ключів для NR-схеми підпису.

Для формування відкритого і відповідного закритого ключа сторона *A* повинна виконати такі дії:

1. Вибрати прості числа p і q такі, де q ділить (p – 1).

2. Визначити генератор Θ циклічної групи порядку q в Z_p^*. Для цього:

2.1. Вибрати елемент $g \in Z_p^*$ і обчислити:

$$\Theta = g^{(p-1)/q} \bmod p$$

2.2. Якщо $\Theta = 1$, повернутися в 2.1.

3. Вибрати випадкове ціле число a таке, що $1 \le a \le q - 1$.

4. Обчислити число $y = \Theta^a \bmod p$.

5. Відкритим ключем сторони A є вектор (p, q, Θ, y), закритим ключем є число a.

Процедура накладання цифрового підпису.

Нехай сторона *A* бажає підписати повідомлення m ∈ M. Тоді необхідно:

1. Обчислити $\bar{m} = R(m)$ (уведення надмірності).

2. Вибрати випадкове секретне ціле число k, $1 \le k \le q - 1$ і обчислити параметр:

$$r = \Theta^{-k} \bmod p.$$

3. Обчислити:

$$e = \tilde{m} r \bmod p;$$
$$s = ae + k \bmod q.$$

4. Цифровим підписом повідомлення m є пари чисел (e, s).

Процедура зняття цифрового підпису.

Для того щоб верифікувати ЕЦП і відновити повідомлення, об'єкт B повинен:

1. Одержати автентичну копію відкритого ключа (p, q,Θ, y) сторони A.

2. Перевірити, що 0 < e < p. Якщо умова не виконується, то підпис відкидається.

3. Перевірити, що 0 < s < q. Якщо умова не виконується, то підпис відкидається.

4. Обчислити:

$$v = \Theta^{s} y^{-e} \bmod p;$$
$$\tilde{m} = ve \bmod p.$$

5. Перевірити, що $\tilde{m} \in M_R$. Якщо $\tilde{m} \notin M_R$, то підпис відкидається.

6. Відновити повідомлення $m = R^{-1}(\tilde{m})$.

Коректність ЕЦП випливає зі справедливості такого ствердження. Якщо абонент A сформував підпис, то:

$$v \equiv \Theta^{s} y^{-e} \equiv \Theta^{s-ae} \equiv \Theta^{k} (\bmod p).$$

У такий спосіб $ve \equiv \Theta^{k} \tilde{m} \Theta^{-k} \equiv \tilde{m} (\bmod p)$, що і потрібно було довести.

Стійкість *NR*-схеми ґрунтується на складності розв'язання задачі дискретного логарифма. Зловмисник може спробувати підробити підпис

шляхом вибору числа k і обчислення параметрів r. Така атака рівноцінна тривіальному випадковому підбору підпису s. Імовірність успіху такої атаки складає 1/p, що при великих значеннях p складає досить малу величину.

Для запобігання визначення особистого ключа необхідно на кожне повідомлення обирати різне k. Ці числа повинні формуватися з урахуванням спеціальних вимог, мати рівномірний закон розподілу і бути непередбачуваними. Для формування числа k можна використовувати генератор, визначений у ANSI X 9.17. Оскільки *NR*-схема підпису забезпечує відновлення повідомлення, то пред'являються тверді вимоги до функції введення надмірності R(m). При виборі функції надмірності R необхідно враховувати мультиплікативну природу схеми підпису. Припустимо, що $m \in M$, $\tilde{m} = R(m)$ і (e, s) – цифровий підпис для m. Тоді $e = \tilde{m}\Theta^{-k} \bmod p$ для деякого цілого k і $s = ae + k \bmod q$. Нехай $\tilde{m}^* = \tilde{m}\Theta^{e} \bmod p$ для деякого e. Якщо $s^* = s + e \bmod q$ і $\tilde{m} \in M_R$, то (e, s^*) є коректним підписом для нового повідомлення $m^* = R^{-1}(\tilde{m}^*)$. Це випливає з того, що:

$$v \equiv \Theta^{s} y^{-e} \equiv \Theta^{s+e}\Theta^{-ae} \equiv \Theta^{k+e} (\bmod p);$$

$$ve \equiv \Theta^{k+e}\tilde{m}\Theta^{-k} \equiv \tilde{m}\Theta^{e} \equiv \tilde{m}^* (\bmod p).$$

Оскільки $\tilde{m} \in M_R$, то підроблений підпис (e, s^*) буде прийнятий як коректний для повідомлення m^*.

Нарешті, перевірка умови 0 < e < p є обов'язковою. Припустимо (e, s) є підписом сторони A для повідомлення m. Тоді $e = \tilde{m}r \bmod p$ і $s = ae + k \bmod q$. Зловмисник може використовувати цей підпис для формування підпису під обраним ним самим повідомленням m^*. Він визначає e^* таке, що $e^* \equiv \tilde{m}^* r (\bmod p)$ і $e^* \equiv e (\bmod q)$. Якщо не перевірити умову, що $0 < e^* < p$, то пара (e, s^*) буде прийнята як справжній підпис.

Розглянемо основні стандарти ЕЦП, застосовані у комплексних системах захисту банківської інформації, проведемо зіставлення основних характеристик ЕЦП (довжину ключів, довжину цифрового підпису, складність (час) обчислення і складність (час) перевірки дійсності цифрового підпису) з метою виявлення їх переваг і недоліків за умови, що рівень стійкості підпису стосовно будь-яких методів фальсифікації не нижче, ніж 10^{21} (або 30 років безперервної роботи мережі з 1000 суперкомп'ютерів).

У якості "базової" довжини ключів і довжини самого цифрового підпису будемо розглядати довжину в 64 байти.

Алгоритм RSA ґрунтується на NP-повній задачі факторизації числа (розкладання цілого параметра n у вигляді добутку двох різних простих чисел приблизно рівних один по одному величини, тобто n = p × q на ці прості множники). За сучасними оцінками складність задачі розкладання на прості множники при цілих числах n з 64 байтів становить порядок 10^{17} – 10^{18} операцій, тобто перебуває десь на грані досяжності для серйозного "зловмисника". Тому звичайно в системах цифрового підпису на основі алгоритму RSA застосовують більш довгі цілі числа n (звичайно від 75 до 128 байтів). Це відповідно приводить до збільшення довжини самого цифрового підпису відносно 64-байтного варіанта приблизно на 20 – 100 % (у цьому випадку її довжина збігається з довжиною запису числа n), а також від 70 до 800 % збільшує час обчислень при підписуванні і перевірці.

Крім того, при генерації й обчисленні ключів у системі RSA необхідно перевіряти велику кількість досить складних додаткових умов на прості числа p і q, а невиконання кожного з них може уможливити фальсифікацію підпису з боку того, хто виявить невиконання хоча б однієї із цих умов (при підписуванні важливих документів допускати, навіть теоретично, таку можливість небажано). Схему створення і перевірки підпису за алгоритмом RSA наведено на рис. 2.6.

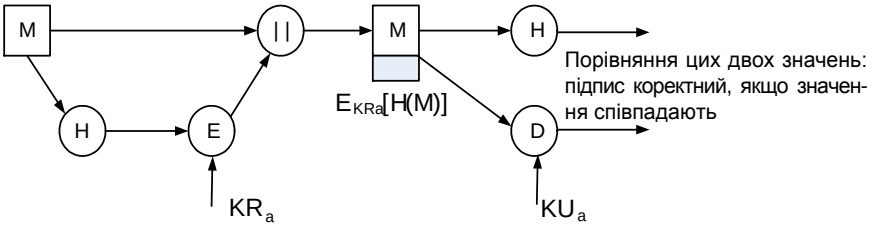


Рис. 2.6. **Створення і перевірка підпису за алгоритмом RSA**

Алгоритм НОТАРІУС-1

Алгоритм НОТАРІУС-1 [98] – аналог алгоритму Ель Гамаля. Основна відмінність алгоритму полягає в тому, що замість звичайної операції множення цілих чисел за модулем великого простого p використовується множення за модулем простого числа, ця операція набагато ефективніше обчислюється на розповсюджених процесорах.

Процедури підписування електронних документів і перевірки цифрових підписів за алгоритмом НОТАРІУС-1 виглядають аналогічно відповідним процедурам алгоритму Ель Гамаля та забезпечують той же рівень стійкості підпису, але виконуються швидше. Алгоритм НОТАРІУС-S дозволив при збереженні стійкості підпису скоротити її довжину ще на

32,5 %. Для базового варіанта з ключами з 64 байтів довжина підпису скоротилася відносно DSA і НОТАРІУСА-D з 40 байтів до 27 байтів. Відповідно зменшився час обчислення і перевірки підпису. Стійкість залишилася на тому ж рівні – 10^{21} [46].

Алгоритм EGSA

Алгоритм EGSA [91]. Істотним кроком уперед у розробці сучасних алгоритмів цифрового підпису був новий алгоритм Ель Гамаля. У цьому алгоритмі ціле число n покладається рівним спеціально обраному великому простому числу p, за модулем якого і виробляються всі обчислення.

Такий вибір дозволяє підвищити стійкість підпису при ключах з 64 байтів приблизно в 1000 разів, тобто при такій довжині ключів забезпечується необхідний нам рівень стійкості порядку 10^{21}.

При цьому довжина самого цифрового підпису збільшується у два рази і становить 128 байтів. Головна "заслуга" алгоритму Ель Гамаля полягала в тому, що надалі він послужив основою для прийняття декількох стандартів цифрового підпису, у тому числі національного стандарту США DSS і державного стандарту РФ ГОСТ Р-34.10-2001 [9].

Алгоритм ЕЦП ГОСТ34.10–2001

Алгоритм обчислення і перевірки підпису в ГОСТ 34.10 улаштований аналогічно алгоритму DSA, але попередня обробка електронних документів перед підписуванням (так зване гешування) виконується інакше, повільніше [9].

В основу протоколу електронному цифрового підпису ГОСТ Р 34.10 покладено протокол Ель-Гамаля. Цей протокол має обчислювальні морфізми, що дозволяють на підставі одного підписаного повідомлення створювати довільне число формально правильних пар повідомлення-підпис, тому підпис обчислюється для значення геш-функції від повідомлення.

Нехай m – повідомлення, що підписується, h – геш-функція за ГОСТ Р 34.11–94, $\{E(F_p),Q,P\}$ – відкритий ключ, число l, – секретний ключ, при цьому P = lq. Для формування підпису виконуються такі дії:

1) обчисляється геш-функція $e \equiv h(m)(\mathrm{mod} r)$, причому $e \neq 0$;

2) виробляється випадковий показник k, 0 < k < r, и обчислюється крапка $R = (X_R, Y_R) = kQ$, причому $X_R \neq 0(\mathrm{mod} r)$;

3) обчислюється частина підпису $s \equiv (lx_R + ke)(\mathrm{mod} r)$, причому $s \neq 0$.

Підписане повідомлення становить трійку $(m, x_R(\mathrm{mod} r), s)$.

Для перевірки підпису виконуються такі дії:

1) перевіряються умови $0 < x_R < (\mathrm{mod}\, r)$ і $0 < s < r$;

2) обчислюється $e \equiv h(m)(\mathrm{mod}\, r)$ і якщо e = 0, то вважають e = 1;

3) обчислюється точка $R' = (se^{-1}(\mathrm{mod}\, r))Q - (x_R e^{-1}(\mathrm{mod}\, r))Q$;

4) перевіряється порівняння $x_{R'} \equiv x_R (\mathrm{mod}\, r)$. Якщо порівняння виконується, то підпис правільний.

Безпека протоколу підпису безпосередньо залежить від складності поводження геш-функції й складності обчислення колізій геш-функції, тому що в першому випадку можна обчислити повідомлення для наявного підпису, а в другому – заготовити пари повідомлень із однаковими значеннями e, підписати одне з них, а потім замінити одне повідомлення іншим. Складність обчислення колізій геш-функції методом Полларда рівна. Однак складність обчислення колізій крокової функції гешування, за допомогою якої обчислюється h, рівна es.

Чим менше складністна неоднорідність криптосистеми, тем потенційно більш стійкою є ця криптосистема (додавання нового завдання не може підвищити стійкість, а може її тільки знизити). Тому при проектуванні криптосистем слід дотримуватися принципу максимальної складністої однорідності: кількість завдань, покладених в основу безпеки криптосистеми, таких, що розв'язок кожної з них порушує безпеку, повинно бути мінімальним. Оптимальною є криптосистема зі складністої неоднорідністю 1 (її безпека заснована на єдином завданні).

Безпека електронного цифрового підпису базується на таких припущеннях [30]:

1) завдання дискретного логарифмування на еліптичній кривій є складною (розв'язок цієї задачі приводить до розкриття ключа);

2) ентропія генератора випадкових чисел не нижче ентропії генератора ключів (а якщо ні, то стійкість підпису буде знижена порівняно з розрахунковою оцінкою);

3) імовірність повтору двох випадкових чисел протягом терміну дії ключа зневажливо мала (а якщо ні, то можна розраховувати на розкриття секретного ключа з низькою складністю);

4) геш-функція є обчислювально незворотньою (а якщо ні, то можна підмінити підписане повідомлення іншим);

5) колізії геш-функції є важко обчислювальними (а якщо ні, то можна заготовити пари повідомлень, що складають колізію, і після підпису замінити підписане повідомлення іншим).

Ці припущення визначають список задач, покладених в основу безпеки, неоднорідність стандартного протоколу підпису дорівнює 5. Можна вважати, що стійкість підпису не перевищує мінімуму складностей дискретного логарифмування, знаходження використовуваного випадкового числа, поводження геш-функції, обчислення колізій геш-функції.

Протокол підпису може бути легко посилений, причому це посилення є строгим, тобто не вносить слабкостей порівняно зі стандартом за інших рівних умов. Посилення полягає в зчепленні повідомлення m і координати $x_R(\mathrm{mod} r)$ з R через геш-функцію, замість $e \equiv h(m)(\mathrm{mod} r)$ слід використовувати $e \equiv h(m \| x_R(\mathrm{mod} r))(\mathrm{mod} r)$.

Це виключить вразливість протоколу, пов'язану із заготівлею колізій (задача п. 5) і викликану цим необхідність періодичної зміни стартового вектора геш-функції. Таке посилення протоколу підпису знизить його неоднорідність до 4, при цьому завдання обігу геш-функції заміняється задачею її обмеженого поводження: частина розрядів аргументу, обумовлена точкою R, повинна мати заданий вигляд. Очевидно, що задача обмеженого поводження не може бути простіше задачі поводження (протилежне припущення негайно одразу призводить до протиріччя).

Алгоритм ЕЦП ДСТУ-4145

Алгоритм електроного цифрового підпису стандарту ДСТУ 4145-2002 [14] заснований на еліптичних кривих (несуперсінгулярні криві) над полем $GF(2^m)$. У стандарті використовуються криві в поліноміальному базисі і криві в оптимальному нормальному базисі (ОНБ). Причому останні (ОНБ) двох різних типів: другого і третього типу. Основою для розробки даного стандарту послужив міжнародний стандарт IEEE P1363a [14]. Тому принциповою відмінністю ДСТУ 4145 є тільки використання "своєї" функції гешування. Для гешування використовується алгоритм – міждержавний стандарт ГОСТ 34.311-95 (раніше ГОСТ 28147-89 у режимі імітовставки).

Даний стандарт встановлює механізм цифрового підпису, заснований на властивостях груп точок еліптичних кривих над полями $GF(2^m)$, і правила застосування цього механізму до повідомлень, які передаються по каналах зв'язку й/або обробляються в комп'ютеризованих системах загального призначення. Застосування даного стандарту гарантує цілісність підписаного повідомлення, автентичність його автора й незаперечність авторства при дотриманні визначених правил використання цифрового підпису, які не є об'єктом даного стандарту.

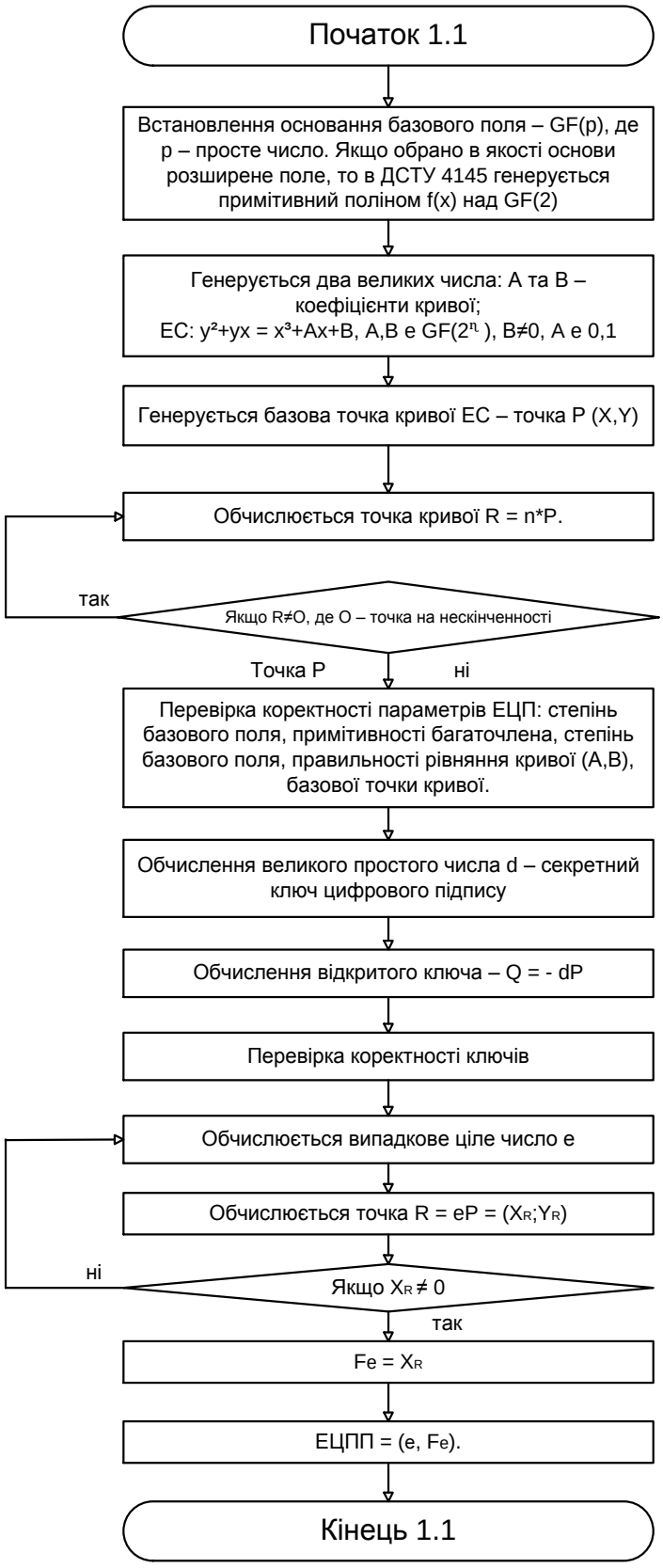
У табл. 2.1 наведено основні позначення, що застосовуються у цьому стандарті.

Таблиця 2.1

Основні позначення ДСТУ-4145

a mod b	ціле число, що дорівнює залишку від ділення цілого числа a на ціле число b;
[.]	ціла частина виразу, що стоїть у дужках;
L()	довжина двійкового представлення цілого числа, що стоїть у дужках;
T	повідомлення, для якого обчислюється цифровий підпис;
LT	довжина повідомлення T;
H	функція гешування, геш-функція;
H(T)	результат обчислення функції гешування за повідомленням T (геш-код);
L_H	довжина результату обчислення функції гешування (геш-коду);
D	цифровий підпис;
LD	довжина цифрового підпису, число LD кратне 16, $L_D \geq 2L(n)$;
GF(2)	просте поле, що складається з двох елементів 0 і 1;
$GF(2^m)$	базове поле, розширення простого поля степені m;
m	степінь базового поля – непарне просте число, $163 \leq m \leq 509$;
0	нульовий елемент базового поля;
1	одиничний елемент базового поля;
X^{-1}	зворотний елемент для елемента базового поля x;
tr(x)	слід елемента базового поля x;
htr(x)	напівслід елемента базового поля x;
f(t)	примітивний тричлен або пятичлен степені m, задаючий поліноміальний базис базового поля;
(A,B)	коефіцієнти рівняння еліптичної кривої, $A \in \{0,1\}, B \neq 0$;
O	нескінченно віддалена точка еліптичної кривої;
P	базова точка еліптичної кривої;
$(X_p, У_p)$	координати базової точки еліптичної кривої P;
P	стисле представлення точки P, елемент базового поля;
n	порядок базової точки еліптичної кривої, просте число, $n \geq \min(2^{160}, 4\sqrt{2^m})$;
GF(p)	просте поле, що складається з p елементів, p - просте число
d	секретний ключ цифрового підпису, елемент GF(n);
Q	відкритий ключ цифрового підпису, точка еліптичної кривої;
(Xq,Yq)	координати відкритого ключа цифрового підпису;
Q	стисле представлення точки Q, елемент базового поля;
(e,Fe)	цифровий предпідпис, $e \in GF(n), Fe \in GF(2^m)$;
(a←b)	присвоювання змінній a значення виразу b;
i_H	ідентифікатор геш-функції;
S||R	конкатенація рядків S и R.

На рис. 2.7 та 2.8 наведено алгоритм електронного цифрового підпису стандарту ДСТУ 4145.



Незведений поліном – це поліном, який ділиться тільки на себе та на одиницю. Примітивним поліномом називається поліном f(x), корінь якого є примітивним елементом поля

Початок генерації базових параметрів ЕЦП

Початок генерації ЕЦП

Цифровий предпідпис

Рис. 2.7. **Частина 1 алгоритму електронного цифрового підпису ДСТУ 4145**

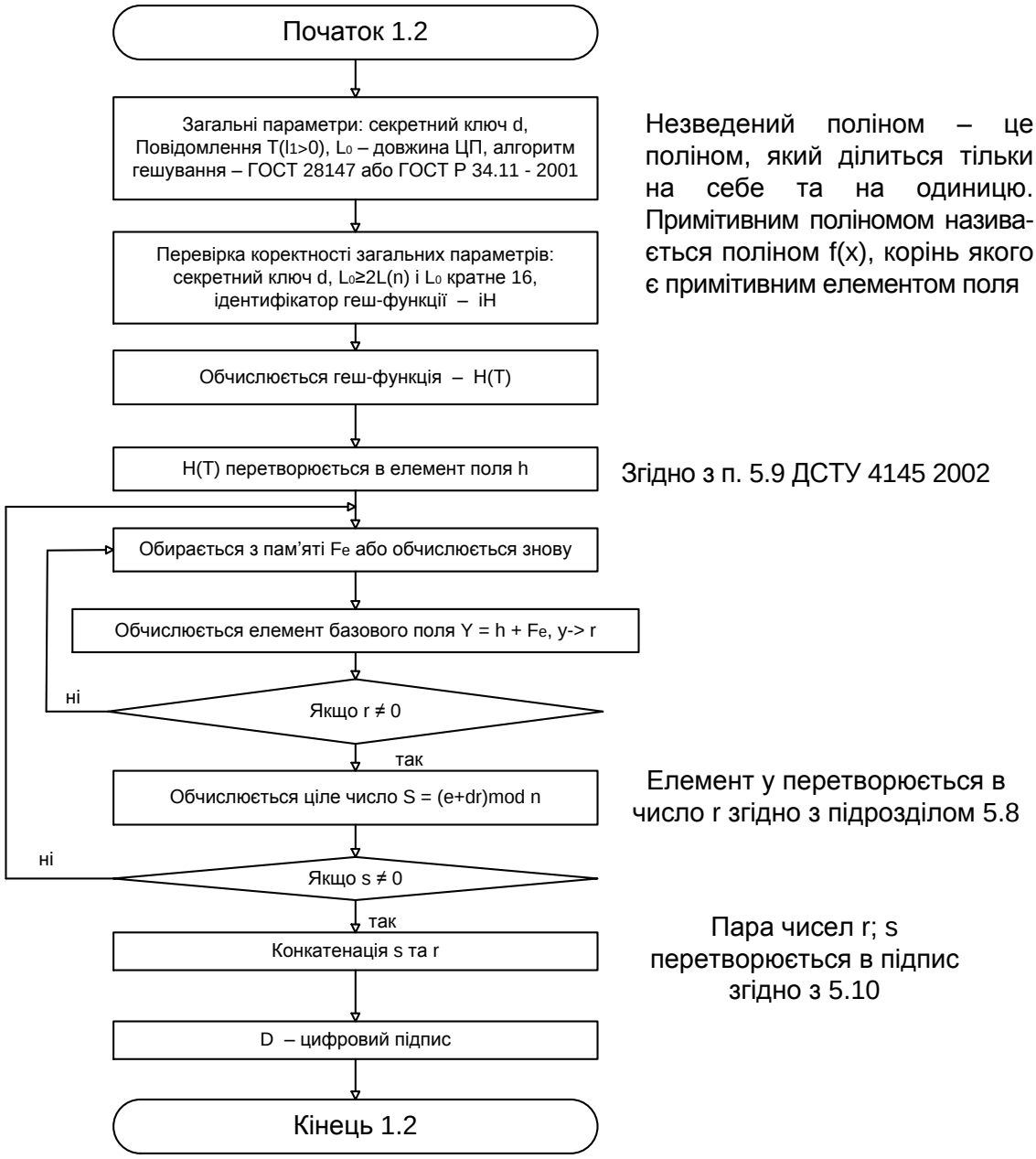


Рис. 2.8. **Частина 2 алгоритму електронного цифрового підпису ДСТУ 4145**

Протоколи колективного цифрового підпису на еліптичних і гіпереліптичних кривих

У якості джерела абелевої групи для протоколу колективного підпису на основі ДСТУ-4145, запропонованого в [14], можна обрати групу дивізорів гіпереліптичної кривої. Основна перевага використання гіпереліптичних кривих полягає в тому, що розмір основного поля, над яким визначена крива, зменшується пропорційно роду кривої без втрати стійкості, хоча сама формула групового додавання виглядає більш громіздко.

Нехай F – кінцеве поле й нехай $\bar{F}$ – алгебраїчне замикання поля F. Гіпереліптична крива 3 роду $g \le 1$ над F становить [65] набір розв'язків $(x,y) \in F \times F$ рівняння:

$$C: y^2 + h(x)y = f(x), \qquad (1)$$

де $h(x) \in F[x]$ – поліном степені не більш $g, f(x) \in F[x]$ – нормований поліном степені 2g + 1 і не існує розв'язків $(x,y) \in \bar{F} \times \bar{F}$, які б одночасно задовольняли рівнянню (1) і рівнянням 2y + h(x) = 0 і h`(x)y – f`(x) = 0. Автори вважають, що нескінченно віддалена точка ∞ також належить кривій.

Згідно із роботою [65], у якості групової структури у випадку гіпереліптичних кривих розглядається якобіан кривої C. Кожний елемент якобіана – це клас еквівалентності дивізорів, який може бути представлений унікально наведеним дивізором у вигляді пари поліномів у формі Мамфорда. На якобіане визначені групові операції додавання й дублювання дивізорів.

Згідно із роботою [51], порядок якобіана гіпереліптичної кривої обмежений інтервалом Хассе – Вейла:

$$\left|\left(\sqrt{q}-1\right)^{2g}\right| \le \#J/F_q \le \left[\left(\sqrt{q}+1\right)^{2g}\right],$$

де q – характеристика поля, над яким визначена крива, g – рід кривої.

Можна вважати, що порядок якобіана:

$$J/F_q \approx q^g.$$

Більшість криптографічних додатків базуються на еліптичних або гіпереліптичних кривих з довжиною ключа не менш 160 біт, тобто з порядком групи не менше 2^{160}. Отже, для криптосистем на гіпереліптичних кривих над полем F_q повинне виконуватися як мінімум:

$$g \times \log_2 q \approx 160.$$

Зокрема, для кривої роду 2 необхідно вибрати основне поле F_q з $|F_q| \approx 2^{80}$, з довжиною операндів 80 біт. Для кривої 3 роду потужність основного поля $|F_q| \approx 2^{54}$ для кривої 4 роду $|F_q| \approx 2^{40}$.

Оскільки елементи підпису належать основному полю, над яким визначена крива, у випадку гіпереліптичних кривих розмір підсумкового колективного підпису зменшується пропорційно роду кривої. Так, при використанні гіпереліптичної кривій другого роду розмір колективного підпису виявиться приблизно у два рази менше, чим при використанні еліптичної кривої, що володіє аналогічним рівнем криптостійкості. Відповідно, при використанні кривої третього, роду розмір колективного підпису зменшиться приблизно в три рази і т. д.

У криптографічних цілях використовуються гіпереліптичні криві роду 2 і 3. Криві більш високого роду не є стійкими.

Протокол колективного підпису на основі стандарту ДСТУ 4145-2002 на гіпереліптичних кривих [14].

Слід ввести позначення:

D – базовий дивізор гіпереліптичної кривої;

l – кількість користувачів;

n – порядок циклічної підгрупи якобіана гіпереліптичної кривої;

d_i – секретний ключ і-го користувача;

h – геш-образ повідомлення.

$\Psi(R)$– функція перетворення дивізора в елемент основного поля. Авторами пропонується таке перетворення: коефіцієнти першого полінома дивізора R представлені у вигляді числа в системі числення з основою, що дорівнює модулю основного поля, над яким визначена крива (у випадку простого поля). А потім переводиться це представлення в десяткову систему числення.

Генерація відкритого колективного ключа.

Кожний і-й користувач (i = 1…l) формує відкритий ключ виду:

$$Q_i = -d_i D.$$

Колективний відкритий ключ обчислюється як сума відкритих ключів групи з l користувачів:

$$Q = \sum_{i-1}^{l} Q_i = \sum_{i-1}^{l} d_i D.$$

Формування колективного підпису.

1. Кожний користувач розраховує дивізор R_i у такий спосіб:

а) вибирає випадковий параметр k_i, $1 < k < n$;

б) обчислює $R_i = k_i D$.

2. По представленим користувачами дивізорам R_i обчислюється загальний дивізор:

$$R = \sum_{i=1}^{l} R_i$$

3. Обчислюється значення функції $\Psi(R)$.

4. Перша частина підпису визначається формулою:

$$r = h\Psi(R) \bmod n.$$

5. Кожний користувач обчислює свій параметр s_i:

$$s_i = (k_i + d_i r) \bmod n.$$

5. Друга частина підпису визначається формулою:

$$s = \sum_{i=1}^{l} s_i \bmod n$$

Колективним підписом є пара чисел – (r,s).

Перевірка колективного підпису.

Перевіряючий обчислює геш-образ h` загального повідомлення.

Використовуючи відкритий колективний ключ Q, обчислює дивізор R` = sD + rQ, R` і значення $v = h`\Psi(R`)$. Якщо v = r, то підпис визнається справжнім.

Протокол колективного підпису на основі протоколу ECRP на еліптичних кривих

Протокол електронного цифрового підпису з перед обчисленнями ECRP [65] було запропоновано з метою зменшити трудомісткість операції верифікації підпису в корпоративній мережі за рахунок множення тільки на базову точку, яке можна виконати з передобчисленнями.

Модифікуємо цей протокол для реалізації колективного підпису. Слід ввести позначення:

P – базова точка еліптичної кривої;

l – кількість користувачів;

n – порядок циклічної підгрупи точок еліптичної кривої;

d_i – секретний ключ i-го користувача;

h – геш-образ повідомлення;

$\pi(R) = X_R \bmod n$ – виділення x-координати точки R = (X_R, Y_R) еліптичної кривої.

Генерація відкритого колективного ключа.

Кожний i-й користувач (i = 1…l) формує відкритий ключ виду:

$$Q_i = -d_i P.$$

Колективний відкритий ключ обчислюється як сума відкритих ключів групи з l користувачів:

$$Q = \sum_{i=1}^{l} Q_i = \sum_{i=1}^{l} - d_i P.$$

Формування колективного підпису.

Кожний i-й користувач (i = 1…l) розраховує точку R_i таким чином:

а) вибирає випадковий параметр k_i, $1 < k_i < n$;

б) обчислює значення $t_i = \frac{k_i}{n} \bmod n$;

в) і точку $R_i = t_i P$.

За представленими користувачами точками R_i обчислюється загальна точка $R = \sum_{i=1}^{l} R_i = (X_R, Y_R)$ і значення $\omega = \pi(R) = X_R \bmod n$.

Формується точка $\omega R = (x, y)$ і перша частина колективного підпису $r = \pi(x, y) = x \bmod n$.

Кожний користувач обчислює свій параметр s_i:

$$s_i = (\omega k_i + h d_i) \bmod n$$

і надає його для обчислення другої частини колективного підпису $s = \sum_{i=1}^{l} s_i$. Колективним підписом є пара чисел – (r,s).

Перевірка колективного підпису.

Перевіряючий обчислює геш-образ h` $h'$ загального повідомленняі значення $t = \frac{s}{h`} \bmod n$.

Використовуючи відкритий колективний ключ Q формує точку tP + + Q = (x,y) і обчислює значення $v = \pi(x, y) = x \bmod n$.

Якщо v = r, то підпис визнається справжнім.

Обґрунтування коректності представленого протоколу.

Оскільки при формуванні підпису значення r визначається формулою $r = \pi(\omega R) = \pi\left(\omega \sum_{i-1}^{l} \frac{k_i}{h} P\right)$, а при перевірці підпису перевіроч-ний вираз tP + Q дає точку ωR:

$$tP + Q = \frac{s}{h} P - \sum_{i=1}^{l} d_i P = \frac{\sum_{i=1}^{l} (wk_i + hd_i)}{h} P - \sum_{i=1}^{l} d_i P = \omega \sum_{i-1}^{l} \frac{k_i}{h} P + \sum_{i=1}^{l} d_i P - \sum_{i=1}^{l} d_i P =$$

$$= \omega \sum_{i-1}^{l} \frac{k_i}{h} P = \omega R$$

У підсумку маємо: $v = \pi(tP + Q) = \omega R$, що відповідає r при форму-ванні підпису.

Протокол колективного підпису на основі протоколу ECPR на гіпереліптичних кривих

У якості джерела абелевої групи для запропонованого протоколу можна також обрати групу дивізорів гіпереліптичної кривої, як це було зроблено вище для ДСТУ-4145. У цьому випадку базовій точці, відкритим ключам користувачів, і проміжній точці R відповідають дивізори гіпереліптичної кривої [51].

Генерація відкритого колективного ключа.

Кожний і-й користувач (i = 1…l), формує відкритий ключ виду:

$$Q_i = -d_i D.$$

Колективний відкритий ключ обчислюється як сума відкритих ключів групи з l користувачів:

$$Q = \sum_{i-1}^{l} Q_i = \sum_{i-1}^{l} d_i D.$$

Формування колективного підпису.

Кожний і-й користувач (i = 1…I) розраховує дивізор R_i таким чином:

1) вибирає випадковий параметр:

$$k_i,\ 1 < k < n;$$

2) обчислює значення:

$$t_i = \frac{k_i}{n} \bmod n;$$

3) $R_i = t_i P$;

4) за представленими користувачами дивізорами R_i обчислюється загальний дивізор:

$$R = \sum_{i-1}^{I} R_i$$

і значення $\omega = \Psi(R)$;

5) формується дивізор ωR іперша частина колективного підпису:

$$r = \Psi(\omega R).$$

Кожний користувач обчислює свій параметр s_i:

$$s_i = (\omega k_i + h d_i) \bmod n$$

і надає його для обчислення другої частини колективного підпису $s = \sum_{i=1}^{I} s_i$. Колективним підписом є пара чисел – (r,s).

Перевірка колективного підпису.

Перевіряючий обчислює геш-образ h` загального повідомлення і значення $t = \frac{s}{h`} \bmod n$.

Використовуючи відкритий колективний ключ Q, формує дивізор tD + + Q = (x,y) і обчислює значення $v = \Psi(tD + Q)$. Якщо v = r, то підпис визнається справжнім.

Алгоритм ECDSA

ECDSA (Elliptic Curve Digital Signature Algorithm) – алгоритм із відкритим ключем для створення цифрового підпису, аналогічний по своїй побудові до DSA, але визначений, на відміну від нього, не над полем цілих чисел, а в групі точок еліптичної кривої [102].

Особливості.

Стійкість алгоритму шифрування ґрунтується на проблемі дискретного логарифма в групі точок еліптичної кривої. На відміну від проблеми простого дискретного логарифма й проблеми факторизації цілого числа, не існує субекспонеціального алгоритму для проблеми дискретного логарифма в групі точок еліптичної кривої. Із цієї причини "сила на один біт ключа" істотніше вище в алгоритмі, який використовує еліптичні криві. Д. Брауном (Daniel R. L. Brown) було доведено, що алгоритм ECDSA не є більш безпечним, ніж DSA. Ним було сформульоване обмеження безпеки для ECDSA, яке привело до такого висновку: "Якщо група еліптичної кривої може бути змодельована основною групою, та її геш-функція задовольняє певному обґрунтованому припущенню, то ECDSA стійка до chosen-message атаки з існуючою фальсифікацією". Алгоритм ECDSA в 1999 р. був прийнятий, як стандарт ANSI, в 2000 р. – як стандарт IEEE і NIST [51]. Також у 1998 р. алгоритм був прийнятий стандартом ISO. Незважаючи на те, що стандарти ЕЦП створені зовсім недавно й перебувають на етапі вдосконалювання, одним найбільш перспективних з них на сьогоднішній день є ANSI X9.62 ECDSA від 1999 – DSA для еліптичних кривих [51].

Вибір параметрів.

Для підписування повідомлень необхідна пара ключів – відкритий і закритий. При цьому закритий ключ повинен бути відомий тільки тому, хто підписує повідомлення, а відкритий – будь-якому бажаючому перевірити дійсність повідомлення. Також загальнодоступними є параметри самого алгоритму.

Параметри алгоритму.

1. Вибір геш-функції H(x). Для використання алгоритму необхідно, щоб повідомлення, що підписується, було числом. Геш-функція повинна перетворити будь-яке повідомлення в послідовність біт, яке можна потім перетворити в число.

2. Вибір великого простого числа q – порядок однієї із циклічних підгруп групи точок еліптичної кривій. Якщо розмірність цього числа в бітах менше розмірності в бітах значень геш-функції H(x) то використовуються тільки ліві біти значення геш-функції.

3. Простим числом *p* позначається характеристика поля координат F_p.

Генерування ключів ECDSA.

Для простоти розглядаються еліптичні криві над полем F_p, де F_p – кінцеве просте поле. Причому, якщо необхідно, конструкцію можна легко адаптувати для еліптичних кривих над іншим полем. Нехай E – еліптична крива, певна над F_p, і P – точка простого порядку q кривої $E(F_p)$. Крива E і точка P є системними параметрами. Число p – просте. Кожна користувачка Аліса конструює свій ключ за допомогою таких дій:

1. Вибирає випадкове або псевдовипадкове ціле число x з інтервалу [1, q − 1].
2. Обчислює добуток Q = x × P.

Відкритим ключем користувача A є число Q, а закритим – x. Замість використання E і P у якості глобальних системних параметрів, можна фіксувати тільки поле F_p для всіх користувачів і дозволити кожному користувачеві вибирати свою власну еліптичну криву E і точку $P \in E(F_p)$. У цьому випадку певне рівняння кривої E, координати точки P, а також порядок q цієї точки P повинні бути включені у відкритий ключ користувача. Якщо поле F_p фіксоване, то апаратна й програмна складові можуть бути побудовані так, щоб оптимізувати обчислення в тому полі. У той же час є величезна кількість варіантів вибору еліптичної кривої над полем F_p.

Обчислення цифрового підпису.

Для того, що підписати яке-небудь повідомлення, для якого підрахований значення h геш-функції H, користувач A повинен зробити таке:

1. Вибрати випадкове ціле число k в інтервалі [1, q − 1].
2. Обчислити $k \times P = (x_1, y_1)$ й припустити в $r = x_1 \pmod q$, де r отримується із цілого числа x_1 між 0 і (p − 1) зведенням по модулю q.

Зауваження: якщо r = 0, то рівняння підпису $s = k^{-1}(h + xr) \pmod q$ не залежить від секретного ключа a, і отже, (r, s) не підходить у якості цифрового підпису. Виходить, у випадку r = 0 необхідно повернутися до кроку 1.

3. Обчислити $k^{-1} \pmod q$ й припустити $s = k^{-1}(h + xr) \pmod q$.

Зауваження: якщо s = 0, то значення $s^{-1} \pmod q$, потрібне для перевірки, не існує.

Виходить, у випадку s = 0 необхідно повернутися до кроку 1. Підписом для повідомлення є пара цілих чисел (r, s).

Перевірка цифрового підпису.

Для того, щоб перевірити підпис користувача A (r, s) на повідомлення, користувач B повинен зробити таке:

1) одержати підтверджену копію відкритого ключа Q користувача A;

2) перевірити, що числа r і s є цілими числами з інтервалу [1, q − 1], і обчислити значення геш-функції h від повідомлення;

3) обчислити $u_1 = s^{-1}h(\mathrm{mod}\ q)$і$u_2 = s^{-1}r(\mathrm{mod}\ q)$;

4) обчислити $u_1P + u_2Q = (x_0,y_0)$, відносно x_0, як цілого числа між 0 і (p − 1), покласти $v = x_0(\mathrm{mod}\ q)$;

5) прийняти підпис, якщо і лише якщо v = r;

6) прийняти підпис, якщо й тільки якщо v = r.

Слід визначити, що, якщо користувач A обчислив свій підпис правильно, те $u_1P + u_2Q = (u_1 + xu_2)\ P = kP$, тому що $k = s^{-1}(h + xr)(\mathrm{mod}\ q)$, і тому v = r.

ECDSA відповідно до стандарту ANSI X9.62

Для практичного застосування алгоритму ECDSA накладають обмеження на поля, у яких визначені еліптичні криві. Більш того, для запобігання деяких відомих атак, обмеження накладаються й на рівняння, що задають еліптичні криві, і на порядок базових точок [51]. Для простоти в даному розділі розглядаються тільки кінцеві F_p.

Вимоги до еліптичної кривої.

Для того, щоб уникнути відомих атак, заснованих на проблемі дискретного логарифма в групі точок еліптичної кривої, необхідно, щоб число точок еліптичної кривої E ділилося на досить велике просте число n. Стандарт ANSI X9.62 вимагає $n > 2^{160}$. Рівняння еліптичної кривої складається специфічним чином, використовуючи випадкові або псевдовипадкові коефіцієнти.

Головними параметрами при побудові еліптичної кривої є:

1) розмірність поля p, де p є непарним простим числом;

2) два елементи поля F_p – a і b, визначені рівнянням еліптичної кривої E, де E має вигляд:

$$y^2 = x^3 + ax + b,$$

де $a,b \in F_p$, і $4a^3 + 27b^2 \neq 0(\mathrm{mod}\, p)$.

3) два елементи поля F_p – xg і yg, які визначають кінцеву точку $G = (x_g, y_g)$ – генератор $E(F_p)$;

4) порядок q точки G, де $q > 2^{160}$ і $q > 4\sqrt{p}$;

5) співмножник $h = \#E(F_p)/q$, де позначення $\#E(F_p)$ означає порядок групи точок еліптичної кривої $E(F_p)$.

Генерація головних параметрів.

Один зі способів генерування криптографічно надійних параметрів полягає в такому:

1. Обираються коефіцієнти a і b специфічним чином, використовуючи в обчисленнях випадкові/псевдовипадкові числа. Нехай E – еліптична крива – $y^2 = x^3 + ax + b$.

2. Обчислюється $N = \#E(F_p)$.

3. Перевіряється, що N має дільник, який є найбільшим простим числом q($q > 2^{160}$ і $q > 4\sqrt{p}$). Якщо ні, то потрібно повернутися на крок 1.

4. Перевіряється, що q не ділить $p_k - 1$ для кожного k, $1 \le k \le 100$. Якщо ні, то потрібно повернутися на крок 1.

5. Перевіряється, що $q \neq p$. Якщо ні, то потрібно повернути на крок 1.

6. Береться випадкова точка $G` \in E(F_p)$ і слід вважати G = (N / q)G'. Потрібно повторювати доти, поки $G \neq 0$.

У 1985 р. Л. Скооф (Schoof) представив алгоритм, що працює за поліноміальний час, для підрахунку $\#E(F_p)$ число точок еліптичної кривої, визначеної над полем F_p (p – непарне просте число). Алгоритм Скоофа є досить неефективним на практиці для значень p, які дійсно становлять інтерес, тобто $p > 2^{160}$. В останні кілька років було зроблено багато роботи з поліпшення й прискорення алгоритму Скоофа, зараз він називається SEA (Schoof-Elkies-Atkin) алгоритм. З таким поліпшенням криптографічно придатні еліптичні криві, визначені над полями, чиї порядки більш, ніж 2^{200}, можуть бути сгенеровані за кілька годин на робочих станціях.

У табл. 2.2. наведені порівняльні характеристики алгоритмів RSA і ECDSA (непарний випадок) при створенні і перевірці цифрових підписів. Обидва алгоритми тестовано на паралельних процесорах Motorola 56303 DSP (66 Мгц). При цьому функція перевірки підпису RSA використовує e= = 65 537.

Таблиця 2.2

Порівняльні характеристики алгоритмів RSA і ECDSA (непарний випадок) при створенні і перевірці цифрового підпису

Алгоритм (довжина ключа, біти)	Час виконання, мс	
	Створення підпису	Перевірка підпису
RSA (1024)	25	< 2
ECDSA (160)	32	33
RSA (2048)	120	5
ECDSA (216)	68	70

Як видно з табл. 2.2, при збільшенні розмірів ключа створення підписів за допомогою ECDSA виробляється значно швидше, ніж у системах RSA. Це розходження ще більшою мірою проявляється для однопроцесорних систем. З іншого боку, перевірка підпису за допомогою ECDSA робиться набагато повільніше, ніж ця ж процедура в системах RSA і знову ж розходження підсилюється для систем з одним процесором. Обробка ECDSA може трішки прискоритися в "парному" випадку. Потужність процесора, витрачена на перевірку підпису ECDSA, може сповільнити виконання інших додатків у системі. У деяких випадках системи RSA (навіть більші ключі, що використовують), можливо, будуть більш прийнятні, ніж криптосистеми на основі еліптичної кривої. Проте криптосистеми на основі еліптичної кривої одержують все більше поширення скоріше як альтернатива, а не заміна систем RSA, оскільки системи ECDLP мають деякі переваги, особливо при використанні в пристроях з малопотужними процесорами та маленькою пам’яттю [9; 51].

2.2. Коди автентифікації повідомлень

Особливе місце серед механізмів забезпечення цілісності і автентичності займають функції гешування: безключові та ключові, що дозволяють забезпечити широкий спектр послуг безпеки інформації згідно з ISO 7498 [115]. Односторонні геш-функції визначені в окремому міжнародному стандарті ISO/IEC 10118 [110; 111; 112].

До ключових геш-функцій відносяться *MAC*-коди.

Визначення автентифікуючих кодів повідомлення згідно Пренилем (Preneel) [78; 79]:

MAC-функція h, що задовольняє таким умовам:

1. Аргумент X може бути довільної довжини й результат $h(K;X)$, має фіксовану довжину n біт, де вторинний вхід K позначає секретний ключ.

2. При наявності даних h і X (але з невідомим K), повинне бути складно визначити $h(K;X)$ з імовірністю успіху значно більшою $1/2^n$.

Навіть при великій кількості відомих пар $\{X_i; h(K; X_i)\}$ складно визначити ключ K або обчислити $h(K; X')$ для будь-якого $X' \neq X_i$.

Більшість MAC становлять повторювані конструкції, у тому розумінні, що вони засновані на функції стиску з фіксованим розміром вхідних значень; вони обробляють кожний блок повідомлень аналогічниим способом. Вхід X становить однозначне заповнення, кратне розміру блоку. Звичайно, це також включає збільшення загальної довжини на бітах вхідних значень. Заповнений вхід потім розділяється на t блоків, що позначають X_1 через X_t. MAC включає функцію стиску f і єднальну змінну H_i між етапом $i-1$ і етапом i:

$$H_0 = IV_K$$
$$H_i = f_K(H_{i-1}, X_i), 1 \le i \le t,$$
$$h(K; X) = g_K(H_t).$$

Тут IV позначає початкове значення й g – вихідне перетворення. Секретний ключ K може бути застосований у IV, у функції стиску, і/або у вихідному перетворенні.

Класифікація атаки.

Залежно від інформації, доступної криптоаналітику, розрізняють такі типи атаки на MAC-коди [79]:

Атака по відомому тексту. Криптоаналітик може вивчити деякі відкриті тексти й відповідні величини MAC.

Атака по обраному тексту. Криптоаналітик може вибрати набір текстів і згодом одержати список величин MAC, відповідних до цих текстів.

Адаптивна атака по обраному тексту. Це найбільш загальний вид атаки, де криптоаналітик вибирає текст і негайно одержує відповідну величину MAC. Вибір тексту може залежати від результату попередніх запитів.

Злам алгоритму MAC може мати різні цілі:

Підробка існуючого тексту. Криптоаналітик може визначити величину MAC принаймні для одного тексту. Доти, поки він не володіє цим текстом, злам може бути випадковим або безглуздим.

Вибіркова підробка. Криптоаналітик може визначити величину MAC для конкретного попередньо обраного ним тексту. Слід пам'ятати, що практична атака часто вимагає підробки, що перевіряється, тобто криптоаналітик знає, що підроблений MAC правильний з імовірністю близькою до 1.

Відновлення ключа. Означає, що криптоаналітик може визначити секретний ключ K. Такий злам є сильнішим, ніж підробка, оскільки він враховує довільні вибіркові підробки.

На практиці, адаптивна обрана текстова атака не завжди може бути реалізована; крім того, підробки звичайно повинні мати специфічну надмірність, для практичного використання. Проте, у загальному випадку краще бути консерватором і вимагати, щоб алгоритми MAC були стійкі проти можливої більш сильної атаки.

Проста атака в алгоритмі MAC полягає у виборі довільного нового повідомлення, і згодом вгадуванні величини MAC [119]. Це може бути зроблено двома шляхами: або шляхом безпосереднього вгадування величини MAC з імовірністю успіху $2-n$, або вгадування ключа з таким обчисленням величини MAC з імовірністю успіху $2-k$. Тут n позначає розмір у бітах величини MAC і k – розмір у бітах секретного ключа. Це атака, що не піддається перевірці: нападаючий не знає заздалегідь про правильність свого припущення. Можливість атаки залежить від кількості спроб, які можуть бути виконані від очікуваного значення успішної атаки; і те, і інше значною мірою залежить від програмного забезпечення.

Повний пошук по ключу.

Це ще один приклад простої атаки, яка може бути застосована до будь-якого алгоритму. Атака вимагає приблизно k/n пари відомих значень текст-MAC для даного ключа; ключ намагаються визначити, перебираючи послідовно всі варіанти. Очікувана кількість спроб 2^{k-1}. На відміну від попередньої атаки, ця атака виконується оффлайн і дає повний контроль над алгоритмом MAC.

Атака, заснована на внутрішній помилці.

Зміст цієї атаки в тому, що якщо можна визначити внутрішню помилку (помилка, яка відбувається перед вихідним перетворенням g), вона може бути використана для створення підробки MAC, що піддається перевірці й заснованого на єдиному обраному тексті.

Преніл (Preneel) і Ван Уоршот (van Oorschot) [79] показали, що внутрішня помилка для h може бути виявлена з використанням u відомих пар текст-MAC і v обраних текстів, де очікувані величини для u і v – визначаються таким чином: $u = \sqrt{2} \times 2^{t/2}$ і $v = 0$, якщо вихідне перетворення g є перестановкою, а якщо ні, то:

$$v \approx 2(2^{t-n} + \left\lfloor \frac{l-n}{n-1} \right\rfloor + 1).$$

Подальші вдосконаення цієї атаки можливі, якщо набір пар текст-MAC має однакову послідовність s кінцевих блоків.

Відновлення ключа, засноване на внутрішній помилці.

Для деяких функцій стиску, можна розширити атаку по внутрішній помилці до атаки відновлення ключа [78]. Ідея в тому, щоб ідентифікувати одну або більш внутрішніх помилок; наприклад, якщо f – не перестановка для встановленого H_l, внутрішня помилка після того, як перший блок повідомлення дасть рівняння $f_k(H_0; X_1) = f_k(H_0; X_1')$, у якому K і можливо H_0 невідомі (допускається, що $H_0 = IV$ залежні від ключа). Для деяких функцій стиску f можна одержати інформацію про секретний ключ, ґрунтуючись на таких співвідношеннях.

Атака "Розділяй і пануй".

Ця атака є особливим випадком відновлення ключа на основі внутрішньої помилки. Для деяких функцій стиску, що використовують два окремих ключа, можна використовувати внутрішні помилки для атаки відновлення ключа "розділяй і пануй" [78]. Нехай ключі позначені K_1, K_2. Загальна ідея така, що криптоаналітик спочатку шукає деякі внутрішні помилки, потім робить повний перебір для ключа K_1, який створює ці помилки. Як тільки K_1 визначений, повний пошук буде використаний для пошуку K_2.

Отже, стійкість такого MAC походить від власних ключів і не пов’язана з їхньою загальною довжиною (хоча атака менш реальна, ніж простий повний ключовий пошук, оскільки вимагає великої кількості відомих пар текст-MAC).

EXOR-підробка.

Цей тип підробки працює тільки якщо величина H_i обчислена як функція $H_{i-1} \oplus X_i$, і якщо не є присутнім ніяке вихідне перетворення. Найлегший варіант атаки вимагає тільки єдиної відомої пари текст-MAC. Нехай вхід X і заповнена версія $\overline{X}$ складаються з єдиного блоку. Припустимо, що відомо $h(K; X)$ із цього випливає, що:

$$h(K, \overline{X} \| (X \oplus h(K, X))) = h(K, X).$$

Мається на увазі, що можна створити підроблене нове повідомлення з тою ж величиною MAC [119].

Канальні атаки.

Канальна атака є основною загрозою для всіх реалізацій шифрувальних алгоритмів. Канальна атака в алгоритмах MAC аналогічна атакам інших симетричних примітивів.

Огляд загальних проектів.

Існує кілька алгоритмів, розроблених для специфічної мети автентифікації повідомлення. У більшості випадків код автентифиікації повідомлення створюється із блокового шифру або з геш-функції. Різні методи є сімействами універсальних геш-функцій. Також поєднують процедури, розпочаті декількома організаціями для стандартизації кодів автентифікації повідомлення.

MAC на основі блокового шифру

Найбільш загальним шляхом використання MAC у блоковому шифрі є використання шифру в режимі CBC (блокове з'єднання шифру): ключ MAC використовується як код на кожному кроці ітерації, і блок повідомлення, що обробляється на поточній ітерації, виступає у якості перекладу відкритого тексту в шифр, після побітового додавання з вихідним значенням шифротекста з попереднього кроку [119]:

$$H_1 = E_k(X_1), H_i = E_K(X_i \oplus H_{i-1})(2 \le i \le t).$$

Тут можна допустити, що повідомлення X (після заповнення), ділиться на $X_1..., X_t$ блоків з довжиною, що відповідає використо-

вуваному блокового шифру. E_K означає шифрування із секретним ключем K і H_t формує вихідне значення алгоритму MAC.

Окремо можна виділити три підходи побудови MAC-кодів:

1. MAC-коди, побудовані на основі БСШ (CBC-MAC).

2. MAC-коди, побудовані на основі безключових геш-функцій (HMAC, MDX-MAC).

3. MAC-коди, побудовані на основі сімейства універсальних геш-функцій.

Основна конструкція CBC підходить для атаки EXOR-підробки, отже може використана в додатках, де повідомлення мають фіксовану довжину. Трохи більш безпечні зміни в схемі, проте, існують.

Прикладом є *алгоритм EMAC* – це використання додаткового шифрування як вихідного перетворення, де ключ шифрувальної операції може бути похідний від ключа MAC. Інший приклад, відомий як пере-розподіл MAC, заміняє останнє шифрування двоключовим потрійним шифруванням. Безпека цих конструкцій може бути доведена на основі припущення, що основний блоковий шифр – псевдовипадковий [191].

Усі ці схеми включені в ISO/IEC 9797-1 [118], стандарт для MAC, що використовує (невизначений) блоковий шифр.

Розробка нових методів виробітку CBC-MAC обумовила появу нових атак на CBC-MAC, що докладно описується в додатку ISO/IEC 9797-1. В оновлений стандарт увійшли новий метод доповнення повідомлення і новий алгоритм формування MAC-коду зі спеціальною обробкою першого блоку (такий же як і обробка останнього блоку). Це робить більш трудомістким вичерпний пошук ключа. Крім того стандартом вводяться два нових "рівнобіжних" варіанта формування CBC-MAC.

Сутність нового методу доповнення полягає в такому. Рядок даних *x* доповнюється праворуч необхідною кількістю нулів (можливе доповнення і не нулів) до одержання рядка, довжина якого буде кратною *n* бітам. Потім отриманий рядок доповнюється ліворуч одним *n-розрядним* блоком *L*, що містить двійкове представлення довжини, вираженої в бітах, нерозширеного рядка даних *x*. При необхідності додатковий блок заповнюється ліворуч нулями до довжини *n*.

У новій версії ISO/IEC 9797-1 пропонується шість алгоритмів виробітку MAC-кодів. Перші три алгоритми вже були описані стандартом 1994 року.

Алгоритм 1. Перший алгоритм є просто виробітком CBC–MAC без якої-небудь додаткової обробки.

Алгоритм 2 є виробітком CBC–MAC з додатковою обробкою другого типу (з додатковим шифруванням останнього блоку на ключі k'

Алгоритм 3 є виробітком CBC–MAC з додатковою обробкою першого типу (з додатковим дешифруванням на ключі k' і шифруванням останнього блоку на ключі *k*). Таким чином, останній блок піддається потрійному шифруванню.

Алгоритм 4. У цьому алгоритмі перший і останній блоки повідомлення піддаються подвійному шифруванню.

Алгоритм 5. Алгоритм будується на принципі рівнобіжного застосування двох алгоритмів 1 з різними ключами. Два вихідних значення потім складаються за модулем 2, утворюючи результуючий MAC-код.

Алгоритм 6. Алгоритм також будується на принципі рівнобіжного застосування двох алгоритмів з різними ключами. Тільки як основа береться алгоритм 4. Два вихідних значення складаються за модулем 2, утворюючи результуючий MAC-код.

MAC на основі геш-функцій

Код автенфікації повідомлення може бути також отриманий з геш-функції, включаючи секретний ключ в обчисленні. Це найбільш загальний метод, оскільки такі MAC звичайно швидше, ніж MAC на основі блокового шифру. Проте, просто додаючи або додаючи ключовий фрагмент на вхід повідомлення геш-функції не дає безпечний MAC.

HMAC – вкладена конструкція, яка обчислює MAC для основної функції геша h, повідомлення X і секретний ключ K.

Альтернативою для HMAC є конструкції MDx-MAC [79], які можуть бути засновані на MD5, SHA, RIPEMD або аналогічних функціях геша. Тут основна геш-функція перетворена в MAC невеликими модифікаціями, включаючи секретний ключ на початку, в остаточному підсумку й у кожній ітерації функції геша. Це досягається ключовими залежними модифікаціями початкової величини й константи значення, що додається, яке використовується геш-функцією і вихідним перетворенням, залежними від ключа. Безпека MDx-MAC може бути доведена на основі припущення, що основна функція стиску псевдодовільна.

Як HMAC, так і MDx-MAC включені в ISO/IEC 9797-2 [119], стандарт для MAC, що використовував власну геш-функцію.

MAC, засновані на універсальному гешуванні

Сімейство геш-функцій геша $H = \{h : D \to R\}$ – це кінцева множина функцій із загальною сферою визначення D і (кінцевим) діапазоном R. Його можна також позначити через $H : K \times D \to R$ де $H_k : D \to R$ – функція множини для кожного $K \in K$ [56]. В останньому випадку, можна вибрати довільну функцію h з множини $K \in K$ й $h = HK$.

Множина універсальних геш-функцій є сімейством геш-функцій з деякою комбінаторною властивістю. Наприклад, сімейство геш-функцій $H = \{h : D \to R\}$ є майже універсальним, якщо для будь-якого різного $X, X' \in D$, імовірність, що $h(X) = h(X')$ – не більше, ніж коли $h \in H$ вибирається довільно [90; 100].

Це може використовуватися для автентифікації повідомлення, наприклад, при гешуванні повідомлення з функцією з сімейства універсальних геш-функцій функції, що кодує вихід геша, й потім здійснює закодований вихід геша як величину MAC. Це підтверджується тим, що безпека результуючої схеми MAC залежить від безпеки шифру, використаного для кодування вихідного геша. Комбінаторні властивості сімейства універсальних геш-функцій часто нескладно довести, і схеми, що отримуються на виході, є найшвидшими серед MAC.

Різновид NESSIE UMAC – приклад MAC, заснований на універсальному геші.

Сучасні стандарти.

Кілька організацій здійснюють ініціативи стандартизації автентифікайії кодів повідомлення. ISO/IEC розробив стандарт 9797 для MAC у двох окремих частинах. Частина 9797-1 [118] описує MAC, заснований на блоковому шифрі, а саме CBC-MAC для невизначеного блокового шифру (з деякими додатковими розширеннями, включаючи EMAC і перерозподіл MAC). Розділ 9797-2 [119] деталі MACs засновані у відданій функції геша, а саме HMAC і конструкції MDx-MAC для невизначеної геш-функції (і варіант MDx-MAC для коротких вхідних значень).

ANSI прийняв базований DES CBC-MAC (включаючи перерозподіл MAC) у своєму стандарті X9.19 [50] і HMAC (з невизначеною геш-функцією) у стандарті X9.71 [51]. NIST розробив FIPS 113 [123] для DES CBC-MAC і FIPS 198 [123] для SHA-1 HMAC.

Рівень безпеки залежить від розміру внутрішнього блоку (розмір блоку функції гешування), від довжини ключа і від довжини значення MAC-коду. Серед основних недоліків алгоритмів, розглянутих вище, є погане розсіювання та використання для генерації ключа алгоритму DES (AES). Щодо використання алгоритмів DES та AES для отримання стійкої ключової послідовності, то це накладає обмеження на швидкість гешування самого алгоритму MAC-коду.

У табл. 2.3 наведено аналіз тестування швидкості роботи алгоритмів гешування.

Таблиця 2.3

Аналіз тестування швидкості роботи алгоритмів гешування

Функція гешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
Whirlpool	10	C	28,013 Мбіт/с	46,961 Мбіт/с
SHA-2 (512)	80	C	41,159 Мбіт/с	68,701 Мбіт/с
SHA-2 (256)	64	C	81,308 Мбіт/с	135,557 Мбіт/с
ГОСТ 34311-95	-	C+Assembler	49,408 Мбіт/с	83,056 Мбіт/с
HAVAL	96 (128, 160)	C	337,842 Мбіт/с	564,809 Мбіт/с
SHA-1	80	C, Assembler	206,285 Мбіт/с 361,581 Мбіт/с	344,433 Мбіт/с 605,558 Мбіт/с
RIPEMD-160	160	C	147,465 Мбіт/с	246,568 Мбіт/с
MD5	64	C	278,715 Мбіт/с	574,635 Мбіт/с
MD4	48	C	344,086 Мбіт/с	467,793 Мбіт/с
UMAC	-	C C+Assembler	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с
ГОСТ 28147-89	16	C+Assembler	189,559 Мбіт/с	315,270 Мбіт/с

2.3. Коди детектування маніпуляцій

До безключових геш-функцій відносяться коди виявлення змін повідомлення (MDC-код, modification detection code), також відомі як коди виявлення маніпуляцій над повідомленнями або коди цілісності повідомлень. MDC-коди призначені для формування стислого образу

або геш-коду повідомлення, який задовольняє спеціальним властивостям. Вони поділяються на блокові шифри та модульну арифметику [69]. Зрештою, MDC-коди забезпечують, спільно з іншими механізмами, цілісність даних. Безключові геш-функції є одним із складових елементів цифрових підписів.

Геш-функції засновані на використанні принципів, закладених у геш-функціях сімейства MDx (MD2, MD4, MD5), які спеціально розроблялися для реалізації на 32-розрядних ЕОМ. На цей час геш-функції MD4, MD5 є найпоширенішими в практичних додатках геш-функціями. Однак деякі їх недоліки не дозволили стандартизувати їх [96]. Європейський консорціум RIPE, опираючись на свої дослідження властивостей цих алгоритмів, запропонував посилену версію MD4, що одержала назву RIPEMD. Геш-функція RIPEMD по суті складається із двох паралельно працюючих і модифікованих функцій MD4 (тобто функція має дві лінії). Суть модифікації полягає в зміні аргументів операторів циклічних зрушень і порядку проходження на вхід циклів геш-функції слів повідомлення, яке гешується. Слід розглянути ці алгоритми детальніше.

Алгоритм MDC-2

Даний метод виконує дві операції шифрування на обробку кожного вхідного блоку і, таким чином, швидкість геш-функції дорівнюватиме 1/2. Стандарт орієнтований на використання алгоритму DES як основи. Однак загальна конструкція може бути використана і для побудови геш-функції на основі інших блокових шифрів [5].

Геш-функція MDC-2 використовує такі компоненти: алгоритм DES як блоковий шифр E з довжиною блоку n = 64 біти і 56-бітним ключем K; функції g і $\tilde{g}$, що відображають 64-бітні рядки в 56-бітні ключі для DES. Функції визначаються в такий спосіб. Нехай $U = u_1u_2...u_{64}$ буде 64-бітним рядком. Тоді в рядку U вилучається кожен восьмий біт, починаючи з u_8, а другий і третій біти встановлюються в "10" для g і "01" для$\tilde{g}$, тобто:

$$g(U) = u_1 10 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} ... u_{63},$$
$$\tilde{g}(U) = u_1 01 u_2 u_3 u_4 u_5 u_6 u_7 u_8 u_9 u_{10} ... u_{63}.$$

При використанні алгоритму DES як блокового шифру геш-функція MDC-2 має такий вигляд (рис. 2.9):

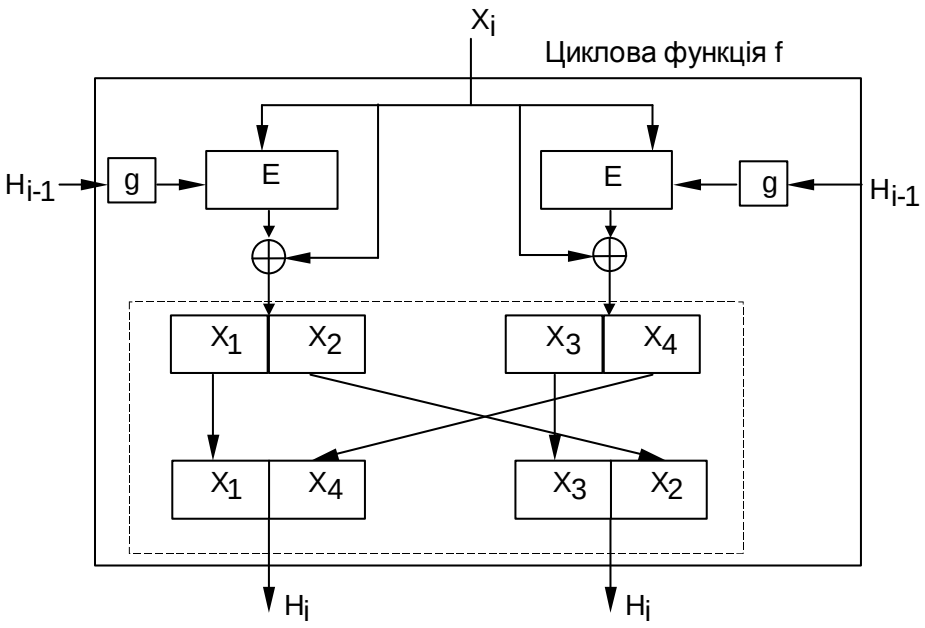


Рис. 2.9. **Схема роботи геш-функції MDC-2**

На основі MDC-2 побудована більш складна геш-функція MDC-4. Геш-функція MDC-4 як циклову функцію використовує геш-функцію MDC-2.

Одна ітерація циклової функції MDC-4 містить у собі послідовне виконання двох циклових функцій MDC-2. Таким чином, швидкість MDC-4 дорівнюватиме 1/4. Схема геш-функції MDC-4 зображена на рис. 2.10. Нижче наведений алгоритм MDC-4.

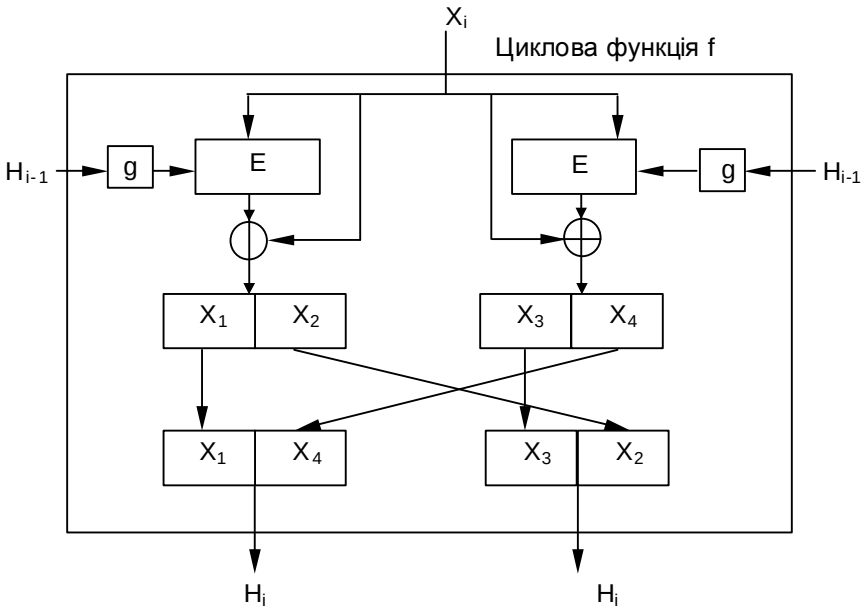


Рис. 2.10. **Структура геш-функції MDC-4**

Порівняльний аналіз геш-функцій наведено в табл.2.4.

Таблиця 2.4

Порівняльний аналіз геш-функцій

Геш-функція	Клас функції	Базові перетворення	Довжина гешу, біти
Whirlpool	однонаправлена	в кінцевих полях Галуа	512
SHA-2	однонаправлена	логічні та арифметичні	256, 384, 512
ГОСТ 34311-95	однонаправлена	БСШ	256
HAVAL	однонаправлена	логічні та арифметичні	128, 160, 192, 256
SHA-1	однонаправлена	логічні та арифметичні	160
RIPEMD-160	однонаправлена	логічні та арифметичні	160
MD5	однонаправлена	логічні та арифметичні	128
MD4	однонаправлена	логічні та арифметичні	128
UMAC	однонаправлена, MAC	в кінцевих полях Галуа	128, 64
Rijndael,CBC-MAC	виробка MAC	БСШ	128
ГОСТ28147	виробка MAC	БСШ	64

Проведений аналіз табл. 2.4 показує, що використання MDC-коду за даним вхідним повідомленним геш-коду може обчислити будь-який суб’єкт, а при використанні MAC-коду обчислити геш-код за даним вхідним повідомленням може тільки суб’єкт, що володіє секретним ключем. Таким чином, використання MAC-кодів дозволяє інтегровано вирішувати завдання гешування і забезпечення стійкості отриманих геш-кодів [37; 63; 55; 56; 119].

Проаналізувавши характеристики MDC-кодів і MAC-кодів, можна зробити висновок, що MAC-коди є більш стійкими і, на відміну від MDC-кодів, не потребують додаткових алгоритмів для шифрування повідомлення.

Розділ 3. Функції гешування інформації

Розглянуто методи й агоритмі формування геш-кодів на основі використання бесключових і ключових геш-функцій на основі вимог міжнародних стандартів MAC ISO/IEC 6796, ISO/IEC 10118.

3.1. Визначення та загальні вимоги до функцій гешування

Вибір та реалізація механізмів забезпечення цілісності та справжності інформаційних ресурсів у сучасних автоматизованих системах є одними з важливих етапів проектування та розробки підсистем захисту інформації. Це пов'язано з постійним зростанням послуг, які надаються різними мережними службами. Більшість послуг надаються при відсутності фіксованих мережених адрес клієнтів та їх особливостей. У зв'язку з чим, ризик порушення цілісності та автентичності інформації збільшується. Для захисту від таких загроз безпеки інформації, як правило, використовують механізми гешування даних – ключові та безключові геш-функції. Геш-функції також можуть використовуватись у складі електронного цифрового підпису, який є потужним механізмом забезпечення автентифікації в сучасних автоматизованих системах [32].

До обговорення аспектів безпеки визначимо більш точні визначення геш-функції та її криптографічних властивостей.

Геш-функція – це функція $h: D \to R$, де сфера визначення $D = \{0,1\}^*$ і сфера значень $R = \{0,1\}^n$ для деякого $n \geq 1$.

Компресійна функція – це функція $y_1 = h(x_1)$, де $D = \{0,1\}^a \times \{0,1\}^b \times$ $\times R = \{0,1\}^n$ для деяких a, b та $n \geq 1$, з $a + b \geq n$.

Ітеративний геш компресійної функції $f: \left(\{0,1\}^n \times \{0,1\}^b \to \{0,1\}^n\right)$ – це геш-функція $h: \left(\{0,1\}^b\right)^* \to \{0,1\}^n$, визначена як $h(X_1 \ldots X_t) = H_t =$ $= H_i = f(H_{i-1}, X_i)$ для $1 \leq i \leq t$ (множина $H_0 IV$).

Далі наведені визначення для стійкості по (другому) прообразу та стійкості до колізій.

Стійкість по прообразу. Геш-функція $h: \{0,1\}^* \to R$ є стійкою по прообразу степені $(t, \in)$, якщо не існує імовірнісного алгоритму I_h який приймає вхід $Y \in_R R$ і виводить значення $X \in \{0,1\}^*$ під час виконання не

більше t, де $h(X) = Y$ з імовірністю щонайменше ε, отриманої над випадковими виборами I_h і Y.

Стійкість по другому прообразу. Нехай S буде кінцеве підмножина з $\{0,1\}^*$. Геш-функція $h : \{0,1\}^* \to R$ є стійкою по другому прообразу степені $(t \in S)$, якщо не існує імовірнісного алгоритму S_h, який приймає вхід $X \in_R S$ і виводить значення $X' \in \{0,1\}^*$ під час виконання не більшеt, де $X' \neq X$, $h(X') = h(X)$ ймовірністю щонайменше ε, отриманої над випадковими виборами S_h і X.

Геш-функції використовуються як будівельний блок у різних криптографічних додатках. Найважливіше їх використання для захисту автентифікації інформації і як інструмент для схем цифрових підписів. Геш-функція – це функція, яка відображає вхід довільної довжини в фіксовану кількість вихідних біт – геш-значення. Для того щоб бути корисною в криптографічних додатках, геш-функція повинна задоволь-няти деяким вимогам. Геш-функції можуть поділятися на односторонні геш-функції та стійкі до колізій геш-функції.

Одностороння функція повинна бути стійкою по прообразу і другому прообразу, тобто повинно бути "важко" знайти повідомлення із заданим гешем (прообразом) або яке гешується в одне і те ж зна-чення, що і задане повідомлення (другий прообраз).

Стійка до колізій геш-функція це одностороння геш-функція, для якої "важко" знайти два різні повідомлення, для яких геш-значення однакове.

Для деяких програм можуть знадобитися додаткові властивості до геш-функцій, наприклад, псевдо випадковість виходу, що генерується. У контраст до інших криптографічних примітивів, обчислення геш-функції не залежить від будь-якої секретної інформації.

Одностороння геш-функція – це функція h, яка задовольняє таким умовам [83; 84]:

аргумент X може бути довільної довжини, а результат $h(X))$ має фіксовану довжину n біт;

геш-функція повинна бути односторонньою в тому сенсі, що по заданому Y в образі h складно знайти повідомлення X, таке що $h(X) = Y$ (стійкі по прообразу) і за заданим повідомленням X і значенням $h(X)$ важко знайти повідомлення $X' \neq X$, таке що $h(X') = h(X)$ (стійкі по другому прообразу).

Стійка до колізій геш-функція – це функція h, яка задовольняє таким умовам [87]:

аргумент X може бути довільної довжини, а результат $h(X)$ має фіксовану довжину n біт;

геш-функція повинна бути односторонньою, тобто стійкою по прообразу і стійкою по другому прообразу.

Для того, щоб геш-функція H вважалася *криптографічно стійкою*, вона повинна задовольняти трьом основним вимогам, на яких заснована більшість застосувань геш-функцій в криптографії:

незворотність або стійкість до відновлення прообразу: для заданого значення геш-функції m має бути обчислювально неможливо знайти блок даних x, для якого $h(x) = m$;

стійкість до колізій першого роду або відновлення другого прообразів: для заданого повідомлення m повинно бути обчислювально неможливо підібрати інше повідомлення n, для якого $h(n) = h(m)$;

стійкість до колізій другого роду: має бути обчислювально неможливо підібрати пару повідомлень, що мають однаковий геш.

Дані вимоги не є незалежними:

оборотна функція нестійка до колізій першого і другого роду;

функція, нестійка до колізій першого роду, нестійка до колізій другого роду; зворотне неправильно.

Односторонні геш-функції можуть застосовуватися для вирішення інших завдань, наприклад, вироблення ключів та псевдовипадкових чисел. Для застосування в таких завданнях геш-функція повинна задовольняти таким вимогам:

відсутність кореляції – вхідні і вихідні біти не повинні корелювати, тобто зміна будь-якого вхідного біта призводить до великих непередбачуваним змінам вихідних біт;

стійкість до близьких колізій – для заданої односторонньої функції обчислювально неможливо знайти два прообрази x і x', для яких геш-значення $h(x)$ і $h(x')$ відрізнялися б на малу кількість біт;

стійкість до часткової односторонності – обчислювально неможливо відновити будь-яку частину вхідного повідомлення так само, як і всі повідомлення. Більш того, за будь-якої відомої частини вхідного повідомлення обчислювально неможливо відновити частину (віднов-лення t невідомих біт вимагає не менш ніж 2^{t-1} операцій);

можливість роботи в режимі розтягування – можливість обчислення геш-функції при довжині вхідного повідомлення менше, ніж довжина геш-значення.

Вимога до застосовуваних у криптографії геш-функцій з секретним ключем [41]:

обчислювальна стійкість – неможливість знаходження геш-значення для заданого повідомлення без відомого секретного ключа, тобто для заданої ключовою геш-функції й однієї або більше коректних пар прообразів і геш-значень $(x_i, h(x_i, k))$ і невідомому секретному ключі k обчислювально неможливо знайти іншу коректну пару $(x, h(x,k))$ для будь-кого $x \neq x_i$.

Вимога обчислювальної стійкості передбачає виконання вимоги стійкості ключа (по одній або більше коректних пар прообразів і геш-значення $(x_i, h(x_i, k))$ обчислювально неможливо відновити секретний ключ), однак, вимога стійкості ключа k не передбачає виконання вимоги обчислювальної стійкості.

Більшість геш-функцій мають ітеративні конструкції, в тому сенсі, що вони базуються на функції компресії з фіксованими входами, вони обробляють кожен блок повідомлення подібним чином. Введення X доповнюється по однозначного правилом доповнення до кратності розміру блоку. Зазвичай це також включає додавання загальної довжини входу в бітах. Доповнений вхід потім ділиться на t блоків, які охоплюють від X_1 до X_T. Геш-функція включає компресійну функцію f і зв'язує змінну H_i між стадією $i-1$ і стадією i.

Класифікацію геш-функцій наведено на рис. 3.1.

До безключових геш-функцій відносяться коди виявлення змін повідомлення (MDC-код, modification detection code), також відомі як коди виявлення маніпуляцій над повідомленнями або коди цілісності повідомлень.

Суттєвим недоліком безключових геш-функцій є те, що вони не захищені від можливості по підбору такого ж самого повідомлення з однаковим гешем, та мають відсутність властивості обчислювальної стійкості. Зрештою MDC-коди забезпечують, спільно з іншими механізмами, цілісність даних.

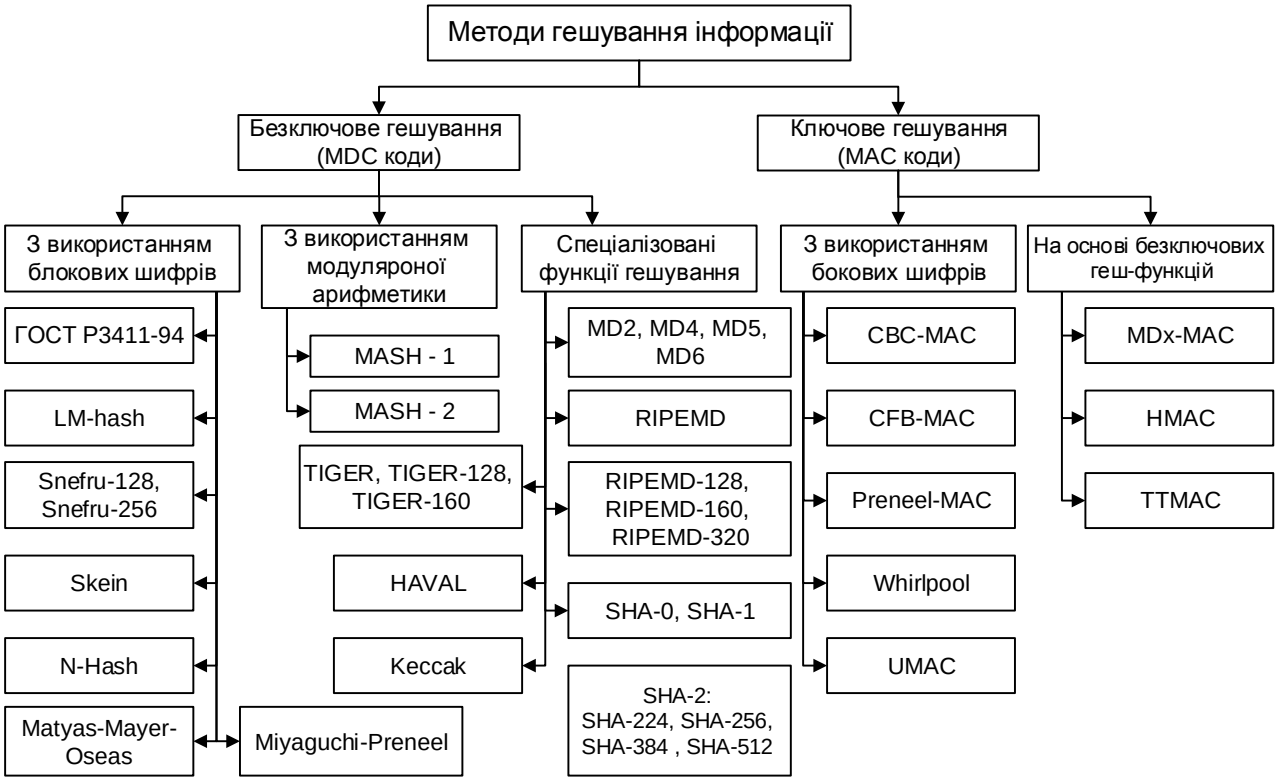


Рис. 3.1. **Класифікація геш-функцій**

3.2. Ітеративні геш-функції

Призначенням ключових ітеративних схем гешування є вироблення образу гешованого повідомлення (тексту) для визначення і доказу його достовірності і приналежності дійсному власникові (об'єкту). Ключові геш-функції також прийнято називати методами безпечного гешування. Суть безпечного гешування полягає у формуванні стислого образу відкритої послідовності, параметризованого секретним ключем. Довжина геш-коду і довжина ключа впливає на стійкість схеми гешування до нав'язування помилкових даних.

Ключова геш-функція представляє сімейство ітеративних геш-функцій F_k, параметризованих за допомогою секретного ключа.

Усі геш-функції, визначені в ISO/IEC 10118, як і більшість безключових геш-функцій побудовані на основі ітеративної моделі. На рис. 3.2 наведена загальна модель ітеративної геш-функції [107].

Відповідно до даної моделі формування геш-коду здійснюється таким чином. На вхід геш-функції h надходить вихідне повідомлення x – рядок даних довільної довжини. Даний рядок представляється у вигляді послідовності *r-розрядних* блоків x_i. Перед обробкою, за необхідністю,

останній блок доповнюється до r бітів. Крім того, з міркування підвищення стійкості, вихідний рядок може бути доповнений новим *r-розрядним* блоком, що, наприклад, може містити двійкове представлення числа, яке характеризує довжину вихідного повідомлення x. Стандарт ISO/IEC 10118 не визначає методи доповнення, але передбачається, що можуть бути використані методи, визначені в ISO/IEC 9797 [107; 118].

Основою геш-функції є так називана циклова функція $f(x,y)$ або функція стиску, що, у загальному випадку, бере на вхід два рядки x і y, довжиною відповідно r і s розрядів і формує на виході *s-розрядний* рядок. Кожен блок x_i слугує вхідним аргументом для циклової функції. Іншим аргументом є *s* розрядне проміжне значення (змінна зчеплення), отримане на попередньому кроці гешування [39; 40]. Нехай H_i позначає частковий результат гешування на *i*-му кроці (ітерації), тоді загальна модель ітеративної геш-функції зі входом $x = x_1x_2...x_t$ може бути описана такими співвідношеннями:

$$H_0 = IV;$$
$$H_i = f(x_i, H_{i-1}), 1 \le i \le t;$$
$$h(x) = g(H_{t,}),$$

де H_{i-1} – *s*-розрядна змінна зчеплення між i-1-м і i-м кроками обробки;

H_0 – вектор ініціалізації.

Нехай $\sum^n = \{0,1\}^n$ – множина всіх двійкових рядків довжини n, $\sum = \{0,1\}^r$ – множина всіх двійкових рядків довжини r, $\{H_i\}$ – множина функцій відображення $\sum^n \to \sum$, причому відображення сюрективне.

Інформація подається у вигляді послідовності блоків M_i визначеної довжини l_m, якщо необхідно, то доповнюється до розміру, кратного довжині блоку. Сукупність блоків $\{M_i\}$ послідовно обробляється в циклові функції (рис. 3.3). Результат останньої ітерації поступає на вихід геш-функції у вигляді образу: $h = H(M_i)$, де $h \in \Sigma^r, \Sigma^r$ – множина значень геш-функції.

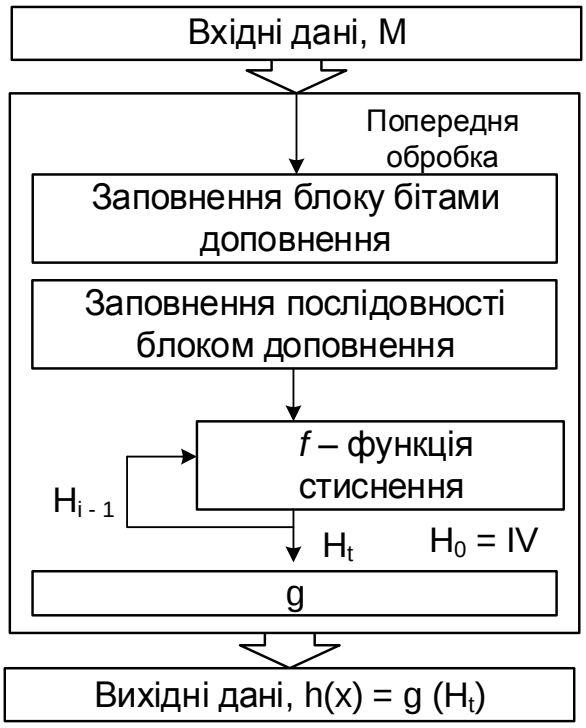


Рис. 3.2. **Загальна схема ітеративної геш-функції**

На виході циклової функції f формується *s*-розрядний геш-код. При необхідності він може бути усічений до заданої довжини за допомогою додаткової обробки $g(H_t)$. Найчастіше $g(H_t) = H_t$. Таким чином, для формування геш-коду відповідно до ітеративної моделі геш-функції за ISO/IEC 10118 необхідно вибрати такі параметри: r – довжина блоку даних; s – довжина виходу циклової функції, так називана змінна зчеплення, що визначає максимально можливу довжину вироблюваного геш-коду; IV – s-розрядний вектор ініціалізації; $L_H \leq s$ – довжина геш-коду.

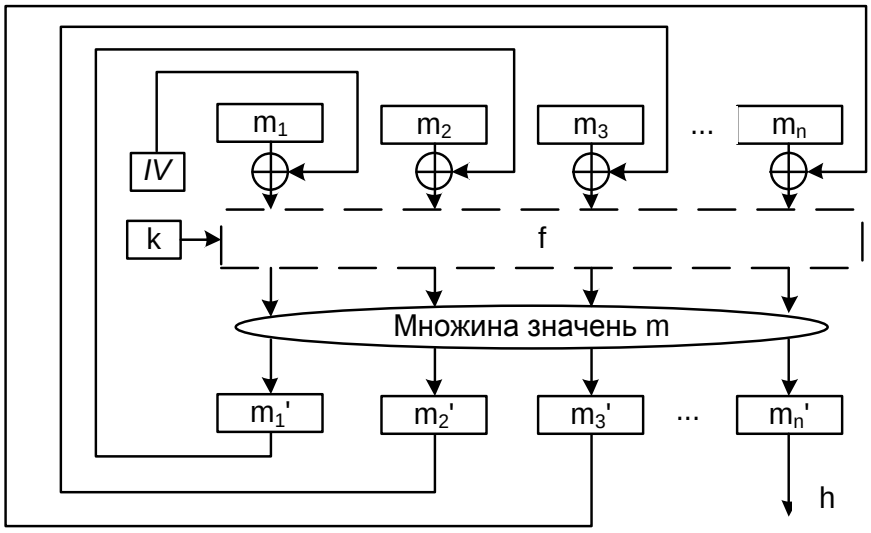


Рис. 3.3. **Структура геш-функції CBC-MAC ISO/IEC 6796**

3.3 Безключове гешування

Уже відзначалося, що геш-функції повинні володіти як мінімум двома властивостями, а саме властивістю стиску і легкістю обчислення. Безключові геш-функції, крім цього, повинні мати додаткові властивості. Варто їх розглянути.

Нехай h – безключова геш-функція, вхідними аргументами якої є рядки x і x', а вихідними значеннями рядка y і y'. Безключова геш-функція повинна мати такі властивості [67].

1. Нехай дане деяке y отримане за невідомим вхідним рядком x Тоді обчислювально неможливо знайти деяке $x' = x$ таке, що $h(x') = y$. Власне кажучи, для всіх заздалегідь визначених вихідних значень геш-функції обчислювально знайти яке-небудь вхідне значення, що відображається в задане вихідне значення. Така властивість називається стійкістю до прообразу (preimage resistance) або однобічністю (one-way).

2. Стійкість до другого прообразу (2nd-preimage resistance). Нехай відомо деякий рядок x. Тоді обчислювально неможливо знайти другий рядок (другий прообраз) $x' = x$ такий, що $h(x) = h(x')$. Дана властивість зветься слабкою стійкістю до зіткнення (weak collision resistance).

3. Стійкість до зіткнення (collision resistance). Обчислювально неможливо знайти будь-які два різні рядки даних x і x' для яких $h(x) = h(x')$. На відміну від попередньої властивості, стійкість до зіткнення розглядається в умовах, коли може здійснюватися вільний вибір обох вхідних рядків. Дана властивість ще відома як сильна стійкість до зіткнення (strong collision resistance).

У залежності від властивостей, якими володіють геш-функції, безключові геш-функції можна розділити на два класи *однобічні геш-функції* і *вільні від зіткнень геш-функції*.

Однобічна геш-функція (ОБГФ) або слабка однобічна геш-функція – це геш-функція, що має властивості стиску, легкості обчислення, стійкості до прообразу і стійкості до другого прообразу.

Вільна від зіткнення геш-функція (ВЗГФ) або сильна однобічна геш-функція – це геш-функція, що має властивості стиску, легкості обчислення, стійкості до другого прообразу і стійкості до зіткнення.

Зловмисник у загальному випадку при реалізації атак на геш-функції може вирішувати такі задачі:

атака на ОБГФ – даний геш-код y, знайти повідомлення x таке, що $y = h(x)$ або дано пару $(x, h(x))$, знайти друге повідомлення x' таке, що $h(x') = h(x)$.

атака ВЗГФ – знайти будь-які два вхідних повідомлення x і x' таких, що $h(x') = h(x)$.

Однобічні геш-функції є одним з фундаментальних криптографічних примітивів і використовуються для забезпечення цілісності даних, у цифрових підписах, протоколах підтвердження знань, схемах установлення (вироблення) загального ключа, а також як елементи генераторів псевдовипадкових чисел і в багатьох інших додатках.

У ході використання геш-функцій були визначені додаткові практичні властивості, а саме [83; 84]:

відсутність кореляції між вхідними і вихідними бітами. Бажано, щоб будь-який вхідний біт впливав на декілька вихідних біт . Іншими словами геш-функція повинна мати досить гарний лавинний ефект;

ускладнена стійкість до зіткнень, що полягає у тому, що важко знайти два будь-яких вхідних значення x і x' таких, що $h(x)$ і $h(x')$ відрізнялися б у невеликій кількості біт;

часткова стійкість до прообразу і локальна однобічність, що полягають у тому, що відновлення частини повідомлення є такою же складною задачею, як і відновлення всього повідомлення. Більш того, навіть якщо відома частина вхідного повідомлення, відновлення залишку повідомлення також є трудомісткою задачею (наприклад, для відновлення t невідомих вхідних біт, необхідно виконати в середньому 2^{t-1} операцій гешування).

Однобічні геш-функції визначені в окремому міжнародному стандарті ISO/IEC 10118. Стандарт ISO/IEC 10118 складається з чотирьох частин. Перші дві частини були опубліковані в 1994 році, третя частина – у 1998 році, а четверта частина в даний час перебуває в стадії розробки [107].

Перша частина ISO/IEC 10118-1:1994 – загальна частина, у якій вводяться загальні визначення і поняття для інших частин стандарту, а також модель ітеративних геш-функций.

ISO/IEC 10118-2:1994 визначає геш-функції, що використовують *n-розрядний* алгоритм блокового шифрування. У даній частині описуються методи формування геш-коду на основі *n-розрядного* блокового шифру.

У третій частині ISO/IEC 10118-3:1998 визначаються спеціалізовані геш-функції (dedicated hash function). Стандарт описує три типи таких геш-функцій, а саме американський стандарт SHA-1, і європейський стандарт RIPEMD-128, RIPEMD-160 [108].

Нарешті в четвертій частині стандарту ISO/IEC 10118-4 визначені геш-функції, що використовують модулярну арифметику. Стандарт уводить дві геш-функції MASH-1 і MASH-2, що використовують модульне зведення в степінь для побудови геш-коду [109].

3.3.1. На основі блокових шифрів

Друга частина стандарту ISO/IEC 10118-2 визначає методи формування геш-коду на основі використання *n-розрядного* блокового шифру. Практичною мотивацією побудови таких геш-функцій є те, що використання як основного компонента геш-функції досить ефективної програмної або апаратної реалізації блокового шифру, що застосовується в системі, дозволить забезпечити велику функціональність системи за менші витрати. ISO/IEC 10118-2 визначає два методи формування геш-коду на основі використання *n-розрядного* блокового шифру. Перший метод дозволяє виробляти геш-код однократної довжини, тобто довжиною геш-кода $L_н \le n$, а другий метод виробляє геш-код подвійної довжини, тобто довжиною $L_н \le 2n$. Геш-функції однократної довжини використовуються для побудови однобічних геш-функцій на основі блокових шифрів з довжиною блоку $n = 64$ біта або побудови вільних від зіткнень геш-функцій на основі блокових шифрів з довжиною блоку $n = 128$ біт. У даний час геш-функції подвійної довжини будуються тому, що сучасні блокові шифри є 64-розрядними, а геш-функції однократної довжини не є вільними від зіткнень [109].

Оскільки геш-функції спираються на алгоритм блокового шифрування, то одним з аргументів геш-функції є ключ. Тут циклова функція задається як $f(n,k,s)$, де n – довжина вхідного блоку даних, k – довжина ключа, s – довжина вихідного блоку. У найпростішому випадку ключ має таку ж довжину, як і оброблюваний блок, тобто n розрядів. В інших випадках геш-функції можуть використовувати ключі більшої (наприклад, подвійної) довжини.

Нарешті, для таких геш-функцій вводиться така характеристика як швидкість, тобто кількість операцій блокового шифрування, які необхідно виконати для формування геш-коду довжиною, рівній довжині блоку шифру. Нехай h – ітеративна геш-функція, побудована на основі блокового шифру з цикловою функцією f, що виконує t операцій блокового шифрування для обробки n-*розрядного* блоку повідомлення. Тоді швидкість геш-функції визначається як $1/t$.

Стандарт функції гешування ГОСТ Р 34.11-94

Стандарт ГОСТ Р 34.11-94 визначає алгоритм і процедуру обчислення геш-функції для будь-яких послідовностей двійкових символів, що застосовуються в криптографічних методах обробки і захисту інформації [22]. Цей стандарт базується на блоковому алгоритмі шифрування ГОСТ 28147-89, хоча, в принципі, для бізнес-структур можна використовувати й інший блоковий алгоритм шифрування з 64-бітовим блоком і 256-бітовим ключем, який має меншу обчислювальну складність.

Геш-функція ГОСТ Р 34.11-94 формує 256-бітове геш-значення.

Функція стиску $H_i = f(M_i, H_{i-1})$ (обидві операнди M_i і H_{i-1} є 256-бітовими величинами) визначається таким чином [22]:

1. Генеруються 4 ключі шифрування K_j, $j = 1...4$, шляхом лінійного змішування M_i, H_{i-1} і деяких констант C_j.

2. Кожен ключ K_j, використовують для шифрування 64-бітових підслів h_i слова H_{i-1} у режимі простій заміни: $S_j = E_{k_1}(h_j)$. Результуюча послідовність S_4, S_3, S_2, S_1 завдовжки 256 біт запам'ятовується в тимчасовій змінній S.

3. Значення H_i є складною лінійною функцією змішування S, M_i і H_{i-1}.

При обчисленні остаточного геш-значення повідомлення M враховуються значення трьох зв'язаних між собою змінних:

H_n – геш-значення останнього блоку повідомлення;

Z – значення контрольної суми, що отримується при складанні по модулю 2 всіх блоків повідомлення;

L – довжина повідомлення.

Ці три змінні і доповнений останній блок M_n повідомлення об'єднуються в остаточне геш-значення таким чином:

$$H = f(Z \oplus M', f(L, f(M' H_n))).$$

Ця геш-функція (рис. 3.4) визначена стандартом ГОСТ Р 34.11-95 для використання спільно зі стандартом електронного цифрового підпису ДСТУ 4145 або ГОСТ Р 34.310-95 [21]. За оцінками провідних спеціалістів практична реалізація алгоритму гешування даних ГОСТ Р 34.11-94, ГОСТ 28147-89 (у 4-му режимі) є низькошвидкісною. На сучасному ПК досягає до 100 Кбіт/с, та має розмір MAC-коду 64 біти, що також вже є недостатнім для багатьох практичних задач. Тому в арсенал сучасного криптографа повинні входити більш гнучкі алгоритми гешування, які володіють теоретичними границями стійкості та дозволяють отримувати високі показники швидкості гешування даних.

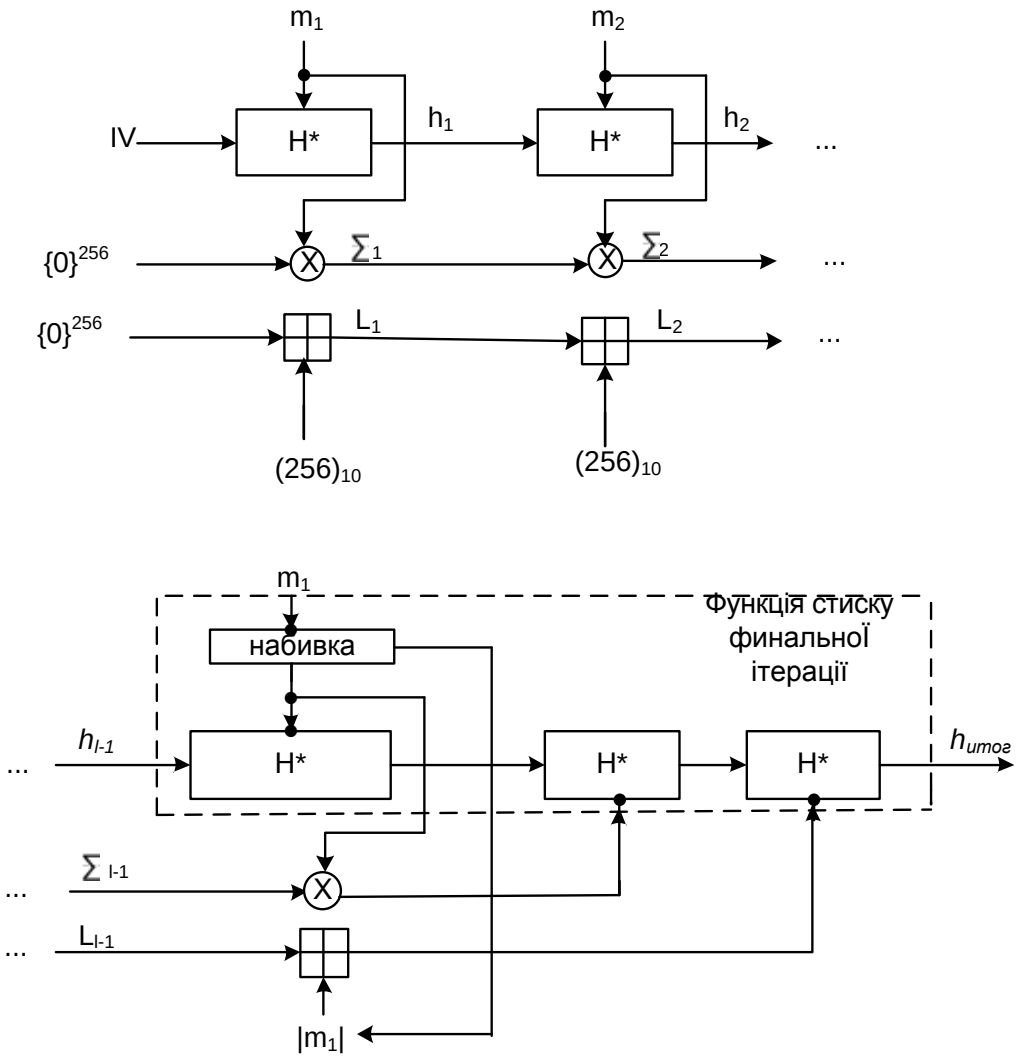


Рис. 3.4. **Геш-функція ГОСТ Р 34.11-94**

Особливості ГОСТ Р 34.11-94.

При обробці блоків використовуються перетворення за алгоритмом ГОСТ 28147-89; обробляється блок довжиною 256 біт, і вихідне значення теж має довжину 256 біт; застосовані заходи боротьби проти пошуку колізій, на основі неповноти останнього блоку; обробка блоків відбувається по алгоритму шифрування ГОСТ 28147-89, який містить перетворення на S-блоках, що істотно ускладнює застосування методу диференціального криптоаналізу до пошуку колізій [22].

Формат виводу.

Згідно зі стандартом, результатом геш-функції є 256-бітове число. Стандарт не вказує, як воно повинно виводитися. Різні реалізації використовують різні формати виводу, що разом з двома поширеними S-блоками посилює плутанину. ГОСТ Р 34.11-94 оперує з Little-endian числами. Багато реалізації (зокрема rhash, mhash library, консольна утиліта openssl) виводять 32 байта результуючого геша в шістнадцятковому поданні, в порядку, в якому вони розташовуються в пам'яті – молодші байти першими. Це подання виправдовується тим, що воно ж використовується при виведенні геш сум широко поширених західних алгоритмів MD5, SHA1, Tiger, Whirlpool і ін.

GOST("Thisismessage,length=32bytes")=
B1C466D37519B82E8319819FF32595E047A28CB6F83EFF1CЯ6916A815A
637FFFA.

У наведених у стандарті прикладах, результуючий геш записується як шістнадцяткове подання 256-бітного Little-endian числа. Тим самим, виходить зворотний порядок байт (старші розряди першими). Такий же порядок використовує, зокрема, програма gostsum, що постачається з оригіналами бібліотеки OpenSSL.

H = FAFF37A6 15A81669 1CFF3EF8 B68CA247 E09525F3 9F811983
2EB81975 D366C4B1

Оцінка криптостійкості.

У 2008 р. командою експертів з Австрії та Польщі була виявлена технічна вразливість, що скорочує пошук колізій у 223 разів. Кількість

операцій, необхідних для знаходження колізії, таким чином, становить 2105, що однак на даний момент практично не реалізовується. Проведення колізійної атаки на практиці має сенс тільки у випадку цифрового підпису документів, причому якщо зломник може змінювати непідписаний оригінал.

Використання.

Функція використовується при реалізації систем цифрового підпису на базі асиметричного крипто-алгоритму за стандартом ГОСТ Р 34.10 – 2001. Спільнота російських розробників СКЗІ погодило використовувані в Інтернеті параметри ГОСТ Р 34.11-94, див. RFC 4357:

використання в сертифікатах відкритих ключів;

використання для захисту повідомлень у S/MIME (Cryptographic Message Syntax, PKCS # 7);

використання для захисту з'єднань у TLS (SSL, HTTPS, WEB);

використання для захисту повідомлень в XML Signature (XML Encryption);

захист цілісності Інтернет-адресів та імен (DNSSEC).

Геш-функції однократної довжини

В основі стандартизованого методу виробітку геш-кода лежить схема Матіаса – Мейєра – Озіса (Matyas – Mayer – Oseas), яка наведена на рис. 3.5. У даному розділі розглядається дуальна даній схемі, схема Девіса – Мейєра (Davies – Meyer), яка наведена на рис. 3.6 і схема Міагуччи – Пренеля (Miyaguchi – Preneel), яка наведена на рис. 3.7.

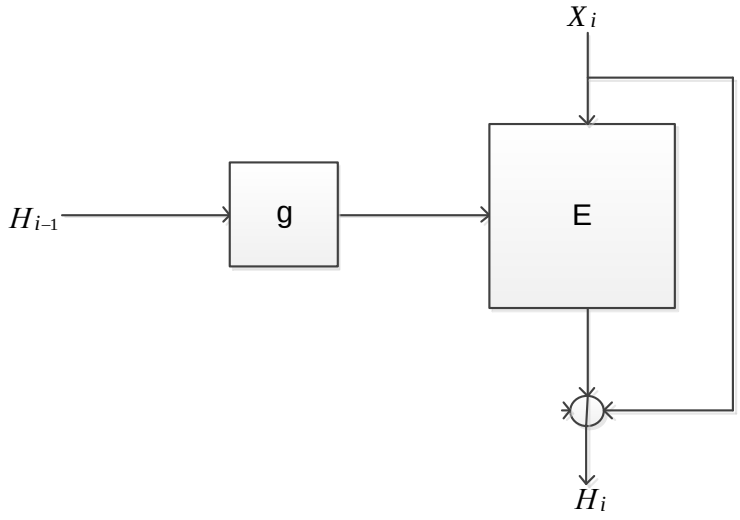


Рис. 3.5. **Схема Матіаса – Мейєра – Озіса**

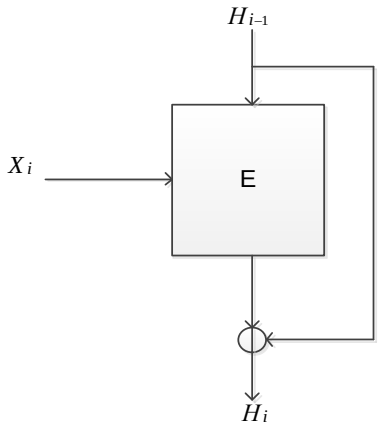


Рис. 3.6. **Схема Девіса – Мейєра**

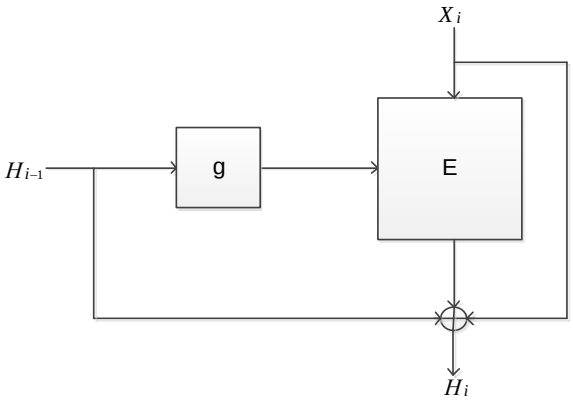


Рис. 3.7. **Схема Міагуччи – Пренеля (Miyaguchi – Preneel)**

Таким чином, для однократних геш-функцій на основі блокових шифрів використовуються такі компоненти:

n-розрядний блоковий шифр E_K із секретним ключем K;

функція g, яка відображає n-*розрядний* рядок у ключ K, придатний для використання у відповідному блоковому шифрі E;

фіксований *n-розрядний* вектор ініціалізації IV для певного алгоритму E [78].

Геш-функція Matiaca – Мейєра – Озіса (ISO/IEC 10118-2)

Вхід: двійковий рядок *x* довільної довжини.

Вихід: *n-розрядний* геш-код за рядком x.

1. Рядок даних X, який гешується, розбивається на *n-розрядні* блоки. При необхідності останній блок доповнюється необхідною кількістю символів. Слід позначити доповнене повідомлення, яке склада-

ється з *tn-розрядних* блоків, як $X_1X_2...X_t$. Визначається деяка *n-розрядна* константа, що виступає як вектор ініціалізації IV.

2. Вихідне значення H_t визначається відповідно до виразу:

$$H_0 = IV;$$
$$H_i = E_{g(H_{i-1})}(x_i) \oplus x_i, 1 \le i \le t.$$

Як видно, довжина блоку даних дорівнює довжині відкритого/закритого тексту для блокового шифру. Довжина геш-коду також установлюється рівною довжині блоку блокового шифру, тобто $s = n$. Функція усікання є взяттям молодших значущих L_H бітів рядка H_t.

Геш-функція Девіса – Мейєра

Вхід: двійковий рядок *x* довільної довжини.

Вихід: *n-розрядний* геш-код за рядком x [67].

1. Рядок даних *x*, який гешується, розбивається на *k-розрядні* блоки (за розміром ключа блокового шифру). При необхідності останній блок доповнюється необхідною кількістю символів. Слід позначити доповнене повідомлення, яке складається з *tk-розрядних* блоків, як $X_1X_2...X_t$. Визначається деяка *n-розрядна* константа, що виступає як вектор ініціалізації IV.

2. Вихідне значення H_t визначаться відповідно до виразу:

$$H_0 = IV;$$
$$H_i = E_{x_i}(H_{i-1}) \oplus H_{i-1}, 1 \le i \le t.$$

Дана схема є дуальною попередній у тому значенні, що X_i і H_{i-1} міняються місцями і ролями. Це призводить до того, що в схемі Девіса – Мейєра швидкість може бути і менше одиниці. Так, якщо за основу взяти алгоритм DES з 56-розрядним ключем, то швидкість геш-функції дорівнюватиме 56/64 < 1.

Геш-функція Міагуччи – Пренеля

Вхід: двійковий рядок *x* довільної довжини.

Вихід: *n-розрядний* геш-код за рядком x.

1. Рядок даних x, що гешується, розбивається на *n-розрядні* блоки. При необхідності останній блок доповнюється необхідною кількістю

символів. Потрібно позначити доповнене повідомлення, яке складається з *tn-розрядних* блоків як $X_1X_2...X_t$. Визначається деяка *n-розрядна* константа, що виступає як вектор ініціалізації IV.

2. Вихідне значення H_t визначається відповідно до виразу:

$$H_0 = IV;$$
$$H_i = E_{g(H_{i-1})}(x_i) \oplus x_i \oplus H_{i-1}, 1 \le i \le t.$$

Стійкість даних геш-функцій базується на стійкості використовуваного в їхній основі блокового шифру E, що повинен мати необхідні властивості випадковості формованих вихідних значень. Крім того, для даного алгоритму не повинні реалізуватися атаки, що спираються на використання будь-яких особливих властивостей і особливостей структури алгоритму. Тобто реалізується модель "чорного ящика". Тоді пошук прообразу або зіткнення вимагатиме виконання порядку 2^n і $2^{n/2}$ операцій шифрування, відповідно.

Геш-функція Snefru

Snefru – це криптографічна односпрямована геш-функція, винайдена Ральфом Меркле, яка перетворює повідомлення будь-якої довжини в 128- і 256- бітні геш-суми. Вона була названа на честь єгипетського фараона Sneferu, продовжуючи традиції блокових шифрів Khufu і Khafre.

Нехай M – повідомлення деякої довжини. Функція називається односпрямованою, якщо з рівності $h = H(M)$ легко:

знайти геш-код h, знаючи, що повідомлення M дуже трудомістке;

знайти повідомлення M за відомим геш-кодом h (тобто якщо геш-код пароля став відомий хакеру, то пароль він по ньому не знайде);

знайти відмінне від M повідомлення M', таке, що їх геш-коди $H(M') = H(M)$ збігаються.

Простіше визначення можна записати так:

односпрямованість – це "відбиток пальця";

якщо дана конкретна людина, то можна взяти у нього відбиток пальця;

неможливо знайти людину за відбитком його пальця (якщо немає бази даних з відбитками пальців усіх людей, а її немає);

неможливо знайти другу людину з таким же як у іншого відбитком пальця.

Опис алгоритму:

вхідне повідомлення розбивається на блоки довжиною 512-m біт (де m – довжина вихідного геш-значення).

"Серцем" алгоритму є функція H, що гешує 512-бітове значення → m-бітове.

H (<512 біт>) = <m біт> m біт виходу функції H є геш-значенням блоку. Наступний блок додається до геш-значення попереднього, і знову гешується. До початкового блоку додаються нулі (до 512 біт). Якщо вихідне повідомлення складається не з цілого числа блоків, останній блок доповнюється нулями. Після операцій з останнім блоком, отримані m бітів додаються до бінарного подання довжини повідомлення та гешується останній раз.

Тобто $x_1, x_2, x_3, ... x_n$ – блоки, на які розбито вихідне повідомлення (кожне по 512-m біт, крім останнього, останнє може бути менше 512-m біт). H (x_1 | <m бітових 0>) = y_1, де "|" позначає додаток (дописування) до того, що ліворуч від "|" тим, що праворуч.

$H(y_1 | x_2) = y_2$

$H(y_2 | x_3) = y_3$

...

$H(y(n-1) | x_n)$ < необхідна кількість бітових нулів до 512-m >) = y_n;

$H(y_n | L |$ < необхідне кількість бітових нулів до 512-m >) = y.

Функція H ґрунтується на функції E, де E – оборотна функція блочного шифрування, що працює з 512 бітовими блоками. На виході H – останні m бітів виходу E, об'єднані за допомогою XOR з першими m бітами входу E.

Тобто $H(x) =$ < останні m біт $E(x) >$ XOR < Перші m біт $x >$ E рандомізують дані за кілька проходів. Оригінальний Snefru складався тільки з двох проходів. Кожен прохід складається з 64 рандомізуючих етапів.

Потрібно розглянути один етап.

$E(X)$... Входом E є повідомлення X.

$Y := X$. Під час етапу обробляються слова Y, і повідомлення Y перетвориться.

Використовується S-блок (побудова S-блоку аналогічна побудові в алгоритмі Khafre). Входом S-блоку є байт даних, що залежить від поточного слова. Слова, сусідні з поточними, піддаються операції XOR з вихідним словом S-блоку. Крім того, виконується ряд циклічних зрушень.

Криптоаналіз Snefru.

Використовуючи диференційний криптоаналіз, Біхам і Шамір показали небезпечність двопрохідному Snefru (з 128-бітовим геш-значенням). Їх спосіб розтину за кілька хвилин виявляє пару повідомлень з однаковим геш-значенням.

Для 128 бітового Snefru їх розкриття працюють краще, ніж розтин грубою силою для чотирьох і менше проходів. Розтин Snefru методом "Дня народження" вимагає 2^{64} операцій; диференціальний криптоаналіз може знайти пару повідомлень з однаковим геш-значенням за $2^{28,5}$ операцій для трьохпрохідного Snefru і за $2^{44,5}$ операцій для чотирьох-прохідного Snefru. Знаходження повідомлення, геш-значення якого збігається з заданим, при використанні грубої сили вимагає 2^{128} операцій, при диференційному криптоаналізі для цього потрібно 2^{56} операцій для трьох прохідного і 2^{88} операцій для чотирьохпрохідного Snefru.

Хоча Біхам і Шамір не аналізували 256-бітові геш-значення, вони провели аналіз аж до 224-бітових геш-значень. У порівнянні з розкриттям методу "Дня народження", які вимагають 2^{112} операцій вони можуть знайти повідомлення з однаковим геш-значенням за $2^{12,5}$ операцій для двох прохідного Snefru, за 2^{33} операцій для трьох прохідного Snefru і за 2^{81} операцій для чотирьох прохідного Snefru.

У даний час Меркле рекомендує використовувати Snefru принаймні з вісьмома проходами. Однак з такою кількістю проходів алгоритм стає набагато повільніше, ніж MD5 або SHA.

Геш-функція Khufu

Khufu – у криптографії симетричний блоковий криптоалгоритм, розроблений Ральфом Мерклом у 1990 р. в якості альтернативи DES, що виправляє ряд її недоліків [124].

Принципи алгоритму.

Основним принципом, розглянутим при розробці алгоритму було використання накопиченого досвіду аналізу DES і елементів даного алгоритму з виправленням його недоліків і досягнення максимальної стійкості до криптоанализу.

1. Однозначно, 56-бітовий розмір ключа DES занадто малий і повинен бути збільшений.

2. Інтенсивне використання перестановок у DES зручно тільки для апаратних реалізацій, але затрудняє програмні реалізації. Найбільш швидкі реалізації DES виконують перестановки табличним способом. Перегляд таблиці може забезпечити ті ж характеристики "розсіювання", які властиві перестановці, і може зробити реалізацію значно гнучкіше.

3. S-блоки DES, усього з 64 4-бітовими елементами, занадто малі. Повинні збільшитися й S-блоки. Більше того, всі вісім S-блоків використовуються одночасно. Хоча це й зручно для апаратури, для програмної реалізації це здається непотрібним обмеженням. Повинні бути реалізовані більший розмір S-блоків і послідовне (а не паралельне) їх використання.

4. Початкова й заключна перестановки криптографічно безглузді, тому вони повинні бути усунуті.

5. Усі швидкі реалізації DES заздалегідь розраховують ключі для кожного етапу. Нема сенсу ускладнювати ці обчислення.

6. Критерії проектування S-блоків повинні бути загальнодоступні.

Алгоритм.

Khufu має 64-бітовий (8-байтний) блок: 64-бітовий вихідний відкритий текст розбивається на дві 32-бітові половини, іменовані автором як L і R. Над половинами L і R, так само, як і над деякими частинами ключа проводиться операція XOR. Потім, за аналогією з DES, над результатом даної операції проводиться ряд операцій у процесі ряду етапів. На кожному етапі молодший значущий байт L використовується в якості вихідних вхідних даних S-Блоку. У кожного S-блоку 8 вхідних біт і 32 вихідних біта. Далі над обраним 32-бітовим блоком у S-блоці проводяться операції XOR з R. Потім L циклічно зрушується на кілька з восьми біт, L і R взаємозамінюються – етап завершується. S-блок не є статичним і змінюється кожні вісім етапів. Після останнього етапу над L і R виконується операція XOR з іншими елементами ключа, L і R поєднуються, становлячи блок шифротекста.

Частини ключа використовуються для XOR із блоком шифрування як на початку, так і наприкінці алгоритму, але головна мета 512-бітного ключа – це генерація секретних S-блоків, які, по суті, є частиною ключа. Алгоритм надає спосіб генерації S-блоків по ключу. Кількість етапів алгоритму чітко не визначена, однак 8-етапний Khufu чутливий до розкриття з обраним відкритим текстом і тому рекомендується використовувати 16, 24 або 32 етапів. Кількість етапів повинна бути кратною восьми, що дозволяє застосовувати, приміром, 64-етапний варіант. Структуру алгоритму наведено на рис. 3. 8.

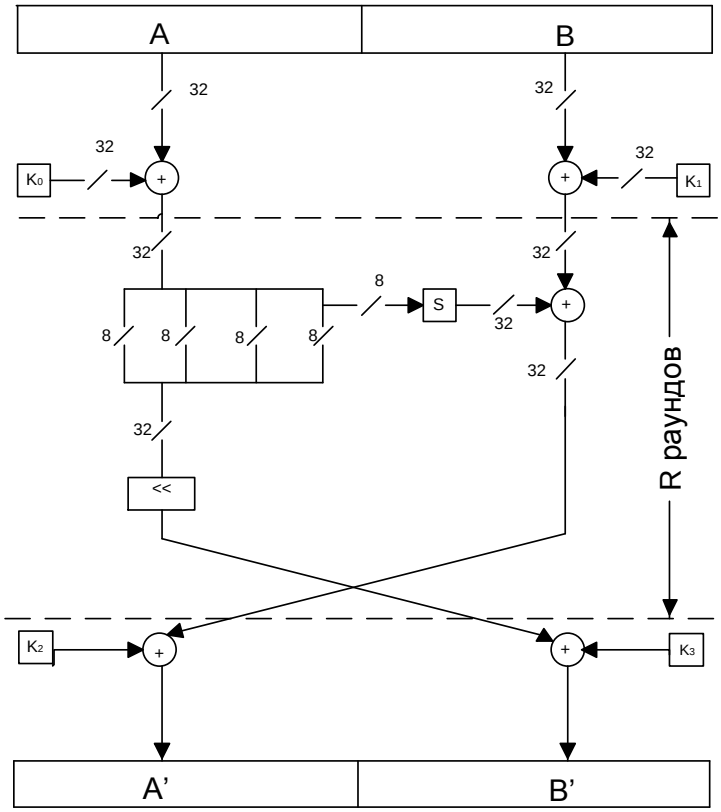


Рис. 3.8. **Структура алгоритму Khufu**

Стійкість.

Стійкість до диференціального криптоаналізу алгоритму Khufu заснована на використанні залежних від ключа й секретних S-блоків. Виконано диференціальне розкриття 16-етапного Khufu, яке відкриває ключ після 231 обраних відкритих текстів, але воно не може бути розширене на більшу кількість етапів. Оскільки кращим способом розкрити Khufu є груба сила, його надійність досить висока. 512-бітовий ключ алгоритму забезпечує необхідну складність, 2512, що виключає практично можливість відновлення відкритого тексту.

Геш-функція Khafre

Khafre – друга із криптосистем, запропонованих Мерклом. (Khufu (Хуфу) і Khafre (Хафр) – це імена єгипетських фараонів). По конструкції цей алгоритм схожий на Khufu, але він спроектований для додатків, що не використовують попередні обчислення. S-блоки не залежать від ключа. Замість цього Khafre використовує фіксовані S-блоки. Блок шифрування зазнає операції XOR із ключем не тільки перед першим етапом і після останнього, але й після кожних 8 етапів шифрування.

Меркл припустив, що з Khafree повинні використовуватися 64- або 128-бітові ключі, і що для Khafre будуть потрібні більше етапів, ніж для Khufru. Це поряд з тим, що кожний етап Khafre складніше етапу Khufu, робить Khafre більш повільним. Проте для Khafre не потрібні ніякі попередні розрахунки, що дозволяє швидше шифрувати невеликі порції даних.

У 1990 р. Біхам і Шамір застосували свій метод диференціального аналізу проти Khafre. Їм вдалося зламати 16-етапний Khafre за допомогою розкриття з обраним відкритим текстом після 1500 різних шифрувань. На їх персональному комп'ютері це зайняло близько години. Перетворення цього розкриття в розкриття з відомим відкритим текстом вимагає близько 2^{38} шифрувань. Khafre з 24 етапами може бути розкритий за допомогою розкриття з обраним відкритим текстом за 2^{53} шифрування, а за допомогою розкриття з відомим відкритим текстом – за 2^{59} шифрування.

У 1990 р. Ральф Меркл (Ralf Merkle) запропонував два алгоритми. В основі їх проектування лежали такі принципи:

1. 56-бітовий розмір ключа DES занадто малий. Тому що вартість збільшення розміру ключа зневажливо мала (комп'ютерна пам'ять недорога й доступна), він повинен бути збільшений.

2. Інтенсивне використання перестановок у DES хоча й зручно для апаратних реалізацій, надзвичайно ускладнює програмні реалізації. Найбільш швидкі реалізації DES виконують перестановкам табличним способом. Перегляд таблиці може забезпечити ті ж характеристики "розсіювання", що й властиво перестановкам, і може зробити реалізацію набагато більш гнучкою.

3. S-блоки DES, усього 64 з 4-бітовими елементами, занадто малі. Тепер зі збільшенням пам'яті повинні збільшитися й S-блоки. Більше того, усі 8 S-блоків використовуються одночасно. Хоча це й зручно для

апаратури, для програмної реалізації це – непотрібне обмеженням. Повинні бути реалізовані більший розмір S-блоків і послідовне (а не паралельне) їх використання.

4. Широко відомо, що початкова й заключна перестановки криптографічно безглузді, тому вони повинні бути усунуті.

5. Усі швидкі реалізації DES заздалегідь розраховують ключі для кожного етапу. При даній умові немає сенсу ускладнювати ці обчислення.

6. На відміну від DES, критерії проектування S-блоків повинні бути загальнодоступні.

На рис. 3.9 наведено структуру алгоритму Khafre.

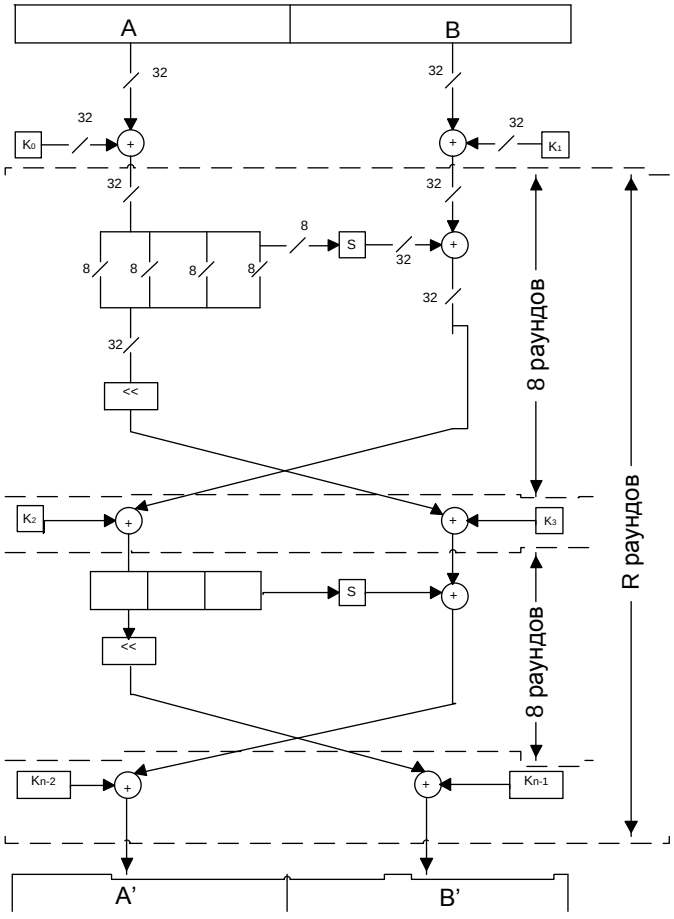


Рис. 3.9. **Структура алгоримту Khafre**

3.3.2. На основі модулярної арифметики

У ISO/IEC 10118-4 визначаються дві геш-функції MASH-1 і MASH-2, засновані на ітеративному зведенні в степінь за модулем. Ці функції є покращеними варіантами функцій, визначених у стандартах CCITT X.509:1988 і ISO/IEC 9594-8:1989, що були визнані недостатньо безпеч-

ними. Унаслідок цього вони були виключені з оновлених версій стандартів ITU-T X.509:1993 і ISO/IEC 9594-8:1995 [111].

Основною ідеєю геш-функцій є використання ітеративної функції як циклової функції, яка використовує модулярну арифметику. Причинами стандартизації і застосування таких геш-функцій є, по-перше, можливість використання існуючих програмних і апаратних засобів модулярної арифметики, застосовуваних у несиметричних криптосистемах, і по-друге, забезпечення необхідного рівня стійкості. Основним недоліком функцій є низька швидкість формування геш-коду.

Геш-функція MASH-1 (Modular Arithmetic Secure Hash-1) використовує RSA подібні модулі N, довжина яких забезпечує необхідну стійкість. Кількість N повинна важко розкладатись, на чому і ґрунтується стійкість алгоритму. Розмір модуля визначає довжину блоків оброблюваного повідомлення, а також розмір геш-коду (наприклад 1025-бітний модуль забезпечує формування 1024-бітного геш-коду). У якості вхідної послідовності використовується двійковий рядок *x* довжиною $0 \le b \le 2^{n/2}$, де *n* розрядність геш-кода.

Алгоритм MASH-1складається з таких етапів [8; 41]:

1. *Системні установки та визначення констант*. Установити RSA-подібний модуль $N = p \times q$ довжиною m біт, де p і q випадково обрані більші прості числа, що зберігаються в секреті. Визначити двійкову довжину n геш-коду як найбільший добуток числа 16 і довжини геш-коду, що задовольняє нерівності $16 \times n < m$.

Як вектор ініціалізації вибрати $H_0 = 0$. Визначити n-бітне ціле кількість як константа $A = 0xf00...00$.

2. *Передобробка*. Доповнити, якщо необхідно, рядок x нульовими бітами, для того, щоб одержати двійковий рядок довжиною $t \times n/2$ для найменшого $t \ge 1$. Розділити доповнений текст на $n/2$-розрядні блоки $x_1,..., x_t$ і додати останній блок x_{t+1}, що містить $n/2$-розрядне подання числа b. Даний етап алгоритму наведено на рис. 3.10.

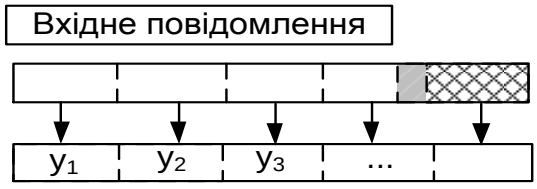


Рис. 3.10. **Етап алгоритма-предобробки повідомлення**

3. *Розширення*. Розширити кожний x_i блок у n-розрядний блок y_i шляхом вставки між 4-розрядними напівбайтами блоку x_i комбінації із чотирьох одиниць (1111). Останній блок y_{t+1} формується аналогічним чином за винятком того, що вставляється комбінація 1010.

4. Циклова функція. Для всіх $1 < i \le t+1$ відобразити два n-розрядних вхідних блоки (H_{i-1}, y_i) в один n-розрядний блок відповідно до виразу:

$$H_i = \left(\left(\left(\left(H_{i-1} \oplus y_i\right) \vee A\right)^2 \bmod N\right) \perp n\right) \oplus H_{i-1},$$

де $\vee$ – операція побітового логічного АБО;

$\oplus$ – додавання по модулю два (XOR);

$\perp$ – збереження молодших n-розрядів m-розрядного результату.

Загальний вид циклової функції наведено на рис. 3.11.

5. Закінчення. У якості геш-коду приймається n-розрядний блок H_{t+1}.

Алгоритм MASH-2 відрізняється від алгоритму MASH-1 тільки показником степені в цикловій функції, що має такий вигляд:

$$H_i = \left(\left(\left(\left(H_{i-1} \oplus y_i\right) \vee A\right)^{2^8+1} \bmod N\right) \perp n\right) \oplus H_{i-1}.$$

Загальний вигляд циклової функції наведено на рис. 3.12.

Пізніша версія стандарту MASH-2 відрізняється від MASH-1 показником степені, що забезпечує велику нелінійність циклової функції, і зниження вірогідності колізій з одного боку, а з іншою викликає зниження швидкості гешування MASH-2. Таким чином використання модулярної арифметики в алгоритмах ключового гешування дозволяє отримати стійкі схеми автентичності, та це призводить до підвищення розрахункової складності криптоперетворень і часу формування геш-коду.

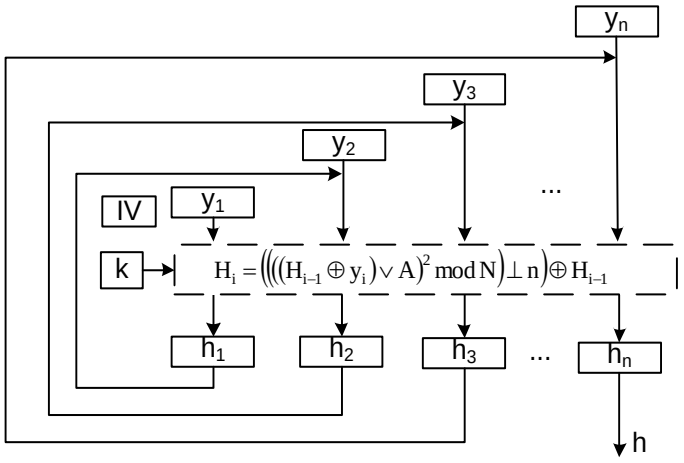


Рис. 3.11. **Загальний вид циклової функції MASH-1**

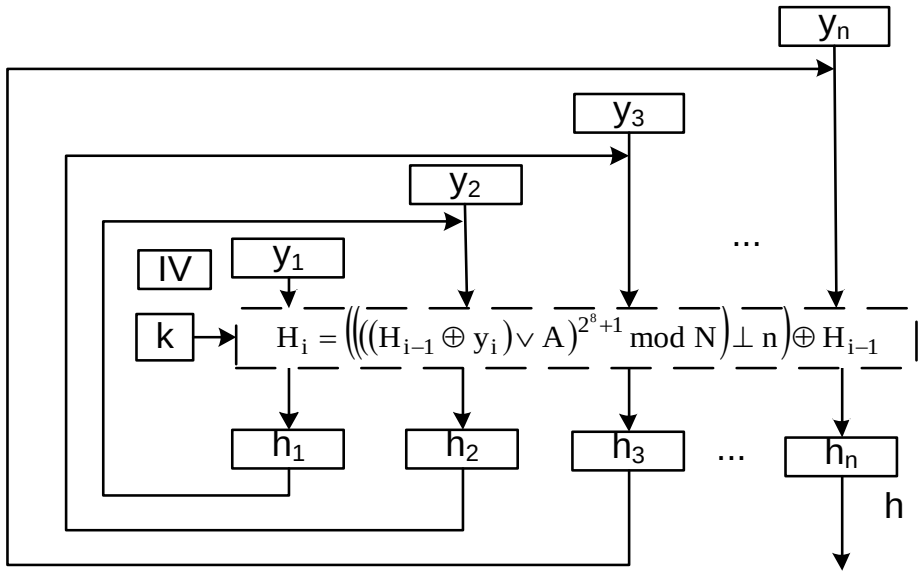


Рис. 3.12. **Загальний вид циклової функції MASH-2**

Потрібно оцінити показники часу формування геш-коду при невеликих розмірах вхідних повідомлень від 100 біт до 2 Кбіт. Варто сформувати геш-код для кожного повідомлення, для цього довжину ключів вибирають 64, 128, 256, 512, 1024 біт. На рис. 3.13 наведені залежності швидкості формування геш-коду від довжини ключа при розмірі повідомлення до 2 Кбіт, на рис. 3.14 залежності при використанні розміру вхідної послідовності до 40 Кбайт для алгоритму MASH-1.

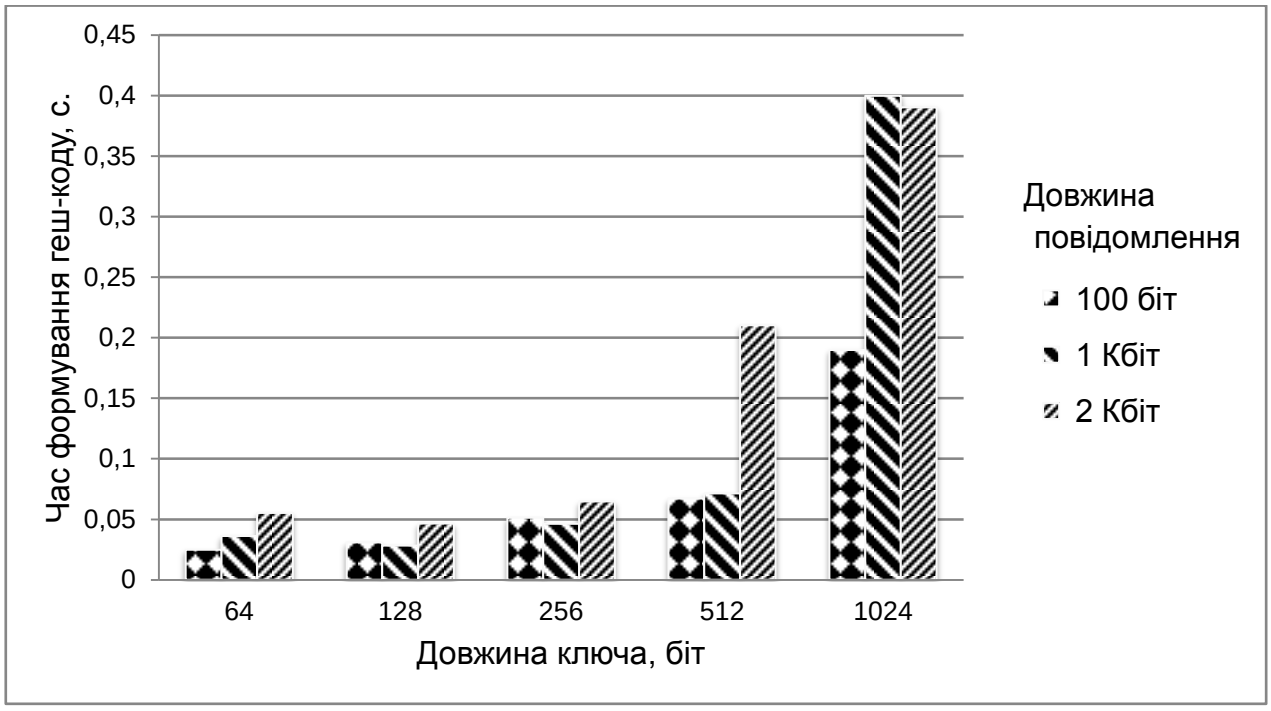


Рис. 3.13. **Залежність швидкості формування геш-коду від довжини ключа при довжині повідомлення від 100 біт до 2 Кбіт**

На рис. 3.15. наведені залежності швидкості формування геш-коду від довжини ключа при розмірі повідомлення до 2 Кбіт, на рис. 3.16 залежності при використанні розміру вхідної послідовності до 40 Кбайт для алгоритму MASH-2.

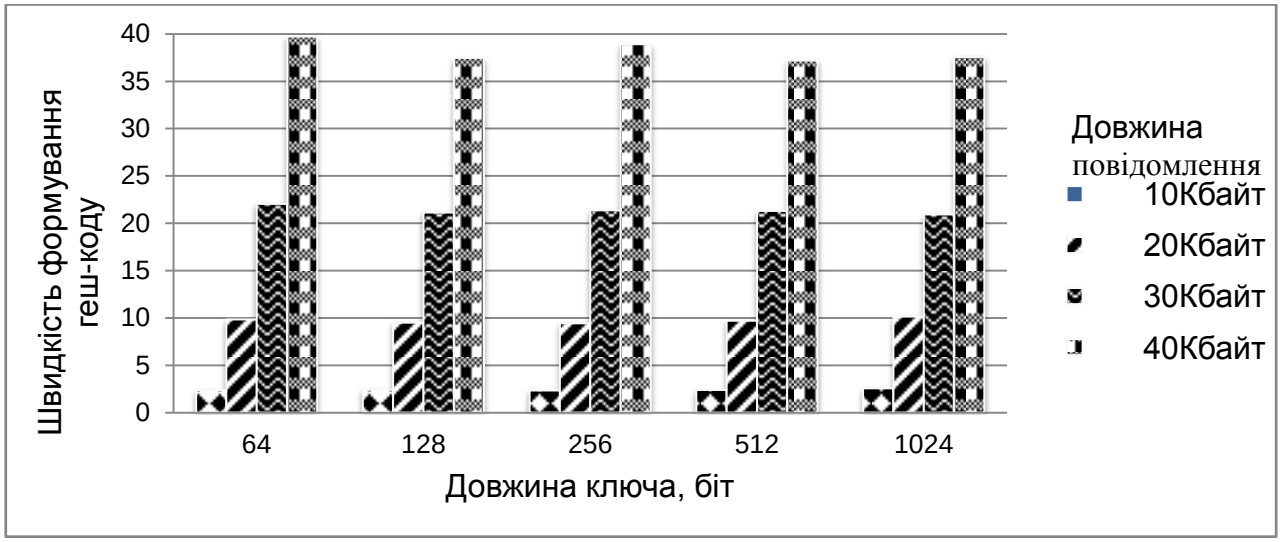


Рис. 3.14. **Залежність швидкості формування геш-коду від довжини ключа при довжині повідомлення від 10 – 40 Кбайт**

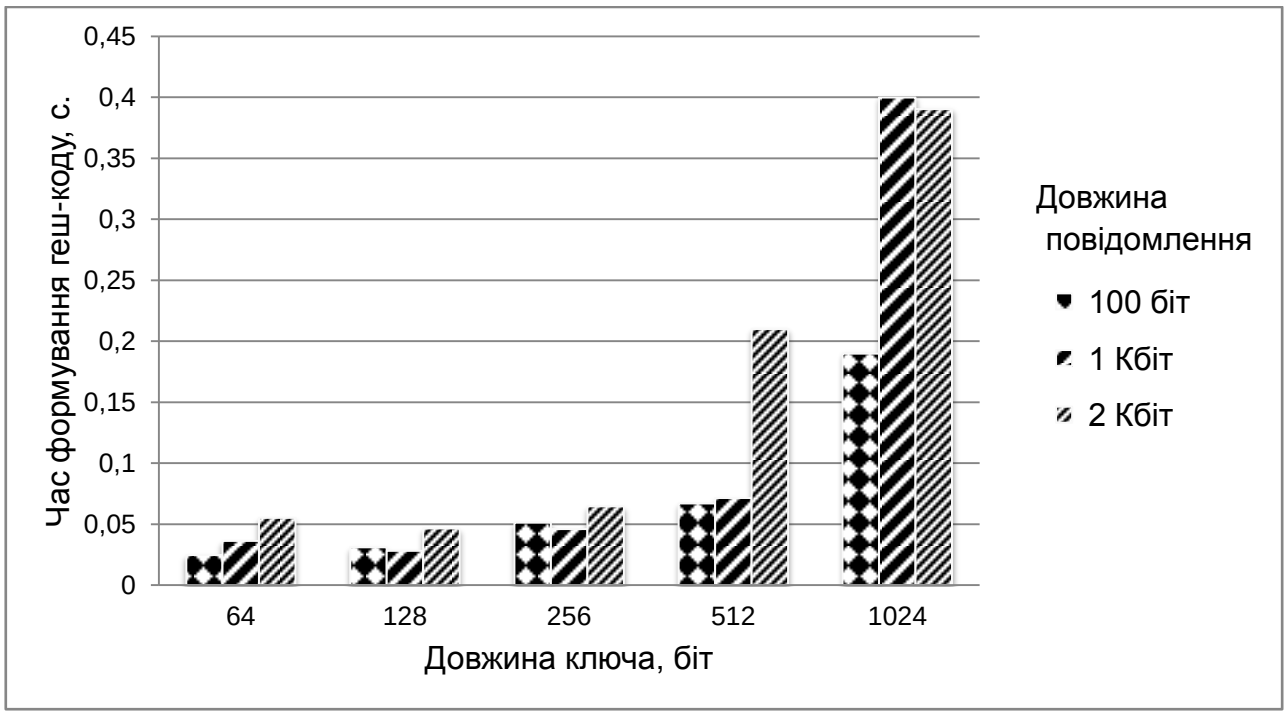


Рис. 3.15. **Залежність швидкості формування геш-коду від довжини ключа при довжині повідомлення від 100 біт до 2 Кбіт**

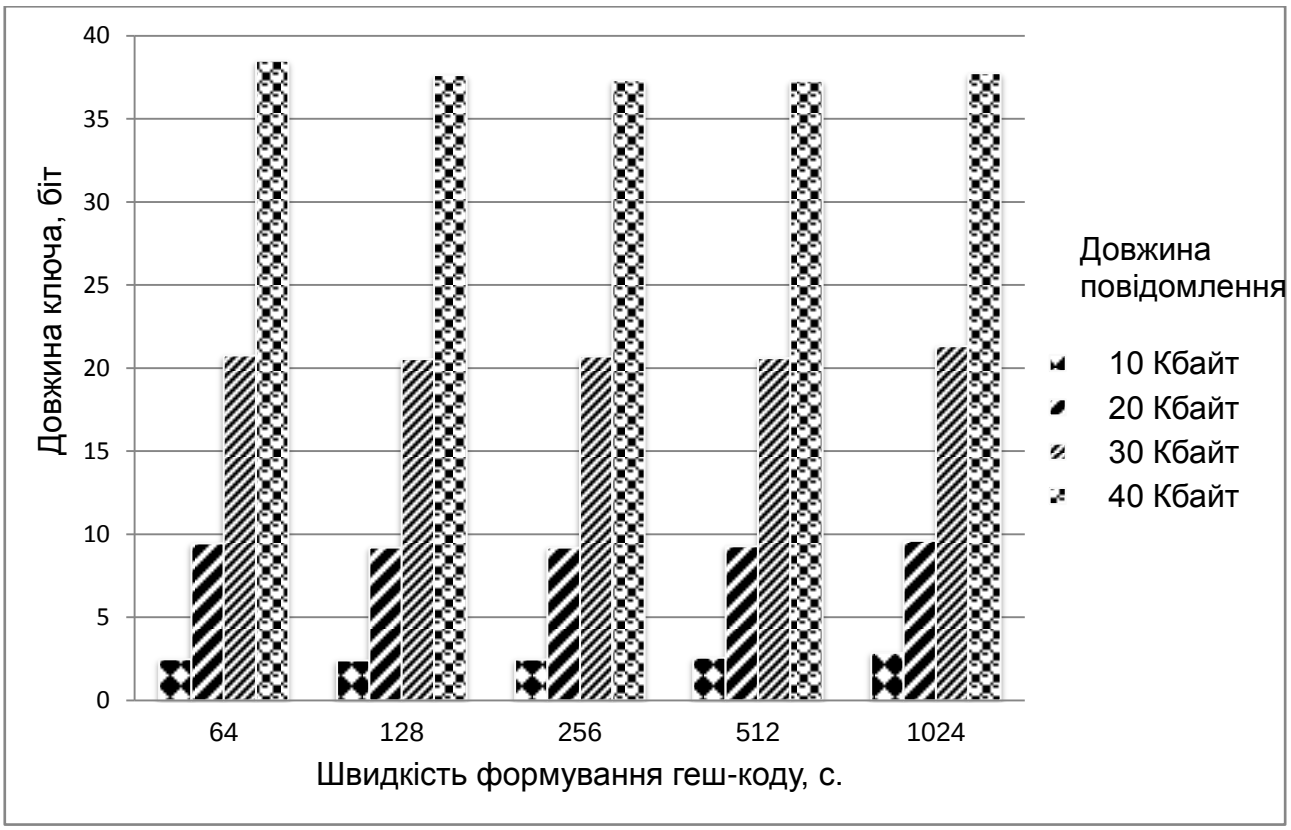


Рис. 3.16. **Залежність швидкості формування геш-коду від довжини ключа при довжині повідомлення від 10 – 40 Кбайт**

Зображені на рис. 3.13 − 3.16 залежності свідчать, що при збільшенні довжин ключів і розмірності повідомлень, час формування геш-коду збільшується.

Потрібно провести дослідження співвідношення загального часу формування геш-коду до часу генерації простих чисел. Зафіксуємо довжину повідомлення в 10 Кбайт, проведемо вимір часу формування простих чисел і модулярних перетворень, результат вимірів поданий на рис. 3.17. Потрібно зафіксувати довжину повідомлення в 40 Кбайт, провести вимір часу формування простих чисел і модулярних перетворень, результат вимірів зображений нарис. 3.18.

Аналіз співвідношення загального часу формування геш-коду до часу генерації простих чисел показує, що процедура пошуку простих чисел необхідної довжини вимагає більших витрат часу. При невеликій довжині послідовності, яка гешується, швидкість формування геш-коду знижується, що призводить до зниження ефективності алгоритму.

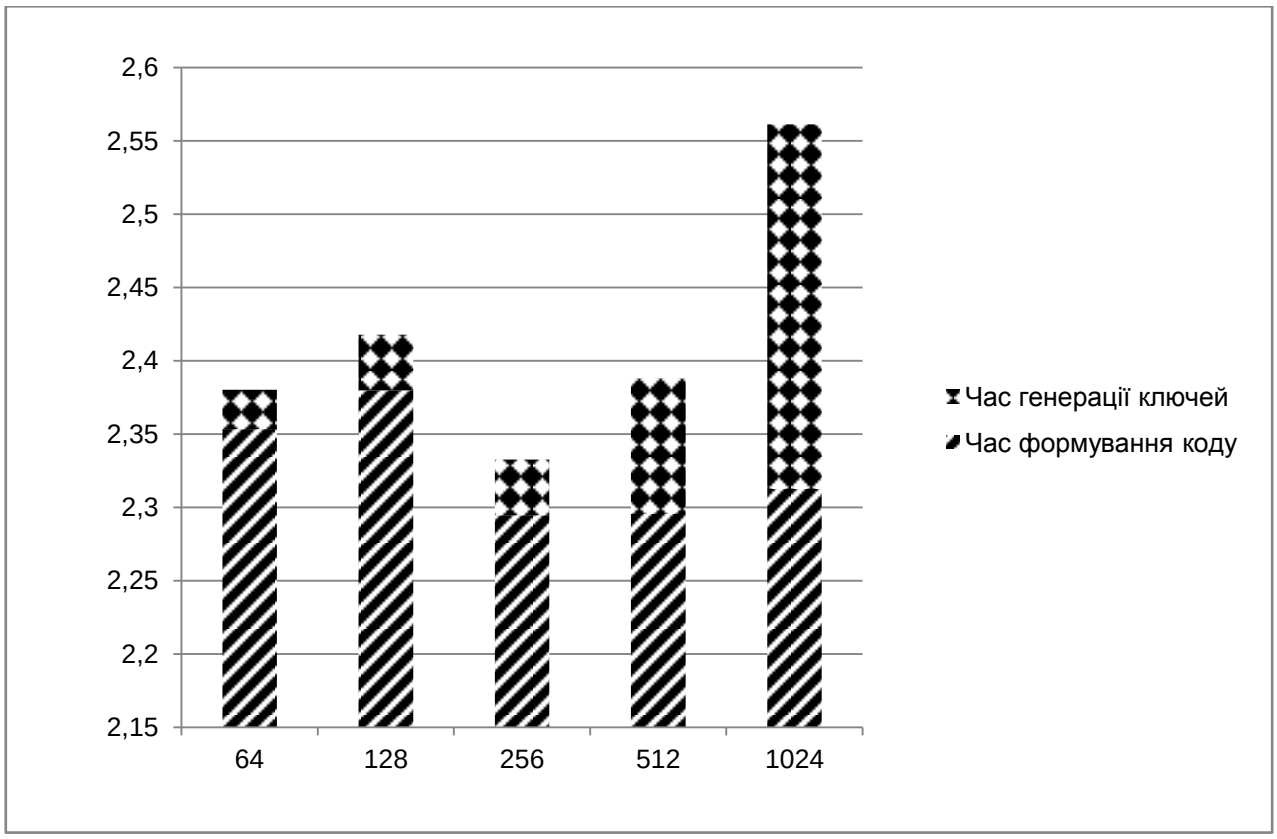


Рис. 3.17. **Співвідношення загального часу формування геш-коду до часу генерації простих чисел при довжині повідомлення 10 Кбайт в алгоритмі MASH-1**

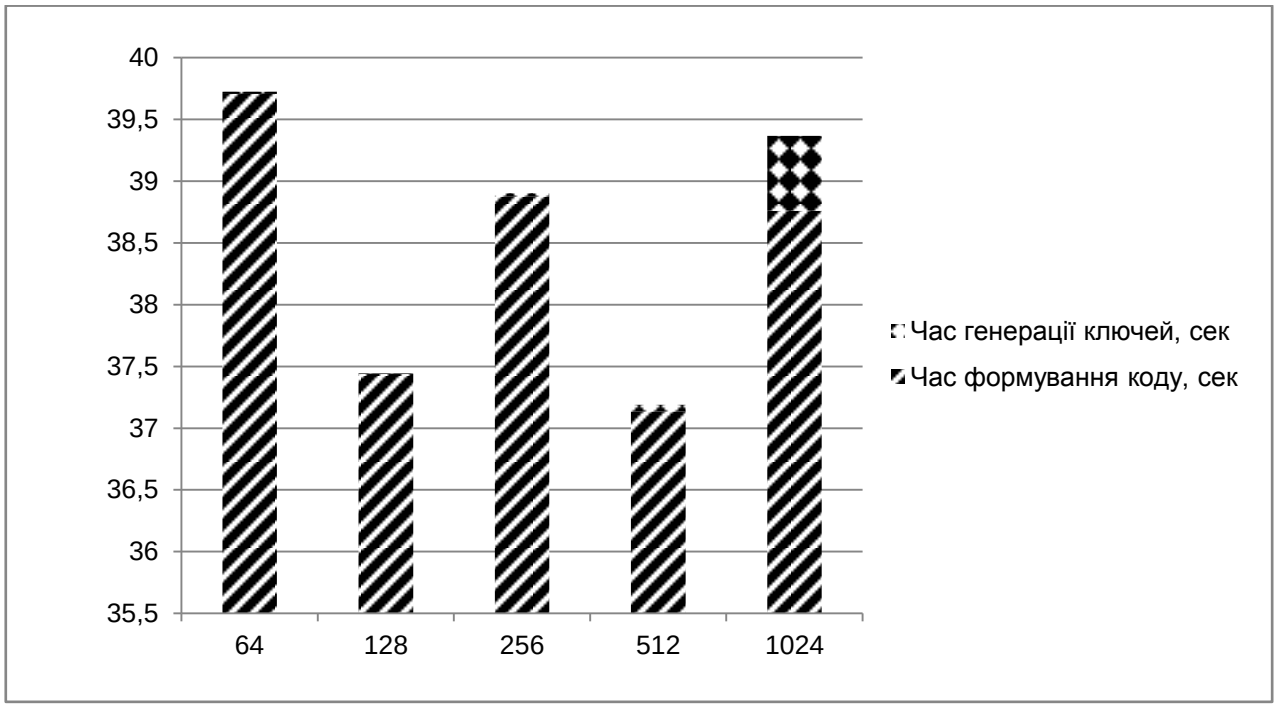


Рис. 3.18. **Співвідношення загального часу формування геш-коду до часу генерації простих чисел при довжині повідомлення 40 Кбайт в алгоритмі MASH-1**

3.3.3. Спеціалізовані геш-функції

Спеціалізовані геш-функції (Customised hash functions або dedicated hash functions) спеціально розроблені тільки для цілей гешування й оптимізовані для виконання даної задачі. Третя частина стандарту ISO/IEC 10118-3 визначає три спеціалізовані геш-функції, а саме функції RIPEMD-128, RIPEMD-160 і SHA-1. Дані геш-функції засновані на використанні принципів побудови, закладених у геш-функції сімейства MD*x* (MD2, MD4, MD5), які спеціально розроблялися для реалізації на 32-розрядних ЕОМ [79]. Алгоритм MD4 був запропонований Р. Райвестом у 1990 р., а в 1991 той же автор запропонував модифіковану версію алгоритму – MD5. У даний час геш-функції MD4, MD5 є найбільш розповсюдженими в практичних додатках геш-функціями. Однак деякі їх недоліки не дозволили стандартизувати їх на міжнародному рівні.

Європейський консорціум RIPE, спираючись на свої дослідження властивостей цих алгоритмів, запропонував посилену версію MD4, що одержала назву RIPEMD. Геш-функція RIPEMD по суті складається з двох паралельно працюючих і модифікованих функцій MD4 (тобто функція має дві лінії). Іншою альтернативою алгоритмам MD*x* є алгоритм SHA-1, розроблений спільно Агентством національної безпеки США і NIST і прийнятий як американський національний стандарт (FIPS 180-1).

MD4

MD4 (Message Digest 4) – геш-функція, розроблена професором Массачусетського університету Рональдом Рівестом у 1990 р., і вперше описана в RFC 1186. Для довільного вхідного повідомлення функція генерує 128-розрядне геш-значення, зване дайджестом повідомлення. Цей алгоритм використовується в протоколі автентифікації MS-CHAP, розробленому корпорацією Майкрософт для виконання процедур перевірки достовірності віддалених робочих станцій Windows [90; 91].

Гешування з MD4 складається з 48 операцій, згрупованих у 3 раунди по 16 операцій. F – нелінійна функція; в кожному раунді функція змінюється. Mi означає 32-бітний блок вхідного повідомлення, а K_i – 32-бітова константа, різна для кожної операції. На рис. 3.19 наведено структуру раунда MD4.

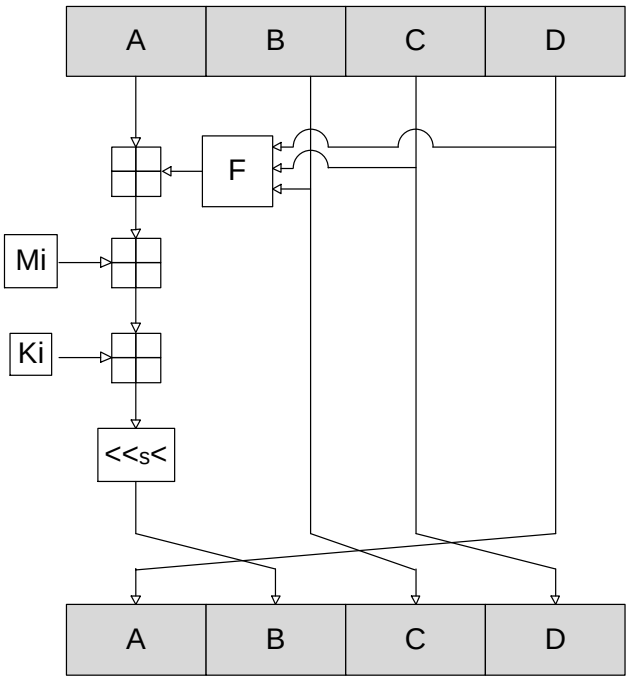


Рис. 3.19. **Структура раунда MD4**

Алгоритм MD4.

Передбачається, що на вхід подано повідомлення, що складається з біт, геш якого потрібно обчислити. Тут – довільне невід'ємне ціле число, воно може бути нулем, не повинно бути кратним восьми, і може бути як завгодно великим. Запишемо повідомлення побітово, у вигляді: $m_0 m_1 ... m_{b-1}$.

Нижче наведено 5 кроків, які використовуються для обчислення гешу повідомлення.

Крок 1. Додавання відсутніх бітів.

Повідомлення розширюється так, щоб його довжина в бітах по модулю 512 дорівнювала 448. Таким чином, у результаті розширення, повідомленням бракує 64 біта до довжини, кратної 512 бітам. Розширення проводиться завжди, навіть якщо повідомлення спочатку має потрібну довжину.

Розширення проводиться таким чином: один біт, що дорівнює 1, додається до повідомлення, а потім додаються біти, рівні 0, до тих пір, поки довжина повідомлення не стане рівної 448 по модулю 512. У результаті, до повідомлення додається як мінімум 1 біт, і як максимум 512.

Крок 2. Додавання довжини повідомлення.

64-бітове уявлення b (довжини повідомлення перед додаванням набивальних бітів) додається до результату попереднього кроку.

У малоймовірному випадку, коли b більше, ніж 2^{64}, використовуються тільки 64 молодших біта. Ці біти додаються у вигляді двох 32-бітових слів, і першим додається слово, що містить молодші розряди.

На цьому етапі (після додавання бітів і довжини повідомлення) отримуються повідомлення довжиною кратної 512 бітам. Це еквівалентно тому, що це повідомлення має довжину, кратну 16-ти 32-бітовим словами. Кожне 32-бітове слово містить чотири 8-бітних, але йдуть вони не підряд, а навпаки (наприклад, з восьми 8-бітових слів (abcdefgh) отримуються два 32-бітових слова (dcba hgfe)). Нехай $M[0...N-1]$ означає масив слів отриманого повідомлення (тут N кратно 16).

Крок 3. Ініціалізація MD-буфера.

Для обчислення гешу повідомлення використовується буфер, що складається з 4 слів (32-бітних регістрів): (A,B,C,D). Ці регістри ініціалізувалися такими шістнадцятковим числами (молодші байти спочатку):

word A: 67 45 23 01
word B: ef cd ab 89
word C: 98 ba dc fe
word D: 10 32 54 76

Крок 4. Обробка повідомлення блоками по 16 слів.

Для початку слід визначити три допоміжні функції, кожна з яких отримує на вхід три 32-бітових слова, і по них обчислює одне 32-бітове слово.

$$F(X,Y,Z) = XY$$
$$F(X,Y,Z) = XY \vee XZ$$
$$G(X,Y,Z) = XY \vee XZ \vee YZ$$
$$H(X,Y,Z) = XYZ$$

На кожну бітову позицію F діє як умовний вираз: якщо X, то Y; інакше Z. Функція F могла б бути визначена з використанням $\oplus$ замість $\vee$, оскільки XY і $\overline{XZ}$ не можуть рівнятися 1 одночасно. G діє на кожну бітову позицію як функція максимального значення: якщо, щонайменше, в двох словах з X,Y,Z відповідні біти рівні 1, то G видасть 1 в цьому біті, а інакше G видасть біт, що дорівнює 0. Потрібно визначити, що якщо

біти X, Y і Z статистично незалежні, то біти $F(X, Y, Z)$ і $G(X, Y, Z)$ будуть також статистично незалежні. Функція H реалізує побітовий XOR, вона володіє такою же властивістю, як F і G.

Крок 5. Формування гешу.

Результат (геш-функція) виходить як ABCD. Тобто, виписується 128 біт, починаючи з молодшого біта A, і закінчуючи старшим бітом D.

Реалізація алгоритму на мові C міститься в RFC 1320.

Порівняння з MD5.

MD4 використовує три цикли з 16 кроків кожен, у той час як MD5 використовує чотири цикли з 16 кроків кожен.

У MD4 додаткова константа в першому циклі не застосовується. Аналогічна додаткова константа використовується для кожного з кроків у другому циклі. Інша додаткова константа використовується для кожного з кроків у третьому циклі. У MD5 різні додаткові константи, $T[i]$, застосовуються для кожного з 64 кроків.

MD5 використовує чотири елементарні логічні функції, по одній на кожному циклі, у порівнянні з трьома в MD4, по одній на кожному циклі.

У MD5 на кожному кроці поточний результат складається з результатом попереднього кроку. Наприклад, результатом першого кроку є змінене слово A. Результат другого кроку зберігається в D і утворюється додаванням A до циклічно зрушеному ліворуч на певне число біт результату елементарної функції. Аналогічно, результат третього кроку зберігається в C і утворюється додаванням D до циклічно зрушеному ліворуч результату елементарної функції. MD4 – це останнє додавання не включає.

Безпека.

Рівень безпеки, який закладався в MD4, був розрахований на створення досить стійких гібридних систем електронного цифрового підпису, заснованих на MD4 і криптосистеми з відкритим ключем. Рональд Рівест вважав, що алгоритм гешування MD4 можна використовувати і для систем, які потребують сильної криптостійкості. Але в той же час він відзначав, що MD4 створювався передусім як дуже швидкий алгоритм гешування, тому він може бути поганий у плані

криптостійкості. Як показали дослідження, він був правий, і для додатків, де важлива насамперед крипостійкість, почав використовуватися алгоритм MD5.

Вразливості.

Вразливості в MD4 були продемонстровані у статті Берта ден Боєра і Антона Босселаріса в 1991 році. Перша колізія була знайдена Гансом Доббертіном у 1996 р. [64].

MD2

MD2 (The MD2 Message Digest Algorithm) – геш-функція, розроблена Рональдом Рівестом (RSA Laboratories) в 1989 р., і описана в RFC 1319. Розмір геша – 128 біт. Розмір блоку вхідних даних – 512 біт [59].

Алгоритм MD2.

Передбачається, що на вхід подано повідомлення, що складається з *b* байт, геш якого потрібно обчислити. Тут *b* – довільне невід'ємне ціле число, воно може бути нулем або яким завгодно великим. Потрібно записати повідомлення побайтово, у вигляді: $m_0 m_1 \ldots mb-1.$

Далі наведено 5 кроків, які використовуються для обчислення гешу повідомлення.

Крок 1. Додавання відсутніх біт.

Повідомлення розширюється так, щоб його довжина в байтах за модулем 16 дорівнювала 0. Таким чином, у результаті розширення довжина повідомлення стає кратною 16 байтам. Розширення проводиться завжди, навіть якщо повідомлення спочатку має потрібну довжину.

Розширення проводиться таким чином: і байт, рівних і, додається до повідомлення, так щоб його довжина в байтах стала рівною 0 по модулю 16. У результаті, до повідомлення додається як мінімум 1 байт, і як максимум 16.

На цьому етапі (після додавання байт) повідомлення має довжину в точності кратну 16 байтам. Нехай $M[0...N-1]$ означає байти отриманого повідомлення (N кратно 16).

Крок 2. Додавання контрольної суми.

16-байтним контрольна сума повідомлення додається до результату попереднього кроку.

Цей крок використовує 256-байтную "випадкову" перестановчу матрицю, що складається з цифр числа пі:

PI_SUBST [256] = {41, 46, 67, 201, 162, 216, 124, 1, 61, 54, 84, 161, 236, 240, 6, 19,98, 167, 5, 243, 192, 199, 115, 140, 152, 147, 43, 217, 188, 76, 130, 202,30, 155, 87, 60, 253, 212, 224, 22, 103, 66, 111, 24, 138, 23, 229, 18,190, 78, 196, 214, 218, 158, 222, 73, 160, 251, 245, 142, 187, 47, 238, 122,169, 104, 121, 145, 21, 178, 7, 63, 148, 194, 16, 137, 11, 34, 95, 33,128, 127, 93, 154, 90, 144, 50, 39, 53, 62, 204, 231, 191, 247, 151, 3, 255, 25, 48, 179, 72, 165, 181, 209, 215, 94, 146, 42, 172, 86, 170, 198, 79, 184, 56, 210, 150, 164, 125, 182, 118, 252, 107, 226, 156, 116, 4, 241, 69, 157, 112, 89, 100, 113, 135, 32, 134, 91, 207, 101, 230, 45, 168, 2, 27, 96, 37, 173, 174, 176, 185, 246, 28, 70, 97, 105, 52, 64, 126, 15, 85, 71, 163, 35, 221, 81, 175, 58, 195, 92, 249, 206, 186, 197, 234, 38, 44, 83, 13, 110, 133, 40, 132, 9, 211, 223, 205, 244, 65, 129, 77, 82, 106, 220, 55, 200, 108, 193, 171, 250, 36, 225, 123, 8, 12, 189, 177, 74, 120, 136, 149, 139, 227, 99, 232, 109, 233, 203, 213, 254, 59, 0, 29, 57, 242, 239, 183, 14, 102, 88, 208, 228, 166, 119, 114, 248, 235, 117, 75, 10, 49, 68, 80, 180, 143, 237, 31, 26, 219, 153, 141, 51, 159, 17, 131, 20}.

Крок 5. Формування гешу.

Геш обчислюється як результат $X[0...15]$, на початку йде байт $X[0]$, а наприкінці $X[15]$.

На цьому завершується опис алгоритму MD2. У RFC 1319 можна знайти реалізацію алгоритму на мові С.

Безпека.

Роже і Шаво в 1997 р. опублікували приклад колізій для MD2, хоча і не змогли представити алгоритм знаходження інших колізій.

У 2004 р. було показано, що MD2 більш схильний до атаки на знаходження колізій зі складністю, еквівалентною 2^{104} операцій гешування (Міллер, 2004). Міллер заявив: "MD2 не може більше розглядатися, як криптостійкий алгоритм гешування".

MD5

MD5 – це поліпшена версія MD4 [60; 64]. Хоча вона складніше MD4, їх схеми схожі, і результатом MD5 також є 128-бітове значення.

На рис. 3.20 наведено структуру раунда MD5.

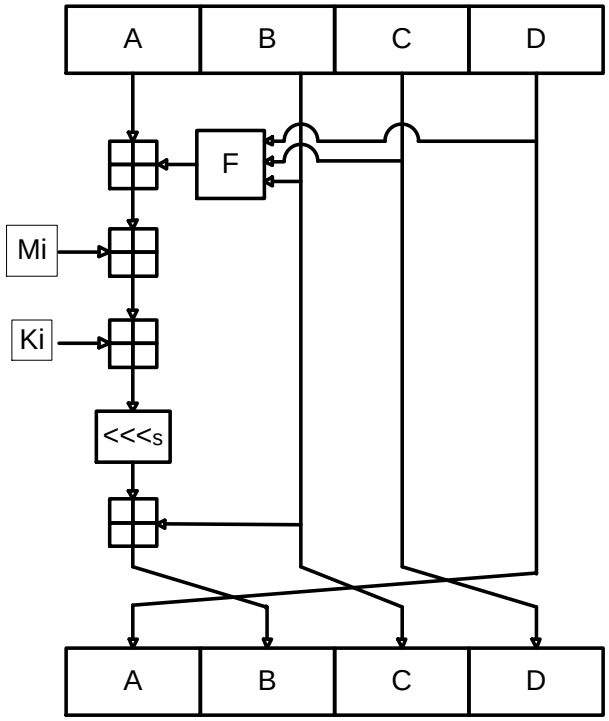


Рис. 3.20. **Структура раунда MD5**

Потрібно розглянути алгоритм отримання дайджеста повідомлення MD5 (RFC 1321), розроблений Роном Рівестом з MIT.

Логіка виконання MD5.

Алгоритм отримує на вході повідомлення довільної довжини і створює в якості виходу дайджест повідомлення довжиною 128 біт. Логіку виконання наведено на рис. 3.21.

Алгоритм складається з таких кроків [60; 64]:

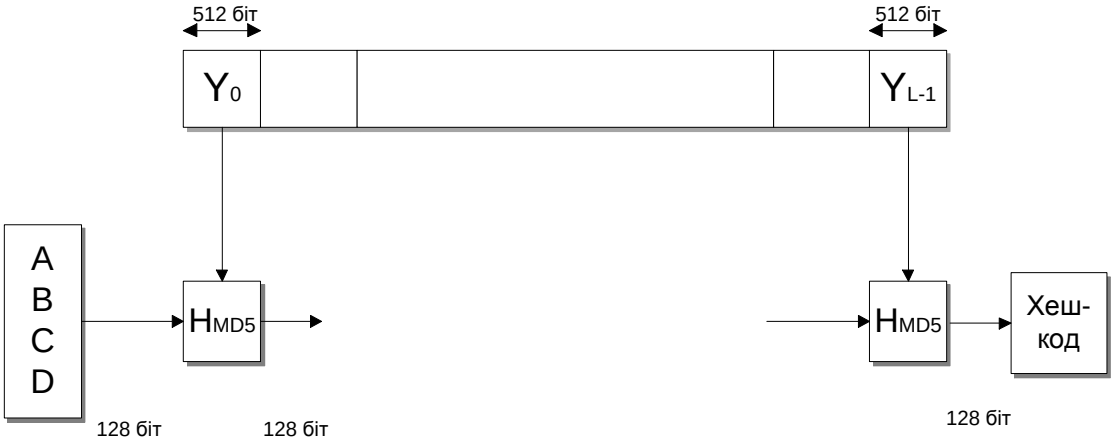


Рис. 3.21. **Логіка виконання MD5**

Крок 1: додавання відсутніх бітів.

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

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

Крок 2: додавання довжини.

64-бітове уявлення довжини вихідного (до додавання) повідомлення в 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}.

У результаті перших двох кроків створюється повідомлення, довжина якого кратна 512 бітам. Це розширене повідомлення, що наведено на рис. 3.22, подається як послідовність 512-бітних блоків $Y_0, Y_1, ..., YL-1$, при цьому загальна довжина розширеного повідомлення дорівнює L × 512 бітам. Таким чином, довжина отриманого розширеного повідомлення кратна шістнадцяти 32-бітовим словам.

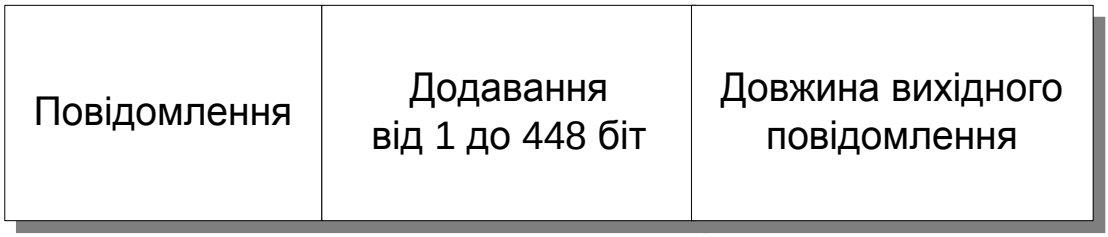


Рис. 3.22. **Структура розширеного повідомлення**

Крок 3: ініціалізація MD-буфера.

Використовується 128-бітний буфер для зберігання проміжних і остаточних результатів геш-функції. Буфер може бути представлений як чотири 32-бітних регістра (A, B, C, D). Ці регістри ініціалізувалися такими шістнадцятковим 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 HMD5. 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. На рис. 3.23 наведено послідовність обробки 512-бітного блоку.

Кожен цикл приймає на вході поточний 512-бітний блок Y_q, що обробляється в даний момент, і 128-бітове значення буферу ABCD, яке є проміжним значенням дайджесту, і змінює вміст цього буфера. Кожен цикл також використовує четверту частину 64-елементної таблиці $T[1...64]$, побудованої на основі функції sin і-ий елемент T, що позначається $T[i]$, має значення, рівне цілої частини від $2^{32} \times abc(sin(i))$, і заданий у радіанах. Оскільки abs(sin(i)) є числом між 0 і 1, кожен елемент T є цілим, яке може бути подано 32 бітами. Таблиця забезпечує "випадковий" набір 32-бітних значень, які повинні ліквідувати будь-яку регулярність у вхідних даних. Для отримання MD_{q+1} вихід чотирьох циклів додається за модулем 2^{32} з MD_q. Додавання відбувається незалежно для кожного з чотирьох слів у буфері [28; 33].

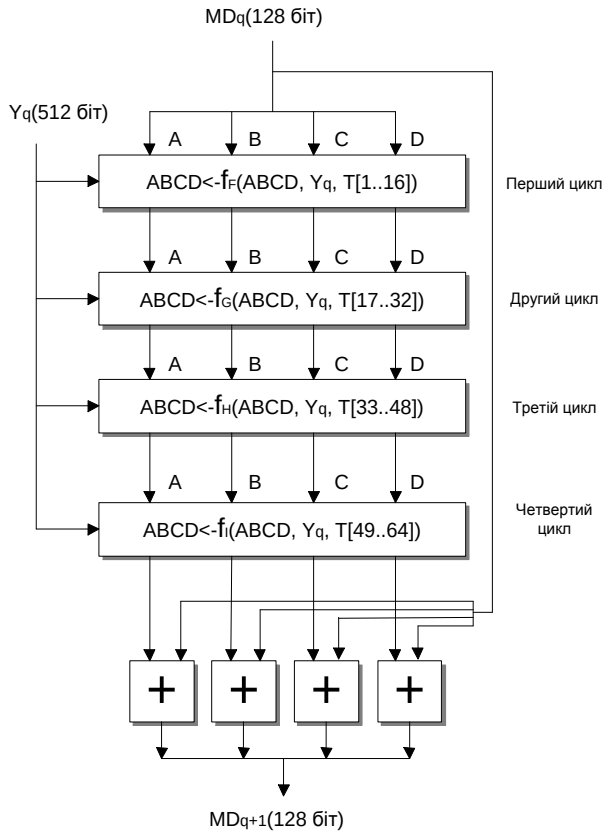


Рис. 3.23. **Обробка чергового 512-бітного блоку**

Крок 5: вихід.

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

Потрібно розглянути більш детально логіку кожного з чотирьох циклів виконання одного 512-бітного блоку. Кожен цикл складається з 16 кроків, що оперують з буфером ABCD.

Логіку виконання окремого кроку наведено на рис. 3.24. Кожен крок можна зобразити у вигляді:

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

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

f – одна з елементарних функцій f_F, f_G, f_H, f_I;

CLS_s – циклічний зсув ліворуч на s біт 32-бітного аргументу;

$X[k] - M[q*16+k] - k - e$ – 32-бітове слово в q-му 512 блоці повідомлення;

$T[i] - i - e$ – 32-бітове слово в матриці T;

$+$ – додавання за модулю 2^{32}.

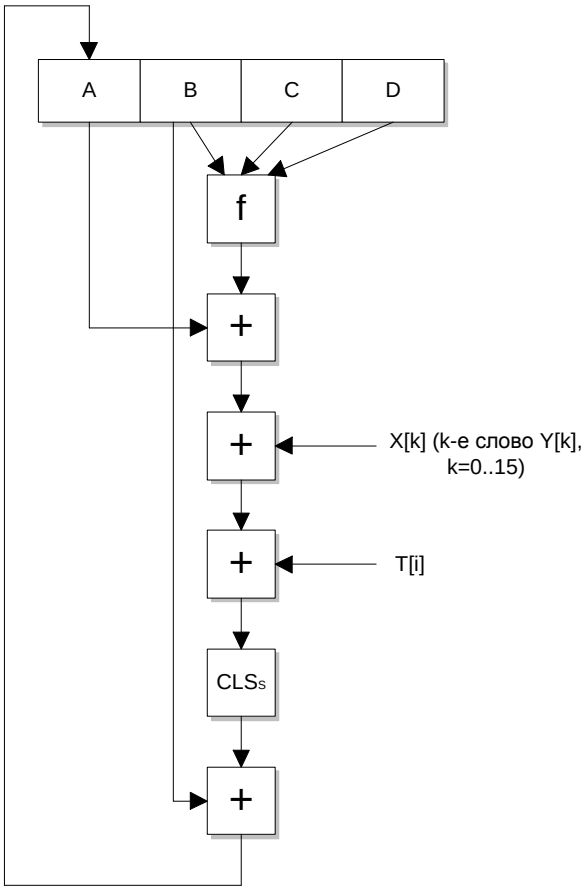


Рис. 3.24. **Логіка виконання окремого кроку**

На кожному з чотирьох циклів алгоритму використовується одна з чотирьох елементарних логічних функцій. Кожна елементарна функція отримує три 32-бітових слова на вході і на виході створює одне 32-бітове слово. Кожна функція є безліччю побітових логічних операцій, тобто n-ий біт виходу є функцією від n-ого біта трьох входів. Елементарні функції такі [61]:

$$f_F = (B \wedge C) \vee (\overline{B} \wedge D)$$

$$f_G = (B \wedge D) \vee (C \wedge \overline{D})$$

$$f_H = B \oplus C \oplus D$$

$$f_I = C \oplus (B \wedge \overline{D})$$

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 схемою, показаною на рис. 3.24, 64 рази. Якщо уявити вхідний 512-бітний блок у вигляді шістнадцяти 32-бітових 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-бітних слів, використовується тільки один разів. Після кожного кроку циклу відбувається циклічний зсув ліворуч чотирьох слів A, B, C I D. На кожному кроці змінюється тільки одне з чотирьох слів буфера ABCD. Отже, кожне слово буфера змінюється 16 разів, і потім 17-ий раз у кінці для отримання остаточного виходу даного блоку. Додавання алгоритмом можна виконувати таким чином:

$$MD_0 = IV$$

$$MD_{q+1} = MD_q + f_I\left[Y_q, f_H\left[Y_q, f_G\left[Y_q, f_F\left[Y_q, MD_q\right]\right]\right]\right]$$

$$MD = MD_{L-1},$$

де IV – початкове значення буфера ABCD, визначене на кроці 3, Y_q – q-ий 512-бітний блок повідомлення, L – кількість блоків у повідомленні (включаючи поля доповнення та довжини).

Геш-функція SHA-1

Алгоритм SHA-1 розроблений у 1992 р. і формує по вхідному двійковому рядку довільної довжини 160-бітний геш-код. Алгоритм має циклічний характер і у своїх циклах використовує тикий набір нелінійних функцій:

$$f(u,v,w) = (u \wedge v) \vee (\bar{u} \wedge w);$$

$$g(u,v,w) = (u \wedge v) \vee (u \wedge w) \vee (v \wedge w);$$

$$h(u,v,w) = u \oplus v \oplus w;$$

де u,v,w – 32-бітні змінні (слова);

$u \wedge v$ – логічне "І" по розрядах;

$u \vee v$ – логічне "АБО" по розрядах;

$\bar{u}$ – логічне "доповнення" по розрядах.

$\oplus$ – додавання за модулем 2.

Далі, при описі роботи алгоритму символ "$+$" позначає додавання за модулем 2^{32}, "<<<*s*" – циклічний зсув ліворуч на *s* розрядів.

SHA-1 реалізує геш-функцію, побудовану на ідеї функції стиснення.

Входами функції стиснення є блок повідомлення довжиною 512 біт і вихід попереднього блоку повідомлення. Вихід є значенням усіх геш-блоків до цього моменту. Іншими словами геш блоку M_i дорівнює $h_i = f(M_i, h_{i-1})$. Геш-значенням усього повідомлення є вихід останнього блоку [41].

Алгоритм SHA-1.

Вхід. Двійковий рядок *x* довжиною $b \geq 0$.

Вихід. 160-бітний геш-код за рядком *x*.

1. *Ініціалізація.* Ініціалізувати п’ять векторів ініціалізації:

h_1 = 67452301x; h_2 = efcdab89x; h_3 = 98badcfex; h_4 = 10325476x;

h_5 = c3d2e1f0x.

Задати чотири додаткові константи.

y_1 = 5a827999x; y_2 = 6ed9eba1x; y_3 = 8f1bbcdcx; y_4 = ca62c1d6x.

2. *Передобробка.* Доповнити рядок x так, щоб його довжина була кратна числу 512. Для цього додати в кінець рядка одиницю, а потім стільки нульових біт, скільки необхідно для одержання рядка довжиною на 64 біта коротше довжини кратної 512. Нарешті, додати останні два

32-х розрядні слова, що містять двійкове представлення числа b. Нехай m – кількість 512-бітних блоків в отриманому доповненому рядку. Таким чином, форматований вхід буде складатися з $16m$ 32-х розрядних слів $x_0 x_1 \ldots x_{16m-1}$. Ініціалізувати вектор змінних зчеплення:

$$(H_1, H_2, H_3, H_4, H_5) = (h_1, h_2, h_3, h_4, h_5)$$

3. *Обробка.* Для всіх i від 0 до m-1 кожний i-й блок із шістнадцяти 32-х бітних слів перетворити у вісімдесят 32-х бітних слів і записати в тимчасовий масив за таким алгоритмом:

$$X_i = x_{16i+j}, \quad 0 \le j \le 15;$$

$$X_i = ((X_{j-3} \oplus X_{j-8} \oplus X_{j-14} \oplus X_{J-16}) <<< 1).$$

Слід зазначити, що у початковій версії, що називалася SHA, алгоритм не містив лівого зрушення на один біт. Ця зміна введена для посилення геш-функції. У результаті отриманий алгоритм SHA-1.

Ініціалізувати робочі змінні $(A, B, C, D, E) = (H_1, H_2, H_3, H_4, H_5)$.

Обчислити для всіх i від 0 до $m - 1$:

Для $j = 0$ до 19

$$t = ((A <<< 5) + f(B, C, D) + E + X_j + y_1);$$

$$(A, B, C, D, E) = (t, A, B <<< 30, C, D).$$

Для j = 20 до 39

$$t = ((A <<< 5) + h(B, C, D) + E + X_j + y_2);$$

$$(A, B, C, D, E) = (t, A, B <<< 30, C, D).$$

Для j = 40 до 59

$$t = ((A <<< 5) + g(B, C, D) + E + X_j + y_3);$$

$$(A, B, C, D, E) = (t, A, B <<< 30, C, D).$$

Для j = 60 до 79

$$t = ((A <<< 5) + h(B, C, D) + E + X_j + y_4);$$

$$(A, B, C, D, E) = (t, A, B <<< 30, C, D).$$

Обновити вектор змінних зчеплення:

$$(H_1, H_2, H_3, H_4, H_5) = (H_1 + A, H_2 + B, H_3 + C, H_4 + D, H_5 + E).$$

4. *Завершення*. Як геш-код прийняти величину: $H_1 \,||\, H_2 \,||\, H_3 \,||\, H_4 \,||\, H_5$.

У порівнянні з 128-розрядними геш-функціями, 160-розрядний геш-код, вироблюваний SHA-1, забезпечує більшу стійкість до силових атак. Геш-функції SHA-1 і RIPEMD-160 за стійкістю приблизно рівні й обидві перевершують MD5. Розширення блоку вхідного повідомлення введено з метою забезпечення більшої відмінності між вхідними блоками. Надмірність, що вводиться, сприяє підвищенню стійкості геш-функції. На рис. 3.25 наведена схема однієї ітерації алгоритму SHA-1.

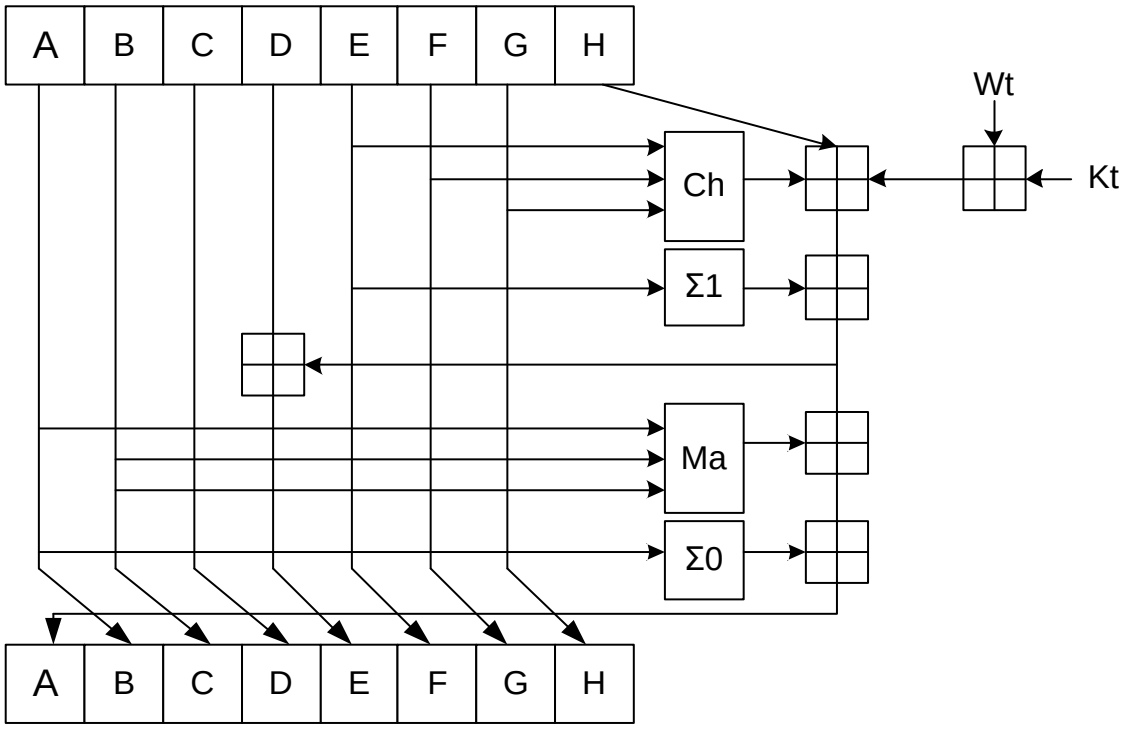


Рис. 3.25. **Одна ітерація алгоритму SHA1**

SHA-2

Secure Hash Algorithm 2 – безпечний алгоритм гешування, версія 2 – збірна назва односпрямованих геш-функцій SHA-224, SHA-256, SHA-384 і SHA-512. Геш-функції призначені для створення "відбитків" або "дайджестів" повідомлень довільної бітової довжини. Застосовуються в різних додатках або компонентах, пов’язаних із захистом інформації.

Геш-функції SHA-2 розроблені Агентством національної безпеки США й опубліковані Національним інститутом стандартів і технологій у федеральному стандарті обробки інформації FIPS PUB 180-2 у серпні 2002 р. У цей стандарт також увійшла геш-функція SHA-1, розроблена в 1995 р. У лютому 2004 р. в FIPS PUB 180-2 була додана SHA-224. У жовтні 2008 р. вийшла нова редакція стандарту – FIPS PUB 180-3 [118].

У липні 2006 року з'явився стандарт RFC 4634 "Безпечні геш-алгоритми США (SHA і HMAC-SHA)", що описує SHA-1 і сімейство SHA-2. Агентство національної безпеки від імені держави випустило патент на SHA-2 під ліцензією Royalty Free.

Загальний опис.

Геш-функції сімейства SHA-2 побудовані на основі структури Меркле – Дамгарда [68].

Початкове повідомлення після доповнення розбивається на блоки, кожен блок – на 8 слів. Алгоритм пропускає кожен блок повідомлення через цикл з 64- чи 80 ітераціями (раундами). На кожній ітерації 2 слова з вось-ми перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням геш-функції.

Алгоритм використовує такі бітові операції:

|| – конкатенація;

+ – додавання;

and – побітове "І";

or – побітове "АБО";

xor – виключає "АБО";

shr – логічний зсув вправо;

rotr – циклічний зсув вправо.

На рис. 3.26 наведена схема однієї ітерації алгоритму SHA-2.

У табл. 3.1 наведено деякі технічні характеристики різних варіантів SHA-2. "Внутрішній стан" означає проміжну геш-суму після обробки чергового блоку даних.

Таблиця 3.1

Характеристики SHA-2

Геш-функція	Довжина дайджесту повідомлення (біт)	Довжина внутрішнього стану (біт)	Довжина блоку (біт)	Максимальна довжина повідомлення (біт)	Довжина слова (біт)	Кількість ітерацій у циклі
SHA-256/224	256/224	256	512	$264 - 1$	32	64
SHA-512/384	512/384	512	1024	$2128 - 1$	64	80

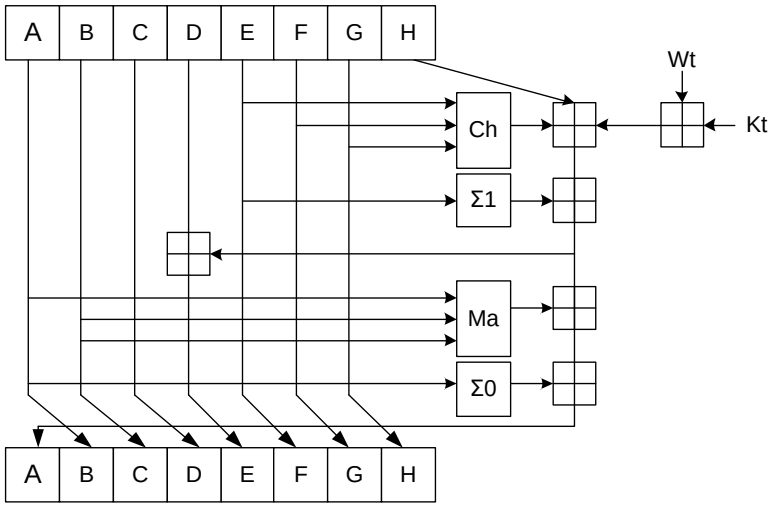


Рис. 3.26. **Схема однієї ітерації алгоритмів SHA-2**

Геш-функції SHA-2 використовуються для перевірки цілісності даних і в різних криптографічних схемах. На 2008 р. сімейство геш-функцій SHA-2 не має такого широкого розповсюдження, як MD5 і SHA-1 незважаючи на виявлені в останніх недоліки.

Геш-функції RIPEMD-160, RIPEMD-128

У 1995 р. були виявлені деякі недоліки в першій версії геш-функції RIPEMD, що формує 128-розрядний геш-код. Зокрема, були виявлені колізії для останніх двох із трьох циклів RIPEMD. Розроблена методика пошуку колізій була успішно застосована і для пошуку колізій у MD4. На основі отриманих результатів були зроблені припущення, що дана методика може бути використана для пошуку колізій у MD5 і в повній версії RIPEMD. Версія RIPEMD (разом з SHA-1) широко використовується в банківських технологіях. Однак ситуація з криптоаналізом геш-функцій сімейства MDx і перспективи розвитку обчислювальних потужностей призвели до висновку про необхідність відновлення RIPEMD. У результаті для стандартизації були запропоновані більш могутні версії геш-функції – RIPEMD-160 і RIPEMD-128. За оцінками розроблювачів геш-функція RIPEMD-160 повинна залишатися безпечною протягом десяти найближчих років. Удосконалена версія RIPEMD-128 замінила геш-функцію RIPEMD [123].

Необхідно розглянути дані функції більш докладно.

Розмір геш-кода і перемінних зчеплень RIPEMD-160 дорівнює 160 бітам (п’ять 32-х розрядних слів). Кількість циклів збільшена з трьох (як у RIPEMD) до п’яти. Крім того, внесені додаткові зміни в паралельні лінії алгоритму. Тепер права і ліва лінії відрізняються не тільки застосовува-

ними константами, але і порядком проходження бульових функцій. Для алгоритму RIPEMD-160 визначені такі булеві функції:

$$f(u,v,w) = u \oplus v \oplus w;$$

$$g(u,v,w) = (u \wedge v) \vee (\bar{u} \wedge w);$$

$$h(u,v,w) = (u \vee \bar{v}) \oplus w;$$

$$k(u,v,w) = (u \wedge w) \vee (v \wedge \bar{w})$$

$$l(u,v,w) = u \oplus (v \vee \bar{w}) .$$

Усі позначення аналогічні раніше використаним позначенням при описі алгоритму SHA-1.

На рис. 3.27 подана загальна структура геш-функції RIPEMD-160.

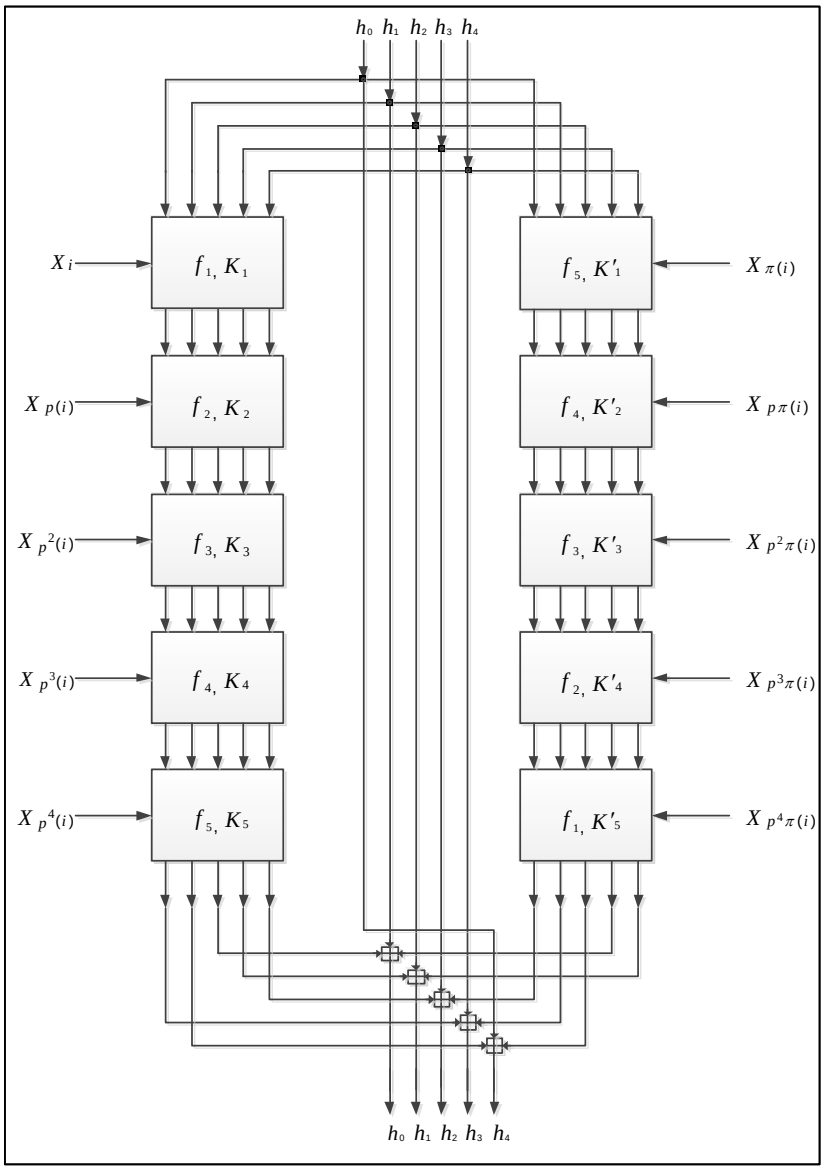


Рис. 3.27.**Схема геш-функції RIPEMD-160**

Алгоритм RIPEMD-160.

Вхід. Двійковий рядок *x* довжиною $b \geq 0$

Вихід. 160-бітний геш-код за рядков x.

1. *Ініціалізація*.

а) ініціалізувати п’ять векторів ініціалізації:

$$h_1 = 67452301_x; h_2 = efcdab89_x; h_3 = 98badcfe_x;$$
$$h_4 = 10325476_x; h_5 = c3d2e1f0_x;$$

б) задати для лівої і правої ліній алгоритму додаткові константи:

– ліва лінія:

$$y_L[j] = 00000000_x, 0 \leq j \leq 15;$$
$$y_L[j] = 5a827999_x = \lfloor 2^{30} \cdot \sqrt{2} \rfloor, 16 \leq j \leq 31;$$
$$y_L[j] = 6ed9eba1_x = \lfloor 2^{30} \cdot \sqrt{3} \rfloor, 32 \leq j \leq 47;$$
$$y_L[j] = 8f1bbcdc_x = \lfloor 2^{30} \cdot \sqrt{5} \rfloor, 48 \leq j \leq 63;$$
$$y_L[j] = a953fd4e_x = \lfloor 2^{30} \cdot \sqrt{7} \rfloor, 64 \leq j \leq 79;$$

– права лінія:

$$y_R[j] = 0 \times 50a28be6 = \lfloor 2^{30} \cdot \sqrt[3]{2} \rfloor, 0 \leq j \leq 15;$$
$$y_R[j] = 0 \times 5c4dd124 = \lfloor 2^{30} \cdot \sqrt[3]{3} \rfloor, 16 \leq j \leq 31;$$
$$y_R[j] = 0 \times 6d703ef3 = \lfloor 2^{30} \cdot \sqrt[3]{5} \rfloor, 32 \leq j \leq 47;$$
$$y_R[j] = 7ad76e9_x = \lfloor 2^{30} \cdot \sqrt[3]{7} \rfloor, 48 \leq j \leq 63;$$
$$y_R[j] = 00000000_x, 64 \leq j \leq 79;$$

в) задати порядок подачі слів повідомлення на ліву $z[j]$і праву $z[j]$ лінії. Порядок подачі слів визначається на основі таких перестановок:

для лівої лінії визначається перестановка ρ, а для циклів порядок слів визначається таким чином:

i	z_L[0..15]	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ρ(i)	z_L[16..31]	7	4	13	1	10	6	15	3	12	0	9	5	2	14	11	8
ρ^2(i)	z_L[32..47]	3	10	14	4	9	15	8	1	2	7	0	6	13	11	5	12
ρ^3(i)	z_L[48..63]	1	9	11	10	0	8	12	4	13	3	7	15	14	5	6	2
ρ^4(i)	z_L[64..79]	4	0	5	9	7	12	2	10	14	1	3	8	11	6	15	13

для правої лінії визначається перестановка $\pi(i) = 9i + 5(\lfloor 16 \rfloor)$ і порядок подачі слів визначається з використанням перестановки ρ у таким чином:

$\pi(i)$	z_R[0..15]	5	14	7	0	9	2	11	4	13	6	15	8	1	10	3	12
$\rho\pi(i)$	z_R[16..31]	6	11	3	7	0	13	5	10	14	15	8	12	4	9	1	2
$\rho^2\pi(i)$	z_R[32..47]	15	5	1	3	7	14	6	9	11	8	12	2	10	0	4	13
$\rho^3\pi(i)$	z_R[48..63]	8	6	4	1	3	11	15	0	5	12	2	13	9	7	10	14
$\rho^4\pi(i)$	z_R[64..79]	12	15	10	4	1	5	8	7	6	2	13	14	0	3	9	11

г) визначити для лівої $S[j]$ і правої $S[j]$ ліній аргументи для операторів циклічного зсуву відповідно до таблиці.

s_L[0..15]	11	14	15	12	5	8	7	9	11	13	14	15	6	7	9	8
s_L[16..31]	7	6	8	13	11	9	7	15	7	12	15	9	11	7	13	12
s_L[32..47]	11	13	6	7	14	9	13	15	14	8	13	6	5	12	7	5
s_L[48..63]	11	12	14	15	14	15	9	8	9	14	5	6	8	6	5	12
s_L[64..79]	9	15	5	11	6	8	13	12	5	12	13	14	11	8	5	6
s_R[0..15]	8	9	9	11	13	15	15	5	7	7	8	11	14	14	12	6
s_R[16..31]	9	13	15	7	12	8	9	11	7	7	12	7	6	15	13	11
s_R[32..47]	9	7	15	11	8	6	6	14	12	13	5	14	13	13	7	5
s_R[48..63]	15	5	8	11	14	14	6	14	6	9	12	9	12	5	15	8
s_R[64..79]	8	5	12	9	12	5	14	6	8	13	6	5	15	13	11	11

2. *Передобробка.* Доповнити рядок x так, щоб його довжина була кратна числу 512. Для цього додати в кінець рядка одиницю, а потім стільки нульових біт, скільки необхідно для одержання рядка довжиною на 64 біта коротше довжини кратної 512. Нарешті, додати останні два 32-х розрядні слова, що містять двійкове представлення числа b. Нехай m – кількість 512-бітних блоків в отриманому доповненому рядку. У такий спосіб форматований вхід складається з 16m 32-х розрядних слів:

$$X_0 X_1 \text{ к } X_{16m-1}.$$

Ініціалізувати вектор змінних зчеплення:

$$(H_1, H_2, H_3, H_4 . H_5) = (h_1, h_2, h_3, h_4, h_5).$$

3. *Обробка*. Для всіх і від 0 до m – 1 кожен і-й блок з шістнадцяти 32-х бітних слів записати в тимчасовий масив:

$$X[j] = x_{16i+j}, 0 \le j \le 15.$$

Потім:

а) виконати п'ять 16-ти крокових циклів лівої лінії.

Ініціалізувати робочі змінні

$$(A_L, B_L, C_L, D_L, E_L) = (H_1, H_2, H_3, H_4, H_5).$$

Виконати:

Для j = 0 до 15

$$t = (A_L + f(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для j = 16 до 31

$$t = (A_L + g(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для j = 32 до 47

$$t = (A_L + h(B_L, C_L, D_L) + X[z_L[j]] + y_L[j]);$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для j = 48 до 63

$$t = (A_L + k(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L).$$

Для j = 64 до 79

$$t = (A_L + l(B_L, C_L, D_L) + X[z_L[j]] + y_L[j])$$

$$(A_L, B_L, C_L, D_L, E_L) = (E_L, E_L + (t <<< s_L[j]), B_L, C_L <<< 10, D_L);$$

б) виконати паралельно з циклами лівої лінії п'ять 16-крокових циклів правої лінії.

Ініціалізувати робочі змінні:

$$(A_R, B_R, C_R, D_R, E_R) = (H_1, H_2, H_3, H_4, H_5).$$

Виконати:

Для j = 0 до 15

$$t = (A_R + I(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для j = 16 до 31

$$t = (A_R + K(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для j = 32 до 47

$$t = (A_R + H(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для j = 48 до 63

$$t = (A_R + G(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R).$$

Для j = 64 до 79

$$t = (A_R + I(B_R, C_R, D_R) + X[z_R[j]] + y_R[j]);$$

$$(A_R, B_R, C_R, D_R, E_R) = (E_R, E_R + (t <<< s_R[j]), B_R, C_R <<< 10, D_R);$$

в) обновити змінні зчеплення:

$$t = H_1;$$

$$H_1 = H_2 + C_L + D_R;$$

$$H_2 = H_3 + D_L + E_R;$$

$$H_3 = H_4 + E_L + A_R;$$

$$H_4 = H_5 + A_L + B_R;$$

$$H_5 = t + B_L + C_R.$$

4. *Завершення*. Як геш-код прийняти величину: $H_1 \| H_2 \| H_3 \| H_4 \| H_5$.

Основні відмінності алгоритму RIPEMD-128 від RIPEMD-160 полягають у такому:

довжина геш-кода складає 128 біт (чотири 32-х бітні слова);

функція складається з чотирьох циклів і, відповідно, використовується чотири 32-х розрядні змінні зчеплення;

основна операція циклу має вигляд:

$$t = (A + f(B,C,D)) + X(z[j] + y[j]);$$
$$(A,B,C,D) = (D, D + (t <<< s[j]), A, B, C);$$

визначені тільки чотири булеві функції f, g, h і k;

з переліку констант виключаються значення $[20^{30}\sqrt{7}]$ і $[20^{30}\sqrt[3]{7}]$, а з переліку векторів ініціалізації виключають h_5.

Деякі додатки, у яких використовують геш-функції, можуть вимагати виробітки більш довгих геш-кодів.

Прямим шляхом розв'язання даної задачі є паралельне виконання однієї і тієї ж геш-функції з різними векторами ініціалізації. Однак у цьому випадку можлива поява залежностей між двома результатами гешування, що і було виявлено при використанні MD4.

Геш-функції RIPEMD-128 і RIPEMD-160 уже мають паралельні лінії і для одержання геш-кодів довжиною, відповідно, 256 і 320 біт досить виключити кінцеву комбінацію результатів гешування по кожній лінії.

3.4. Ключове гешування

До ключових геш-функцій відносяться MAC-коди. Вони призначені для забезпечення цілісності даних і автентифікації повідомлень без використання яких-небудь інших механізмів і дозволяють забезпечити автентифікацію повідомлення на основі використання методів симетричної криптографії.

Алгоритми формування MAC-кодів розглядаються як геш-функції з двома вхідними параметрами, а саме повідомленням і секретним ключем. На виході такого алгоритму формується двійковий рядок фіксованої довжини. При цьому на практиці неможливо сформувати точно такий же рядок без знання ключа. MAC-коди можуть використовуватися як для забезпечення цілісності даних, так і автентифікації джерела даних [26].

У криптографічних додатках загальноприйнятим є те, що алгоритми геш-функцій є відкритими. Таким чином, у випадку використання MDC-коду за даним вхідним повідомленням геш-код може обчислити будь-який суб'єкт, а при використанні MAC-коду обчислити геш-код за даним вхідним повідомленням може тільки суб'єкт, що володіє секретним ключем.

Стандарти кодів автентифікації повідомлень

Першими стандартами на MAC-коди є американські промислові стандарти ANSI X9.9 і ANSI X9.19, опубліковані відповідно у 1982 та 1986 рр. [50 – 52].

У 1986 р. з'явився міжнародний стандарт ISO 8730, що є по суті міжнародним еквівалентом ANSI X9.9 і визначає загальні вимоги для таких механізмів. Після стандарту ISO 8730 розроблені стандарти ISO 8731-1:1987 і ISO 8731-2:1987. Стандарти ANSI X9.9, ANSI X9.19, ISO 8730, ISO 8731-1:1987 визначають алгоритми формування MAC-кодів на основі використання алгоритму блокового шифрування DES у CBC режимі. Такий код дійсності відомий у літературі як CBC-MAC. Стандарт ISO 8731-2:1987 стандартизує алгоритм автентифікації повідомлення (Message Authentificator Algorithm, MAA). Усі перераховані стандарти орієнтовані на застосування стандартизованих методів формування MAC-кодів у банківських технологіях. На основі банківських стандартів у 1989 р. ISO розробляє стандарт MAC-коду загального призначення – ISO/IEC 9797. Цей стандарт також використовує блоковий шифр у CBC режимі, тобто він визначає CBC-MAC. У 1994 р. стандарт ISO/IEC 9797 був доопрацьований і оновлений.

У 1997 р. почався перегляд міжнародного ISO стандарту MAC-коду. Даний стандарт 1994 р. був замінений на стандарт ISO/IEC 9797-1, що містить розширену множину CBC-MAC механізмів. Інша частина стандарту, ISO/IEC 9797-2, також переглянута і поліпшена, містить ряд механізмів формування MAC-кодів на базі геш-функції, включаючи метод HMAC [118].

Варто почати з розгляду CBC-MAC, визначених у другій редакції стандарта ISO/IEC 9797:1994. Потім розглянути, які зміни були введені в ISO/IEC 9797-1 і розглянути механізми формування MAC-коду на основі геш-функції, визначені в ISO/IEC 9797-2. Також потрібно розглянути алгоритм автентифікації повідомлень, визначений у стандарті ISO 8731-2:1987.

Перш ніж безпосередньо розглянути алгоритми формування MAC-кодів, потрібно навести деякі загальні положення.

Алгоритм формування коду автентифікації повідомлення є сімейством функцій h_k, де k – секретний ключ, що володіють такими властивостями [79]:

1. *Простота обчислення* – для відомої функції h_k,заданого значення k і вхідного значення x, легко обчислити $h_k(x)$. Отриманий результат називається MAC-код.

2. *Стиск* – h_k відображає вхідне значення x – кінцевий двійковий рядок довільної довжини у вихідне значення $h_k(x)$ – двійковий рядок фіксованої довжини n.

3. *Стійкість до обчислення* – по відомих парах "текст – МАС-код" $(x_i, h_k(x_i))$ обчислювально неможливо обчислити будь-яку іншу пару "текст – МАС-код" $(x', h_k(x'))$ для будь-якого нового вхідного значення $x' \neq x_i$.

Якщо остання властивість не виконується, то можлива підробка МАС-коду. У той час, як стійкість до обчислення має на увазі і наявність властивості невідновлюваності ключа (тобто обчислювально неможливо відновити ключ k, за відомою однією чи більше пар "текст – МАС-код" $(x_i, h_k(x_i))$, отриманих з використанням цього ключа), властивість не відновлюваності ключа не має на увазі виконання властивості стійкості до обчислення, оскільки не обов'язково відновлювати ключ, для того щоб підробити МАС-код.

Задачу зловмисника можна сформулювати в такий спосіб: без знання ключа k, обчислити нову пару "текст – МАС-код" $(x', h_k(x'))$ для деякого тексту $x' \neq x_i$ за відомою однією чи більше пар "текст – МАС" $(x_i, h_k(x_i))$.

Стійкість до обчислення повинна забезпечуватися й у випадку, коли тексти x_i, для яких доступні відповідні МАС-коди, заздалегідь дані зловмиснику, й у випадку, коли він вільно вибирає ці тексти.

Можна виділити таки атаки на МАС-коди:

1. Атака на основі відкритого тексту має місце тоді, коли доступні одна або більш пар "текст – МАС-код" $(x_i, h_k(x_i))$.

2. Атака на основі обраного відкритого тексту має місце тоді, коли доступні одна або більше пар "текст – МАС-код" $(x_i, h_k(x_i))$ (для обраних зловмисником текстів x_i.

3. Адаптивна атака на основі обраного відкритого тексту має місце тоді, коли відкритий текст x_i вибирається зловмисником в умовах наявності в нього деякої додаткової інформації.

З погляду сертифікації алгоритм формування MAC-коду повинний протистояти адаптивній атаці не залежно від того, можлива чи ні її практична реалізація, оскільки саме ця атака є найбільш небезпечною.

Степінь тяжкості наслідків, що настають унаслідок підробки MAC-коду, залежить від степені контролю зловмисником значень x, для яких можлива підробка MAC-коду. З цього погляду можливі два типи підробки MAC-коду: вибірна підробка, що включає множину атак, за допомогою яких зловмисник здатен утворити нову пару "текст – MAC-код" для тексту, що ним же самим і обраний (або вибирався під його керуванням). Отже, тут обраним значенням є текст, для якого MAC-код підроблений, тоді як в атаці на основі обраного відкритого тексту текст із пари "текст – MAC-код" служить для аналізу, з метою підробки MAC-кодів для інших текстів;

екзистенціальна підробка, що включає множину атак, за допомогою яких зловмисник здатен породити нову пару "текст – MAC-код", але без контролю над значенням цього тексту.

Найбільш небезпечною атакою, з погляду здійснення підробки MAC-коду, є відновлення ключа. При реалізації всіх типів атак, що дозволяють підробити MAC-код, зловмисник може видавати підроблені повідомлення за справжні.

3.4.1. На основі блочних шифрів

Стандарт MAC-коду ISO/IEC 9797:1994

Стандарт ISO/IEC 9797:1994 обумовлює метод використання секретного ключа k і n-розрядного алгоритму блокового шифрування E для обчислення m-розрядного криптографічного значення (суми), що може бути використаний як механізм забезпечення цілісності даних для виявлення неавторизованої зміни даних [118]. Потрібно відзначити, що довжина MAC-кода визначається користувачем з урахуванням обмеження $m \le n$.

Сутність обробки даних полягає в такому. Спочатку дані обробляються таким чином, щоб сформувати послідовність n-розрядних блоків. При необхідності здійснюється операція доповнення повідомлення. Потім дані шифруються з використанням блокового шифру із секретним ключем у CBC режимі.

І нарешті, як MAC-коду після додаткової обробки (optinal process) і усікання (за необхідності, якщо потрібно $m < n$) береться останній шифрований блок [24].

Алгоритм формування CBC – MAC

Вхід: дані x; блоковий шифр E з довжиною блоку n; секретний ключ k.

Вихід: *m*-розрядний MAC-код для даних *x*.

1. *Доповнення і розбивка на блоки.* Доповнити рядок даних *x*, якщо необхідно, використовуючи один з визначених методів доповнення. Розбити доповнений текст на *n*-розрядні блоки $x_1, x_2, ..., x_t ...$.

2. *Зашифрування в режимі CBC.* Нехай E_k позначає процес зашифрування з використання блокового шифру E з ключем k. Тоді слід обчислити блок H_t у такий спосіб:

$$H_1 = E_k(x_1);$$
$$H_i = E_k(H_{i-1} \oplus x_i) \forall 2 \le i \le t.$$

Таким чином, здійснюється стандартне зчеплення блоків з початковим вектором IV = 0.

3. *Додаткова обробка й усікання.* Стандарт ISO/IEC 9797:1994 визначає два типи додаткової обробки.

3.1. Додаткова обробка 1 типу. Використовуючи другий секретний ключ к' ≠ k обчислити $H'_t = E_k(E^{-1}_{k'}(H_t))$.

3.2. Додаткова обробка 2 типу. Використовуючи другий секретний ключ к' ≠ k обчислити $H'_t = E_{k'}(H_t)$. Ключ k' може бути похідним від ключа k.

Після додаткової обробки, результат – *n*-розрядний блок H_t може бути усічений до *m* бітів (якщо $m < n$).

Стандарт ISO/IEC 9797:1994 визначає два можливих методи доповнення [118].

Метод 1 (старий метод). Додати до вихідних даних *x* необхідну кількість нулів до одержання блоку даних, довжина якого буде цілим числом кратним *n*.

Метод 2. Додати до вихідних даних *x* одну одиницю, а потім необхідну кількість нулів.

Перший метод неоднозначний – останні нульові розряди вихідних даних не можуть бути розпізнані від нульових розрядів, що додаються в час доповнення. Другий метод виключає цю неоднозначність. При його використанні кожному доповненому рядку x' буде відповідати унікаль-

ний вихідний рядок x. Якщо довжина вихідного рядка вже кратна *n*, то застосування другого методу наведе до появи нового блоку. Важливо відзначити, що для забезпечення цілісності записаного/переданого блоку даних, процес доповнення не обов'язковий.

Стандарт ISO/IEC 9797:1994 дає такі рекомендації з вибору методів доповнення. Якщо довжина блоку вихідних даних не повідомляється верифікатору шляхом використання деяких достовірних способів, то повинен використовуватися другий метод доповнення, оскільки він дозволяє визначити злочинне вилучення/додавання останніх нулів. Метод 1, хоча і потенційно слабкіше другого, збережений для забезпечення сумісності міжнародного стандарту зі стандартами ANSI X9.9 і ANSI X9.19, що не забезпечують вибір другого методу доповнення [50 – 52; 118].

Додаткова обробка дозволяє зменшити погрозу реалізації вичерпного пошуку ключа, а також запобігти екзистенціальній підробці на основі обраного відкритого тексту, без істотного зниження ефективності роботи алгоритму в порівнянні з використанням двоключового потрійного шифрування.

Припустимо CBC – MAC обчислений без додаткової обробки й усікання. Тоді, по двох наявних повідомленнях з дійсними MAC-кодами (обчисленими з використанням того самого секретного ключа *k*), ми можна обчислити третє "складове" підроблене повідомлення з дійсним MAC-кодом без знання секретного ключа. Слід проілюструвати дану атаку.

1. Нехай відомі MAC-коди для двох одноблокових повідомлень X_1 і X_2, $MAC_1 = E_k(x_1)$ і $MAC_2 = E_k(x_2)$. Тоді MAC_2 також є дійсним MAC-кодом і для нового двоблокового повідомлення вигляду $(x_1 \| (x_2 \oplus MAC_1))$.

2. Нехай відома пара (x_1, MAC_1) і потрібно отримати MAC-код MAC_2 для одноблокового повідомлення $x_2 = MAC_1$. Ясно, що $MAC_2 = E_k(E_k(x_1))$. Однак отриманий код також є дійсним для повідомлення вигляду $(x_1 \| \perp 0)$. Тут $\perp 0$ – n-розрядне представлення нуля.

3. Нехай дані дві відомі пари (x_1, MAC_1) і (x_2, MAC_2). Необхідне значення MAC-коду для обраного тексту $(x_1 \| z)$ визначається як $MAC_3 = E_k(H_1 \oplus z)$. Однак MAC_3 є дійсним кодом і для повідомлення вигляду $(x_2 \| (H_1 \oplus z \oplus H_2))$.

Отже, навіть якщо два MAC-коди ніколи не обчислюються з використанням того самого ключа, ці атаки все-таки застосувані, оскільки можна взяти $X_1 = X_2$.

Підробка MAC-коду можлива при використанні $O(2^{n/2})$ відомих пар "текст – MAC-код" плюс *v* обраних пар "текст – MAC-код". Значення *v* лежать у межах від 1 до 2^{n-m}, залежно від вигляду алгоритму формування MAC-коду. Це верхня границя стійкості MAC-коду, визначена в припущенні можливості здійснення криптоаналізу на основі парадоксу дня народження. Для $n = m = 64$ значення $v = 1$.

Наведені приклади і практика використання MAC-кодів показує необхідність використання додаткової обробки й усікання. Зокрема, стандарт завжди рекомендує використовувати додаткову обробку при m = n.

Механізми, що використовують блоковий шифр (за міжнародним стандартом ISO/IEC ISO/IEC 9797-1:1999(E))

Призначення.

ISO/IEC 9797 визначає шість MAC-алгоритмів, що використовують секретний ключ і n-*розрядний* блоковий шифр для розрахунків m-*розрядного* MAC. Ці механізми можуть бути використані як механізми підтримки цілісності даних, щоб перевірити, що дані не були змінені не уповноваженим способом. Вони також можуть бути використані як механізми автентифікації повідомлення, що гарантують, що повідомлення було породжено об’єктом, що володіє секретним ключем. Міцність механізму підтримки цілісності даних і механізму автентифікації повідомлення залежить від довжини (у бітах) k* і таємності ключа, довжини блоку (у бітах) n і міцності блокового шифру, довжини (у бітах) MAC m, а також від спеціальних механізмів [118].

Перші три механізми, визначені в ISO/IEC 9797 як загальновідомі CBC-MAC (CBC є скороченням від поблочної передачі зашифрованого тексту, Cipher Block Chaining). Обчислення MAC, як описано в ISO 8731-1 і ANSI X9.9, становить окремий приклад цього розділу при n = 64, m = 32, що використовує MAC-алгоритм 1 і метод заповнення 1, блоковий шифр – DEA (ANSI X3.92: 1981). Обчислення MAC, як описано в ANSI X9.19 і ISO 9807, являє окремий приклад цього розділу ISO/IEC 9797 при n = 64, m = 32, що використовує або MAC-алгоритм 1, або MAC-алгоритм 3

(в обох випадках – метод заповнення 1), блоковий шифр – DEA (ANSI X3.92: 1981) [50 – 52].

Четвертий механізм є варіантом CBC-MAC з первинним спеціальним перетворенням. Він рекомендується для додатків, що вимагають, щоб довжина ключа MAC-алгоритму була у два рази більше блокового шифру.

Наприклад, у випадку DEA (ANSI X3.92: 1981) довжина ключа блокового шифру – 56 біт, а довжина ключа MAC-алгоритму – 112 біт.

При використанні DEA (також відомий як DES) цей алгоритм називається Macdes [52].

П'ятий і шостий механізм використовують перший і четвертий механізми відповідно, сполучаючи при цьому результати з операцією побітового, що виключає АБО. Вони рекомендуються для додатків, що вимагають підвищений рівень безпеки від атак. П'ятий механізм використовує одиничну довжину ключа MAC-алгоритму, а шостий механізм подвоює довжину ключа MAC-алгоритму.

Цей розділ ISO/IEC 9797 може бути використаний у службі безпеки будь-якої архітектури безпеки, будь-якого процесу або додатка.

У ISO/IEC 9797 використовуються такі символи й нотації [118]:

D – рядок даних, що є вхідним до MAC-алгоритму.

D – блок, отриманий з рядка даних D після процесу заповнення.

d(C) – розшифрування шифротекста. Із блоковим шифром e, використовуючи ключ K.

$e_k(P)$ – шифрування відкритого тексту P блоковим шифром e, використовуючи ключ K. відображає

g – вихідне перетворення, що блок H_g на блок G.

G – блок, що є результатом вихідного перетворення.

H_j – блок, що використовується MAC-алгоритмом для зберігання проміжних результатів.

I – початкове перетворення.

k – довжина (у бітах) ключа блокового шифру.

k* – довжина (у бітах) ключа MAC-алгоритму.

KK', K'', K''', K_1, K_2, K_1',K_2',K_1'',K_2'' – секретні ключі блокового шифру.

L – довжина блоку, що використовується в методі заповнення 3.

L_D – довжина (у бітах) рядка даних D.

m – довжина (у бітах) MAC.

n – довжина блоку (у бітах) блокового шифру.

q – кількість блоків у рядку даних D після процесу заповнення й розбивки.

j~X – рядок, отриманий з рядка X, використовуючи крайні ліві j біт з X.

$X \oplus Y$ виключаюче АБО бітових рядків X та Y.

X‖Y – зчеплення бітових рядків X та Y (у такому ж порядку).

:= – символ, що означає операцію "прирівняти", використовувану в процедурних специфікаціях MAC-алгоритмів, де вона позначає, що значення рядка ліворуч від символу буде дорівнювати значенню виразу, що знаходиться праворуч від символу.

Вимоги.

Користувачі, що бажають використовувати MAC-алгоритм із розділу ISO/IEC 9797, повинні вибрати:

блоковий шифр *e*;

метод заповнення з визначених;

MAC-алгоритм із визначених;

довжину (у бітах) m MAC;

метод походження загального ключа, якщо використовуються MAC-алгоритми 4, 5 і 6; метод походження загального ключа також може знадобитися для MAC-алгоритму 2.

Угода на ці підбори серед користувачів важливі для роботи механізму цілісності даних.

Довжина m MAC повинна бути додатним цілим, менше або рівним довжині блоку n.

Якщо використовується метод заповнення 3, довжина в бітах рядка даних *D* повинна бути менше ніж 2^n.

Вибір конкретного блокового шифру *e*, методу заповнення, MAC-алгоритму, значення m і методу походження ключа (якщо потрібно) перебуває поза рамками цього розділу ISO/IEC 9797. Ці вибори впливають на рівень безпеки MAC-алгоритму.

Той же ключ буде використаний для розрахунків і перевірки MAC. Якщо рядок даних також шифрується, ключ, що використовується для обчислення MAC, повинен відрізнятися від того, який використовується для розшифрування.

Вважається гарною криптографічною практикою мати незалежні ключі для конфіденційності й цілісності даних.

Модель для MAC-алгоритмів.

Застосування MAC-алгоритму вимагає таких шість кроків: заповнення, розбивка, початкове перетворення, ітераційне використання блокового шифру, вихідне перетворення й усікання. Кроки з 3 по 6 представлені на рис. 3.28.

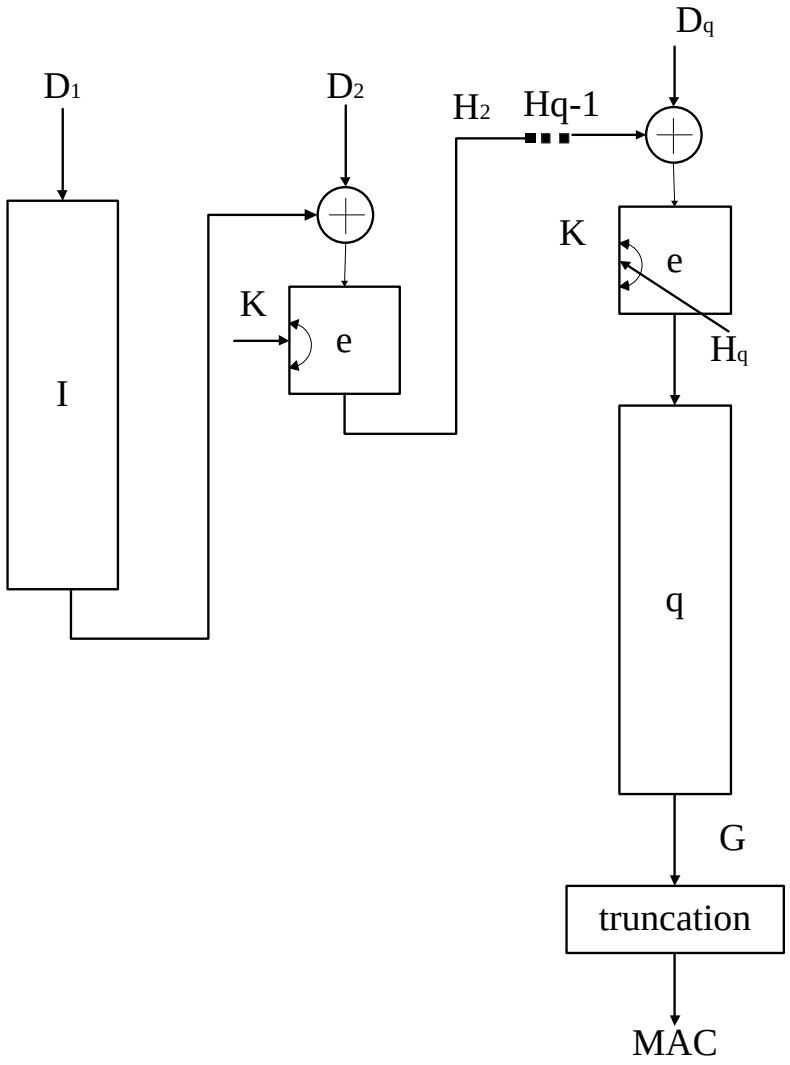


Рис. 3.28. **Використання кроків 3, 4, 5 і 6 MAC-алгоритму**

Крок 1 (заповнення).

Цей крок містить завдання префікса й/або постфікса дата рядка *D* з додатковим заповненням бітами такими, що заповнений варіант рядок даних буде завжди кратним *n* бітам по довжині. Заповнені біти, додані до вихідного рядка даних відповідно до обраного методу заповнення, використовуються тільки для обчислення MAC. Тому немає необхідності зберігати або передавати з даними ці заповнені біти. Верифікатор повинен знати, були чи ні заповнені біти збережені або передані, а також використовуваний метод заповнення.

Цей розділ ISO/IEC 9797 визначає три методи заповнення. Усі з них можуть бути взяті для шести MAC-алгоритмів, позначених у цьому розділі ISO/IEC 9797.

Метод заповнення 1.

Рядок даних D, вхідна для MAC-алгоритму, повинна бути заповнена праворуч мінімальним числом (якщо можливо, нічим не заповнена) "0" – біт, щоб одержати рядок даних, довжина якого (у бітах) є додатним цілим кратним n числом.

MAC-алгоритми, що використовують метод заповнення 1, можуть бути об’єктами тривіальних фальшивих атак. Якщо рядок даних порожній, метод заповнення 1 визначає, що він заповнений праворуч n "0" – бітами.

Метод заповнення 2.

Рядок даних D, вхідний для MAC-алгоритму, повинен бути заповнений праворуч одиничним "1" – бітом. Потім результуючий рядок повинен бути заповнений за необхідністю праворуч мінімальним числом (якщо можливо, нічим не заповнений) "0" – біт, щоб одержати рядок даних, довжина якого (у бітах) є додатним цілим кратним n числом.

Метод заповнення 3.

Рядок даних D, вхідний для MAC-алгоритму, повинен бути заповнений праворуч мінімальним числом (якщо можливо, нічим не заповнений) "0" – біт, щоб одержати рядок даних, довжина якого (у бітах) є додатним цілим кратним n числом. Результуючий рядок повинен бути потім заповнений ліворуч блоком L. Блок L складається з бінарного представлення довжини (у бітах) L_D незаповненого рядка даних D, ліворуч заповненого мінімальним (можливо, нічим незаповненим) "0" – біт, щоб одержати n-бітовий блок. Крайній правий біт блоку L співвідноситься до найменш значного біта двійкового представлення L_D.

Метод заповнення 3 не підходить для використання у випадках, коли довжина рядка даних не доступна до початку MAC-обчислення.

Крок 2 (розбивка).

Заповнений варіант рядка даних D розбивається на q-бітні блоки $D_1,D_2,\ldots,D_q$. Де D_1 представляє перші n біт заповненого варіанта D, D_2 представляє такі n біт і т. д.

Крок 3 (початкове перетворення).

Початкове перетворення I здійснюється над блоком D_1 заповненого рядка даних, щоб одержати блок H_1.

Кожен із шести MAC-алгоритмів, визначених у цьому розділі ISO/IEC 9797, використовує один із двох можливих початкових перетворень.

Початкове перетворення 1.

Це перетворення вимагає тільки один ключ K блокового шифру. Блок H_1 обчислюється використанням блокового шифру із ключем K у такий спосіб: $H_1:=e_k(D_1)$.

Початкове перетворення 2.

Це перетворення вимагає два ключі K і K'' блокового шифру. Блок H_1 обчислюється використанням блокового шифру із ключами *K* і K_1 у такий спосіб: $H_1:=e_{k''}(e_k(D_1))$.

Крок 4 (ітерація).

Блоки H_2, H_3, …, H_q обчислюються ітеративно використанням блокового шифру з побітовим виключаючим АБО блоку даних D_i і попереднього результату H_{i-1} для і від 2 до q: $H_i := e_k(D_1 \oplus H_{i-1})$.

Якщо q дорівнює 1, крок 4 пропускається.

Ця операція відповідає режиму зчеплення блоків по шифротексту (CBC), певному в ISO-IEC 10116.

Крок 5 (вихідне перетворення).

Вихідному перетворенню g підвержено значення H_q, отримане як результат кроку 4 (або кроку 3 у випадку, коли q_1).

Вихідне перетворення 1.

Це вихідне перетворення є тотожною функцією, тобто: $G :=$.

Вихідне перетворення 2.

Дане вихідне перетворення складається з використання блокового шифру із ключем блокового шифру K' стосовно H_q, тобто: $G := (H_q)$

Вихідне перетворення 3.

Дане вихідне перетворення складається з використання блокового шифру (у режимі розшифрування) із ключем блокового шифру K'

стосовно H_q, а потім використовується блоковий шифр із ключем K стосовно результату цієї операції, тобто: $G := (d_{K'}(H_q))$.

Крок 6 (усікання).

MAC m біт отримуються вибором крайніх m лівих біт блоку G, тобто: $MAC := m \approx G$.

MAC-алгоритми.

ISO/IEC 9797 визначає шість MAC-алгоритмів. Початкове перетворення й вихідне перетворення визначаються в кожному випадку. Однак метод заповнення не визначається, тобто кожний MAC-алгоритм може бути використано з кожним із трьох методів заповнення, визначених вище.

Вибір методу заповнення впливає на безпеку MAC-алгоритму.

MAC-алгоритм 1.

MAC-алгоритм 1 використовує початкове перетворення 1 і вихідне перетворення 1. Ключ MAC-алгоритму складається із ключа блокового шифру K. MAC-алгоритм 1 наведений на рис. 3.29.

MAC-алгоритм 2.

MAC-алгоритм 2 використовує початкове перетворення 1 і вихідне перетворення 2. Ключ MAC-алгоритму складається із двох ключів блокового шифру K і K'''. Значення K''' може бути отримане зі значення K таким чином, що K і K''' різні.

Приклад того, як одержати K''' з K – додати по черзі підрядки з чотирьох біт K, починаючи з перших чотирьох біт. Інший приклад – одержати обидва K і K''' із загального головного універсального ключа.

Якщо K і K''' рівні, застосовується проста XOR фальшива атака (криптоаналіз). Подробиці див. приклад, наведений нижче.

Якщо K і K''' незалежні, рівень забезпечення безпеки проти атак відновлення ключа менше, ніж задає розмір ключа MAC-алгоритму.

MAC-алгоритм 2 наведений на рис. 3.30.

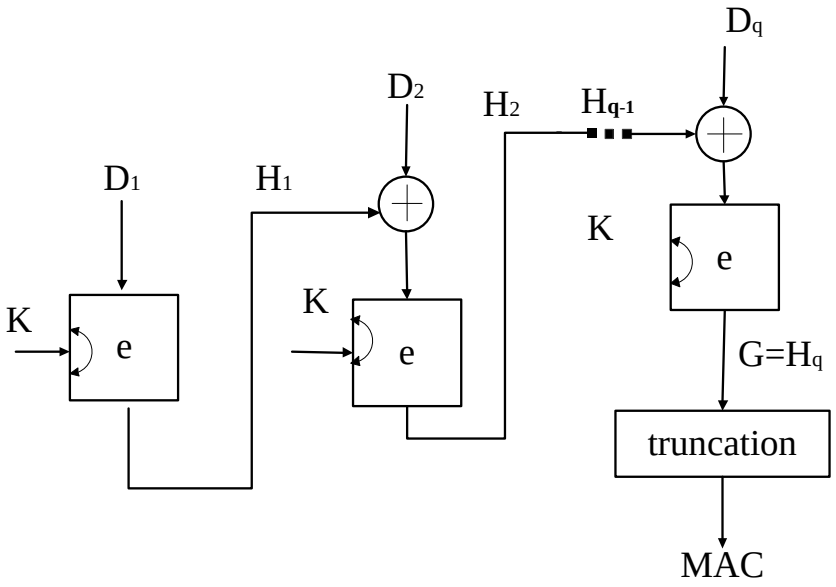


Рис. 3.29. **MAC-алгоритм 1**

MAC-алгоритм 3.

MAC-алгоритм 3 використовує початкове перетворення 1 і вихідне перетворення 3. Ключ MAC-алгоритму складається із двох ключів блокового шифру K і K'. Значення K і K' повинні бути обрані незалежно. Якщо $K' = K$ MAC-алгоритм 3 перетворюється в MAC-алгоритм 1, що не завжди бажано. MAC-алгоритм 3 наведений на рис. 3.31.

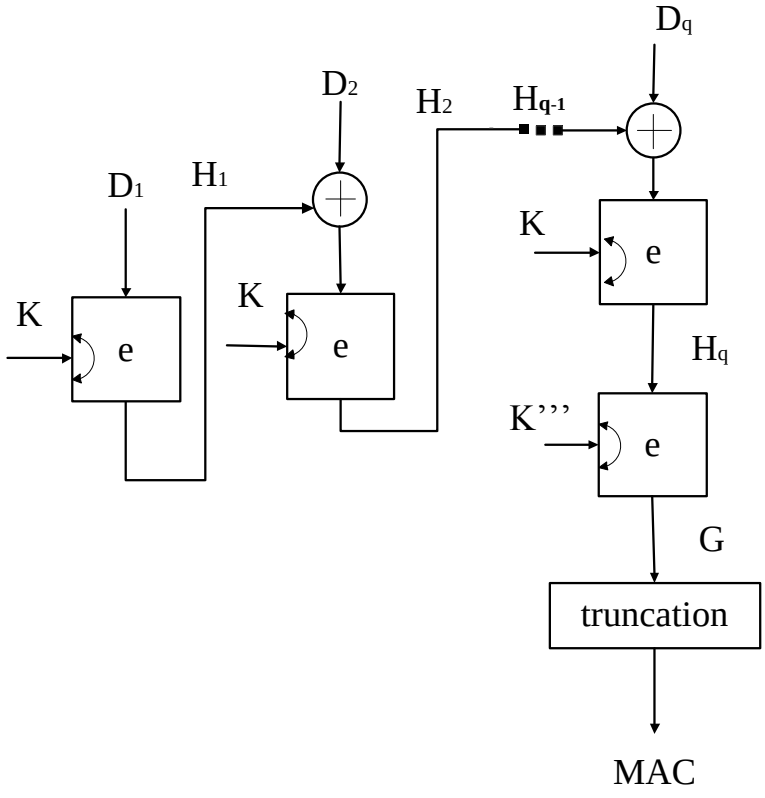


Рис. 3.30. **MAC-алгоритм 2**

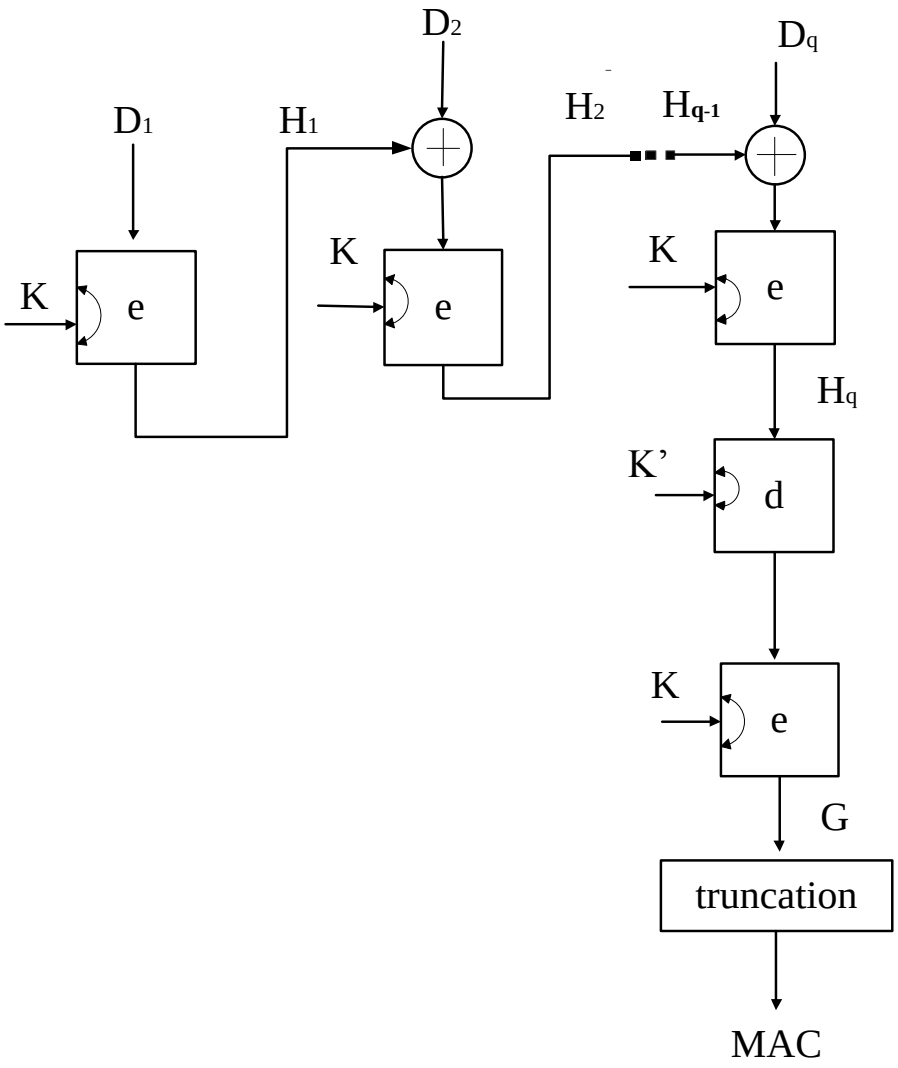


Рис. 3.31. **MAC-алгоритм 3**

MAC-алгоритм 4.

MAC-алгоритм 4 використовує початкове перетворення 2 і вихідне перетворення 2. Ключ MAC-алгоритму складається із двох ключів блокового шифру K і K', які повинні бути обрані незалежно. Третій ключ блокового шифру K'' повинен походити від K'.

Значення KK',K''' повинні бути різними. Ключі K і K'' блокового шифру використовуються з початковим перетворенням 2, а ключі K і K' блокового шифру використовуються з вихідним перетворенням 2.

Приклад того, як одержати K'' з K' – додати почергово підрядки із чотирьох біт ключа K', починаючи з перших чотирьох біт. Інший приклад, як одержати обоє K' і K'', – із загального головного (еталонного) ключа.

Число блоків у заповненому варіанті рядка даних повинне бути більше або дорівнювати двом, тобто $q \geq 2$.

MAC-алгоритм 4 наведений на рис. 3.32.

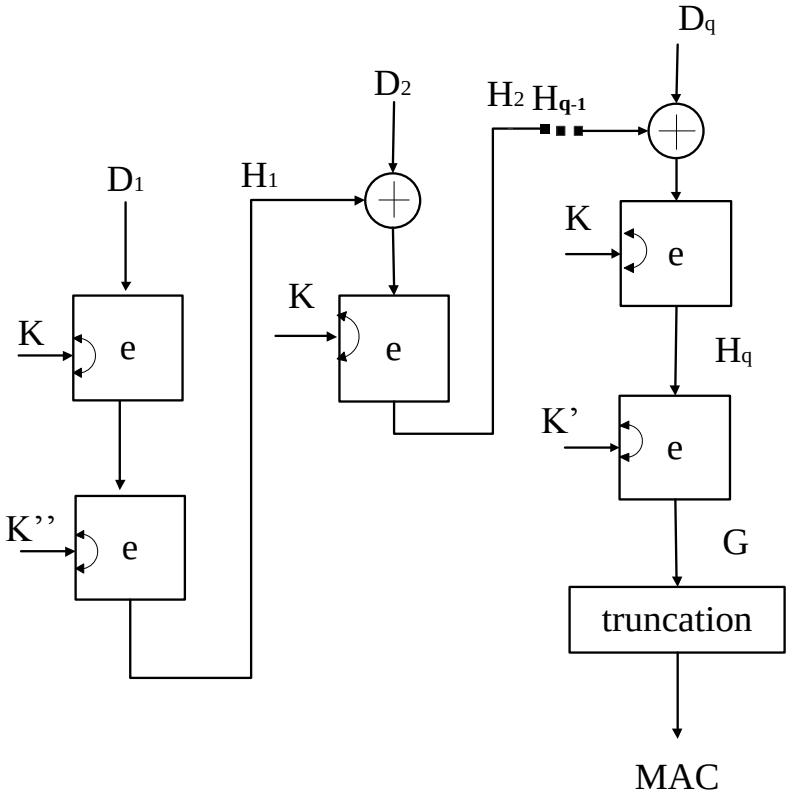


Рис. 3.32. **MAC-алгоритм 4**

MAC-алгоритм 5.

MAC-алгоритм 5 подібний MAC-алгоритму 1. Результатом є два проміжні значення: MAC1 і MAC2 відповідно. Ключ даного MAC-алгоритму складається із ключа блокового шифру K. Ключі K_1 і K_2, використовувані для першого й другого випадків, походять із ключа K. Значення K_1 і K_2 повинні бути різними.

Примітка. Приклад того, як одержати K_1 і K_2 з K – вибрати K_1 рівним K, а K_2 виходить доповненням, підрядків, що чергуються по чотири біти ключа K, починаючи з перших чотирьох біт. Інший приклад – одержати обидва K_1 і K_2 із загального еталонного ключа так, щоб вони були різні.

Кінцевий MAC виходить побітовим виключаючим АБО проміжних значень MAC1 і MAC2, тобто: MAC := MAC·

MAC-алгоритм 6.

MAC-алгоритм 6 подібний MAC-алгоритму 4, результатом якого є два проміжні значення: MAC1 і MAC2 відповідно. Ключ даного алгоритму складається із двох ключів блокового шифру K і K', які повинні бути обрані незалежно. Ключі (K_1,K_1') і (K_2,K_2'), використовувані для першого й другого етапів, відповідно повинні бути отримані із ключів (KK') так, щоб K_1 і K_1' були різними, K_2 і K_2' були різними й пари K_1 (K_1,K_1') і (K_2,K_2') були різними.

Приклад того, як одержати (K_1,K_1') і (K_2,K_2') з (KK') – обрати K_1 рівним $KK_1' = K_1$, а побудувати $K_2(K_2')$ можна доповненням підрядків, що чергуються за всіма бітами ключа $K_1(K_1')$, починаючи з перших восьми біт.

MAC-алгоритм 4 усередині використовує отриманий ключ K'', що означає, що MAC-алгоритм 6 використовує в загальному шість ключів блокового шифру (K_1,K_1',K_1'') і (K_2,K_2',K_2''). Рекомендується перевірити, щоб усі ці ключі були різними.

Число блоків у заповненому варіанті рядка даних повинне бути більше або дорівнює двом, тобто, $q \geq 2$.

Кінцевий MAC виходить побітовим, що виключають АБО проміжних значень MAC1 і MAC2, тобто: $MAC := MAC_.$

Аналіз захищеності MAC-алгоритмів.

Припустимо, що довжина ключа блокового шифру дорівнює k біт, а довжина ключа MAC-алгоритму дорівнює k^* біт. Значення k^*, таким чином, дорівнює k або 2^k.

У цьому додатку MAC k(D) позначає MAC для рядка D, обчислений, використовуючи ключ MAC-алгоритму K.

Для того, щоб визначити рівень безпеки MAC-алгоритму, розглядаються дві стратегії атаки:

Підроблена атака.

Ця атака складається із прогнозованого значення MAC k(D) для рядка даних D без первісного знання K. Якщо зловмисник може зробити

це для єдиного рядка даних, його називають здатним до підробки. Фактичні атаки часто вимагають, щоб зловмисник був таким, що перевіряється, тобто, щоб підроблений MAC був відомий і коректний заздалегідь із імовірністю, близькою до 1. Більше того, у більшості додатків, рядок даних має певний формат, який накладає додаткові обмеження на рядок даних D.

Атака відновлення ключа.

Ця атака складається зі знаходження самого ключа MAC-алгоритму K з пар чисел рядок даних/MAC. Така атака могутніша порівняно з підробленою атакою, оскільки вона дозволяє довільні атаки.

Здійснюваність атаки залежить від числа необхідних пар відомих і обраних рядка даних/MAC, а також від числа off-line шифрувань.

Можливі атаки проти MAC-алгоритмів описані нижче; немає гарантії, що цей список вичерпний. Перші дві атаки – загальні, тобто застосовуються до будь-якого MAC-алгоритму. Наступні атаки застосовуються до ітераційного MAC-алгоритму. Наступні три атаки, застосовувані конкретно для одного або більше MAC-алгоритмів, описаних в розділі ISO/IEC 9797 (докладніше див. [8; 11; 12; 14; 15; 16]).

Вгадування MAC.

Це підробка, яка не може бути перевірена і має ймовірність успіху $\max(1/2^m, 1/2^{k^*})$. Ця атака застосовується до всіх MAC-алгоритмів і може бути попереджена розумним вибором m і k.

Відновлення ключа з використанням грубої сили.

Для цієї атаки в середньому потрібно 2^{k^*-1} операцій; перевірка даної атаки вимагає пар рядка даних/MAC порядку k^*/k. Ця атака також застосовується до всіх MAC-алгоритмів. Може бути попереджена розумним вибором значення k^*. У якості альтернативи, можна запобігти одержанню пар k^*/k рядка даних/MAC, які необхідні для однозначної ідентифікації ключа. Наприклад, якщо $k^* = 64$ і $m = 32$, приблизно 2^{32} ключів відповідають заданій парі рядка даних/MAC; якщо ключ змінюється залежно від рядка даних, відновлення ключа з використанням грубої сили є не більш ефективним, ніж вгадування значення MAC.

Підробка дня народження [14; 16].

Якщо зловмисник знає достатнє число пар рядка даних/MAC, він чекає, щоб знайти два рядки даних D і D', таких що MAC к(D) = MAC к(D'),

а також, щоб значення H_q в обох обчисленнях були рівними; це називається *внутрішня колізія*. Якщо D і D' утворюють внутрішню колізію, MAC к(D||Y) = MAC к(D'||D) для будь-якого рядка Y. Це дозволяє атаку одного обраного рядка даних, коли зловмисник може спрогнозувати MAC для D'||D після спостереження за MAC, що відповідає D||Y. Ця підробка рядків даних визначених форм, яка стосується не всіх додатків, але слід відзначити, що розширення цієї атаки існують, які дозволяють більшу гнучкість у рядках даних. Ця атака вимагає один обраний рядок даних і приблизно $2^{n/2}$ відомих рядків даних, а також 2^{n-m} обраних рядків даних.

Атака підробки дня народження може бути відвернена комбінацією таких заходів: використання методу заповнення 3, приєднання попереду блоку до рядка даних, який містить порядковий номер і робить MAC обчислення оголошеним. Це означає, що ця реалізація повинна гарантувати, щоб кожний порядковий номер використовувався тільки один раз для MAC обчислення протягом часу життя ключа. Це не здійсненне у всіх оточеннях. Якщо тільки одне із двох заходів використовується, то варіанти основної атаки підробки дня народження використовуються як раніше й мають подібні складності; наприклад, якщо метод заповнення 3 використовується без інших заходів, то один варіант атаки (заснований на Лемі 1 з [16]) дня народження вимагає обраних $2^{n/2}$ відкритих текстів.

Тривіальна підробка.

Якщо використовується метод заповнення 1, зловмисник може типово додати або вилучити число замикаючих (задніх) "0" біт рядка даних, не змінюючи MAC. Мається на увазі, що метод заповнення 1 повинен використовуватися тільки в оточеннях, де довжина рядка даних D відома сторонам заздалегідь, або, де рядок даних з різною кількістю замикаючих "0" біт мають ту ж семантику.

Підробка XOR.

Якщо використовується MAC-алгоритм 1 з методом заповнення 1 або 2, а також m = n, те можлива проста XOR-підробка. Припустимо, для простоти, що D або його заповнена версія $\overline{D}$ складається з єдиного блоку. Припустимо, що відомо MAC к(D); із цього випливає, що:

$$MACk(\overline{D} \,||\, (D \oplus MACk(D))) = MACk(D).$$

Це означає, що можливо сконструювати нове повідомлення з тим же значенням MAC, яке є підробкою. Отже, ця атака застосовується навіть якщо ключ MAC-алгоритму використовується тільки один раз. Якщо відомо MAC к(D) і MAC к(D'), подібне обчислення показує, що:

$$MACk(\overline{D} \,||\, (D' \oplus MACk(D))) = MACk(D').$$

Якщо відомо MAC к(D), MAC к(D||Y) і MAC к(D'), то відомо, що $MACk(D' || Y') = MACk(D || Y)$, якщо $Y' = Y \oplus MACk(D) \oplus MACk(D')$ (якщо D і Y потрапляють на межі блоку). Це також дозволяє підробку, коли зловмисник може підробити MAC на $D' || Y'$ у даному значенні MAC-ів для двох відомих рядків даних і одного обраного рядка даних. Отже, перераховані підробки є на рядках даних певної форми, які можуть не відноситися до додатків.

Ця атака може бути відвернена використанням методу заповнення 3.

Ця атака може бути розширена для випадку m < n, але вона стає більш складнішою: у цьому випадку, вона вимагає знання MAC-ів для додаткових $2^{(n-m)/2}$ обраних рядків даних [11].

Ця ж атака застосовується, коли використовується MAC-алгоритм 2 із двома рівними ключами, тобто, K' = K. У цьому випадку дана атака працює, коли Y містить, принаймні, два блоки й перші n біт Y є "0" бітами.

Скорочене відновлення ключа.

Деякі MAC-алгоритми є потенційно вразливими до атак відновлення ключа, заснованим на внутрішній колізії. Прикладами є MAC-алгоритм 3 [12; 15] і MAC-алгоритм 4 у комбінації з методом заповнення 1 або 2 [9] або методом заповнення 3 [10].

Наступні таблиці надають порівняльну характеристику рівня безпеки MAC-алгоритмів, описаних у цьому розділі ISO/IEC 9797. Передбачається, що блоковий шифр не має слабких сторін. Табл. 3.1 показує головні властивості MAC-алгоритмів. Оскільки метод заповнення 1 дозволяє тривіальну підробку, порівняння розглядає тільки методи заповнення 2 і 3. Табл. 3.2 і 3.3 представляють найбільш відомі атаки блокових шифрів з n = 64 і k = 56 (тобто, алгоритм шифрування даних [5]). Більшість цих атак – атаки скороченого відновлення ключа на MAC-алгоритм 3 m = 32 (число 3 у табл. 3.3) виходять неопублікованим (але скоріше простим) розширенням відповідних атак у табл. 3.2. Табл. 3.4 і 3.5 – для блокових шифрів з n =128 і k = 128. У цьому випадку розглядаються тільки MAC-алгоритми 1, 2 і 5, тому що немає необхідності подвоювати довжину ключа MAC-алгоритму. Атака описується як

кортеж – четвірка $[\alpha\beta,\beta\delta]$, де α позначає число off-line шифрувань блокового шифру, β позначає число відомих пар рядок даних/MAC, γ позначає число обраних пар рядок даних/MAC, а δ позначає кількість on-line MAC перевірок.

Таблиця 3.1

Властивості MAC-алгоритмів. Ключі позначають кількість ключів незалежного блокового шифру. Ефективність позначає кількість шифрувань, щоб обробити рядок даних з t_n біт

MAC-алгоритм	Вхідний перетворювач	Вихідний перетворювач	Заповнення	№ Ключа	Ефективність
1	1	1	2	1	t+1
1	1	1	3	1	t+2
2	1	2	2	1	t+2
2	1	2	2	2	t+2
3	1	3	2	2	t+3
4	2	2	2	2	t+3
4	2	2	3	2	t+4
5	1	1	2	1	2t+2
6	2	2	2	2	2t+6

Таблиця 3.2

Оцінені рівні безпеки для n = 64, k = 56, m = 64; рівень безпеки визначається чотирма числами: off-line шифруваннями блокового шифру, відомими парами рядок даних/MAC, обраними парами рядок даних/MAC, а також on-line MAC перевірками

Відновлення ключа		Підробка		
Груба сила	Скорочене	Тривіальна	XOR	День народження
$[2^{56},1,0,0]$		$[0,0,0,2^{56}]$	$[0,1,0,0]$	$[0,2^{32},1,0]\uparrow$
$[2^{56},1,0,0]$		$[0,0,0,2^{56}]$		$[0,2^{32},1,0]\uparrow$
$[2^{56},1,0,0]$		$[0,0,0,2^{56}]$		$[0,2^{32},1,0]\uparrow$
$[2^{112},1,0,0]$	$[2^{57},2,0,0]$	$[0,0,0,2^{64}]$		$[0,2^{32},1,0]\uparrow$
$[2^{112},2,0,0]$	$[2^{57},2^{32},0,0]$ $[2^{56},1,0,2^{56}]$	$[0,0,0,2^{64}]$ $[0,1,0,2^{56}]$		$[0,2^{32},1,0]\uparrow$
$[2^{112},2,0,0]$	$[2^{58},2^{32},2,0]\uparrow$ $[2^{58},1,1,2^{56}]\uparrow$	$[0,0,0,2^{64}]$ $[2^{58},1,1,2^{56}]\uparrow$		$[0,2^{32},1,0]\uparrow$
$[2^{112},2,0,0]$	$[2^{57},2^{32},2^{56},0]$	$[0,0,0,2^{64}]$		$[0,2^{32},1,0]\uparrow$
$[2^{56},1,0,0]$		$[0,0,0,2^{56}]$		$[0,2^{32},1,0]\uparrow$
$[2^{112},2,0,0]$		$[0,0,0,2^{64}]$		$[0,2^{64},2^{64},0]\uparrow$

Може бути відвернене шляхом приєднання спочатку порядкового номера до рядка даних у комбінації з методом заповнення 3.

Таблиця 3.3

Оцінені рівні безпеки для n = 64, k = 56, m = 32; рівень безпеки визначається чотирма числами: off-line шифруваннями блокового шифру, відомими парами рядок даних/MAC, обраними парами рядок даних/MAC, а також on-line MAC перевірками

Відновлення ключа		Підробка		
Груба сила	Скорочене	Тривіальна	XOR	День народження
$[2^{56},2,0,0]$		$[0,0,0,2^{32}]$	$[0,2,2^{16},0]$	$[0,2^{32},2^{32},0]$
$[2^{56},2,0,0]$		$[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{56},2,0,0]$		$[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{112},4,0,0]$	$[2^{57},2^{32},2^{32},0]$ $[2^{88},4,0,0]$	$[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{112},4,0,0]$	$[2^{57},2^{32},2^{32},0]\uparrow$ $[2^{89},2^{32},0,0]$ $[2^{56},2,0,2^{56}]$	$[0,0,0,2^{32}]$ $[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{112},4,0,0]$	$[2^{78},2^{32},2^{56},0]$	$[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{112},4,0,0]$	$[2^{78},2^{32},2^{56},0]$	$[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{56},2,0,0]$		$[0,0,0,2^{32}]$		$[0,2^{32},2^{32},0]$
$[2^{112},4,0,0]$		$[0,0,0,2^{32}]$		$[0,2^{64},2^{96},0]$

Може бути відвернене шляхом приєднання спочатку порядкового номера до рядка даних у комбінації з методом заповнення 3.

Таблиця 3.4

Оцінені рівні безпеки для n = 128, k = 128, m = 64; рівень безпеки визначається чотирма числами: off-line шифруваннями блокового шифру, відомими парами рядок даних/MAC, обраними парами рядок даних/MAC, а також on-line MAC перевірками

Відновлення ключа		Підробка		
Груба сила	Скорочене	Тривіальна	XOR	День народження
$[2^{128},2,0,0]$		$[0,0,0,2^{64}]$	$[0,2,2^{32},0]$	$[0,2^{64},2^{64},0]\uparrow$
$[2^{128},2,0,0]$		$[0,0,0,2^{64}]$		$[0,2^{64},2^{64},0]\uparrow$
$[2^{128},2,0,0]$		$[0,0,0,2^{64}]$		$[0,2^{64},2^{64},0]\uparrow$
$[2^{128},2,0,0]$		$[0,0,0,2^{64}]$		$[0,2^{64},2^{64},0]\uparrow$

Може бути відвернене шляхом приєднання спочатку порядкового номера до рядка даних у комбінації з методом заповнення 3.

Таблиця 3.5

Оцінені рівні безпеки для n = 128, k = 128, m = 32; рівень безпеки визначається чотирма числами: off-line шифруваннями блокового шифру, відомими парами рядок даних/MAC, обраними парами рядок даних/MAC, а також on-line MAC перевірками

Відновлення ключа		Підробка		
Груба сила	Скорочене	Тривіальна	XOR	День народження
$[2^{128},4,0,0]$		$[0,0,0,2^{32}]$	$[0,2,2^{48},0]$	$[0,2^{64},2^{96},0]\uparrow$
$[2^{128},4,0,0]$		$[0,0,0,2^{32}]$		$[0,2^{64},2^{96},0]\uparrow$
$[2^{128},4,0,0]$		$[0,0,0,2^{32}]$		$[0,2^{64},2^{96},0]\uparrow$
$[2^{128},4,0,0]$		$[0,0,0,2^{32}]$		

Може бути відвернене шляхом приєднання спочатку порядкового номера до рядка даних у комбінації з методом заповнення 3.

У [79; 118] нижня межа підтверджується на рівні безпеки CBC-MAC, заснованому на визначених властивостях блокового шифру. Мається на увазі, що множина обговорених атак дня народження є близькою до найкращих можливих атак за умови, що блоковий шифр є сильним. Посилання [13] забезпечує доказ безпеки MAC-алгоритму 2.

Логічне обґрунтування.

Ця частина пояснює вибір MAC-алгоритмів у цьому розділі ISO/IEC 9797. Важливим фактором виборі пропонованого механізму є збереження минулої сумісності з попередніми ANSI, ISO і ISO/IEC стандартами [118]. Порівнюючи зі старими версіями, число схем заповнення було збільшено із двох до трьох. Третя схема заповнення забезпечує опір проти визначених атак за низькою ціною, особливо в додатках, де не проблема приєднати попереду довжину до інформації.

Були включено три нові схеми:

MAC-алгоритм 4 забезпечує поліпшений спосіб збільшення довжини ключа. Наполегливо рекомендується використовувати цей алгоритм у комбінації з методом заповнення 3; у цьому випадку він забезпечує кращу безпеку, ніж MAC алгоритм 3 у порівнянній вартості;

MAC-алгоритм 5 забезпечує збільшену стійкість від атак підробки дня народження;

MAC-алгоритм 6 забезпечує як збільшену довжину ключа, так і збільшену стійкість від атак підробки дня народження.

У той час, як існують кілька альтернативних шляхів поліпшити основний CBC-MAC, запропоновані розв'язки не тільки забезпечують додаткову безпеку, а й відносно прості, що робить їх простими для аналізу й реалізації.

Слід навести приклади генерації MAC, використовуючи DEA (див. ANSIX3.92). Відкриті тексти – 7-бітові ASCII коди (без контролю за парністю) для рядка даних 1: "Now␣is␣the␣time␣for␣all" і рядка даних 2: "Now␣is␣the␣time␣for␣it", де "␣" символізує пробіл. ASCII кодування аналогічне кодуванню, що використовує ISO 646. Використовувані значення двох ключів – = 0123456789 (шістнадцятирічна система) і = 9876543210 (шістнадцятирічна система). Біти парності ключа ігноруються. Отримані ключі обчислюються доповненням, підрядків, що чергуються по чотири біти ключа, починаючи з перших чотирьох біт. Для MAC-алгоритмів 1, 2, 3 і 4 = 32, а для MAC-алгоритмів 5,6 і 4 = 64. Усі значення записані в шістнадцятирічній нотації.

Для рядка даних 1 результати трьох методів заповнення (набивання) такі:

Рядок даних 2 з методом заповнення 2

key (K''')	F1 D3 B5 97 79 5B 3D 1F
G	17 36 AC 1A 63 63 0E FB

MAC = 17 36 AC 1A.

Рядок даних 2 з методом заповнення 3

key (K''')	F1 D3 B5 97 79 5B 3D 1F
G	05 38 26 96 27 4F B4 F0

MAC = 05 38 26 96.

MAC-алгоритм 3

Перші q-кроків ідентичні до MAC-алгоритму 1. Єдина різниця – вихідне перетворення 3 застосовується замість вихідного перетворення 1.

Рядок даних 1 з методом заповнення 1

key (K')	FE DC BA 98 76 54 32 10
Output of d G	B4 8D 36 EC 7A D5 69 4F A1 C7 2E 74 EA 3F A9 B6

MAC = A1 C7 2E 74.

Рядок даних 1 з методом заповнення 2

key (K')	FE DC BA 98 76 54 32 10
Output of d G	79 53 7F EE I8 CF 18 93 E9 08 62 30 CA 3B E7 96

MAC = E9 08 62 30.

Рядок даних 1 з методом заповнення 3

key (K')	FE DC BA 98 76 54 32 10
Output of d G	FE B3 B9 66 ID BE DE CD AB 05 94 63 D7 A7 D1 70

MAC = AB 05 94 63.

Рядок даних 2 з методом заповнення 1

key (K')	FE DC BA 98 76 54 32 10
Output of d G	32 8A C7 8B A1 CA 0B 3F 2E 2B 14 28 CC 78 25 4F

MAC = 2E 2B 14 28.

Рядок даних 2 з методом заповнення 2

key (K')	FE DC BA 98 76 54 32 10
Output of d G	7A 71 AF 2F 5D 15 40 A7 5A 69 2C E6 4F 40 41 45

MAC = 5A 69 2C E6.

Рядок даних 2 з методом заповнення 3

key (K')	FE DC BA 98 76 54 32 10
Output of d G	20 97 B4 05 F1 9E 2D D8 C5 9F 7E ED 32 8D DD 69

MAC = C5 9F 7E ED.

MAC-алгоритм 4

Рядок даних 1 з методом заповнення 1

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
Output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A9 DD 09 4C
H_3	7B 93 0A AE 67 4A C9 24
G	AD 35 02 B7 AC 4A 48 A0

MAC = AD 35 02 B7.

Рядок даних 1 з методом заповнення 2

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
Output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A9 DD 09 4C
H_3	7B 93 0A AE 67 4A C9 24
$D_3 \oplus H_3$	FB 93 0A AE 67 4A C9 24
H_4	26 C4 FA D7 2E 6D D3 A2
G	61 C3 33 E3 42 C5 53 7C

MAC = 61 C3 33 E3.

Рядок даних 1 з методом заповнення 3

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
Output of e	4B B5 82 65 DD 87 B3 05
H_1	71 5A F8 BE DA BE 90 44
$D_2 \oplus H_1$	3F 35 8F 9E B3 CD B0 30
H_2	50 2A 04 42 6A 80 B6 0B
$D_3 \oplus H_2$	38 4F 24 36 03 ED D3 2B
H_3	AF 13 8C 54 99 9B 84 30
$D_3 \oplus H_3$	C9 7C FE 74 F8 F7 E8 10
H_4	7F 90 05 61 B4 2C CE D2
G	95 2A F8 38 98 9B 5C 00

MAC = 95 2A F8 38.

Рядок даних 2 з методом заповнення 1

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
Output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A1 C5 65 6C
H_3	21 FC 35 F2 B2 26 6C 9A
G	05 F1 08 4C 1D E3 A3 3D

MAC = 05 F1 08 4C.

Рядок даних 2 з методом заповнення 2

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
Output of e	3F A4 0E 8A 98 4D 48 15
H_1	EA F0 4B F5 31 ED 33 5E
$D_2 \oplus H_1$	82 95 6B 81 58 80 56 7E
H_2	7E 7F 98 A0 C8 B1 65 6C
$D_3 \oplus H_2$	18 10 EA 80 A1 C5 E5 6C
H_3	8F 76 9B 55 48 42 23 FD
G	A1 BC 09 31 52 BB 3E 0F

MAC = A1 BC 09 31.

Рядок даних 2 з методом заповнення 3

key (K)	01 23 45 67 89 AB CD EF
key (K')	FE DC BA 98 76 54 32 10
key (K'')	0E 2C 4A 68 86 A4 C2 E0
Output of e	DF 9C D6 EA 7E 5A E1 62
H_1	82 61 94 52 C7 6D 04 F1
$D_2 \oplus H_1$	CC 0E E3 72 AE 1E 24 85
H_2	ED 33 1C 07 37 D6 B8 26
$D_3 \oplus H_2$	85 56 3C 73 5E BB DD 06
H_3	7C A1 DE 70 BB 1F 7F 07
$D_3 \oplus H_3$	1A CE AC 50 D2 6B 7F 07
H_4	40 B7 45 2E F3 CF 71 49
G	AF DE E0 F9 50 39 66 3D

MAC = AF DE E0 F9.

MAC-алгоритм 5.

Рядок даних 1 з методом заповнення 1

key (K_1) MAC_1	01 23 45 67 89 AB CD EF 70 A3 06 40 CC 76 DD 8B
key (K_2) MAC_2	FE DC BA 98 76 54 32 10 84 47 04 F6 7B 5A CE 9C
$MAC_1 \oplus MAC_2$	F4 E4 02 B6 B7 2C 13 17

MAC = F4 E4 02 B6 B7 2C 13 17.

Рядок даних 1 з методом заповнення 2

key (K_1) MAC_1	01 23 45 67 89 AB CD EF 10 E1 F0 F1 08 34 1B 6D
key (K_2) MAC_2	FE DC BA 98 76 54 32 10 60 11 AE 38 EC C3 34 F4
$MAC_1 \oplus MAC_2$	70 F0 5E C9 E4 F7 2F 99

MAC = 70 F0 5E C9 E4 F7 2F 99.

Рядок даних 1 з методом заповнення 3

key (K_1) MAC_1	01 23 45 67 89 AB CD EF 2C 58 FB 8F F1 2A AE AC
key (K_2) MAC_2	FE DC BA 98 76 54 32 10 FA 47 AA 7D 1B 00 83 CF
$MAC_1 \oplus MAC_2$	D6 1F 51 F2 EA 2A 2D 63

MAC = D6 1F 51 F2 EA 2A 2D 63.

Рядок даних 2 з методом заповнення 1

key (K_1) MAC_1	01 23 45 67 89 AB CD EF E4 5B 3A D2 B7 CC 08 56
key (K_2) MAC_2	FE DC BA 98 76 54 32 10 EB 7F 87 76 1B EE 07 19
$MAC_1 \oplus MAC_2$	0F 24 BD A4 AC 22 0F 4F

MAC = 0F 24 BD A4 AC 22 0F 4F.

Рядок даних 2 з методом заповнення 2

key (K_1) MAC_1	01 23 45 67 89 AB CD EF A9 24 C7 21 36 14 92 11
key (K_2) MAC_2	FE DC BA 98 76 54 32 10 49 20 D4 60 AC E8 84 1A
$MAC_1 \oplus MAC_2$	E0 04 13 41 9A FC 16 0B

MAC = E0 04 13 41 9A FC 16 0B.

Рядок даних 2 з методом заповнення 3

key (K_1) MAC_1	01 23 45 67 89 AB CD EF B1 EC D6 FC 8B 37 C3 92
key (K_2) MAC_2	FE DC BA 98 76 54 32 10 6C 33 88 2F 84 2F 28 6E
$MAC_1 \oplus MAC_2$	DD DF 5E D3 0F 18 EB FC

MAC = DD DF 5E D3 0F 18 EB FC.

MAC-алгоритм 6.

Рядок даних 1 з методом заповнення 1

key (K_1) key ($K_1^{/}$) key ($K_1^{//}$) MAC_1	01 23 45 67 89 AB CD EF FE DC BA 98 76 54 32 10 0E 2C 4A 68 86 A4 C2 E0 AD 35 02 B7 AC 4A 48 A0
key (K_2) key ($K_2^{/}$) key ($K_2^{//}$) MAC_2	FE 23 BA 67 76 AB 32 EF 01 DC 45 98 89 54 CD 10 F1 2C B5 68 79 A4 3D E0 FA 4B F0 96 B4 84 15 1A
$MAC_1 \oplus MAC_2$	57 7E F2 21 18 CE 5D BA

MAC = 57 7E F2 21 18 CE 5D BA.

Рядок даних 1 з методом заповнення 2

key (K_1)	01 23 45 67 89 AB CD EF
key (K_1')	FE DC BA 98 76 54 32 10
key (K_1'')	0E 2C 4A 68 86 A4 C2 E0
MAC_1	61 C3 33 E3 42 C5 53 7C
key (K_2)	FE 23 BA 67 76 AB 32 EF
key (K_2')	01 DC 45 98 89 54 CD 10
key (K_2'')	F1 2C B5 68 79 A4 3D E0
MAC_2	01 B7 53 5B 9A 05 AE 86
$MAC_1 \oplus MAC_2$	60 74 60 B8 D8 C0 FD FA

MAC = 60 74 60 B8 D8 C0 FD FA.

Рядок даних 1 з методом заповнення 3

key (K_1)	01 23 45 67 89 AB CD EF
key (K_1')	FE DC BA 98 76 54 32 10
key (K_1'')	0E 2C 4A 68 86 A4 C2 E0
MAC_1	95 2A F8 38 98 9B 5C 00
key (K_2)	FE 23 BA 67 76 AB 32 EF
key (K_2')	01 DC 45 98 89 54 CD 10
key (K_2'')	F1 2C B5 68 79 A4 3D E0
MAC_2	68 17 43 56 69 FE 5B 54
$MAC_1 \oplus MAC_2$	FD 3D BB 6E F1 65 07 54

MAC = FD 3D BB 6E F1 65 07 54.

Рядок даних 2 з методом заповнення 1

key (K_1)	01 23 45 67 89 AB CD EF
key (K_1')	FE DC BA 98 76 54 32 10
key (K_1'')	0E 2C 4A 68 86 A4 C2 E0
MAC_1	05 F1 08 4C 1D E3 A3 3D
key (K_2)	FE 23 BA 67 76 AB 32 EF
key (K_2')	01 DC 45 98 89 54 CD 10
key (K_2'')	F1 2C B5 68 79 A4 3D E0
MAC_2	15 06 4F 9D 52 91 61 14
$MAC_1 \oplus MAC_2$	10 F7 47 D1 4F 72 C2 29

MAC = 10 F7 47 D1 4F 72 C2 29.

Рядок даних 2 з методом заповнення 2

key (K_1)	01 23 45 67 89 AB CD EF
key (K_1')	FE DC BA 98 76 54 32 10
key (K_1'')	0E 2C 4A 68 86 A4 C2 E0
MAC_1	A1 BC 09 31 52 BB 3E 0F
key (K_2)	FE 23 BA 67 76 AB 32 EF
key (K_2')	01 DC 45 98 89 54 CD 10
key (K_2'')	F1 2C B5 68 79 A4 3D E0
MAC_2	13 27 93 47 8F A7 07 ID
$MAC_1 \oplus MAC_2$	B2 9B 9A 76 DD 1C 39 12

MAC = B2 9B 9A 76 DD 1C 39 12.

Рядок даних 2 з методом заповнення 3

key (K_1)	01 23 45 67 89 AB CD EF
key ($K_1^{/}$)	FE DC BA 98 76 54 32 10
key ($K_1^{//}$)	0E 2C 4A 68 86 A4 C2 E0
MAC_1	AF DE E0 F9 50 39 66 3D
key (K_2)	FE 23 BA 67 76 AB 32 EF
key ($K_2^{/}$)	01 DC 45 98 89 54 CD 10
key ($K_2^{//}$)	F1 2C B5 68 79 A4 3D E0
MAC_2	59 9B 1B 84 ID 73 24 89
$MAC_1 \oplus MAC_2$	F6 45 FB 7D 4D 4A 42 B4

MAC = F6 45 FB 7D 4D 4A 42 B4.

Алгоритм автентифікації повідомлень (MAA)

Алгоритм MAA був розроблений за замовленням Банківського автоматизованого клірингового центра Великобританії Національною фізичною лабораторією Великобританії в 1983 р. Даний алгоритм формування MAC-коду орієнтований і оптимізований під застосування в 32-розрядних ПК. Схема алгоритму наведена на рис. 3.28. Показано, що MAA має недоліки при застосуванні до довгих повідомлень. У зв'язку з цим у ISO 8731-2 був визначений спеціальний режим роботи MAA для повідомлень довжиною більше 1024 біт [124].

Алгоритм автентифікації повідомлень.

Вхід: рядок даних x довжиною 32 j біт, $1 \le j \le 10^6$; секретний 64-розрядний MAC ключ $Z = Z[1]...Z[8]$.

Вихід: 32-розрядний MAC-код за рядком даних *x*.

1. *Розширення ключа і повідомлення*. Розширити ключ *Z* у шістьох 32-х розрядних величин X, Y, V, W, S, T, де X,Y – значення, які ініціалізують; V, W – змінні основного циклу; S, T – розширення повідомлення. Розширення здійснюється таким чином:

1.1. Спочатку заміняють усі байти 00_x і ff_x в ключі *Z* за таким правилом:

$P = 0;$

для і від 1 до 8 робити

$\{P = 2P;$

якщо $Z[i] = 00_x$ або ff_x то

$\{P = P + 1; Z[i] = Z[i] OR\ P\}\};$

1.2. Нехай J і K будуть, відповідно, першими та останніми чотирма байтами ключа Z. Обчислити:

$$X = J^4 (mod 2^{32} - 1) \oplus J^4 (mod 2^{32} - 2);$$

$$X = J^4 (mod 2^{32} - 1) \oplus J^4 (mod 2^{32} - 2);$$

$$Y = [K^5 (mod 2^{32} - 1) \oplus K^5 (mod 2^{32} - 2)](1 + P)^2 (mod 2^{32} - 2);$$

$$V = J^6 (mod 2^{32} - 1) \oplus J^6 (mod 2^{32} - 2);$$

$$W = K^7 (mod 2^{32} - 1) \oplus K^7 (mod 2^{32} - 2);$$

$$S = J^8 (mod 2^{32} - 1) \oplus J^8 (mod 2^{32} - 2);$$

$$T = K^9 (mod 2^{32} - 1) \oplus K^9 (mod 2^{32} - 2).$$

1.3. Обробити три отримані пари (X, Y),(V, W),(S, T) шляхом заміни усіх байтів 00_x і ff_x, як описано в п. 1.1. для ключа Z. Визначити константи для операцій AND – OR:

$$A = 02040801_x, B = 008004021_x, C = bfef7fdf_x, D = 7dfefbff_x.$$

2. *Ініціалізація і попередні обчислення*.

Ініціалізувати:

вектор циклічного зсуву: $v = V$

змінні зчеплення: $H_1 = X; H_2 = Y.$

Додати до рядка x значення S, T (похідні від ключа). Нехай $X_i \ldots X_t$ позначає результуючий прирощений сегмент 32-бітних блоків. Останні два блоки сегмента, таким чином, включають похідні від ключа дані.

3. *Обробка блоку*.

Обробити кожен 32-бітний блок x_i для і від1 до t таким чином:

$$v = (v << 1), U = (v \oplus W);$$

$$t_1 = (H_1 \oplus x_i)x_1(((H_2 \oplus x_i) + U)OR\ A)\ AND\ C);$$

$$t_2 = (H_2 \oplus x_i)x_2(((H_1 \oplus x_i) + U)\ OR\ B)\ AND\ D);$$

$$H_1 = t_1, H_2 = t_2,$$

де x_1 – множення за $\bmod 2^{32} - 1$;

x_2 – множення за $\bmod 2^{32} - 2$;

$+$ – додавання за $\bmod 2^{32}$;

$<<1$ – циклічний зсув ліворуч на один біт.

Операції AND-OR оперують на 32-розрядних величинах.

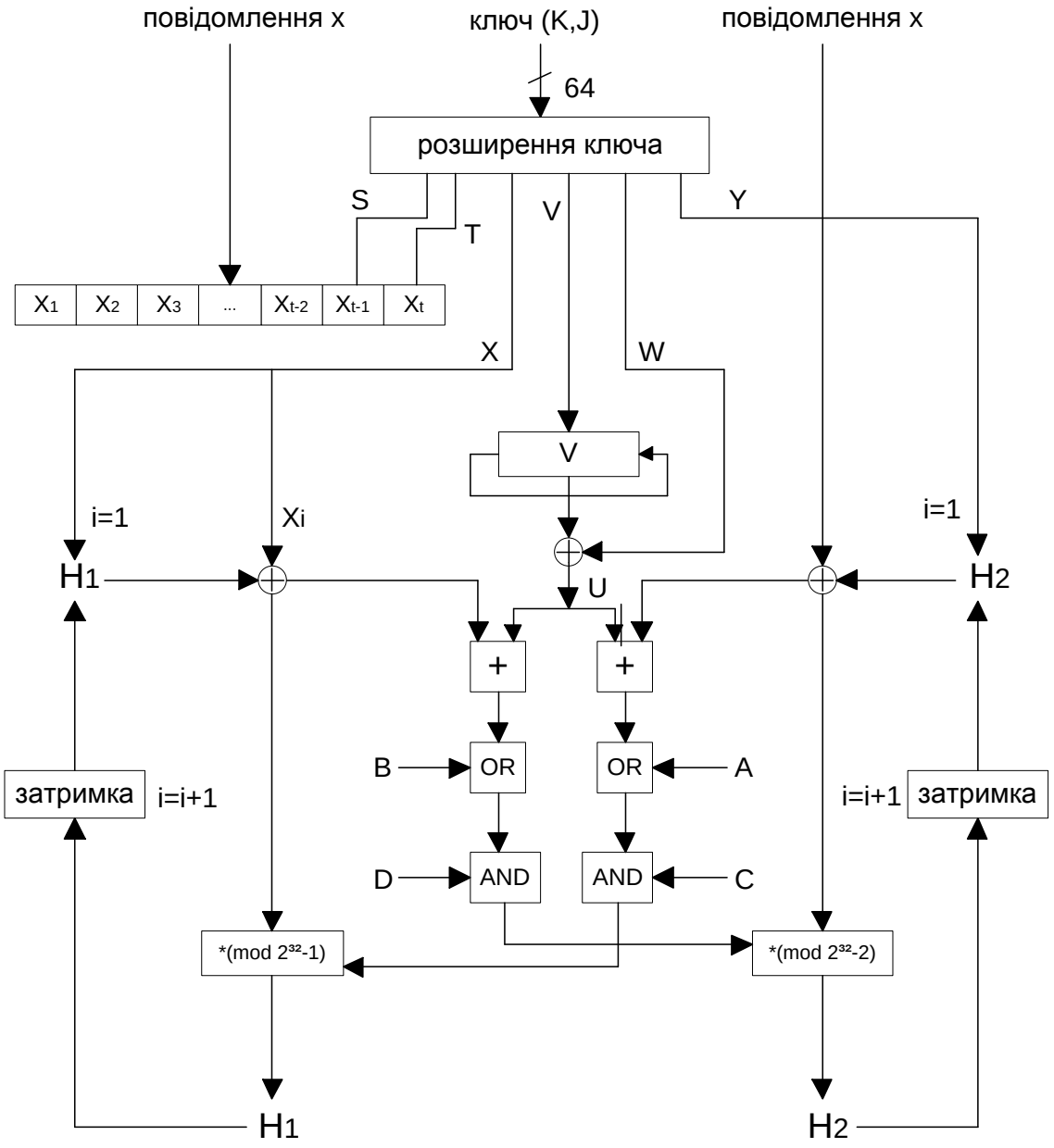


Рис. 3.28. **Алгоритм автентифікації повідомлень**

4. *Кінцева обробка.* Результуючий MAC-код є величина $H = H_1 \oplus H_2$.

Оскільки відносно складна процедура розширення ключа не залежить від повідомлення і може виконуватися тільки один раз для фіксованого значення ключа, це в цілому не знижує ефективність роботи алгоритму. Операції, що перемішують, (арифметика за $\mathrm{mod} 2^{32} - 1$ і $\mathrm{mod} 2^{32} - 2$; XOR; нелінійні обчислювання AND-OR) вводяться для підвищення стійкості алгоритму проти криптоаналітичних атак

3.4.2. На основі безключових геш-функцій

Стандарт ISO/IEC 9797-2. Формування MAC-коду з використанням геш-функції.

Друга частина ISO/IEC 9797-2 містить опис трьох різних методів побудови MAC-коду на основі безключових геш-функцій. У кожному методі рекомендується використовувати одну з трьох геш-функцій, обумовлених стандартом ISO/IEC 10118-3, тобто SHA-1, RIPEMD-128 і RIPEMD-160. Таким чином, ISO/IEC 9797-2 визначає дев'ять різних алгоритмів формування MAC-коду [108; 119].

На перший погляд можна сформувати MAC-код шляхом простої конкатенації секретного ключа з даними, по яких виробляється MAC-код, і такого гешування результату. У цьому випадку можливі три методи формування MAC-коду:

префіксний метод, коли ключ приєднається до початку вихідного рядка даних;

суфіксний метод, коли ключ приєднується до кінця вихідного рядка даних;

префіксно-суфіксний метод з додаванням, коли ключ приєднується в початок і кінець вихідного рядку даних.

Міжнародний стандарт ISO/IEC 9797-2:2002(E) описує механізми, що використовують спеціальну функцію.

ISO/IEC 9797-2: 2002(E) визначає три MAC-алгоритми, що використовують секретний ключ і геш-функцію (або його циклічну функцію (кругла, скорочуюча)) з n-бітовим результатом для обчислення m-бітових MAC [119]. Ці механізми можуть бути використані як механізми захисту даних для перевірки, що дані не були змінені неуповноваженим способом. Вони також можуть бути використані як механізми автенти-

фікації повідомлення, щоб гарантувати, що повідомлення було породжено сутністю, що володіє секретним ключем. Міцність механізму захисту даних і механізму автентифікації повідомлення залежить від довжини (у бітах) k і таємності ключа, від довжини (у бітах) n геш-коду, вироблюваного геш-функцією, від міцності геш-функції, від довжини (у бітах) m MAC, а також від певного механізму.

Три механізми, описані в цьому розділі ISO/IEC 9797, засновані на спеціальних геш-функціях, визначених в ISO/IEC 10118-3. Перший механізм, визначений у цьому розділі ISO/IEC 9797, загальновідомий як MDx-MAC [78]. Він викликає зроблену геш-функцію один раз, а також проводить невелику зміну в циклічній функції шляхом додавання ключа до додаткових констант у циклічній функції. Другий механізм, визначений у цьому розділі ISO/IEC 9797, загальновідомий як HMAC. Він викликає зроблену геш-функцію двічі. Третій механізм, визначений у цьому розділі ISO/IEC 9797, є варіантом MDx-MAC, який бере на вході тільки короткі рядки (у більшості 256 біт). Він пропонує більшу продуктивність додаткам, які працюють тільки з короткими вхідними рядками.

Цей розділ ISO/IEC 9797 може бути використаний у службі безпеки будь-яких архітектур безпеки, будь-якого процесу або додатка.

Цей розділ ISO/IEC 9797 використовує такі символи й нотацію, визначену в ISO/IEC 9797-1 [118]:

D, D' – рядки даних, що є вхідними до MAC-алгоритму;

m – довжина (у бітах) MAC;

q – кількість блоків у рядку даних D після процесу заповнення й розбивки;

$j \sim X$ – рядок, отриманий з рядка X, взявши крайні зліва j біт з X;

$X \oplus y$ – виключаюче АБО бітових рядків X и Y;

X||Y – зчеплення бітових рядків X і Y (у такому ж порядку);

:= – символ, що означає операцію "дорівняти", використовувану в процедурних специфікаціях

MAC-алгоритмів, де вона позначає, що значення рядка ліворуч від символу буде дорівнювати значенню вираження, що знаходиться праворуч від символу.

Для цілей цього розділу ISO/IEC 9797 використовуються такі символи й нотація:

$\overline{D}$ –заповнений рядок даних;

H – геш-функція;

h' – геш-функция h зі зміненими константами й IV;

$\overline{h}$ – спрощена геш-функція h без заповнення й додавання довжини. $\overline{h}$ повинна застосовуватися тільки до вхідних рядків з довжиною додатної цілої й кратної L_1;

H', H'' – рядок з L_2 біт, що використовуються при обчисленні MAC-алгоритму для зберігання проміжного результату;

IV, IV_1, IV_2 – ініціююче значення;

k – довжина (у бітах) ключа MAC-алгоритму;

K – секретний ключ MAC-алгоритму;

$K', K_1, K_2, \overline{K}, \overline{K}_1, \overline{K}_2$ – похідні секретні ключі MAC-алгоритму;

$\tilde{L}$ – бітовий рядок, що кодує довжину повідомлення в MAC-алгоритмі 3;

OPAD, IPAD – постійні рядки, використовувані в MAC-алгоритмі 2;

R, S_0, S_1, S_2 – постійні рядки, використовувані при обчисленні констант для MAC-алгоритму 1 і MAC-алгоритму 3;

T_0, T_1, T_2 – постійні рядки, використовувані при утворенні ключів для MAC-алгоритму 1 і MAC-алгоритму 3;

U_0, U_1, U_2 – постійні рядки, використовувані при утворенні ключів для MAC-алгоритму 1 і MAC-алгоритму 3;

φ' – циклічна функція з константами, що модифікуються;

$K_1[K_i]$ і – е слово 128- бітового рядка K_1, тобто

$$K_1 = K_1[0] \,||\, K_1[1] \,||\, K_1[2] \,||\, K_1[3].$$

Цей розділ ISO/IEC 9797 використовує такі символи й нотацію, визначені в ISO/IEC 10118-1:

H – геш-код;

IV – ініціююче значення;

L_x – довжина (у бітах) бітового рядка X.

Цей розділ ISO/IEC 9797 використовує такі символи й нотацію, визначені в ISO/IEC 10118-3:

C_i, C_i'' – постійні слова, використовувані в циклічних функціях;

L_1 – довжина (у бітах) першої із двох вхідних рядків до циклічної функції φ;

L – довжина (у бітах) другої із двох вхідних рядків до циклічної функції φ вихідного рядка із циклічної функції φ і IV;

φ – циклічна функція, тобто, якщо X та Y – бітові рядки довжин L_1 і L_2 відповідно, то $\varphi(X,Y)$ є рядком, отримані застосуванням φ до X та Y;

$\oplus$ – модуль 2^{32} додаткової операції, тобто, якщо A та B – слова, тоді $A \oplus B$ слово, отримане використанням A та B як бінарними представленнями цілих чисел, що й дають їх сумарний модуль 2^{32}, де результат обмежений і лежить у межах між 0 і $2^{32} - 1$ включно.

Вимоги.

Користувачі, що бажають використовувати MAC-алгоритм із розділу ISO/IEC 9797, повинні вибрати:

MAC-алгоритм із визначених операцій у пп. 6, 7 і 8;

спеціальну геш-функцію з тих функцій, які визначені в ISO/IEC 10118-3;

довжину (у бітах) m MAC.

Згода на ці підбори серед користувачів важливі для роботи механізму цілісності даних.

Для MAC-алгоритмів 1, 2 довжина m MAC повинна бути додатнім цілим, менш ніж або рівній довжині геш-коду $L_н$. Для MAC-алгоритму 3 довжина m MAC повинна бути менше або дорівнювати половині довжини геш-коду, тобто $m \le L_н / 2$.

Довжина в бітах рядка даних D повинна бути якнайбільше $2^{64} - 1$ для MAC-алгоритмів 1 і 2 і 256 для MAC-алгоритму 3.

Вибір конкретного MAC-алгоритму, спеціальної геш-функції й значення для m перебуває за рамками цього розділу ISO/IEC 9797.

Той же ключ буде використаний для розрахунків і перевірки MAC. Якщо рядок даних також шифрується, використовуваний для обчислення MAC ключ повинен відрізнятися від того, який використовується для розшифрування.

Вважається гарною криптографічною практикою мати незалежні ключі для конфіденційності й цілісності даних.

MAC-алгоритм 1.

Цей пункт містить опис MDx-MAC [5]. Більш конкретніше, зі спеціальної геш-функцією 1 цей механізм також відомий як RIPEMD-160-MAC, зі спеціальною геш-функцією 2 цей механізм також відомий як RIPEMD-128-MAC і зі спеціальної геш-функцією 3 цей механізм також відомий як SHA-1-MAC.

MAC-алгоритм 1 вимагає одного застосування геш-функції, щоб обчислити MAC-значення, але вимагає, щоб константи в циклічній функції були змінені.

Геш-функція повинна бути обрана зі спеціальних геш-функцій 1, 2 і 3 з ISO/IEC 10118-3:1998.

Розмір ключа *k* у бітах повинен бути якнайбільше 128 біт.

Опис MAC-алгоритму 1.

MAC-алгоритм 1 вимагає таких 5 кроків: поширення ключа, зміна констант і IV, операції, що гешує, вихідного перетворення й усікання.

Крок 1 (поширення ключа).

Якщо K коротше, ніж 128 біт, конкатенуємо K достатнє число раз і вибираємо 128 біт ліворуч, щоб сформувати 128-бітовий ключ K' (якщо довжина (у бітах) K рівна 128, то $K' := K$

$$K' := 128 \sim (K \| K \| ... \| K).$$

Обчислюємо підключі K_0, K_1, K_2 у такий спосіб:

$$K_0 := \bar{h}(K' \| U_0 \| K'),$$
$$K_1 := 128 \sim \bar{h}(K' \| U_1 \| K'),$$
$$K_2 := 128 \sim \bar{h}(K' \| U_2 \| K').$$

де U_0, U_1, U_2 – 768-бітові константи, визначені в п. 6.3, а $\bar{h}$ позначає спрощену функцію h, тобто без заповнення й додавання довжини.

Заповнення й додавання довжини може бути пропущене, тому що в цьому випадку довжина вхідного рядка завжди $2L_1$ біт.

Похідний ключ K_1 розбивається на чотири слова, позначувані $K_1[i](0 \le i \le 3)$, тобто $K_1 = K_1[0] \| K_1[1] \| K_1[2] \| K_1[3]$.

Для конвертації рядка в слова потрібна угода упорядкування біт. Угода упорядкування біт для цієї конвертації визначене для кожної спеціальної геш-функції в ISO/IEC10118-3 [108].

Крок 2 (зміна констант і IV).

Додаткові константи, використовувані в циклічній функції, змінюються додаванням модуля 2^{32} одного із чотирьох слів K_1, тобто:

$$C_0 := C_0 \oplus K_1[0]$$

Вище визначається, яке слово K_1 додається до кожної константи. Первісне значення IV геш-функції заміняється як $IV' := K_0$. Результуюча геш-функція позначається h', а його циклічна функція позначається φ'.

Крок 3 (операція, що гешує).

Вхідний рядок до зміненої геш-функції h' дорівнює рядку даних D, тобто: $H := h'(D)$.

Крок 4 (вихідне перетворення).

Змінена циклічна функція φ' використовується ще раз з спочатку із входом рядка $K_2 = K_2 \,||\, (K_2 \oplus T_0) \,||\, (K_2 \oplus T_1) \,||\, (K_2 \oplus T_2)$ та з другим входом рядка H' (результат кроку 3), тобто:

$$H'' := \phi'(K_2 \,||\, (K_2 \oplus T_0) \,||\, (K_2 \oplus T_1) \,||\, (K_2 \oplus T_2), H').$$

Тут T_0, T_1, T_2 – 128-бітові рядки, визначені для кожної спеціальної геш-функції.

Вихідне перетворення відноситься до обробки додаткового блоку даних, породженого з K_2 після набивання й додавання поля довжини.

Крок 5 (усікання).

MAC m біт відбувається узяттям ліворуч m біт рядка H'', тобто:

$$MAC := m \sim H''.$$

Ефективність.

Припустимо, що заповнений рядок даних (тут алгоритм заповнення визначається для конкретної геш-функції) містить q блоків; тоді MAC-алгоритм 1 вимагає q + 7 застосувань циклічної функції.

Це може бути скорочено до q + 1 застосуванням циклічної функції, попередньо обчисливши значення K_0, K_1, K_2, і заміною первісного значення IV на IV' у застосуванні геш-функції.

Рекомендується проводити цю зміну коду геш-функції разом із примусовою зміною, необхідною для кроку 2.

Для довгих вхідних рядків, MAC-алгоритм 1 має продуктивність, яка порівнювана з використовуваною геш-функцією.

Обчислення констант.

Константи, описані в цьому пункті будуть використані в обох MAC-алгоритмах – 1 і 3.

Рядки T_i і U_i – фіксовані елементи опису цього MAC-алгоритму. Вони обчислюються (тільки один раз), використовуючи геш-функцію; вони різні для кожної із трьох геш-функцій.

128-бітові константи T_i і 768-бітові константи U_i визначаються в такий спосіб. Визначення T_i включає 496-бітову константу і 16-бітові константи S_0, S_1, S_2, де S_i є 16-бітовим рядком, сформованим повторенням двічі 8-бітного представлення *i* (наприклад, 16-річного представлення S_1, яке дорівнює 3131). В обох випадках використовується ASCII кодування; що еквівалентно кодуванню, що використовує ISO/IEC 646:1991.

$$\text{for } i := 0 \text{ to } 2 \qquad T_i := 128 \sim \overline{h}(S_i \| R)$$

$$\text{for } i := 0 \text{ to } 2 \qquad U_i := T_i \| T_{i+1} \| T_{i+2} \| T_i \| T_{i+1} \| T_{i+2}'$$

де нижній індекс в T_i береться по модулю 3.

Для всіх констант C_i, C_i'' і всіх слів $K_1[K]$ найбільш значущий біт відноситься до самого лівого біта. Константи C_i, C_i'' представлені у 16-річному представленні.

Спеціальна геш-функція 1.

128-бітові постійні рядки T_i для спеціальної геш-функції 1 визначаються в такий спосіб (у 16-річному представленні):

$$T_0 = 1CC7086A046AFA22353AE88F3D3DACEB,$$
$$T_1 = E3FA02710E491D851151CC34E4718D41,$$
$$T_2 = 93987557CO7B8102BA592949EB638F37.$$

Дві послідовності постійних слів $C_0, C_1, ..., C_{79}$ і $C_0', C_1', ..., C_{79}'$ використовуються в циклічній функції спеціальної геш-функції 1. Вони визначаються в такий спосіб:

$$C_i = K_1[0] \oplus 00000000, (0 \le i \le 15),$$
$$C_i = K_1[1] \oplus 5A827999, (16 \le i \le 31),$$
$$C_i = K_1[2] \oplus 6ED9EBA1, (32 \le i \le 47),$$
$$C_i = K_1[3] \oplus 8F1BBCDC, (48 \le i \le 63),$$
$$C_i = K_1[0] \oplus A953FD4E, (64 \le i \le 79),$$

$$C_i = K_1[1] \oplus 50A28BE6, (0 \le i \le 15),$$
$$C_i = K_1[2] \oplus 5C4DD124, (16 \le i \le 31),$$
$$C_i = K_1[2] \oplus 6D703EF3, (32 \le i \le 47),$$
$$C_i = K_1[3] \oplus 7A6D76E9, (48 \le i \le 63),$$
$$C_i = K_1[0] \oplus 00000000, (64 \le i \le 79).$$

Спеціальна геш-функція 2.

128-бітові постійні рядки T_i для спеціальної геш-функції 2 визначаються в такий спосіб (у 16-річному представленні):

$$T_0 = FD7EC18964C36D53FC18C31B72112AAC,$$
$$T_1 = 2538B78ECOE273949EE4C4457A77525C,$$
$$T_2 = F5C93ED85BD65F609A7EB182A85BA181.$$

Дві послідовності постійних слів $C_0, C_1, ..., C_{63}$ і $C_0', C_1', ..., C_{63}'$ використовуються в циклічній функції спеціальної геш-функції 2. Вони визначаються в такий спосіб:

$$C_i = K_1[0] \oplus 00000000, (0 \le i \le 15),$$
$$C_i = K_1[1] \oplus 5A827999, (16 \le i \le 31),$$
$$C_i = K_1[2] \oplus 6ED9EBA1, (32 \le i \le 47),$$
$$C_i = K_1[3] \oplus 8F1BBCDC, (48 \le i \le 63),$$
$$C_i = K_1[1] \oplus 50A28BE6, (0 \le i \le 15),$$
$$C_i = K_1[2] \oplus 5C4DD124, (16 \le i \le 31),$$
$$C_i = K_1[2] \oplus 6D703EF3, (32 \le i \le 47),$$
$$C_i = K_1[3] \oplus 00000000, (48 \le i \le 63).$$

Спеціальна геш-функція 3.

128-бітові постійні рядки T_i для спеціальної геш-функції 3 визначаються в такий спосіб (у 16-річному представленні):

$$T_0 = 1D4CA39FA40417E2AE5A77B49067BBCC,$$
$$T_1 = 9318AFEF5D5A5B46EFCA6BECOE138940,$$
$$T_2 = 4544209656E14F97005DAC76868E97A3.$$

Послідовність постійних слів $C_0, C_1, ..., C_{79}$ використовуються в циклічній функції спеціальної геш-функції 3. Вони визначаються в такий спосіб:

$$C_i = K_1[0] \oplus 5A827999, (0 \le i \le 19),$$
$$C_i = K_1[1] \oplus 6ED9EBA1, (20 \le i \le 39),$$
$$C_i = K_1[2] \oplus 8F1BBCDC, (40 \le i \le 59),$$
$$C_i = K_1[3] \oplus CA62C1D6, (60 \le i \le 79).$$

MAC-алгоритм 2.

MAC-алгоритм 2 вимагає два застосування геш-функції для обчислення MAC-значення.

Геш-функція повинна бути обрана з ISO/IEC 10118-3 з такою вимогою, що L_1 є позитивним цілим кратним 8 і $L_2 \le L_1$.

Спеціальні геш-функції 1, 2 і 3 із ISO/IEC 10118-3: 1998 задовольняють цим умовам.

Розмір ключа k повинен бути не менше L_2, де L_2 – розмір геш-коду в бітах, і не більше L_1 біт, де L_1 – розмір вхідних даних до циклічної функції в бітах, тобто $L_2 \le k \le L_1$.

Опис MAC-алгоритму 2.

MAC-алгоритм 2 вимагає такі чотири кроки: поширення ключа, операція, що гешує, вихідне перетворення й усікання.

Крок 1 (поширення ключа).

Потрібно додати $L_1 - k$ нульових біт праворуч у ключі K; результуючий рядок довжини L_1 позначається $\overline{K}$.

Ключ $\overline{K}$ поширюється на два підключа $\overline{K}_1$ і $\overline{K}_2$:

Визначається рядок IPAD як конкатенація $L_1/8$ разів шістнадцяткового значення "36" (або бінарного значення "00110110"). Потім обчислюється значення K_1 як виключає АБО $\overline{K}$ і рядок IPAD, тобто:

$$\overline{K}_1 := \overline{K} \oplus \text{IPAD}.$$

Визначається рядок OPAD як конкатенацію $L_1 / 8$ разів шістнадцяткового значення "5C" (або бінарного значення "01011100"). Потім обчислюється значення $\overline{K}_2$ що як виключає АБО $\overline{K}$ і рядок OPAD, тобто:

$$\overline{K}_1 := \overline{K} \oplus \text{OPAD}.$$

Крок 2 (операція, що гешує).

Рядок, що є вхідним до зміненої геш-функції, дорівнює конкатенації K_1iD тобто:

$$H := h(\overline{K}_1 \,\|\, D).$$

Крок 3 (вихідне перетворення).

Рядок, який є вхідним до геш-функції, дорівнює конкатенації $\overline{K}_2$ і H', тобто:

$$H'' := h(\overline{K}_2 \,\|\, H').$$

Крок 4 (усікання).

MAC m біт відбувається взяттям ліворуч m біт рядка H'', тобто:

$$\text{MAC} := m \sim H''.$$

Ефективність.

Припустимо, що заповнений рядок даних (тут алгоритм заповнення визначається для конкретної геш-функції) містить q блоків; тоді MAC-алгоритм 2 вимагає q + 3 застосувань циклічної функції.

Це може бути скорочено до q + 1 застосуванням циклічної функції зміненням коду для геш-функції.

Можливо обчислити попереднє значення $IV_1 := \varphi(\overline{K}_1, IV)$ та $IV_2 := \varphi(\overline{K}_2, IV)$ та замінити первісне значення IV та IV_1 у першому застосуванні геш-функції й на IV_2 у вихідному перетворенні (друге застосування геш-функції). Це також вимагає зміни методу заповнення: в остаточному підсумку, фактичне вхідне значення до геш-функції на даний момент на L_1 біт коротше; це означає, що значення L_1 повинне бути додане до значення L_D.

Для довгих вхідних рядків MAC-алгоритм 2 має продуктивність, яка порівнювана з використовуваною геш-функцією.

MAC-алгоритм 3.

Цей пункт містить варіант MAC-алгоритму 1, оптимізованого для коротких входів (не більше ніж 256 біт) [124].

MAC-алгоритм 3 вимагає сім застосувань спрощеної циклічної функції, щоб обчислити MAC-значення, але це також може бути скорочене до єдиного застосування цієї циклічної функції деяким попереднім обчисленням.

Ця геш-функція повинна бути обрана зі спеціальних геш-функцій 1, 2 і 3 з ISO/IEC 10118-3:1998 [118].

Розмір ключа k у бітах повинен бути не більш 128 біт, а довжина MAC m у бітах – не більше $L_H / 2$.

Опис MAC-алгоритму 3.

MAC-алгоритм 3 вимагає такі п'ять кроків: поширення ключа, зміна констант циклічної функції, заповнення, застосування циклічної функції й усікання.

Крок 1 (поширення ключа).

Якщо K коротше, чим 128 біт, конкатенується K стосовно себе достатню кількість раз і беруться 128 біт, які розміщені з лівого краю, щоб сформувати 128-бітовий ключ K' (якщо довжина (у бітах) K дорівнює 128, $K' := K$:

$$K' := 128 \sim (K \| K \| \ldots \| K).$$

Обчислюються підключі K_0, K_1, K_2 у такий спосіб:

$$K_0 := \bar{h}(K' \| U_0 \| K'),$$
$$K_1 := 128 \sim \bar{h}(K' \| U_1 \| K'),$$
$$K_2 := 128 \sim \bar{h}(K' \| U_2 \| K').$$

де U_0, U_1, U_2 – 768-бітові константи, визначені в п. 6.3, а $\bar{h}$ позначає функцію h без заповнення й додавання довжини.

Заповнення й додавання довжини може бути пропущене, тому що в цьому випадку довжина вхідного рядка завжди $2L_1$ біт.

Похідний ключ K_1 розбивається на чотири слова, позначувані $K_1[i] (0 \le i \le 3)$, тобто: $K_1 = K_1[0] \| K_1[1] \| K_1[2] \| K_1[3]$.

Для конвертації рядка в слова потрібне угода упорядкування біт. Згода упорядкування біт для цієї конвертації визначене для кожної спеціальної геш-функції в ISO/IEC 10118-3.

Крок 2 (зміна констант і IV).

Додаткові константи, використовувані в циклічній функції, змінюються додаванням модуля 2^{32} одного із чотирьох слів K_1, тобто:

$$C_0 := C_0 \oplus K_1[0].$$

Вище визначено, яке слово K_1 додається до кожної константи. Первісне значення IV геш-функції замінюється як $IV' := K_0$. Результуюча геш-функція позначається φ'.

Крок 3 (заповнення).

Біти, що заповнюються, додаються до оригінального рядка даних, використовуються тільки для обчислення MAC. У результаті ці біти, що заповнюються (якщо такі є) не мають потреби в тому, щоб бути збереженими або переданими з даними. Верифікатор буде знати, були чи ні біти, що заповнюються, збережені або передані.

Рядок даних D, вхідний для MAC-алгоритму, повинен бути заповнений праворуч як можна меншим (якщо це можливо, нічим) числом "0"-біт, щоб одержати рядок даних $\overline{D}$ довжини 256 біт.

Якщо рядок даних порожній, заповнений рядок даних $\overline{D}$ містить 256 "0" біт.

Крок 4 (застосування циклічної функції).

Бітовий рядок $\tilde{L}$ обчислюється як двійкове представлення довжини (у бітах) L_D рядка даних D, ліворуч заповненого найменшим можливим числом "0"-біт, щоб одержати 128-бітовий рядок. Біт, що розташований праворуч рядка біт $\tilde{L}$ відноситься до найменш значущого біта двійкового представлення L_D.

Рядок, вхідний до циклічної функції φ' (зі зміненими константами), дорівнює конкатенації $K_2, \overline{D}$ виключаючого АБО $K_2, \tilde{L}$:

$$H'' := \phi'(K_2 \parallel \overline{D} \parallel (K_2 \oplus \tilde{L}), IV').$$

Крок 5 (усікання).

MAC *m* біт відбувається узяттям ліворуч m біт рядка H′, тобто:

$$MAC := m \sim H''.$$

Ефективність.

MAC-алгоритм 3 вимагає сім застосувань циклічної функції.

Це може бути скорочене до єдиного застосування циклічної функції попереднім обчисленням значень K_0, K_1, K_2.

Приклади.

Вхідні рядки для тестових значень

no.	input string
1	"" (empty string)
2	"a"
3	"abc"
4	"message digest"
5	"abcdefghijklmnopqrstuvwxyz"
6	"abcdbcdecdefdefgefghfghighijhijkijkljklmklmnlmnomnopnopq"
7	"ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz0123456789"
8	"1234567890" repeated 8 times to give 80 characters
9	"a" repeated 1,000,000 times to give 1 million characters

Загальне.

Цей додаток дає приклади обчислення MAC-алгоритмів 1 і 2, що використовують спеціальні геш-функції 1, 2 і 3 з ISO/IEC 10118-3:1998. Дев’ять прикладів обчислення геш-коду даються для кожної геш-функции.

Вхідні рядки, пронумеровані від 1 до 9, які знаходяться в прикладі. Скрізь у цьому додатку пропонується ASCII-кодування рядків даних; це рівноцінно кодуванню, що використовує ISO/IEC 646.

Два використовувані ключові значення є такими 128-бітовими рядками.

MAC-алгоритм 1.

Для прикладів у цьому розділі було обране значення $mL_2/2$, а саме m/80 для спеціальної функції 1 та 3 і m/64 для спеціальної функції 2.

Спеціальна геш-функція 1.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	80-bit MAC result: leftmost 80 bits of
1	B7F4508111EB8C3B5229C6AED406DE9ECA640133
2	BC78F55933BCEB1EE85A906F9E18374F23E310F9
3	6300DC20E97A5AA29DB9C7D607D23D126FA36863
4	3A2AC89B78EEAB8759F5112BCAD4CD405EEB5D35
5	16DC174925BBC27E0C939426C346846F97F8BC69
6	E062210BA5C9C947377BF3A6E85B3B5664FBD1D4E
7	9B462D5CBDAE1485FFE10BC001EF9E3AF6D128B5
8	88E73A01A1DE36C92D6F9E41F7278D407B4A4CCD
9	E7B128E4A1842B70F1E61A486C867C4887A4B21

Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	80-bit MAC result: leftmost 80 bits of
1	B45D6CA84CFB9020E0D5ABA2A7609D3D81F3F57F
2	8844375992037D1BCD0D118EE548D70C3F19CBBB
3	917C59B8AC7FC19DC25BEF82766412FA16BBC6A7
4	E0737CC7976D8F424390CB8798D623D751AFE15A
5	D57FAE836870718EFA4BD4A5F2F322A179A8735E
6	42B20D4C8FDE8672760CF83C0478D7BF8021404
7	42B20D4C8FDE8672760CF83C0478D7BF8021404
8	10441DF4F68CE8815818DC0FB370ABF87BCA4464
9	E06AD21D2AF04DD4217AB03B1A578F036997D01A

Спеціальна геш-функція 2.

Key1: 01122334455667788 99AABBCCDDEEFF	
No.	64-bit MAC result: leftmost 64 bits of
1	A47A64E9EDE0741B3FDDE33E5C1C6D78
2	51355051852FDC79FB228EAC905633AD
3	D83940DAFFBD4CBBE6BA30A6F9E63F5F
4	1A7CFE2BB26E973E213C1CB96FA4C2EF
5	798AEAC6046B31907C197BD68E59D376
6	0B8E1D4A571F32657189E22A1F2F4A53
7	B814730F482300C6E474F9255A66D680
8	9060A30758EBE3368D939AC168F1A9FD
9	20763FDEDF01E56FF5756954302C7DEO
Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	64-bit MAC result: leftmost 64 bits of
1	35FA3AC39F50F2A4E3FFC7AF5776B4EB
2	A89E25E67967467B630A2A00B802EA53E
3	66339027A36608EBD932DD55161E7B2
4	1F8779BAD84B50373931211A2761EAD3
5	31BF5B5B7ABAC2567DC0E02F1C3A25D7
6	B5B8BA3B8EA895FBC83CB7588FBD2656
7	8D27BBEC257C848D5CF375EB5EDA4CC7
8	B40B5BF6727DE90B26F770850F059C89
9	C6C7BC831B0BCE593DFD44E8E054A373

Спеціальна геш-функція 3.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	80-bit MAC result: leftmost 80 bits of
1	C8A8B3C75E6CE7C6C4F79CC19853CC954ABCB079
2	8DD9AE643BF10BBB7B978EF13EE6C0F480618FB0
3	A738B26A8BD318184E76707A99CAE14C670B9711
4	1EBFE413E55D6B288A2BD01D294A21FD8D4B20BF
5	0CE7BF40A73D977AB4999CF3A9BD1C5BEDC442E9
6	12A6823CC181294F95109073A6AA0C8961B14386
7	9369EE4A043AF1CA6E078D0B8A9CE5C1545440BA
8	B00D37D70A84B762FC0A8A9BC1B15F0E517B5EDF
9	DDDF44613E8559D12C150D022D5FE33F9E0FBACE

Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	80-bit MAC result: leftmost 80 bits of
1	C3A5ECD1E715C7272CFE78BC278086587B040422
2	D5D50FFA7EFDF1B17E96E2EC14DBC4412F7B771F
3	01FDD568008D412158F5B0C90AE2730DCFB77FB
4	9982E0EE91DB89AE7E7618AD1D649BA43406DBDD
5	ACD04E1004FCE53DECA9EE7AB95DAF97B7C44AA8
6	FADF62DCE789E86E60756AA819EF62C3E5C25E94
7	46DB9A49FB4976D007B14B1574843D019CA99445
8	4EF5BED3E816C530B23F491583C038596BB76FDB
9	BAC6BE6BE6153FECE2891FDA03824DD4D535D19

MAC-алгоритм 2.

Для прикладів у цьому розділі було обране значення m/80.

Спеціальна геш-функція 1.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	80-bit MAC result: leftmost 80 bits of
1	9EBEA41FBC24CD80BF2ECFD5B8C8CC8181D3FCAE
2	75CB722C50024C0E8A7A0DBA7D5C36B86D9D1DD5
3	5B48C1749DDED71EDFE0ADE2B944E808E4A65820
4	F9033064567F541235C3944EE95CB476055985D1
5	B37885405B71E025AF0CB574021A562A62733628
6	5C6429B982C8054B5B3348A0D7D2CE24D7032BC1
7	B0A4A451D092655E52428E16D1FEAA241C4DD9B
8	1CCEEC5122F08A76EBCD8E3DE8610D942D8A5F6
9	45D61908BFF6039E6DE3C037FDCE619F19F6410
Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	80-bit MAC result: leftmost 80 bits of
1	2FDE5DAF7050D14E6D7ACD2254D17FA3A8CFBCDD
2	239C4020610429A8662BF81A2CAAEA47F8EA0A44
3	89EFFB9F5A6BCEAE3C65D0C9803F3464E5E9E349
4	F5FC87FD5702F5D4E7BB634DA4CB4B41CD505B6C
5	5686C00F69E6C868732C67402AA107CEAB513439
6	525EC4893A221EFD9B6DD351059B40C05B4CE2D3
7	B975ED3893FC8D535376EF49211E2E6B1BB30B90
8	BC201FFA581357C271DAE25104167F3DCC97BADC
9	95A875AD64D55E677D8E4455E1445E7E940F758

Спеціальна геш-функція 2.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	64-bit MAC result: leftmost 64 bits of
1	AD9DB2C1E22AF9AB5CA9DBE5A86F67DC
2	3BF448C762DE00BCFA0310B11C0BDE4C
3	F34EC0945F02B70B8603F89E1CE4C78C
4	E8503A8AEC2289D82AA0D8D445A06BDD
5	EE880B735CE3126065DE1699CC136199
6	794DAF2E3BDEEA2538638A5CED154434
7	3A06EEF165B23625247800BE23E232B6
8	9A4F0159C0952DA43A89466D46B0AF58
9	19B1B3AF333B894DD86D09427116D0AD
Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	64-bit MAC result: leftmost 64 bits of
1	8931EEEE56A6B257FD1AB5418183D826
2	DBBCF169EA7419D5BA7BD8EB3673FF2D
3	2C4CD07D3162D6A0E338004D6B6FBC9A
4	75BFB25888F4BB77C77AE83AD0817447
5	B1B5DC0FCB7258758855DD1840FCDCE4
6	670D0F7A697B18F1A8AB7D2A2A00DBC1
7	54E315FDB34A61C0475392E5C7852998
8	AD04354D8AA2A623E72E3594EE3535C0
9	6F9B1C0FC06753618D6DB4B007733795

Спеціальна геш-функція 3.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	80-bit MAC result: leftmost 80 bits of
1	86C2962E58B3498A2608935AF7726311F2BFB538
2	0497FF21DAE3251DA0ED2F47F5A3B74ABA6B2560
3	6EE2A25F943E3F3EC05225FBB86BA73E2E5D51D2
4	CD4C0D1328DC4A8DC2801001B129AEFC6E0CF9CE
5	89ECE303FAD1E4313950CC3B008CB239B5B85844
6	9DF741057D075D3C4E1533E38A5FF469647194B4
7	188A58390A6EF9827035B81CDF1B5069211F0EE5
8	98A98D6A81FD361030856D2C19742AD8DBC468E7
9	D2986310BA18A78786534882F9C6BCBF06CCE9E3

Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	80-bit MAC result: leftmost 80 bits of
1	2739B6BE63F539EB70FE250346F6382A2DFA345F
2	A0C2711A6B1DA4CD8F85EF1E6FF7BF70B412B477
3	18F570E864FF903D2773D53C2E114E1A62152953
4	A80845A89BA15E941A2457084BC431F3E47759E1
5	14143EA1057B02D20C0157216190A006E30F3D41
6	DAB4B41BA639B4715889406FE18E0C037017E063
7	AEAEA5415B4F266CB15CBEB844E56AEC2DABAD6D
8	3DBA11471EB4FCCF21BAEB0BFF7E20150132C6CF
9	3BB917B8BD8560E89FF9054FBE096CBACA109D5F

MAC-алгоритм 3.

Для прикладів у цьому розділі було обране значення $mL_2/2$, а саме, m/80 для спеціальної функції 1 і 3 і m/64 для спеціальної функції 2.

Спеціальна геш-функція 1.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	80-bit MAC result: leftmost 80 bits of
1	6606EF2D3BBD010F516C65372C3CF0ACF111B3F7
2	F0BC0C81307E17A71F4C40AE0B2AC39FCB23CE12
3	7720FD23925B854F963E8812573CD86EBA61EB66
4	2683D6CE053BA0420E76130EAE2367734B7D2D53
5	DE532D156CBE12464BB6147E99470C471D9F1C6
Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	80-bit MAC result: leftmost 80 bits of
1	4BD390E9EC460AD4866CCB32D091AFBF73E5B6DA
2	CD2847BAB4636CDBCEADCF3D187122A9199DA670
3	15C3910C42638E5EE6DEBD506BD8C4DB94713A3A
4	04148DCB47728E3E57B836A66043D5145879796E
5	829A24010704DBD0EE34A6D607F7B34829E04E95

Спеціальна геш-функція 2.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	64-bit MAC result: leftmost 64 bits of
1	aeb2c45f13c0c6f5f10be2f1e3e9c322
2	16874d0e17e4f1c290dd749cceff7834
3	a289aa06aeb8fc99b989c377baadd9d4
4	0d80db68bbf99442dc3d6b83d038def3
5	11dc4a6bd375c64f78bb78ad265ed7cd
Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	64-bit MAC result: leftmost 64 bits of
1	7248481816b8d3af29f5c002ff769ea5
2	dfe1e36ce9792476e0889ef1becef18a9
3	9b4f1d21320f4a327f023947554bfc3b
4	3d2d658d0196e4339f42ada50dfcfa6f
5	0a34452d9da70c70183dffddb8eec056

Спеціальна геш-функція 3.

Key1: 00112233445566778899AABBCCDDEEFF	
No.	80-bit MAC result: leftmost 80 bits of
1	708F4A226CDE708820643CBEEEFDCBE6CB36FB269
2	EAB87BE709D1E5CB62C7489C2B1407130B772760
3	C1BD6F9C908132FEF5187CBE681B42A8C785FBF6
4	F34DEB241D46C6448D67ACE6B8CD4DF00DA23EBC
5	669DED2BD6A1AE0BCFF7D3B74494C1D8161FA0D8
Key2: 0123456789ABCDEFFEDCBA9876543210	
No.	80-bit MAC result: leftmost 80 bits of
1	EAF6F9DDBAFD299320FF0FC8E02E2BE62879F341
2	2AAE9DE0A555E7CD7383C27506A467B8DF4E3A33
3	FE6031710329D12090F73F55CFFCC6215F9BEAE9
4	0CCDD9DAB6B0126800EC1CC7A02656E12EDEA42C
5	ABDBC8AAAE4A8CE734432188740A149BDF2D2D15F

Аналіз захищеності MAC-алгоритмів.

Цей додаток обговорює рівень безпеки MAC-алгоритмів цього розділу ISO/IEC 9797. Його метою є допомогти користувачу цього розділу ISO/IEC 9797 у виборі одного з наданих механізмів [118].

У цьому додатку MAC k(D) позначає MAC для рядка D, обчислений з використанням ключа MAC-алгоритму K.

Для того, щоб визначити рівень безпеки MAC-алгоритму, розглядаються дві стратегії атаки.

Атака підробки. Ця атака складається із прогнозованих значень MAC k(D) для рядка даних D без первісного знання K. Якщо зловмисник може зробити це для єдиного рядка даних, його називають здатним до підробки. Фактичні атаки часто вимагають, щоб зловмисник був таким, що його можливо перевірити, тобто, щоб підроблений MAC був відомий і коректний заздалегідь з імовірністю, близькою до 1. Більше того, у множині додатків рядок даних має певний формат, який накладає додаткові обмеження на рядок даних D.

Атака відновлення ключа. Ця атака складається зі знаходження самого ключа MAC-алгоритму K з пар чисел рядок даних/MAC. Така атака сильніша в порівнянні з підробленою атакою, оскільки вона дозволяє довільні атаки.

Виконуваність атаки залежить від кількості необхідних пар відомих і обраних рядком даних/MAC, а також від кількості off-line шифрувань.

Можливі атаки проти MAC-алгоритмів описані нижче; немає гарантії, що цей список вичерпний. Перші дві атаки – загальні, тобто застосовуються до будь-якого MAC-алгоритму. Наступні атаки застосовуються до будь-якого ітераційного MAC-алгоритму [5].

Вгадування MAC. Це підробка, яка не може бути перевірена і має ймовірність успіху $\max(1/2^m, 1/2^k)$. Ця атака застосовується до всіх MAC-алгоритмів і може бути попереджена розумним вибором m і k.

Відновлення ключа з використанням грубої сили. Для цієї атаки в середньому потрібно 2^{k-1} операцій; перевірка даної атаки вимагає пар рядків даних/MAC порядку k/m. Ця атака також застосовується до всіх MAC-алгоритмів. Вона може бути попереджена розумним вибором значення k. У якості альтернативи, можна запобігти одержанню пар k/m рядків даних/MAC, які необхідні для ідентифікації однозначного ключа. Наприклад, якщо k = 128 і m = 64, приблизно 2^{32} ключів відповідають заданій парі рядків даних/MAC; якщо ключ змінюється залежно від рядка даних, відновлення ключа з використанням грубої сили не є більш ефективним, ніж вгадування значення MAC.

Підробка Дня народження [5]. Якщо зловмисник знає достатню кількість пар рядків даних/MAC, він чекає, щоб знайти два рядки даних *D* і D′, таких що MACк(D) = MACк(D′), а також, щоб вхідні значення при вихідному перетворенні в обох обчисленнях були рівними; це називається

внутрішня колізія. Якщо *D* і D′ утворюють внутрішню колізію, *MACк(D||Y)*= = *MACк (*D′||Y*)* для будь-якого рядка *Y*. Це дозволяє атаку одному обраному рядку даних, коли зловмисник може спрогнозувати MAC для D′ ||D після спостереження за MAC, що відповідає D||Y*.* Ця підробка рядків даних визначених форм, яка стосується не всіх додатків, але варто відзначити, що розширення цієї атаки існують, які дозволяють більшу гнучкість у рядках даних. Ця атака вимагає один обраний рядок даних і приблизно $2^{n/2}$ відомих рядків даних, а також 2^{n-m} обраних рядків даних.

Атака підробки Дня народження може бути відвернена приєднання попереду блоку до рядка даних, який містить порядковий номер або робить MAC обчислення оголошеним. Це означає, що ця реалізація повинна гарантувати, що кожний порядковий номер має використовуватися тільки один раз для MAC обчислення протягом часу життя ключа. Це не здійсненне у всіх оточеннях.

Скорочене відновлення ключа. Деякі MAC-алгоритми є потенційно вразливими до атак відновлення ключа, заснованим на внутрішній колізії. Не надане скорочених атак для MAC-алгоритмів, описаних у цьому розділі ISO/IEC 9797 [118].

Доказ безпеки.

Було доведено, що MAC-алгоритм 1 є безпечним, якщо зберігається таке припущення [4]: циклічна функція φ, що отримується від первісного значення IV і додаткової константи, є псевдовипадковою функцією.

Псевдовипадкова функція є функцією із секретним ключем, який швидше за все поводить себе як випадкова функція (тобто важко відрізнити від випадкової функції) для того, хто не знає секретний ключ.

Було доведено, що MAC-алгоритм 2 є безпечним, якщо зберігаються такі три допущення [3]:

геш-функція h є стійкою до колізій, коли IV – секретне;

циклічна функція φ, що отримується від первісного значення IV, є сильнішою за MAC-алгоритм (тобто її вихід важко передбачити);

ключі $\overline{K}_1$ і $\overline{K}_2$ не можуть відрізнятися від істинно випадкових ключів; це відноситься до вимоги, що циклічна функція φ, що отримується від первісного значення IV, є "слабкою" псевдовипадковою функцією ("слабка", тому що опонент має тільки непрямий доступ до значень $\overline{K}_1$ і $\overline{K}_2$).

Безпека MAC-алгоритму 3 подібна припущенням, зробленим стосовно циклічної функції φ для доведення безпеки MAC-алгоритмів 1 і 2.

Розділ 4. Універсальне гешування

Розглянуто основні поняття і загальні властивості універсального гешування. Наведено алгоритми формування геш-коду з використанням функцій універсального гешування на основі лінійно-блокових кодів та на основі модулярних перетворень. Окремо розглянуто алгоритми формування геш-коду на основі використання каскадної схеми універсального гешування.

4.1. Визначення та загальні властивості універсального гешування

Універсальні класи геш-функцій були вперше запропоновані Картером і Вегманном [55; 56; 90], далі вивчені Сарватом [80] і Стінсоном [86; 87]. Слід розглянути основні визначення для опису універсальних класів.

Нехай A і B будуть кінцевими множинами з відповідних елементів $a = |A|$, і $b = |B|$б, де $a \geq b$. Функція $h :\rightarrow B$ буде названа геш-функцією. Для геш-функції h і для $x, y \in A, x \neq y$, потрібно визначити $\sigma h(x, y) = 1$ якщо $h(x) = h(y)$ і $\sigma h(x, y) = 0$ у зворотному випадку. Тобто, $\sigma h(x, y) = 1$, тоді і тільки тоді, якщо гешування значеннь x і y викликають колізію. Для кінцевої множини H геш-функцій визначимо $\sigma H(x, y) = \sum_{h \in H} \sigma h(x, y)$. Звідси $\sigma H(x, y)$ підраховує кількість геш-функцій у H, при яких значення елементів x і y викликають колізію.

Ідея універсального класу геш-функцій полягає у визначенні набору елементів кінцевої множини H геш-функцій таким чином, що випадковий вибір функції забезпечував би низьку ймовірність того, що для будь-яких різних входів x і y, ймовірність того, що (ймовірність колізії) не може бути більше $\varepsilon : P_{кол} = P(h(x) = h(y)) \leq \varepsilon$, і може бути підрахована як $\sigma H(x, y) / |H|$.

Визначення універсального класу геш-функцій еквівалентно визначенням такого алгоритму формування коду автентифікації, при якому виконується така умова: кількість різних правил формування коду автентифікації (кількість ключів), при яких існує колізія (збіг кодів автен-

тифікації) для двох довільних вхідних послідовностей, обмежена. Кількість таких ключів не може перевершувати значення $P_{кол} \cdot H$, де $P_{кол}$ – ймовірність колізії, H – кількість всіх правил (ключів) [86; 87].

Теорема 4.1. [4].

Для будь-якого класу H геш-функцій від A до B існують різні елементи $x, y \in A$, такі що $\sigma H(x, y) \geq |H|(a-b)/(b(a-1))$, де $a = |A|$, і $b = |B|$.

Варто навести два визначення класів геш-функцій.

Нехай $0 < \varepsilon < 1$. H є ε – універсальним геш-класом (скорочено $\varepsilon - U(H, a, b)$, якщо для двох різних елементів $x_1, x_2 \in X$ існує не більше ніж $H \cdot \varepsilon$ функцій таких, що $h(x) = h(y)$, якщо $\sigma H(x, y) \leq s|H|$ для всіх $x, y \in A, x \neq y$.

Нехай $0 < \varepsilon < 1$. H є ε – універсальним геш-класом (скорочено $\varepsilon - U(H, a, b)$, якщо виконуються такі умови:

Для кожного $x_1 \in A$ і для кожного $y_1 \in B$,

$$|\{h \in H : h(x_1) = y_1\}| = |H| / |B|.$$

Для кожного $x_1, x_2 \in A (x_1 \neq x_2)$ і для кожного $y_1, y_2 \in B$,

$$|\{h \in H : h(x_1) = y_1, h(x_2) = y_2\} \leq \varepsilon |H|.$$

Визначення строго універсального класу геш-функцій еквівалентно визначенню алгоритма формування кодів автентифікації. Використовуючи суворо універсальний клас геш-функцій можна побудувати схему автентифікації, при якому будуть виконуватися такі правила:

1. Кількість правил формування коду автентифікації (кількість ключів), при яких для довільної вхідної послідовності значення коду автентифікації не змінюється, обмежена. Кількість таких ключів не може перевершувати значення $\frac{H}{2^a}$, де H – кількість всіх ключів, a – кількість біт коду автентифікації;

2. Кількість правил формування коду автентифікації (кількість ключів), при яких для двох довільних вхідних послідовностей відповідні їм значення коду автентифікації не змінюються, обмежена. Кількість таких ключів не може перевершувати значення $P_{кол}\frac{H}{2^a}$ де $P_{кол}$ – ймовірність колізії, H – кількість всіх ключів, a – кількість біт коду автентифікації.

Ймовірність колізії кодів автентифікації в такій схемі буде визначається таким твердженням $P_{кол} \le \varepsilon$.

Нижні межі на розмір класів геш-функцій.

З точки зору того факту, що класи геш-функцій призводять до автентифікаційних кодів, є інтерес порахувати нижні межі числа необхідних геш-функцій. Потрібно розглянути деякі раніше відомі межі.

Теорема 4.2. [9].

Якщо існує $\varepsilon - SU(H,a,b)$ клас H геш-функцій від A до B, де a = |A| і b = = |B|, тоді $\left|H \ge \frac{a(b-1)}{a(\varepsilon b-1)+b^2(1-\varepsilon)}\right|$.

Підставивши $\varepsilon = 1/b$ і $\varepsilon = (a-b)/(ab-b)$ у наведену вище межу, відповідно можна отримати наслідок.

Наслідок 4.1. [9].

Припустимо, H це клас геш-функцій від A до B, де a = |A| і b = |B|. Якщо H є $\varepsilon - U(H,a,b)$, тоді |H| ≥ a / b. Якщо H є $\varepsilon - U(H,a,b)$ тоді |H| ≥ (a – 1)/ (b – 1).

Далі представимо нижню границю на число геш-функцій у $\varepsilon - U(H,a,b)$ – класі.

Теорема 4.3. [9].

Якщо існує $\varepsilon - U(H,a,b)$ клас H геш-функцій від A до B, де a = |A| і b = |B|, тоді:

$$\left|H \ge 1+\frac{a(b-1)^2}{b\varepsilon(a-1)+b-a}\right|.$$

У випадку ε = 1/b одержуємо такий наслідок.

Наслідок 4.2. [9].

Якщо існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A до B, де a = |A| і b = |B|, тоді |H| ≥ 1 + a(b – 1). Якщо |H| = 1 + a(b – 1) тоді існує OA (b, a, λ), де $\lambda = (a(b-1)+1)/a^2$ (OA(b, a, λ) – ортогональний масив).

3. Прямі й рекурсивні конструкції для побудови універсальних класів геш функцій.

Теорема 4.4. [2; 9].

Нехай q буде ступенем простого числа. Тоді існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A до B, де |A| = q^2, |B| = q і |H|= q (звідси ε = 1/q).

Теорема 4.5. [2; 9].

Нехай q буде ступенем простого числа. Тоді існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A до B, де |A| = q^2, |B| = q і |H|= q^3 (звідси ε = 1/q).

Доведення. Нехай A = GF(q) × GF(q), B = GF(q) і

$H = \left\{ h_{xyz} : x, y, z \in GF(q) \right\}$, де $h_{xyz}(a,b) = x + ay + bx$.

Теорема 4.6. [2; 9].

Нехай q буде ступенем простого числа. Тоді існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A до B, де |A|= q^2, |B| = q і |H| = q^2 (звідси ε = 1/q).

Теорема 4.7. (Декартов добуток) [2; 9].

Якщо існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A до B, тоді, для будь-якого цілого i ≥ 1 існує $\varepsilon - U(H, a, b)$ клас H_i геш-функцій від A_i до B_i с |H| = $|h_i|$.

Теорема 4.8. (Композиція 1) [2; 9].

Для i = 1, 2 припустимо, що існує $\varepsilon - U(H, a, b)$ клас H_i геш-функцій від A_i до B_i, де $A_2 = B_1$. Тоді існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A_1 до B_2, де ε = ε_1 + ε_2 і $|H| = |H_1| \times |H_2|$.

Теорема 4.9. (Композиція 2) [2; 9].

Припустимо H_1 є $\varepsilon - U(H, a, b)$ класом геш-функцій від A_1 до B_1, і припустимо H_2 є $\varepsilon - U(H, a, b)$ класом геш-функцій від B_1 до B_2. Тоді існує $\varepsilon - U(H, a, b)$ клас H геш-функцій від A_1 до B_2, де ε = ε_1 + ε_2 і $|H| = |H_1| \times |H_2|$.

Застосування універсального гешування до автентифікації.

Потрібно розглянути отримані конструкції для одержання автентифікаційних кодів.

Теорема 4.10. [2; 9].

Нехай q буде степінь простого числа й нехай і ≥ 1 буде цілим числом. Тоді існує $\frac{i}{q} - U(H, a, b)$ q_i геш-функцій від A до B, де $|A| = q^{2^i}$ й $|B| = q$.

Теорема 4.11. [2; 9].

Нехай q буде степінь простого числа й нехай і ≥ 1 буде цілим числом. Тоді існує $\frac{i+1}{q} - SU(H, a, b)$ клас q^{i+2} геш-функцій від A до B, де $|A| = q^{2^i}$ та $|B| = q$.

Теорема 4.12. [2; 9].

Нехай q – степінь простого числа й нехай і ≥ 1 буде цілим числом. Тоді існує $\frac{i}{q} + \frac{1}{q} - SU(H, a, b)$ клас q^{2i+3} геш-функцій від A до B, де $|A| = q^{2^i}$ та $|B| = q$.

Таким чином, наведені конструкції дозволяють будувати конструкції сімейств універсальних і строго універсальних класів.

Сімейства універсальних геш-функцій

A (D; N; M) сімейство геш-функцій є набором F функцій D, що f: X →Y для кожного f ∈ F, де |X | = N та |Y | = M [86].

A (D; N; M) – сімейство геш-функцій F, где δ – універсальна функція [20] за умови, що для будь-яких двох різних елементів x_1, x_2 ∈ X, існує множина δD функцій f ∈ F таких, що f (x_1) = f (x_2). Параметр δ часто використовується, як ймовірність колізії сімейства геш-функції. Використовуватися буде позначення δ-U, як скорочення для δ – універсальної функції.

A (D; N; M) – сімейство геш-функций F, строго універсальних [6] за умови, що, для будь-яких двох різних елементів x_1, $x_2 \in X$, і для будь-яких двох (не обов'язково різних) елементів y_1, $y_2 \in Y$, справедливо, що:

$$\left|\{f \in F : f(x_i) = y_i, i = 1,2\}\right| = \frac{D}{M^2}.$$

Буде використовуватися позначення SU як скорочення для строго універсальної функції.

Буде часто зображатися (D; N; M) сімейство гешфункцій, його назва F, у вигляді масиву, розмірністю D × N, що складається з M символів, де рядки проіндексовані відповідно до функцій у F, а стовпці проіндексовані відповідно до елементів у X, і елемент рядка f і стовпця x є f (x) (для будь-якого f ∈ F і будь-якого x ∈ X). Кожен рядок масиву відповідає одній з функцій у сімействі. Позначається цей масив A (F) і називається це подання семейства геш-функції F у вигляді масиву. Якщо F є δU (D; N; M) сімейством геш-функції, то, отже, в будь-яких двох колонках із A (F), існують множини δ D рядків A (F) таких, що елементи в двох даних стовпцях рівні.

Припустимо, Y – алфавіт q символів (n, K, d, q) код – набір C з K вектора (званий кодовим словом) в Y_n таким, що відстань Хеммінга між будь-якими двома різними векторами в C є d.

Якщо код лінійний (тобто, якщо q – степінь простого числа, Y = F_q C – підмножина (F_q) n), можна вважати, що, код [n, k, d, q] і є код, де k = $\log_q$ K є розміром коду.

Наступна рівність була вперше виведена Бієрбрауером, Джохансоном, Кабатіанські і Смітом у роботах [85; 99; 100].

Теорема 4.13.

Якщо існує (n, k, d, q) код, то тоді існує $(1 - \frac{D}{N})$-U (n; k, q) сімейство геш-функцій. Навпаки, якщо існує δ-U (D; N; M) сімейство геш-функцій, тоді існує (D; N; D (1 – δ), M) код.

Теорема 4.14.

Доведення теореми полягає в тому, щоб дозволити ключовим словам в установленому коді, відповідної колонки A F), де F – встановлене сімейство геш-функцій.

Приклад 4.1. Наступне сімейство геш-функцій $\frac{1}{3}$ – U (3; 9; 3), $\{f_i: i \in Z_3\}$ еквівалентно (3, 9, 2, 3) коду:

	(0, 0)	(0, 1)	(0, 2)	(1, 0)	(1, 1)	(1, 2)	(2, 0)	(2, 1)	(2, 2)
f_0	0	0	0	1	1	1	2	2	2
f_1	0	1	2	1	2	0	2	0	1
f_2	0	2	1	1	0	2	2	1	0

Ортогональний масив OA_λ (N, M) M_2 від масиву N, що складається з M символів таких, що, в будь-яких двох колонках масиву, кожна впорядкована пара символів перебувала точно в λ рядку. Якщо F є SU (D; N; M) – сімейство геш-функцій, тоді однозначно A (F) OA_λ (N, M), де $\lambda = D/M_2$. Зворотнє доводиться так само, отже є така теорема, яка була вперше виведена в роботі [19].

Теорема 4.15.

SU (D; N; M) – сімейство геш-функцій, еквівалентне OA_λ (N, M), де λ = D/M2.

Приклад 4.2.

Наступне – SU (9; 3, 3) сімейство геш-функцій $\{f_i, j: i, j \in Z_3\}$ еквівалентно OA_1 (3, 3):

f_i	0	1	2	$f_{1,1}$:	2	2	1
$f_{0,0}$:	0	1	1	$f_{1,2}$:	0	0	2
$f_{0,1}$:	1	2	2	$f_{2,0}$:	1	0	1
$f_{0,2}$:	2	0	0	$f_{2,1}$:	2	1	2
$f_{1,0}$:	1	1	0	$f_{2,2}$:	0	2	0

Деякі схеми сімейств геш-функцій.

У цьому розділі наводиться декілька схем семейств геш-функцій. Основна ідея в наведених схемах використовувалася Рао у 1947 р. [58] з використанням мови ортогональних масивів.

Теорема 4.16.

Нехай ℓ – позитивне ціле число і нехай q – степінь простого числа. Нехай $X \subseteq (F_q)\ell$ – будь-яка сукупність попарних лінійно незалежних

векторів F_q. Для кожного $\vec{r} \in (F_q)^\ell$ визначається функція $f_{\vec{r}}: X \to F_q$ за правилом:

$$f_{\vec{r}}(\vec{x}) = \vec{r} \times \vec{x}.$$

Нарешті, можна визначити:

$$F(q,l,X) = \{f_{\vec{r}} : \vec{r} \in (F_q)^l\}.$$

Тоді F (q, ℓ, X) SU (qℓ; |X|, q) – сімейство геш-функцій.

Доведення.

Відомо, що є (qℓ; |X|, q) – сімейство геш-функцій. Потрібно довести, що воно SU. Нехай $\vec{x}_1, \vec{x}_2 \in (F_q)^l$, де $\vec{x}_1 \neq \vec{x}_2$, і нехай $y_1, y_2 \in F_q$. Необхідно підрахувати кількість векторів $\vec{r} \in (F_q)^l$ таким чином, що $\vec{r} \times \vec{x}_1 = y_1$, та $\vec{r} \times \vec{x}_2 = y_2$.

Тепер $\vec{x}_1$ і $\vec{x}_2$ – лінійно незалежні вектори за F_q. Якщо позначається, $\vec{r} = (r_1,...,r_l)$, то можна отримати лінійно незалежну систему, в F_q, що складається із двох рівнянь із ℓ невідомими $r_1,...,r_l$. Тому однозначно q^{l-2} – розв'язки для вектора $\vec{r}$, і сімейство геш-функцій – SU.

Висновок 4.1.

Нехай q степінь простого числа. Для $a,b \in F_q$, знаходимо f_a, $f_{a,b}: F_q \to F_q$ за правилом:

$$f_{a,b}(x) = ax + b$$

Тоді $\{f_{a,b} : a,b \in F_q\}$ $SU(q^2; q, q)$ – сімейство геш-функцій.

У схемах наведено, сімейства всіх лінійних геш-функцій. Схеми для SU сімейства квадратичних геш-функцій представлені в роботі [8].

Теорема 4.17.

Нехай q не кратний степінь простого числа. Для $a,b \in F_q$, потрібно визначити f_a, b:

$F_q \to F_q$ за правилом:

$$f_{a,b}(x) = (x + a)^2 + b.$$

Тоді $\{f_{a,b} : a,b \in F_q\}$ $SU(q^2; q, q)$ – сімейство геш-функцій.

Теорема 4.18.

Ураховуючи теорему 4.14, можна побудувати багато δ-u сімейств геш-функцій за кодами. Наприклад, використовуючи код Ріда – Соломона можна одержати таку схему, описану в роботі [53].

Теорема 4.19.

Нехай q – степінь простого числа й нехай $1 \le k \le q - 1$. Для $a \in F_q$, потрібно визначити f_a: $(F_q)^k \to F_q$ за правилом:

$$f_a(x_0,...,x_{k-1}) = x_0 + \sum_{i=1}^{k-1} x_i a_i .$$

Тоді $\{f_a : a \in F_q\}$ є $\frac{k-1}{q} - U(q : q^k : q)$ є сімейством геш-функцій.

Починаючи з полінома не нульового степені й до полінома степені $k - 1$ над полем F_q отриманий не менше, ніж $k - 1$ корінь, отже, існує не менше, ніж $k - 1$ значень a для яких справедливо це рівняння. Тому сімейством геш-функцій є $\frac{k-1}{q} - U$. Дана схема наведена в роботі [51].

Теорема 4.20.

Нехай q – простий степінь числа й нехай s і t – позитивні цілі числа, такі, що $s \ge t$. Нехай $\phi : F_{q^s} \to (F_q)^t$ – будь-які q-лінійні сюр'єктивні відображення (тобто, $\phi(x+y) = \phi(x) + \phi(y)$ для всіх $x, y \in F_{q^s}$ і $\phi(ax) = a\phi(x)$ для всіх $a \in F_q, x \in F_{q^s}$). Для кожного $a \in F_q$, потрібно визначити $f_a : F_{q^s} \to (F_q)^t$ по правилу $f_a(x) = \phi(ax)$.

Тоді $\{f_a(x) : a \in F_{q^s}\}$ є $a\frac{1}{q^t} - U(q^s; q^s, q^t)$ сімейство геш-функцій.

Можна брати композицію двох сімейств геш-функцій кожний раз, коли сфера визначення функцій в одному сімействі належить до того ж діапазону, що й сфера визначення функцій іншого сімейства. Композиція схеми з роботи [20] дозволяє комбінувати δ_1-u сімейство геш-функцій з δ_2-u сімейством геш-функцій та одержати $(\delta_1 + \delta_2)$-U сімейство геш-функцій. Цю процедуру можна вважати схемою каскадного кодування.

Теорема 4.21.

Припустимо, що F_1 – $\delta_1 - U(D_1;N,M_1)$ сімейство геш-функцій від X до Y_1, і $F_2\delta_2 - U(D_2;M_1,M_2)$ сімейство геш-функцій від Y_1 до Y_2. Для будь-якого $f_1 \in F_1$, $f_2 \in F_2$, слід визначити $f_1 \circ f_2(x): X \to Y_2$ за правилом $f_1 \circ f_2(x) = f_2(f_1(x))$.

Тоді:

$\{f_1 \circ f_2 : f_1 \in F_1, f_2 \in F_2\}$ і є $(\delta_1 + \delta_2) - U(D_1D_2;N,M_2)$ сімейством геш-функцій.

4.2. Універсальне гешування на основі лінійно-блокових кодів

Забезпечення імітостійкості систем передачі даних можливо на основі рішення взаємозв'язаної сукупності задач захисту інформації. Якість рішення задач захисту значною мірою визначається криптографічними алгоритмами шифрування, цифрового підпису, гешування і формування MAC кодів, що використовуються.

MAC коди, відомі також як коди автентичності повідомлень, є криптографічними примітивами, які використовуються для забезпечення цілісності й автентичності даних.

Для універсального гешування із великим коефіцієнтом стиснення інтерес представляють схеми на алгеброгеометричних (АГ) кодах. Практичні коди великої довжини є АГ кодами. Вперше АГ підхід до побудови кодів був запропонований у 1981 р. [6]. З метою викладу теорії універсального гешування з АГ кодами в розділі виконаний аналіз методів побудови схем універсального гешування із кодами Ріда – Соломона і кодів на кривих Ерміта, досліджені основні форми кривих Ерміта, отримані точні оцінки колізійних властивостей універсальних геш-функцій з кодами Ерміта, розроблений алгоритм прискореного обчислення MAC кодів на кривих Ерміта і досліджені практичні схеми гешування із використанням кодів Ріда – Соломона і Ерміта [43 – 45].

Коди автентифікації повідомлень з великим коефіцієнтом стиснення.

У поданні Прінеля [78], MAC код – це функція відображення h: $K \times M \to R$, де простір ключів $K = \{0,1\}^k$, простір повідомлень $M = \{0,1\}^*$ і простір MAC значень $R = \{0,1\}^n$ для k, $n \geq 1$. Для заданих значень ключа $k \in K$ і повідомлення $X \in M$, функція виробляє MAC значення $Y \in R$.

Підхід, що знижує суперечність між обчислювальними витратами у великих полях і необхідністю забезпечити на великій довжині повідомлення мале значення ймовірності колізії є застосування універсального гешування з АГ кодами [25]. У схемах з АГ кодами (n,k,d) імовірність колізії визначається значенням $1-d/n$. Для відомих до теперішнього часу АГ кодів імовірність колізії обмежується в кращому випадку значенням обернено пропорційним квадрату розмірності поля Zq. Тому інтерес викликає поведінка параметрів геш-функції на основі АГ кодів особливо при різних умовах їх використання.

Застосування АГ кодів для побудови універсальних класів геш-функцій забезпечує найкращі співвідношення між розміром геш-кодів та ймовірністю колізії при обчисленнях у кінцевих полях, а також мінімізує витрати за ключовими даними. Зв'язок між універсальним класом геш-функцій і кодовими схемами (теорема 4.14), вперше був відзначений Бієрбрауером, Джохансоном, Кабатіанські і Смітом [85; 99; 100].

Аналіз загальних характеристик алгеброгеометричних кодів показав [26], що найкращими властивостями з точки зору їх кодової відстані та складності обчислення є розширений код Ріда – Соломона, коди на кривих Ерміта та коди на кривих Сузукі [29].

Параметри універсальних геш-функцій з використанням кодів Ріда – Соломона.

Розглянемо розширений код Ріда – Соломона (RS), який є лінійним з параметрами:

$$[q, k, q-k+1]q,$$

де $k \le q$ і q це степінь простого числа. Застосовуючи теорему 4.14, можна отримати таке [26].

Теорема 4.22. [26].

Нехай q – степінь простого числа і $1 \le k \le q$, тоді існує $\frac{k-1}{q} - U(q, q^k, q)$ родина геш-функцій (RSh_q).

Універсальний RSh_q еквівалентний поліноміальному гешуванню. Відмінність полягає в тому, що обчислення геш-коду може бути реалізовано не тільки в кінцевому полі за модулем простого числа, але і в розширеному полі Галуа. У ряді випадків така ситуація може переважати, оскільки спрощується розбиття повідомлення на слова, які повинні бути приведені до розміру поля обчислення геш-коду.

На рис. 4.9 наведені графіки залежності ймовірності колізії $\varepsilon = \frac{k-1}{q}$ для RSh_q гешування від довжини даних $1 \le k \le q$ і розміру поля обчислень q.

Аналіз графіків показує, що найкращим значеннями q для гешування даних, які лежать у діапазоні розрядності сучасних процесорів, є 2^{32} і 2^{64}. Імовірність колізії зростає лінійно із зростанням об'єму даних. Для ймовірності колізії в діапазоні значень 10^{-3} $(2^{-10}) \div 10^{-9}$ (2^{-30}) розмір даних, для яких обчислюється геш-код зменшується від 2^{22} до 2^3 32-х розрядних слів.

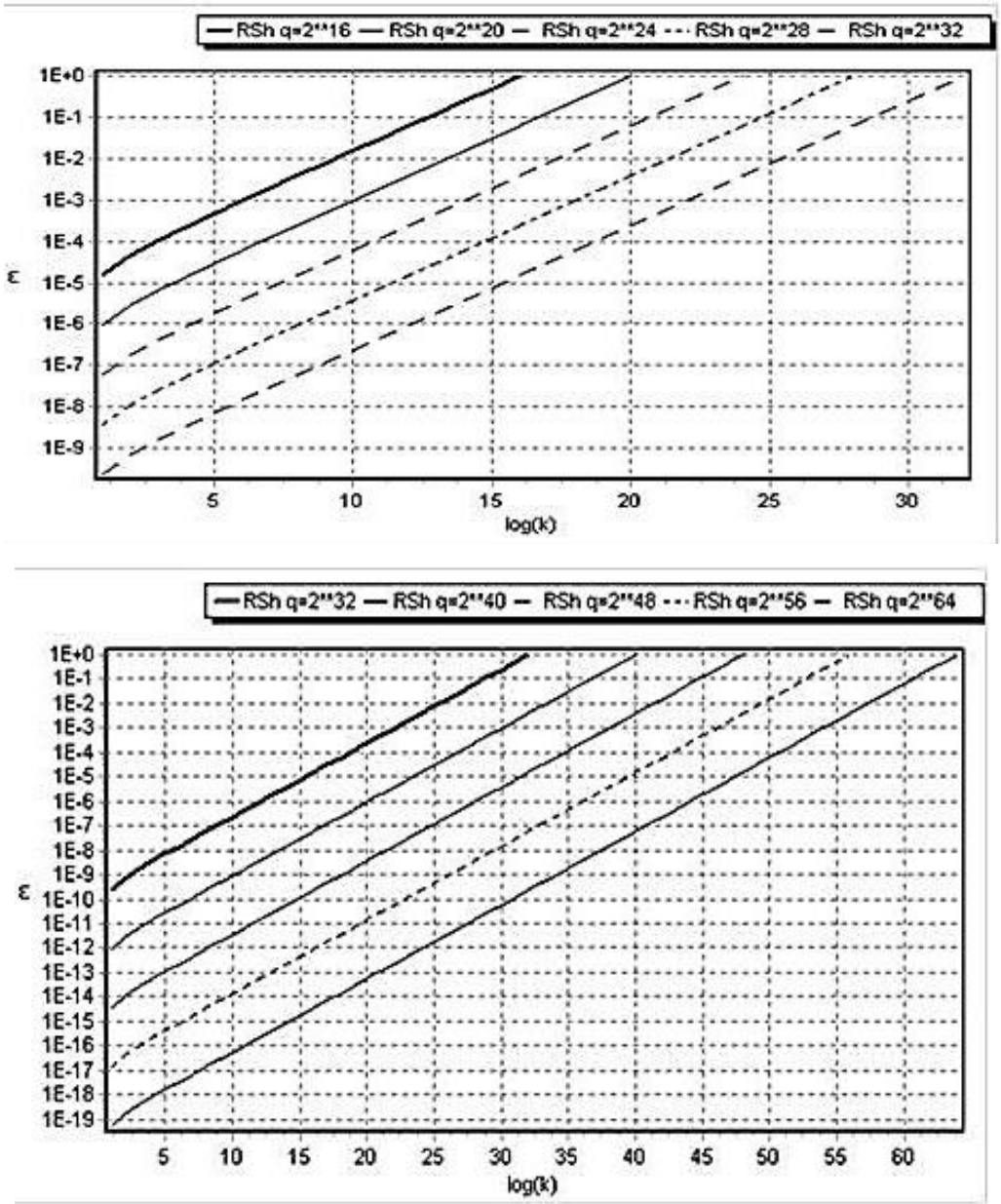


Рис. 4.9. **Графіки залежності ймовірності колізії для RSh_q від довжин даних при різних значеннях розміру кінцевого поля**

При RSh_q,обчислення MAC кодів у кінцевому полі Fq визначається таким виразом:

$$h_x(m) = \sum_{i-0}^{k} m_i x^i,$$

де x – ключове слово, $m = (m_1, m_2, ..., m_k)$ – повідомлення, x, $m_i \in Fq$, k об'єм повідомлення, $k < q$.

Варто розглянути випадок, коли $q = p$, де p при RSh_q,обчислення MAC кодів у кінцевому полі Fq визначається таким виразом:

$$h_x(m) = \sum_{i-0}^{k} m_i x^i,$$

де x – ключове слово, $m = (m_1, m_2, ..., m_k)$ – повідомлення, x, $m_i \in Fq$, k – об'єм повідомлення, $k < q$ – просте число.

Обчислення геш-кодів у простому полі визначається за правилами медулярної арифметики. Далі наведені визначення для RSh_q гешування у простому полі. Нехай p – просте число і k ціле число $k > 0$. RSh_q у простому полі визначається виразом:

$$h_x(m) = \sum_{i-0}^{k} m_i x^i \bmod p,$$

де $x \in Zq$, $m = (m_1, m_2, ..., m_k)$, $m_i \in Zq$.

Обчислення RSh_q можна здійснити по ітераційній схемі Горнера, з однією операцією множення і складання в кінцевому полі на кожному кроці [87]. Важливим аспектом практичної реалізації, який враховує 32-розрядну машинну арифметику, є вибір простого поля $Zq(w)$, де $p(w)$ – розрядність поля (бітове просте число). Кращий результат досягається при арифметичних перетвореннях у полі $Zq(w)$ при якомога більшому простому числі $p(w) \le 2^w$ [65]. У табл. 4.1 наведені прості числа, які були розраховані за допомогою програми пошуку і тестування простих чисел із використанням алгоритму Рабінера – Міллера.

Таблиця 4.1

Прості числа для RSh_q алгоритму і значення зниження ймовірності колізії при обчисленнях у простому полі

w	p(w)	p(w)	Оцінка ймовірності колізії
16	2^{16}-15	0xFFF0	$k2^{-12}$
20	2^{20}-3	0x000FFFF3	$k2^{-18}$
24	2^{24}-3	0x00FFFFF3	$k2^{-21}$
28	2^{28}-57	0x0FFFFF C3	$k2^{-22}$
32	2^{32} –5	0xFFFFFFFB	$k2^{-29}$
64	2^{64} –59	0xFFFFFFFF FFFFFFC5	$k2^{-58}$
128	2^{128} –159	0xFFFFFFFF FFFFFFFF FFFFFFFF FFFFFF61	$k2^{-120}$

Ітераційна схема Горнера обчислення геш-функції передбачає обчислення $y \leftarrow xy + m \bmod p(w)$.

Як витікає із аналізу залежностей імовірності колізії від поля в якому виконуються обчислення, розмір кінцевого поля F_q повинен бути якомога більшим. Ураховуючі 32-розрядну машинну арифметику, доцільно використовувати модуль $p(32)$. Звичайне застосування асемблерної операції $\bmod p(32)$ потребує 12.4cpb (циклів процесора на байт) Pentium II(400).

У роботі [44] відмічається, що можливо використовувати більш ефективний алгоритм для обчислення. Як наведено в табл. 4.1, простий модуль p(w) можна подати у вигляді $p(w) = 2^w - c$. Використовуючи подання $xy = a2^w + b$, і враховуючі, що $a2^{32} = ca$ в $Zq(w)$, можна одержати:

$$y = xy + m \bmod p(w) = ca + b + m \bmod p(w).$$

З аналізу останнього виразу виходить, що при обчисленнях на одному кроці ітерації значення y буде потрібно одне множення і два додавання на w розрядних регістрах і декілька команд, які дозволяють контролювати вихід результату обчислень за межі діапазону подання

чисел у полі $Zq(w)$. Результати моделювання обчислень показали, що другий алгоритм виконується на порядок швидше в арифметиці $p(32)$.

Недоліком наведеного алгоритму є зменшення ключового простору до величини $d = 2^w / c$ і збільшення ймовірності колізії до $k2^{-d}$ через високу швидкість обчислень.

Для максимального значення модуля $p(32) = 23^2 - 5$ запропонований алгоритм використовує 8 рядків асемблерної програми і досягає продуктивності 3.69cpb Pentium II(400). Зменшення ключового простору приводить до величини $23^2 / 5 = 2^{29}$ біт і збільшення ймовірності колізії до $k2^{-29}$. Практична схема ефективного алгоритму обчислення RSh_q повинна включати ще одне додаткове перетворення. Вектори чисел з w бітами, які мають елементи ззовні діапазону подання $Zq(w)$, необхідно перетворити у вектор, який їх не має, за допомогою так званого подвійного подання, як це зроблено в UMAC алгоритмі, що приводить до подвоєння розміру даних і відповідно до збільшення ймовірності колізії до величини $k2^{-(d-1)}$ і відповідно зниження швидкості обчислень. З оцінками розробників UMAC алгоритму швидкість обчислень для $p(32)$ знижується до значення 3.86cpb Pentium II(400) [88].

Подальше підвищення розрядності регістрів обчислення $RSh_q p(w)$ за значення $w - 32$ призводить до значного зниження швидкості обчислень. Так реалізація RSh_q для $p(64) = 2^{64} - 59$ має імовірність колізії $k2^{-49}$, використовує 40 рядків асемблерної програми і має пікову продуктивність 6.86cpb Pentium II(400) [25]. Зниження всього в 2 рази швидкості обчислень визначається тим, що автори використовували MMX технологію, яка базується на 4-векторному представленні 16-розрядних даних і відповідних командах швидкісного множення.

Аналіз показує, що зменшення ключового простору досягає величини $\approx 2 \div 3$ біт і приводить відповідно до збільшення ймовірності колізії в $2^2 \div 2^8$ раз залежно від модуля, що використовується.

Лінійне зростання ймовірності колізії для RSh_q обмежує розмір повідомлення, що гешуєтья (див. теорему 4.22). Чим більше розмір ключового простору, тим більшу кількість слів можна гешувати до досягнення допустимої ймовірності колізії. Потужність множини ключів

визначається значенням простого числа p(w), яке визначає поле Zq(w). При збільшенні p(w) зростають часові витрати на обчислення багаточленів у Zq(w). При збільшенні p(w) від 2^{32} до 2^{64} часові витрати збільшуються у два рази.

Арифметика $GF(2^w)$ є менш зручною для сучасних мікропроцесорів [26; 28; 29], хоч і забезпечує легке розділення бітових рядків, що гешуються, на w бітові підрядки. При цьому відсутні втрати з боку ймовірності колізії і з боку об'єму ключових даних, котрі виникають при гешуванні в полі $Zq(w)$. Обчислення RSh_q визначаються виразом (4.9) у розширеному кінцевому полі GF_q характеристики $p = 2$, оскільки й у Zq(w), реалізується за схемою Горнера.

Ключовим моментом даної схеми є обчислення множення елементів поля $GF(2^w)$. Відомо декілька ефективних алгоритмів швидкого множення в $GF(2^w)$ [24 – 27], орієнтованих на табличне множення елементів поля або на модифікації схеми Монтгомері. Залежно від вимог до програмної реалізації вибирається або просте поле обчислень, або розширене $GF(2^w)$.

З теореми 4.1 виходить, що при гешуванні за схемою із АГ кодами доцільно використовувати коди великої довжини і з великою відносною кодовою відстанню. До класу таких кодів відносяться коди на кривих Ерміта (HC) і Сузуки (SC).

Параметри універсальних геш-функцій з використанням кодів на кривих Ерміта.

Важливою конструкцією АГ кодів є HC. HC визначені над розширеним полем GF_q, $q = p^2$ і мають параметри [67]:

$$\left[q\sqrt{q}, k, q\sqrt{q} - k + 1 - g\right],$$

де рід кривої $g = \sqrt{q}(\sqrt{q} - 1)/2$.

Застосування теореми 4.22, дозволяє отримати універсальний геш-клас для кодів на кривих Ерміта (HCh_q).

Теорема 4.23. [60].

Нехай GF_q, $q = p^2$ – розширенекінцеве поле і $1 \le k \le q\sqrt{q}$, тоді існує:

$$\frac{1}{\sqrt{q}}\left(\frac{k-1}{q}+1-\frac{1}{\sqrt{q}}\right) - U(q\sqrt{q}, q^k, q)\, HCh_q .$$

Твердження 4.1. [99]. Нехай GF_q, $q = p^2$ – розширене кінцеве поле. Для універсальної HCh_q де $k*_{HC}$ визначається виразом:

$$k*_{HC} = \sqrt{q} + 1$$

і ймовірність колізії при $k = k*_{HC}$ дорівнює:

$$\varepsilon(k*_{HC}) = 1/\sqrt{q} .$$

Доведення очевидно і не приводиться.

При малих довжинах даних, коли $k < \sqrt{q} + 1$, схема RSh_q за асимптотичною оцінкою має більшу перевагу, бо в неї найменша ймовірність колізії.

Аналіз графіків показує, що HCh_q забезпечує збільшення об'єму гешованих даних у $\sqrt{q}$ раз.

Для $q = 2^{32}$ об'єм гешованих даних може досягати 2^{39} 32-розрядних слів, що перекриває весь діапазон практичних додатків. При цьому ймовірність колізії обмежується значенням 10^{-3} (2^{-10}). Зниження ймовірності колізії можливо шляхом збільшення q. Так, при $q = 2^{64}$ ймовірність колізії знижується до асимптотичної границі не менше, ніж на 5 порядків до 10^{-8}.

HC є конструктивними, реалізуються відносно просто в тій же арифметиці, що і RS. HCh_q забезпечують меншу ймовірність колізії в порівнянні з RSh_q. Для асимптотичної оцінки ймовірність колізії можна ввести так зване порогове значення довжини повідомлення $k*$. Порогове значення $k*$ показує, при якій довжині даних імовірність колізії в HCh_q

менше, ніж у схемі RSh_q. Порогові значення довжини повідомлення для HCh_q визначається твердженням 4.6.

Серед довгих АГ кодів теоретичний інтерес мають SC.

Параметри універсальних геш-функцій з використанням кодів на кривих Сузукі.

Відомо, що SC [29; 43; 44] визначені над розширеним полем GF_q, $q = p^{2f+1}$ і мають параметр:

$$[q^2, k, q^2 - k + 1 - g],$$

де рід кривої $g = q\sqrt{q} / \sqrt{2}$.

Користуючись результатами теореми 4.22, можна одержати універсальний геш-клас для кодів на кривих Сузукі (SCh_q).

Теорема 4.24. [59].

Нехай GF_q, $q = p^{2f+1}$ – розширене кінцеве поле і $1 < k < q^2$, тоді існує $1/q[(k-1)/q + \sqrt{q/2}] - U(q^2, q^k, q)$ SCh_q.

Для $q = 2^{32}$ об'єм даних, для яких обчислюється SCh_q може досягати 2^{54} 32-розрядних слів, що перекриває на даний момент весь діапазон практичних додатків. При цьому ймовірність колізії обмежується значенням 10^{-3} (2^{-10}). Зниження ймовірності колізії можливо шляхом збільшення q. Так, при $q = 2^{64}$ ймовірність колізії знижується по асимптотичній границі не менше, ніж на 7 порядків до 10^{-10}. Це істотно краще, ніж при RSh_q і HCh_q. Порогове значення довжини k^*_{SC} для гешування повідомлень у SCh_q визначається таким твердженням.

Твердження 4.2 [29]. Нехай GF_q, $q = p^{2f+1}$ – розширене кінцеве поле. Для універсального класу із SC k^*_{SC} визначається виразом:

$$k^*_{SC} = \frac{\sqrt{q}}{2} + \frac{\sqrt{q}}{2(q-1)} + 1,$$

і ймовірність колізії при $k = k*_{SC}$ дорівнює:

$$\varepsilon(k*_{SC}) = \frac{1}{2\sqrt{q}} + \frac{1}{2\sqrt{q}(q-1)}.$$

Граничне значення для SCh_q практично співпадає з граничним значенням для HCh_q. Універсальне SCh_q також по асимптотичній границі програє на малих довжинах RSh_q.

Аналіз залежності асимптотичних границь імовірності колізії $\varepsilon(k)$ для конструкцій з АГ кодами показує, що при $k < \sqrt{q}$ перевагу мають RS. При $k > \sqrt{q}$, слід віддати перевагу HC і SC. Універсальне RSh_q має меншу ймовірність колізії в порівнянні з універсальним HCh_q на всіх довжинах повідомлень. Разом з тим, відсутність практичного, ефективного алгоритму обчислення геш-коду для SCh_q обмежує їх застосування зараз у схемах універсального гешування.

HCh_q при $k = q+1$ мають значення $\varepsilon(q+1) = \frac{2}{\sqrt{q}} - \frac{1}{q}$ і при подальшому зростанні $k \to q\sqrt{q}$, ε практично лінійно наближається до 1. Таким чином, застосування АГ кодів для гешування даних дозволяє забезпечити гешування повідомлень завдовжки $k \le q$ з імовірністю колізії, яка не перевищує значення $\varepsilon = \frac{2}{\sqrt{q}}$ і складністю обчислень, яка визначається арифметикою поля GF_q і алгоритмом побудови кодових слів.

Складність обчислення геш-функції RSh_q для повідомлень завдовжки k слів складе k операцій додавання і множення. Застосування HCh_q потребує, це не складно відобразити, приблизно $k + \sqrt{q}$ операцій додавання і множення [99; 100].

Таким чином, RSh_q ефективніше щодо витрат на обчислення, але при довжинах даних, які перевищують значення $k > \sqrt{q}$, програє по ймовірності колізії HCh_q. Основні результати за схемами гешування з розглянутими кодовими конструкціями наведені в табл. 4.2.

Таблиця 4.2

Основні параметрі схем гешування із АГ кодами

№	Назва схеми	Асимптотичні оцінки для ймовірності колізії ε	Поле обчислень	Довжина гешованих даних
1	RSh_q	$\frac{k-1}{q}$	$GF_q,\ q=p^m$	$1 \le k \le q$
2	HCh_q	$\frac{1}{\sqrt{q}}\left(\frac{k-1}{q}+1-\frac{1}{\sqrt{q}}\right)$	$GF_q,\ q=p^2$	$1 \le k \le q\sqrt{q}$
3	SCh_q	$\frac{1}{q}\left[\frac{k-1}{q}+\sqrt{\frac{q}{2}}\right]$	$GF_q,\ q=p^{2f+1}$	$1 \le k \le q^2$

Недолік гешування з АГ кодами полягає в тому, що ймовірність колізії лінійно збільшується зі зростанням довжини повідомлення, а її зменшення можливо шляхом обмеження довжини гешованого повідомлення, або збільшення розміру поля GF_q обчислення геш-коду.

Як наслідок із проведеного аналізу, для побудови геш-функцій найбільшу цікавість представляють АГ коди, які ефективні в обчислювальному відношенні та мають велику довжину. До таких практичних кодів відносяться НС.

Побудова довгих АГ кодів на кривих Ерміта.

При гешуванні по схемі із АГ кодами, HC коди мають параметри котрі визначаються такою теоремою.

Теорема 4.25. [43].

Нехай, зафіксовані: гладка АГ крива X роду g над F_q, набори точок $P=\{P_1,P_2,...,P_n\} \subseteq X(F_q)$, $Q=\{Q_1,Q_2,...,Q_s\} \subseteq X(F_q)$, $Q \cap P = \varnothing$ (умова не суттєва і знімається використанням пучкової конструкції), ефективний дивізор $D=\sum u_Q Q$ степені $\deg(D)=\sum u_Q$ при цьому $\deg(D) > 2g-2$. З дивізором D асоціюється лінійний простір L(D) із раціональних функцій $f_0, f_1, ..., f_{k-1}$ на X як множина всіх функцій f таких, що порядок f у кожній точці Q задовольняє умові $\mathrm{div}(f) \ge -u_Q$.

Тоді лінійний код C над F_q визначається як:

$$C=\{f(P_1),...,f(P_n) \mid f \in L(D)\}.$$

АГ код має параметри:

$$[n, \deg(D) - g + 1, d],\ d \geq n - \deg(D),$$

а дуальний до нього код також є АГ з параметрами:

$$[n, n - \deg(d) + g - 1, d^{\perp}],\ d \geq n - \deg(D) - 2g + 2.$$

Практичні НС мають довжину q^3. Криві Ерміта визначені над полем F_{q^2} / F_q за допомогою відповідних форм, котрі включають представлення для норм $N(x) = x^{q+1}$ і треків $tr(y) = y^q + y$.

Варто вивчити основні форми кривих Ерміта. З цією метою потрібно розглянути криві над полем GF_{16}. Рівняння кривих Ерміта в розширеному полі GF_{16} мають степінь $d = q + 1 = 5$. Криві, що не приводяться, від змінних x, y, z із кількістю точок рівною 65, подані в табл. 4.3. Усі криві Ерміта є формами із норм і слідів. Це легко показати, якщо перейти від проективного подання кривих до афінного, фіксуючи одну з координат, наприклад, $z = 1$. У табл. 4.4 наведені всі можливі форми неприводних кривих Ерміта. Єдина відсутня в цьому списку форма $N(x) + tr(x) + N(y) + tr(y)$, як легко показати, містить співмножник $(x + y)$. Слід розглянути останнє рівняння:

$$X^4Y + X^4Z + XY^4 + XZ^4 + Y^4Z + YZ^{\$} + Z^5 = 0.$$

У аффінних координатах при z = 1:

$$(x^4 + 1)y + (x + 1)y^4 + x^4 + x + 1 = 0.$$

За допомогою заміни змінних $t = x + 1$ і $h = yt^4$ одержується:

$$t^4y + ty^4 + t^4 + t + 1 = 0 \text{ та } h^4 + h + t^4 + t + 1 = 0.$$

Як результат маємо, що дана форма є кривою степені $d = 4$.

Таким чином, існують незвідні криві степені $d=4$ над полем GF_{16} з кількістю точок, що дорівнює кривим Ерміта. Усі форми кривих степені 4 подані в табл. 4.4. Рівняння 2, 3, 4 табл. 4.4 інваріантні поліноміальним формам і є мінімальними багаточленами, що не приводяться, над GF_{16}.

Таблиця 4.3

Форми кривих Ерміта для поля GF_{16}

№	Рівняння кривих у проективних координатах	Підстановки і заміни	Рівняння кривих у афінних координатах	Інваріантні рівняння
1	$X^5+Y^5+Z^5$	$z=1$	x^5+y^5+1	$N(x)+N(y)+1$
2	$X^5+Y^4Z+YZ^4$	$z=1$	x^5+y^4+y	$N(x)+tr(y)$
3	$X^5+Y^4Z+YZ^4+Z^5$	$z=1$	x^5+y^4+y+1	$N(x)+tr(y)+1$
4	$X^5+Y^4Z+X^4Z+XZ^4+YZ^4$	$z=1$	$x^5+x^4+x+y^4+y$	$N(x)+tr(x)+tr(y)$
5	$X^5+X^4Z+XZ^4+Z^5+Y^4Z+YZ^4$	$z=1$	$x^5+x^4+x+y^4+y+1$	$N(x)+tr(x)+tr(y)+1$
6	$X^5+X^4Z+XZ^4+Z^5+Y^4Z+YZ^4+Y^5$	$z=1$	$x^5+x^4+x+y^4+y+1$	$N(x)+tr(x)+N(y)+$ $+tr(y)+1$
7	$X^4Y+X^4Z+Y^4X+Z^4X+Y^4Z+Z^4Y+Z^5$	$z=1,$ $t=x+1$ $h=yt^4$	h^4+h+t^4+t+1	$tr(h)+tr(t)+1$

Таблиця 4.4

Форми неприводимих кривих степені $d=4$ **над полем** GF_{16}

№	Рівняння кривих у проективних координатах	Підстановки і заміни	Рівняння кривих у афінних координатах	Інваріантні рівняння
1	$X^4+Y^4+XZ^3+YZ^3+Z$	$z=1$	x^4+x+y^4+y+1	$tr(x)+tr(y)+1$
2	$X^4+Y^4+X^3Z+X^2YZ+$ $XY^2Z+Y^3+Z+Z^4$	$z=1,$ $x+y=t$	$x^4+y^4+x^3+x^2y+xy^2+y^3+1$	t^4+t^3+1
3	$X^4+X^3Y+X^2Y^2+XY^3+$ $Y^4+X^3Z+X^2YZ+$ $XY^2Z+Y^3Z+Z^4$	$x=1,\ y+1=t,$ $z+1=h,\ t^3h=s$	$1+y+y^2+y^3+y^4+z+$ $yz+y^2z+y^3z+z^4$	s^4+s^3+1
4	$X^4+Y^4+X^3Z+X^2YZ+$ $XY^2Z+Y^3+Z+X^2Z^2+$ $Y^2Z^2+XZ^3+YZ^3+Z^4$	$z=1,$ $x+y=t$	$x^4+y^4+x^3+x^2y+xy^2+$ $y^3+x^2+y^2+x+y+1$	$t^4+t^3+t^2+t+1$

4.3. Універсальне гешування на основі модулярних перетворень

Ці схеми сконструйовані для використання доступного апаратного забезпечення для виробництва цифрових схем. Їх стійкість частково залежить від складності певної кількості теоретичних проблем і вони легко масштабуються (класифікуються по складності). Недоліком є те, що повинні бути передбачені застереження проти атак, які використовують математичні структури, подібні фіксованим точкам у модулярній експоненціалізації [10] (тривіальні приклади – 0 і 1), мультиплікативні атаки й атаки з малими числами, для яких не відбувається модулярного зменшення (значення).

Схеми з малим модулем.

Перший тип схем використовує тільки малий модуль (близько 32 біт). Дженеман запропонував першу версію в роботі [70], але деякі атаки призвели до деяких нових версій. Остання схема була зламана Копперсмітом.

Схеми з великим модулем.

Другий клас схем використовує великий модуль (розмір модуля n звичайно дорівнює 512 біт і більше). У цьому випадку операндами є елементи кільця, відповідного до модуля RSA, тобто добуток двох великих простих чисел, або елементи поля Галуа GF(p) чи GF(2^n). Для зручності ці схеми будуть позначені як схеми з більшим модулем, хоча ця термінологія застосовується строго тільки до схем, що базуються на RSA. Останній клас може бути розділений на практичні схеми й схеми, які доказово стійкі. У випадку модуля RSA необхідно розв’язати таку практичну проблему: особа, яка згенерувала модуль, знає його розкладання на множники, і тому вона має потенційну перевагу над іншими користувачами геш-функции. Розв’язок – модуль може бути згенерований довіреною третьою стороною (у цьому випадку не можна використовувати модуль користувача), або можна сконструювати геш-функцію таким чином, що перевага (того, хто генерує) обмежена.

Найбільш ефективні схеми базуються на модулярному зведенні у квадрат. Більше того, деякі теоретичні результати припускають, що інвертування модулярного зведення у квадрат без знання розкладання

модуля є важкою проблемою. Знову можна вивчити всі можливі схеми, які використовують просте зведення у квадрат і операції XOR із внутрішньою пам'яттю тільки в 1 блок. Деякі схеми цього типу були оцінені в попередніх роботах [54; 99; 100]. Можна показати, що оптимальна схема має таку форму: $f = (X_i \otimes H_{i-1})^2 \bmod N \oplus H_i$.

Найбільш привабливі пропозиції використовують добре відомий режим зчеплення блоків (Cipher Block Chaining (CBC)).

$$f = (X_i \otimes H_{i-1})^2 \bmod N.$$

З метою уникнути вразливостей (легко можна пройти у зворотному напрямі), додана додаткова надмірність у повідомлення. Перша пропозиція була зафіксувати 64 найбільш значущих бітів у 0. Однак у роботах [54; 76] було показано, що це небезпечно. У новій пропозиції, яка з'явилася в національних та інтернаціональних стандартах надмірність була розсіяна. Копперсміт показав, що можна створити два повідомлення, такі що їх геш-код буде кратним один до іншого [26; 77]. Якщо геш-функція комбінована з мультиплікативною схемою підпису подібною RSA [103], можна використовувати цю атаку для підробки підписів. Як наслідок, були визначені нові методи додавання надмірності в роботах [25; 26; 70], однак ці методи ще вивчаються.

Інші схеми, що базуються на зведенні у квадрат, що є нестійкими, – це схема Дамгарда, яка була зламана Боером і схема Дженга Матсумого й Імаі, яка вразлива до атаки, яку описав Дамгард, що виділив деякі слабкості в MDC (і MAC), що базуються на арифметиці в GF(2^{593}).

Були запропоновані більш сильні схеми, що вимагають більше операцій. Прикладами є використання двох операцій зведення у квадрат [24]: $f = (H_{i-1} \oplus (x_i)^2)^2 \bmod N$ і заміна зведення у квадрат зведенням у більші степені (3 або 2^{16+1}) у попередніх схемах. Це дозволяє спростити надмірність [6; 24].

Можна припустити, що було б бажано знайти надійну схему надмірності для геш-функції, що базується на модулярній арифметиці й замінити режим CBC більш стійким режимом. Якщо прийнятна більш повільна схема, показник степені може бути збільшений.

У схемах з доказовою стійкістю Дамгард припускає, що знаходження колізії еквівалентно розкладанню на множники модуля RSA або знаходженню дискретного логарифма за модулем великого простого числа. Конструкція Гібсона [64] допускає стійку до колізій функцію, що

базується на дискретному логарифмі з модулем – складним числом. Проблема факторизації та знаходження дискретного логарифму відомі як складні числові теоретичні проблеми. Недоліками цих схем є те, що вони дуже неефективні.

Основною ідеєю геш-функцій заснованих на модулярній арифметиці, є використання у якості циклової функції ітеративної функції, що використовує модулярну арифметику. Причинами стандартизації та застосування таких геш-функцій є забезпечення необхідного рівня стійкості. Основним недоліком функцій є низька швидкість формування геш-кода [3].

Ураховуючи всі слабкі сторони та обмеження алгоритмів на основі перетворень у групі цілих чисел, потрібно розглянути досить відомі і більш потужні перетворення в групі точок еліптичної кривої.

Арифметика еліптичних кривих. У якості основного криптографічного примітива в несиметричних криптосистемах побудованих над групою точок еліптичної кривої використовують групові операції додавання та подвоєння точок. Загальні положення криптографії на еліптичних кривих наведені у роботі [2]. На практиці сьогодні застосовують, як правило, гладкі (несуперсингулярні) еліптичні криві у полі GF(2^m) $EC: y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_5$.

Нехай задана несуперсингулярна еліптична крива над полем GF(2^m) в аффінному поданні: $y^2 + xy = x^3 + ax^2 + b(\mathrm{mod}\, f(x), 2)$, де a і b – параметри ЕС.

Вона визначена множиною точок (x, y) ∈ GF (2^m) × GF (2^m), a і b ∈ GF (2^m),b ≠ 0 (mod f(x), 2). Точки на ЕС, включаючи точку нескінченності O, утворять групу з операцією додавання.

Якщо точки $P_1 = (x_1, y_1)$ й $P_2 = (x_2, y_2)$ належать еліптичній кривій, тобто $P_i \in E$ (GF(2^m), то для кожної з них існує зворотна точка, відповідно – $P_1 = (x_1, x1+y1)$ й – $P_2 = (x_2, x_2 + y_2)$, а також точка $P_3 = (x_3, y_3)$, така що $P_1 + P_2 = P_3$. Координати точки $P_3 = (x_3, y_3)$ визначаються з використанням співвідношень:

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \ (\mathrm{mod} f(x), 2),$$

$$y_3 = \lambda(x_1 + x_3) + x_3 + y_1 \ (\mathrm{mod} f(x), 2),$$

$$\lambda = \begin{cases} \frac{y_1 + y_2}{x_1 + x_2} (\mathrm{modf}(x), 2), & \text{якщо} \quad P_1 \neq P_2; \\ \frac{x_1^2 + y_1}{x_1} (\mathrm{modf}(x), 2), & \text{якщо} \quad P_1 = P_2. \end{cases}$$

скалярне множення визначається для декількох точок G ϵ E(GF(2^m)) як:

$$d \cdot G(\mathrm{modf}(x), 2) = \underbrace{G + G + G + \ldots + G}_{d\,\text{раз}}(\mathrm{modf}(x), 2)$$

Ця операція реалізується за рахунок застосування операцій додавання ($P_1 \neq P_2$) або подвоєння ($P_1 = P_2$).

Точка G має порядок n на ЕК, якщо $n \cdot G(\mathrm{modf}(x), 2) = O$, де O є точка на нескінченності (нуль).

Найважливішим завданням при виконанні операцій є мінімізація складності. У такий спосіб використання несуперсингулярної кривої вимагає високої обчислювальної складності та накладає додаткові обмеження. Удосконалення методу ключового гешування з застосуванням перетворень у групі точок еліптичної кривої.

Спосіб 1. (MASH(EC1)).

Алгоритм удосконаленого способу ключового гешування даних MASH-2, в основі якого лежить теоретико-складна проблема взяття дискретного логарифма в групі точок еліптичної кривої реалізується за таким алгоритмом. Текст mi, де i = 0; n, подається координатою X точки еліптичної кривої EC, після обчислюється координата Y. Точка кривої $P_1(x_1, y_1)$ множиться на таємний скаляр k (особистий ключ користувача): $P_k = k * P_1$.

Цикловою функцією алгоритму гешування є скалярний добуток секретного числа на точку кривої.

Наступна точка, якою подається такий блок вихідного тексту, множиться на результат модифікації скаляра k: $k_i = k_{i-1} \oplus X[P_{i-1}]$.

Результатом гешування є координати точки кривої, яка отримується на останньому раунді: $h(M) = X[P_n] \oplus Y[P_n]$.

У загальному вигляді геш-функцію можна подати таким виразом:

$$h_i = X[k_i * P_i] = X[(k_{i-1} \oplus X[P_{i-1}]) * P_i] \oplus Y[(k_{i-1} \oplus X[P_{i-1}]) * P_i],$$

де $P_i = \varphi_{EC}(M_i)$ – результат відтворення з блоку вихідного повідомлення M_i точки кривої.

Алгоритм ключового гешування наведений на рис. 4.12.

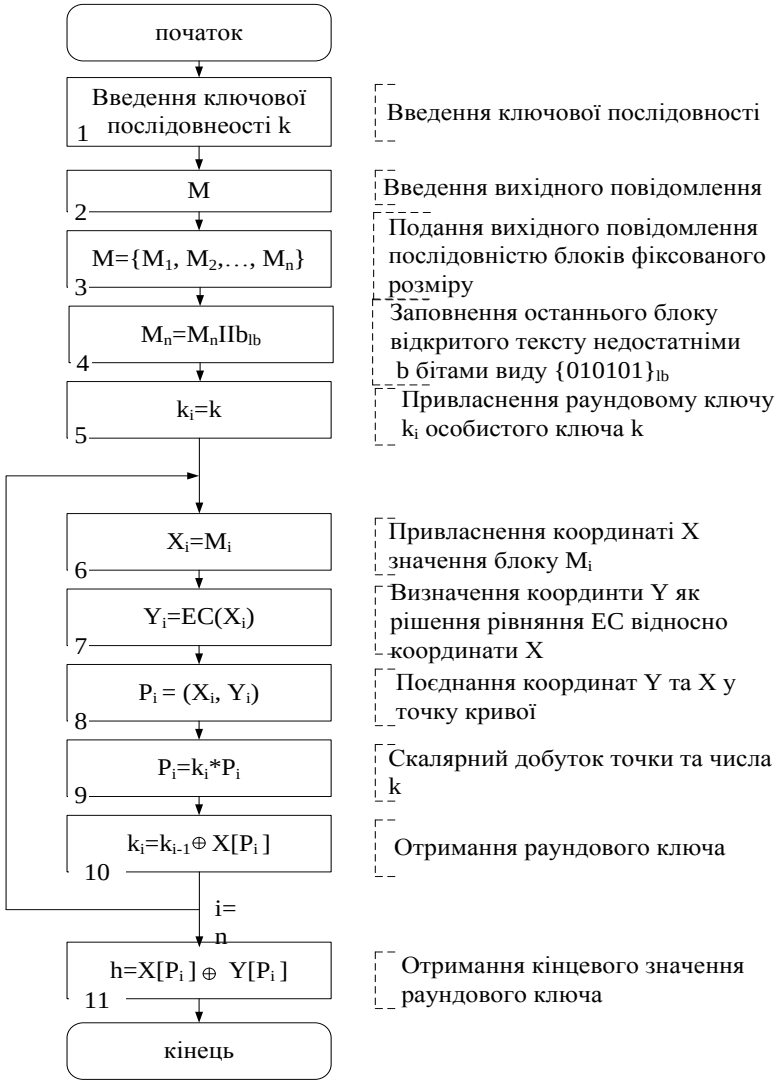


Рис. 4.12. **Алгоритм ключового гешування MASH(EC)**

Першим кроком необхідно встановити ключові параметри геш-функції (крок 1). Розмір ключової послідовності l_k біт є також розміром блоку вихідного повідомлення.

Після розбиття повідомлення на блоки (крок 3) здійснюється доповнення неповного блоку відкритого тексту M_b недостатніми b бітами, кількість яких визначається як $l_b = l_k - l_{M_n}$. Наступним кроком присвою-

ється раундовому ключу значення ключової послідовності. Після присвоєння значення і-го блоку координаті X еквівалентної точки кривої (крок 5) знаходиться координата Y (крок 6) та створюється точка (крок 7). Потім здійснюється скалярний добуток (крок 8) та встановлюється ключ такого раунду (крок 9).

Таким чином, алгоритм з 5 по 8 крок виконується n разів, стільки скільки блоків відкритого тексту.Останнім кроком виконується додавання за модулем 2 координат X та Y. На виході геш-функції отримується послідовність довжиною l_k. Останній крок дозволяє позбутися випадку, коли функція може приймати нульове значення, якщо використовувати в якості геш-кода значення однієї з координат.

Більша частина обчислювальної складності основного кроку запропонованого способу (рис. 4.12) доводиться, на операцію скалярного множення точки еліптичної кривої.

Для зниження обчислювальної складності основного кроку може бути використанний інший спосіб ключового гешування.

Спосіб 2. (MASH(EC2)).

Аналогічним чином представляється текст точками кривої $m_i \rightarrow x_i$, по координаті X визначається точка кривої, потім точки кривої складаються та множаться на секретний скаляр, а результат гешування представляється координатою точки кривої: $h = X[k * (P_1 + P_2 + \ldots + P_n)]$.

Аналіз алгоритмів показує, що при використанні алгоритму формування геш-коду за допомогою першого способу (множення кожного блоку повідомлення, який перетворений в точку еліптичної кривої, на скаляр (секретний ключ)) значно знижує його швидкість, що не дозволяє його використовувати при передачі повідомлень вільної довжини, але стійкість геш-коду значно вища, тому що основується на обчислювально-складній задачі знаходження дискретного алгоритму в групі точок еліптичної кривої.

При використанні другого алгоритму (множення суми точок еліптичної кривої, які є представленням блоків повідомлення, на скаляр (секретний ключ)) який дозволяє формувати геш-код вільної довжини за час, який задовольняє вимогам до сучасних ЦП (час формування ЦП на еліптичних кривих (ДСТУ-4145)).

Пропонуються два методи вдосконалення алгоритмів ключового гешування. Перший метод (множення кожного блоку повідомлення, що перетворений в точку еліптичної кривої, на скаляр (секретний ключ)) пропонується використовувати при передачі транзакцій фіксованої довжини до 1,5 – 2 Кбіта, другий (множення суми точок еліптичної кривої,

які є представленням вихідного повідомлення, на скаляр) – при передачі повідомлення вільної довжини.

Проведені дослідження показали, що застосування еліптичних кривих у алгоритмах ключового гешування дозволяє забезпечити необхідну стійкість та автентичність передачі даних в інформаційних системах.

Метод формування кодів контролю цілісності й автентичності даних на основі модулярних перетворень

В основі методу ключового універсального гешування доказової стійкості лежить використання модулярних перетворень, що забезпечують зведення задачі знаходження прообразу або секретного ключа в схемі гешування до однієї з відомих обчислювально-складних задач, наприклад, до задачі факторизації (RSA-подібні схеми) або дискретного логарифмування. Подібне обґрунтування стійкості за класифікацією моделей безпеки NESSIE прийнято вважати доказовою безпекою, підкреслюючи тим самим зведення задачі криптоаналізу до однієї з добре відомих обчислювально-складних задач [124].

Обґрунтування безпеки доказово стійких криптосистем базується на прийнятті припущення про існування так званих однобічних функцій. Однобічна функція f: x → y, задана на множині x й сферою значень y множині y має дві властивості:

існує поліноміальний алгоритм обчислення f(x);

не існує поліноміального алгоритму інвертування функції f(x), тобто розв’язку рівняння f(x) = y.

Виконання другої зазначеної властивості на сьогоднішній день не доведене ні для одного з відомих кандидатів на однобічну функцію, тобто саме існування однобічних функцій не доведене. У той час, на використанні деяких з них будуються практично всі відомі несиметричні криптоалгоритми [71; 72]. Слід проаналізувати теоретико-складні завдання (обчислювально-складні задачі) на предмет побудови циклової функції при ітеративному формуванні геш-кодів. У табл. 4.5 наведені результати досліджень циклових функцій: у першому стовпчику зазначено теоретико-складні завдання, які покладені в основу їх побудови, у другому стовпчику наведено аналітичний запис циклової функції, у третьому стовпчику – оцінка складності обчислення значення циклової функції, у четвертому – оцінка обчислювальної складності її інвертування (оцінка стійкості).

Проведені дослідження показують, що найбільш доцільним розв’язком можна вважати використання циклової функції, задача інвертування якої

сполучена з розв'язком теоретико-складної задачі обчислення квадратного кореня за модулем n. При визначених обмеженнях на значення складеного модуля n це завдання по обчислювальній складності інвертування порівняне із проблемами факторизації й дискретного логарифмування. У той же час пряме обчислення значення функції $a \equiv (x^2) \bmod(n)$ вимагає значно меншого числа операцій. Варто відзначити, що використання квадратичної циклової функції не приводить до побудови універсального гешування.

Наступною за обчислювальною складністю є циклова функція:

$$f(x_i, H_{i-1}) = (x_i \oplus H_{i-1})^e \bmod(N),$$

завдання інвертування якої пов'язане з розв'язком теоретико-складного завдання RSA, де $\gcd(e, \varphi(p,q)) = 1$, $N = pq$, $\gcd(x, y)$ – найбільший загальний дільник чисел x і y.

Оцінки обчислювальної складності знаходження значень цієї циклової функції, а також складності її інвертування свідчать про те, що при виконанні відповідних обмежень на значенні експоненти вдається будувати доказово стійке перетворення, використовуване для шифрування/розшифрування й/або формування/перевірки цифрового підпису в криптосистемі RSA.

Використання подібного перетворення з метою ключового гешування застосовується в алгоритмі MASH-2, однак виконання умови специфікацією алгоритму не гарантоване.

Таким чином, застосування циклової функції на основі модулярного зведення в степінь дозволяє будувати доказово безпечне універсальне гешування тільки при виконанні обмежень на значення модульної експоненти й значення модуля перетворень.

Ще одним кандидатом на циклову функцію в ітеративній схемі гешування є функція виду: $f(x_i, H_{i-1}) = (\alpha^{x_i \oplus H_{i-1}}) \bmod(p)$, завдання інвертування якої сполучена з розв'язком теоретико-складні завдання дискретного логарифмування, де α – генератор кільця цілих чисел Z_p, а p – велике просте ціле число.

Використання такої циклової функції забезпечує побудову доказово безпечного гешування, колізійні властивості якого задовольняють умовам універсальності.

Таблиця 4.5

Кандидати на побудову циклової функції ітеративного гешування інформації

Теоретико-складні завдання	Кандидати на побудову циклової функції схеми гешування	Оцінка складності обчислення	Оцінка складності інвертування
Проблема цілочисельної факторизації	$f(x_i, H_{i-1}) = x_i H_{i-1}$, функція визначена над більшими простими числами $x_i = p$ та $H_{i-1} = q$	$O(n^2)$, де $n = \lceil \log_2 p \rceil + \lceil \log_2 q \rceil$	$L_N(\alpha, \beta) = \exp\left((\beta + o(1))(\log N)^{\alpha}(\log\log N)^{1-\alpha}\right)$. Для поля чисел загального виду складність інвертування становить $L_N\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right)$. Для поля чисел спеціального виду $N = a^b + c$ складність інвертування становить $L_N\left(\frac{1}{3}, \sqrt[3]{\frac{32}{9}}\right)$
Проблема RSA	$f(x_i, H_{i-1}) = (x_i \oplus H_{i-1})^e \bmod(N)$, $\gcd(e, \varphi(p,q)) = 1$, $N = pq$	$O(\log_2 e)$ множень, алгоритм швидкого зведення в степінь	
Проблема дискретного логарифмування	$f(x_i, H_{i-1}) = \left(\alpha^{x_i \oplus H_{i-1}}\right) \bmod(p)$, α – генератор Z_p	$O(\log_2 n)$ множень, алгоритм швидкого зведення в степінь, $O(n^3)$ для $\alpha = 2$, де $n = \lceil \log_2 p \rceil$	$\min\left\{\sqrt{p}, L_N(\alpha, \beta)\right\}$, де $L_N(\alpha, \beta) = \exp\left((\beta + o(1))(\log N)^{\alpha}(\log\log N)^{1-\alpha}\right)$. Для примітивного поля GF(p) складність інвертування становить $\min\left\{\sqrt{p}, L_N\left(\frac{1}{3}, \sqrt[3]{\frac{64}{9}}\right)\right\}$. Для розширеного поля $GF(2^m)$ складність інвертування становить $L_N\left(\frac{1}{3}, 1{,}4\right)$
Проблема Діффі – Хеллмана	$f(x_i, H_{i-1}) = \left(\alpha^{x_i \oplus H_{i-1}}\right) \bmod(p)$, α – генератор Z_p	$O(n^3)$ для $\alpha = 2$, де $n = \lceil \log_2 p \rceil$	

При виконанні відповідних обмежень ітеративне формування геш-кодів дозволяє з однієї сторони забезпечити виконання умов моделі доказової безпеки, тобто забезпечити високу криптографічну стійкість, з іншого боку – забезпечити виконання умов універсального гешування, тобто забезпечити високі колізійні властивості схеми гешування.

Платою за досягнення таких властивостей гешування є порівняно висока обчислювальна складність формування геш-коду.

Проведений аналіз показує, що найбільш витратною з обчислювальної точки зору операцією при реалізації циклових функцій є операція модульного зведення в степінь. При безпосередньому зведенні в степінь через ланцюжок операцій множення обчислювальна складність реалізації таких циклових функцій зростає пропорційно показнику степені, тобто для зведення числа x в степінь n у загальному випадку потрібно виконати $n-1$ множень: $x^n = \underbrace{x \cdot x \cdot x \cdot ... \cdot x}_{n-1 \text{ умножений}}$.

Асимптотична оцінка обчислювальної складності такої реалізації операції зведення в степінь є $O(n)$ *множень.*

Для зниження обчислювальної складності реалізації схем гешування з використанням циклових функцій застосований алгоритм швидкого зведення в степінь , в основі якого лежить представлення числа x^n в такому вигляді:

$$x^n = x^{((...((m_k \cdot 2 + m_{k-1}) \cdot 2 + m_{k-2}) \cdot 2 + ...) \cdot 2 + m_1) \cdot 2 + m_0} =$$

$$= ((...(((x^{m_k})^2 \cdot x^{m_{k-1}})^2 ...)^2 \cdot x^{m_1})^2 \cdot x^{m_0},$$

де $(m_k, m_{k-1}, ..., m_0)$ – двійкове представлення числа n, тобто $m_i \in \{0,1\}$ й $n = m_k \cdot 2^k + m_{k-1} \cdot 2^{k-1} + ... + m_1 \cdot 2 + m_0$.

Перегрупувавши співмножники у представленні числа x^n можна одержати такий вираз: $x^n = x^{m_0} \cdot (x^2)^{m_1} \cdot (x^{2^2})^{m_2} \cdot (x^{2^3})^{m_3} \cdot ... \cdot (x^{2^k})^{m_k}$, звідки випливає, що для зведення числа x в степінь n потрібно реалізувати не більше k операцій зведення у квадрат і не більше k операцій множень, де $k+1$ – кількість елементів у двійковому записі числа n, тобто $k = (\log_2 n) - 1$. Таким чином, асимптотично обчислювальну складність обчислення x^n можна оцінити як $O(\log_2 n)$.

Реалізація алгоритму швидкого зведення числа x в степінь n зводиться до такої ітеративної процедури.

Вхід. Число x й двійковий вектор $(m_k, m_{k-1}, ..., m_0)$ як двійкове представлення показника ступеня n.

Вихід. Число $a = x^n$.

1. Встановити $a = 1$, $a_k = x$, $i = k$.
2. Якщо $m_i = 1$ обчислити $a = a \cdot a_i$.
3. Обчислити $a_i = (a_i)^2$.
4. Прийняти $i = i - 1$.
5. Якщо $i \geq 0$ перейти до кроку 2.
6. Якщо $i < 0$ в якості результату прийняти число $a = x^n$.

Наведений алгоритм дозволяє суттєво прискорити процедуру обчислення циклових функцій.

У табл. 4.6 наведені залежності складності реалізації операції зведення в степінь через ланцюжок множень і через представлення із зазначенням порядку модуля перетворення, мінімально необхідного для забезпечення необхідного рівня безпеки.

Таблиця 4.6

Оцінки обчислювальної складності реалізації операції зведення в степінь різними методами

Метод зведення в степінь	Порядок модуля перетворень/еквівалентна довжина ключа симетричного криптоалгоритму		
	1024 / 80	3072 / 128	15360 / 256
Через ланцюжок добутків	10308	10924	104623
Швидкий алгоритм зведення в степінь	2046	6142	30718

Для порівняння з іншими схемами ключового гешування за показниками стійкості й швидкодії можна прийняти такі припущення. Нехай одна операція множення над числами, порядку 2^m вимагає $\left\lceil \frac{m}{L} \right\rceil$ операцій порозрядного додавання за модулем два (XOR), де L – розрядність процесора використовуваної обчислювальної системи, $\lceil x \rceil$ – округлене

до більшого цілого число x. Подібне припущення найбільш часто застосовується при оцінці складності реалізації криптоалгоритмів [88]. У цьому випадку оцінка $\left\lceil \frac{m}{L} \right\rceil$ дає приблизне число циклів L – розрядного процесора, необхідних для реалізації одного множення над числами, бітова довжина яких не перевершує m. У той же час, при гешуванні з використанням модулярних перетворень обробляється відразу $m/8$ байт інформаційних даних.

У табл. 4.7 наведені результати порівняльних досліджень швидкодії схем ключового гешування для фіксованих показників безпеки. Показник швидкодії виражений у кількості S циклів 32-розрядного процесора, необхідних для формування одного байта вихідних даних. Показник безпеки фіксувався через довжину секретного ключа, який необхідно зламати зловмисникові. Для схем на модулярній арифметиці наведена еквівалентна довжина ключа блокового симетричного криптоалгоритму.

Таблиця 4.7

Оцінка складності алгоритмів гешування в кількості *S* циклів 32-розрядного процесора на один байт оброблюваних даних

Функція гешування	Рівень стійкості (довжина ключа)	Кількість циклів S
SHA-2 (512)	512	80
SHA-2 (256)	256	64
SHA-1	160	80
RIPEMD-160	160	160
MD5	128	64
Гешування на модулярній арифметиці	80	512
	128	1536
	256	7680

Наведені дані в табл. 4.7 свідчать, що використання модулярних перетворень для розв'язку завдань ключового гешування суттєво підвищує складність обчислень, швидкодія алгоритмів знижується на 1 – 2 порядки. У той же час, такі схеми ключового гешування мають доказово стійкий рівень безпеки (завдання знаходження ключа гешування або прообразу зводиться до розв'язку відомого теоретико-складністного завдання).

Крім того, такі схеми автентифікації задовольняють властивостям універсального гешування, чим забезпечуються високі колізійні характеристики формованих МАС.

4.4. Каскадні схеми універсального гешування

МАС коди, побудовані з використанням класу універсальних геш-функцій, мають високу швидкодію і секретність, що доводиться. Для гешування за схемою з АГ кодами доцільно використовувати коди великої довжини із великою відносною кодовою відстанню. До класу таких схем гешування відносяться конструкції на основі RS і HC. Разом з тим існуюче обмеження на розмір поля обчислення геш-значень приводить до обмеження довжини повідомлення. RSh_q більш ефективне на малих довжинах повідомлення і програє HCh_q на великих довжинах повідомлення [25].

Це протиріччя знімається в методі імітозахисту, на основі каскадної схеми з АГ кодами.

Дослідження властивостей композиційних схем універсального гешування.

Визначення композиційних схем універсального гешування та їх властивості розглянуті у [25].

Нехай $H = \{h : \{0,1\}^a \to \{0,1\}^b\}$ – це клас геш-функцій. Очевидний спосіб збільшення значення образу повідомлення полягає в конкатенації проміжних геш-кодів. Потрібно визначити клас геш-функцій $H^m = \{h : \{0,1\}^{am} \to \{0,1\}^{bm}\}$ у вигляді:

$$h(x_1x_2...x_m) = h(x_1) \| h(x_2) \| ... \| h(x_m), \qquad (4.1)$$

де $|x_i| = a, h(x_i)$ є функцією H класу.

Якщо H це $\varepsilon - AU$ універсальний клас геш-функцій, тоді H^m також є $\varepsilon - AU$. H^m клас забезпечує m кратне збільшенню геш-коду без збільшення складності обчислень і без зміни ймовірності колізії.

Для зменшення ймовірності колізії слід використати гешування повідомлень по декількох незалежно обраних геш-функціях (Картер – Вегман [55 – 57; 90].

Нехай $H_1 = \left\{h : \{0,1\}^a \to \{0,1\}^b\right\}$ і $H_2 = \left\{h : \{0,1\}^a \to \{0,1\}^c\right\}$ це клас геш-функцій. Потрібно визначити клас геш-функцій $H_1 \cap H_2 = \left\{h : \{0,1\}^a \to \{0,1\}^{b+c}\right\}$ у вигляді:

$$(h_1, h_2)(x) = h_1(x) \,||\, h_2(x). \qquad (4.2)$$

Якщо H_1 це $\varepsilon_1 - AU$ універсальний клас і H_2 це $\varepsilon_2 - AU$, тоді $H_1 \cap H_2$ є $\varepsilon_1\varepsilon_2 - AU$.

Розповсюдження конструкції (4.1) на m кратне гешування приводить до схеми з ймовірністю колізії, пропорційної степені ε^m. Значення геш-коду збільшується в m раз і складність обчислень збільшується в m раз.

Для зменшення розміру геш-коду використовується композиційна конструкція Стінсона [87]. Композиційні схеми є ефективним механізмом побудови класів геш-функцій із заданими комбінаторними властивостями.

Нехай $H_1 = \left\{h : \{0,1\}^a \to \{0,1\}^b\right\}$ і $H_2 = \left\{h : \{0,1\}^a \to \{0,1\}^c\right\}$ є клас геш-функцій. Визначимо клас геш-функцій $H_1H_2 = \left\{h : \{0,1\}^a \to \{0,1\}^c\right\}$ у вигляді:

$$(h_1, h_2)(x) = h_2(h_1(x)). \qquad (4.3)$$

Якщо H_1 це $\varepsilon_1 - AU$ універсальний клас і H_2 це $\varepsilon_2 - AU$, тоді $H_1 \cap H_2$ є $(\varepsilon_1 + \varepsilon_2) - AU$.

Композиційне включення геш-функцій з метою зменшення розміру геш-коду приводить до збільшення ймовірності колізії та зростання складності обчислень і, можливо, обсягу ключових даних.

Основні результати згідно з розглянутими композиційними схемами наведено у табл. 4.8.

Таблиця 4.8

Основні параметри композиційних схем гешування

№	Назва схеми	Формула обчислень	$Pr_{h\in H}[h(x_1) = h(x_2)]$	Складність обчислень
1	Гешування з конкатенацією геш-кодів	$h(x_1x_2...x_m) = = h(x_1) \,||\, h(x_2) \,||\, ... \,||\, h(x_m)$	$\varepsilon - AU$	m
2	Багатократне гешування	$(h_1, h_2)(x) = h_1(x) \,||\, h_2(x)$	$\varepsilon_1\varepsilon_2 - AU$	m^2
3	Композиційна конструкція Стінсона	$(h_1, h_2)(x) = h_2(h_1(x))$	$(\varepsilon_1 + \varepsilon_2) - AU$	трохи більше m

Техніка зміни властивостей класу геш-функцій H визначається схемами включень (4.2), (4.3) є загальною для всіх класів геш-функцій і спрямована на те, щоб при обчисленнях у кінцевих полях забезпечити певну ймовірність колізії геш-коду при обмеженні на складність обчислень і обсягу ключових даних.

Проведений аналіз показав [55 – 57; 86; 87; 99; 100], що практичні схеми імітозахисту повинні будуватися на основі класів універсальних геш-функцій. При цьому забезпечується найбільша швидкодія і доведена секретність. Побудова універсальних класів геш-функцій із використанням довгих АГ кодів дозволяє забезпечити великий коефіцієнт стиснення для вхідних повідомлень великого обсягу. При цьому через обмеження на поле обчислень геш-коду ймовірність колізії лінійно зростає з довжини вхідних повідомлень. Використання композиційних схем істотно розширює практичні можливості по розробці схем імітозахисту з заданими властивостями. Однак ті пропозиції, котрі на даний момент існують, не задовольняють вимогам щодо формування геш-коду з обчисленнями в малому кінцевому полі, з низькою ймовірність колізії і для даних з гнучким діапазоном значень довжини. Для вирішення цього протиріччя в роботі пропонується каскадна схема гешування.

Метод імітозахисту на основі каскадної схеми гешування із застосуванням АГ кодів.

Запропонована конструкція каскадного гешування будується на основі техніки композиційних схем. Каскадна схема імітозахисту запропонована в роботах [26; 39; 44].

Визначення каскадної схеми універсального гешування має таке подання.

Нехай $H_1 = \left\{h : \{0,1\}^a \to \{0,1\}^b\right\}$ і $H_2 = \left\{h : \{0,1\}^c \to \{0,1\}^d\right\}$ є родини геш-функцій. Потрібно визначити каскадний клас геш-функцій $H_1H_2 = \left\{h : \{0,1\}^a \to \{0,1\}^d\right\}$ у вигляді:

$$h_1, h_2(x) = h_2(h_1(x_1) \| x_2), \tag{4.4}$$

де $|x_1| = a$, $|x_2| = c$, $a < c$.

Структурна схема каскадного гешування наведена на рис 4.13.

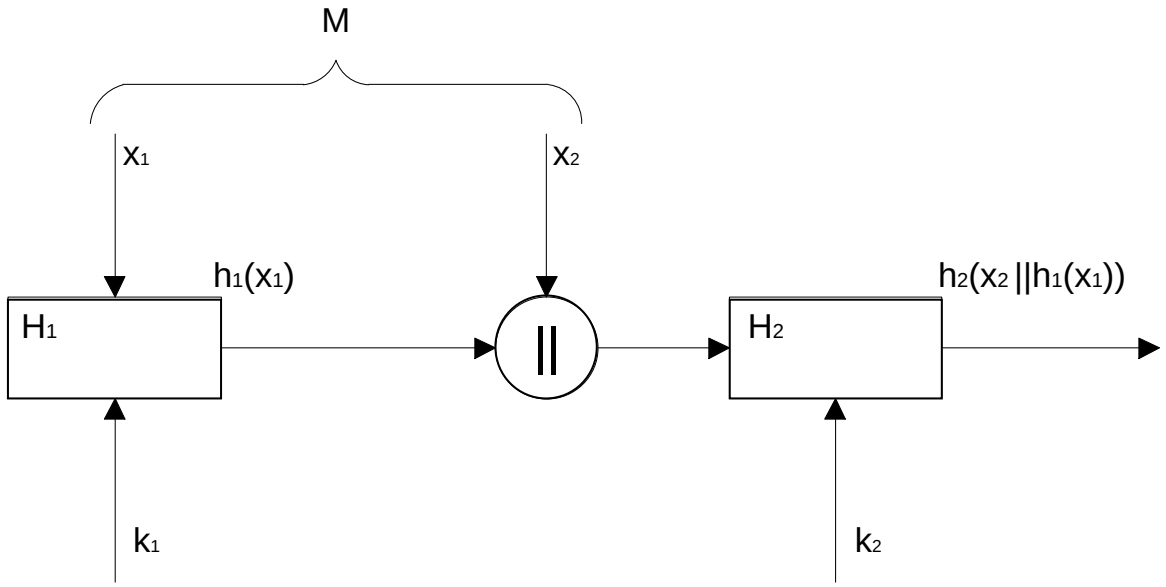


Рис. 4.13. **Структурна схема каскадного гешування**

Колізійні властивості каскадної конструкції визначаються таким твердженням.

Якщо H_1 є $\varepsilon_1 - AU$ універсальний клас і H_2 – це $\varepsilon_2 - AU$, тоді H_1H_2 є ε–AU, де $\varepsilon = \max(\varepsilon_1, \varepsilon_2) + 1/|H^2|$.

Нехай $z = z_1 \| z_2$, $|z_1| = a, |z_2| = c$. За визначенням (4.1) геш-код обчислюється спочатку над першим блоком даних z_1 за допомогою $h_1(x)$, потім за допомогою функції $h_2(x)$ обчислюється результуючий геш для рядка з конкатенацією попереднього результату гешування і рядка даних, що залишився $h_1(z_1) \| z_2$. Припустимо, що ймовірність колізії в цьому випадку $Pr_{k \in H_1H_2}[h(x) = h(y)] \le \varepsilon_2$.

Нехай $x = x_1 \| x_2$, $|x_1| = a, |x_2| = c$, $y = y_1 \| y_2$, $|н_1| = a, |н_2| = c$, причому x, y – різні і $|x| \ne |y|$.

Якщо $|x| \le a$ і $|y| \le a$, тоді виникає колізія $h_1(x) = h_1(y)$, що визначається значенням ε_1 для першого каскаду гешування.

Якщо $|x| > a$ і $|y| > a$, тоді колізія визначається $h_2(x) = h_2(y)$. За причини, що ймовірності колізії різні, вона буде дорівнювати ε_2 для другого каскаду гешування.

Нехай $|x| > a$ і $|y| \le a$, тоді колізія виникає у випадку, коли $h_2(h_1(x_1) \| x_2) = h_1(y_1)$. При довільному виборі геш-функції h_1 гешування

одержується геш-код $h_1(x_1)$ і $h_1(y_1)$. Оскільки значення $h_2(h_1(x_1) \| x_2)$ обчислюється h_2 гешуванням, ймовірність колізії буде дорівнювати ε_2.

Розглянемо випадок, коли $|x| > |y|$ і $|x| > a$, $|y| > a$. Нехай $x_2 \neq y_2$ і тому як $h_1(x_1) \| x_2$ і $h_1(y_1) \| y_2$ є різними, тоді $\varepsilon = \varepsilon_2$.

Нехай $x_1 \neq y_1$ і $x_2 = y_2$ тоді колізія визначається рівністю, таким чином $\varepsilon = \varepsilon_1$. У випадку, якщо $h_1(x_1) \neq h_1(y_1)$, тоді $h_1(x_1) \| x_2$ і $h_1(y_1) \| y_2$ відрізняються тільки у першому слові. Для колізії необхідно, щоб виконувалась умова $h_2(h_1(x_1) \| x_2) = h_2(h_1(y_1) \| y_2) = 0$. Імовірність цієї події дорівнює $1/|H^2|$ і значення ймовірності колізії для цього випадку буде дорівнювати $\varepsilon = (\varepsilon_1) + 1/|H^2|$.

Таким чином, імовірність колізії ε при каскадному гешуванні визначається максимальним значенням ε_1 або ε_2.

Розглянемо умови мінімізації ймовірності колізії у каскадній схемі. З цього приладу справедливе таке твердження.

Нехай H_1 і H_2 відповідно $\varepsilon_1 - AU$ і $\varepsilon_2 - AU$ універсальні класи геш-функцій. Каскадна конструкція H_1H_2, буде мати меншу ймовірність колізії, якщо $\varepsilon_1 = \varepsilon_2$.

При каскадному гешуванні набір даних розбивається на підблоки $x = x_1 \| x_2$, причому $|x_1| = a, |x_2| = c$, $|x| = a + c$. Ймовірності колізій ε_1 і ε_2 для кожного підблока є монотонними зростаючими функціями від довжин даних $|x_1|$ і $|x_2|$. Тому як $\varepsilon = \max(\varepsilon_1, \varepsilon_2)$, тобто, мінімальне значення ε досягається при умові мінімуму $\varepsilon_1(|x_1|)$ і $\varepsilon_2(|x_2|)$, котрий досягається при рівності $\varepsilon_1(|x_1|) = \varepsilon_2(|x_2|)$.

Як витікає з визначення каскадного гешування і результатів твердження, перевагою запропонованої схеми є обмеження ймовірності колізії найбільшим значенням ймовірності колізії одного з каскадів.

При цьому допускається збільшення об’єму даних приблизно в два рази і також зростають витрати на об’єм ключових даних у два рази. Порівняння каскадної схеми гешування з розглянутими раніше композиційними схемами подано в табл. 4.9.

На відміну від композиційної конструкції Стінсона каскадне гешування не збільшує ймовірність колізії і майже не збільшує складність обчислень, при цьому можна ефективно змінювати комбінаторні влас-

тивості геш-кодів. Найбільш важливим є те, що можна збільшувати об'єм гешованих даних і відповідно розмір ключових даних без збільшення ймовірності колізії і складності обчислень.

Застосування каскадної конструкції для гешування на основі АГ кодів дозволяє посилити традиційні універсальні кодові схеми. АГ коди для універсальних класів геш-функцій забезпечують найкраще співвідношення між розміром геш-коду, довжиною даних і ймовірністю колізії при обчисленнях в кінцевих полях, а також мінімізує витрати на формування ключових даних і витрат на обчислення [26; 44].

Таблиця 4.9

Порівняння параметрів композиційних схем гешування

№	Назва схеми	Формула обчислень	$\Pr_{h\in H}\left[\begin{matrix}h(x_1)=\\=h(x_2)\end{matrix}\right]$	Складність обчислень	Об'єм ключів
1	Гешування із конкатенацією геш-результатів	$h(x_1x_2...x_m)=$ $=h(x_1)\|\|h(x_2)\|\|...\|\|h(x_m)$	$\varepsilon - AU$	m	$\|H\|$
2	Багатократне гешування	$(h_1,h_2)(x)=$ $=h_1(x)\|\|h_2(x)$	$\varepsilon_1\varepsilon_2 - AU$	m^2	$\|H_1\|+\|H_2\|$
3	Композиційна конструкція Стінсона	$(h_1,h_2)(x)=h_2(h_1(x))$	$(\varepsilon_1+\varepsilon_2)-AU$	трохи більше m	$\|H_1\|+\|H_2\|$
4	Каскадне гешування	$h_1,h_2(x)=$ $=h_2(h_1(x_1)\|\|x_2)$	$\max(\varepsilon_1,\varepsilon_2)+$ $+1/\|H^2\|-AU$	$m+1$	$\|H_1\|+\|H_2\|$

Метод імітозахисту на основі каскадної схеми гешування з використанням кодів Ріда – Соломона і Ерміта.

Відповідне визначення каскадної схеми гешування на основі АГ кодів має такий вигляд.

Нехай F_q – кінцеве поле, M – повідомлення і $M=M_1\|M_2$. Алгоритм обчислення геш коду в каскадній конструкції визначається виразом [27; 43]:

$$Ch_q(M)=AGh_q(AGh_q(M_1)\|M_2), \qquad (4.5)$$

де AGh_1, AGh_2 це універсальні схеми гешування на основі довгих АГ кодів.

Наступна теорема описує універсальні і колізійні властивості $Ch_q(M)$ конструкції.

Нехай AGh_1, AGh_2 є $\varepsilon_1 - AU$ і $\varepsilon_2 - AU$, відповідно, універсальні схеми гешування на основі довгих АГ кодів. Тоді $Ch_q(M)$ визначає універсальний клас геш-функцій $\varepsilon - AU$, де $\varepsilon = \max(\varepsilon_1, \varepsilon_2) + 1/|H^2|$, $\varepsilon_1, \varepsilon_2$ відповідно, ймовірності колізії для AGh_1, AGh_2 гешування.

Далі розглянуто властивості каскадної схеми гешування з використанням RSh_q, HCh_q.

Для розв'язання протиріччя між довжиною гешованих даних, імовірністю колізії і складністю обчислень пропонується каскадна конструкція RSh_q і HCh_q.

Нехай F_q, $q = p^2$ – розширене кінцеве поле, M – повідомлення і $M = M_1 \| M_2$. Алгоритм обчислення геш-коду в каскадній конструкції визначається виразом:

$$HCh_q(RSh_q(M_1) \| M_2). \qquad (4.6)$$

Наступна теорема описує результуючі характеристики практичної $Ch_q(M)$ конструкції.

Теорема 4.26.

Нехай F_q, $q = p^2$ – розширене кінцеве поле, $M = M_1 \| M_2$, $|M| \le q\sqrt{q}$, $|M_1| \le \sqrt{q} + 1$, $0 < k \le q\sqrt{q} + \sqrt{q}$. Тоді $Ch_q(M)$ визначає універсальний клас геш-функцій $\varepsilon - U(q^2\sqrt{q}, q^k, q)$, $\varepsilon = \max(\varepsilon_{RS}, \varepsilon_{HC}) + 1/|q\sqrt{q}|$, $\varepsilon_{RS}, \varepsilon_{HC}$ – відповідно, ймовірності колізії для RSh_q і HCh_q гешування.

Нехай $|M| \le \sqrt{q} + 1$, тоді $Ch_q(M)$ є універсальною в силу теореми 4.14. Якщо $|M| > \sqrt{q}$, і $M = M_1 \| M_2$, $|M_1| \le \sqrt{q}$, тоді HCh_q гешування, застосовується до геш-коду $RSh_q(M_1)$ і M_2, що також за теоремою 4.14 є універсальним класом геш-функцій. Оскільки параметри HC і RS визначаються співвідношеннями (4.5), (4.6), кількість геш-функцій $HCh_q(RSh_q(M_1) \| M_2)$ дорівнює $N = qq\sqrt{q} = q^2\sqrt{q}$.

Для оцінки імовірності колізії каскадної схеми слід розглянути такі випадки.

Нехай M та M′ гешоване повідомлення, $M \neq M'$ і $M = M_1 \| M_2$, $M' = M'_1 \| M'_2$. Якщо $|M|$ і $|M'| < \sqrt{q}$ тоді $M = M_1$, $M' = M'_1$а M_2, M'_2 – пусті рядки. Результати гешування повідомлення M на RSh_q позначаються h_1 і на HCh_q, потрібно позначити h_2, для M' відповідно h'_1 і h'_2.

Необхідно розглянути випадок, коли $M \neq M'$.

Якщо $|M| \leq \sqrt{q}+1$ і $|M'| \leq \sqrt{q}+1$, тоді колізія означає, що $h_1 = h'_1$ і визначається значенням ε_{RS} для RSh_q гешування.

При $|M| > \sqrt{q}+1$ і $|M'| > \sqrt{q}+1$, колізія виникає, якщо $h_2 = h'_2$. Оскільки $h_1(M_1) \| M_2$ і $h'_1(M'_1) \| M'_2$ різні, ймовірність колізії визначається значенням ε_{CH} для HCh_q гешування.

Нехай $|M| > \sqrt{q}+1$ та $|M'| \leq \sqrt{q}+1$, колізія виникає у випадку, коли $h_2 = h'_1$. При довільному виборі геш-функції RSh_q гешування одержується геш-код h_1 і h'_1. Оскільки значення h_2 обчислюється HCh_q гешуванням, імовірність колізії буде дорівнювати ε_{CH}.

Варто розглянути випадок, коли $|M| = |M'|$ і $|M| > \sqrt{q}+1$, $||M'| > \sqrt{q}+1$. Нехай $M_2 \neq M'_2$ і оскільки $h_1(M_1) \| M_2$ і $h'_1(M'_1) \| M'_2$ різні, тоді $\varepsilon = \varepsilon_{HC}$.

Нехай $M_1 \neq M'_1$ і $M_2 = M'_2$, тоді колізія визначається рівністю $h_1 = h'_1$ тобто $\varepsilon = \varepsilon_{RS}$. У випадку, якщо $h_1 \neq h'_1$, тоді $h_1(M_1) \| M_2$ і $h'_1(M'_1) \| M'_2$ відрізняються у першому q-м слові. Для колізії необхідно, щоб виконувалась умова $h_1(f(x,y)) = h_2(f(x,y))$, де $f(x,y) = \dfrac{F(x,y)}{G(x,y)}$ – це раціональна функція на кривій Ерміта. Раціональні функції HC можуть бути визначені як добуток раціональних функцій $f_1(x,y) = x$ і $f_2(x,y) = y$. Рівняння Ерміта містить особливу точку $Q(X,Y,Z) = Q(0,1,1)$ і точки (x,y) які визначаються коренями (α,β) рівняння $N(\alpha) = tr(\beta)$, де $N(x) = x^{q+1}$норма, а $tr(\beta) = x + x^q$ – слід розширеного поля F_q^2 / F_q. Значення функції $Ch_q(M) = HCh_q(RSh_q(M_1) \| M_2)$ буде дорівнювати нулю тільки у випадку $\alpha = 0$ і відповідно $\beta = 0$. Імовірність такої події дорівнює $1/q\sqrt{q}$ і значення ймовірності колізії для цього випадку буде дорівнювати $\varepsilon = \varepsilon_{RS} + 1/q\sqrt{q}$.

Таким чином, імовірність колізії при каскадному гешуванні ε визначається максимальним значенням ε_{RS} або ε_{HC}.

Умови мінімізації ймовірності колізії у каскадній схемі визначається таким твердженням [27].

Нехай F_q, $q = p^2$ – розширене кінцеве поле, $|M| \le q\sqrt{q}+1$.

При $|M| \le \sqrt{q}+1$ імовірність колізії при $Ch_q(M)$ буде найменшою, якщо $Ch_q(M) = RCh_q(M)$ і $\varepsilon = \varepsilon_{RS}$.

При $|M| > \sqrt{q}+1$, ε для $Ch_q(M) = HCh_q(RSh_q(M_1) \| M_2)$ гешування буде мінімальним, якщо $\varepsilon_{HC} = \varepsilon_{RS}$.

Дійсно, при $|M| \le \sqrt{q}+1$, $M = M_1 \| M_2$, M_2 – пустий рядок, $\varepsilon_{HC} < \varepsilon_{RS}$ згідно з теоремам 4.2 і 4.3. Тоді $Ch_q(M) = RCh_q(M)$ і $\varepsilon = \varepsilon_{RS}$, якщо $|M| > \sqrt{q}+1$, маємо $M = M_1 \| M_2$ і M_2 – не пустий рядок:

$$Ch_q(M) = HCh_q(RSh_q(M_1) \| M_2).$$

Імовірність колізії ε_{RS} – це монотонно зростаюча функція від довжини даних k. Так як $\varepsilon = \max(\varepsilon_{RS}, \varepsilon_{HC})$, тобто, мінімальне значення ε досягається при умові мінімуму $\varepsilon_{RS}(k)$ і $\varepsilon_{HC}(k)$, котрий досягається при виконанні рівності $\varepsilon_{RS}(k) = \varepsilon_{HC}(k)$.

Аналіз залежностей $\varepsilon(k)$ показує, що при $k \le \sqrt{q}+1$, ε є мінімальним, якщо використовується тільки RSh_q і при $k > \sqrt{q}+1$, ε мінімальне, якщо $|M_1| = \sqrt{q}+1$і $|M_2| = k - \sqrt{q} - 1$.

Результуюча характеристика $\varepsilon(k)$ для каскадної схеми буде визначатися співвідношенням:

$$\varepsilon(k) = \begin{cases} \dfrac{k-1}{q} & k \le \sqrt{q}+1, \\ \dfrac{1}{\sqrt{q}}\left[\dfrac{k-1}{q} + 1 - \dfrac{1}{\sqrt{q}}\right] & k > \sqrt{q}+1. \end{cases} \qquad (4.7)$$

На рис 4.14 наведені графіки залежностей імовірності колізії для $Ch_q(M)$ гешування від довжини даних при різних значеннях поля

обчислень. Де ймовірності колізії для $Ch_q(M)$ гешування від довжини даних буде приймає значення згідно з виразом 4.7; імовірності колізії для $HCh_q(M)$ гешування від довжини даних.

Як показує попередній аналіз, істотне зниження імовірності колізії при збереженні обмежень на розмір поля обчислень можливе за рахунок застосування багатократного гешування на незалежно обраних ключах.

Імітозахист на основі використання багатократного каскадного гешування.

Для зменшення імовірності колізії потрібно скористатися гешуванням повідомлень по декількох незалежно обраних геш-функціях (Картер – Вегман [56]). Розповсюдження конструкції (4.1) на m кратне гешування приводить до такої схеми.

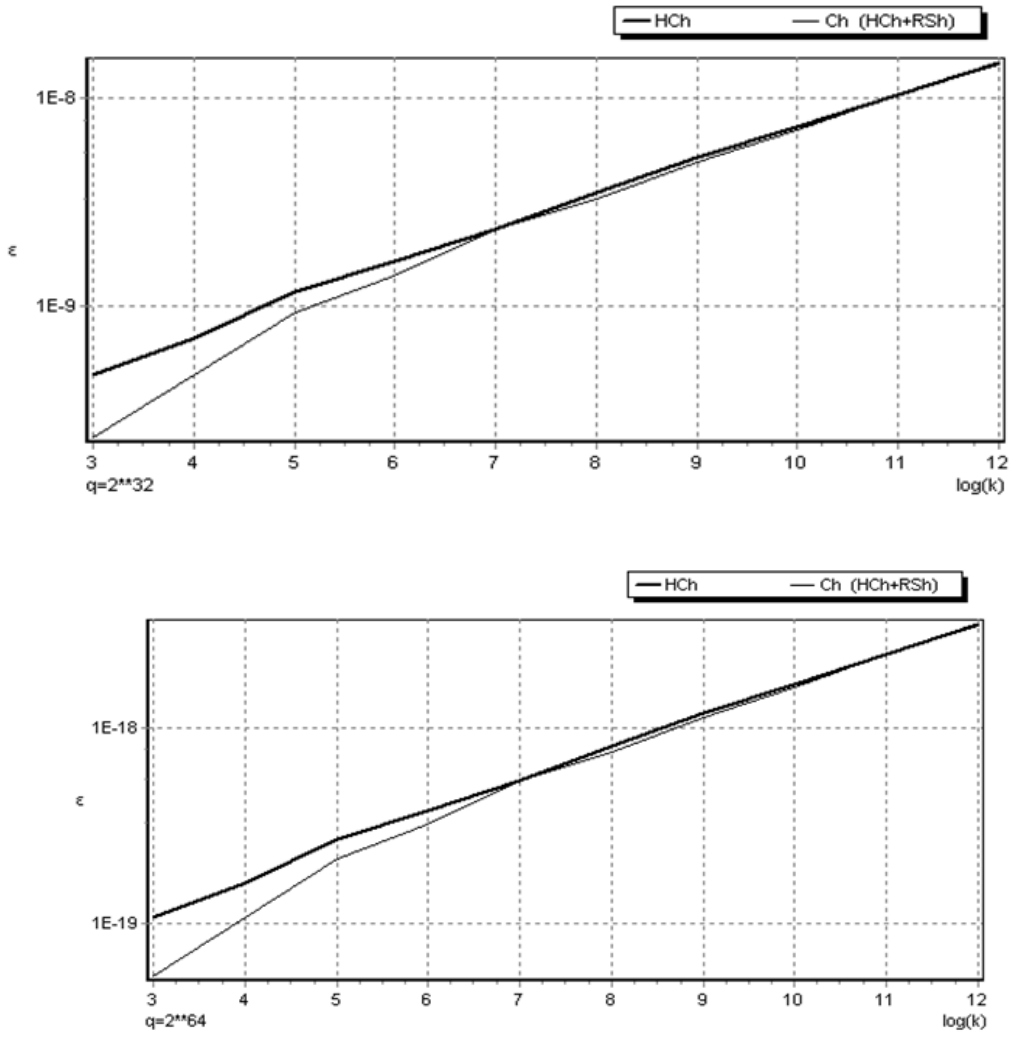


Рис. 4.14. **Графіки залежностей імовірності колізії для** $Ch_q(M)$ **від довжини даних при різних значеннях поля обчислень**

Нехай $H_1 = \{h : \{0,1\}^a \to \{0,1\}^b\}$, …, $H_m = \{h : \{0,1\}^a \to \{0,1\}^d\}$ – це родина геш-функцій.

Слід визначити клас геш-функцій $H_m = \{h : \{0,1\}^a \to \{0,1\}^{b+c+...+d}\}$ у вигляді:

$$(h_1, h_2, ..., h_m)(x) = h_1(x) \,||\, h_2(x) \,||\, ... \,||\, h_m(x). \qquad (4.8)$$

Схему обчислень m кратного геш-коду наведено на рис 4.15.

Аналіз графіків на рис. 4.14 показує, що $Ch_q(M)$ має перевагу по імовірності колізії у порівнянні із схемою HCh_q на малих довжинах даних, що визначається розбиттям повідомлення на два підблоки і, відповідно, зменшенням імовірності колізії при гешуванні кожного із підблоків.

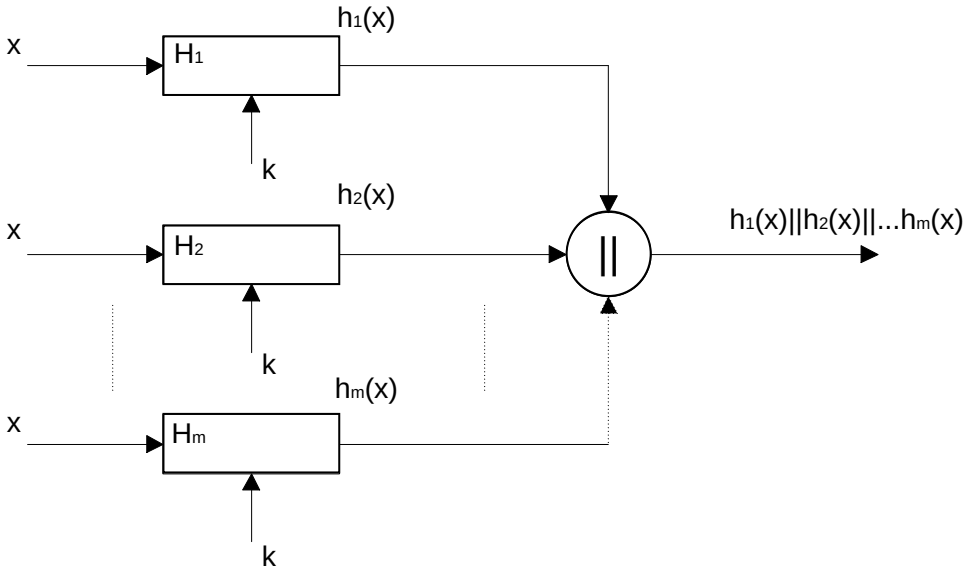


Рис. 4.15. **Схема обчислень** m **кратної геш-функції** H_m

Отримуються оцінки для імовірності колізії схем з багатократним гешуванням. З цією метою потрібно розглянути t-зв'язані схеми імітозахисту.

Визначення t-зв'язаних схем імітозахисту.

Визначення t-зв'заності є узагальненням поняття ортогональних масивів сили t. У вітчизняній літературі із ортогональними масивами ототожнюються t-схеми, ортогональні таблиці [95], t-універсального класу геш-функцій [96].

Нехай $0 \le \varepsilon \le 1$. Масив $(n,k)_p$ є t-зв'язним (t-wise), ε – зміщеним, якщо будь-яка нетривіальна лінійна комбінація не більш ніж t-стовпчиків має зміщення $\text{bias} \le \varepsilon$.

Визначення зміщення було введено Бієрбрауером [96] для оцінки степені відхилення розподілення елементів масиву від рівномірного закону. Наступне визначення на практиці є більш корисним.

Нехай $0 \le \varepsilon \le 1$. Масив $(n,k)_p$ є t-зв'язним, ε-залежним (ε-dependent), якщо для будь-якого набору U із $s \le t$ стовпчиків і кожного вектора $a \in F_p^s$ частота $v_U(a)$ появи в стовпчиках значення a задовольняє умові:

$$\left| \frac{v_U(a)}{n} - \frac{1}{p^s} \right| \le \varepsilon. \qquad (4.9)$$

Для незалежного (0-залежного) масиву $(n,k)_p$, одержується $\frac{v_U(a)}{n} = \frac{1}{p^t}$. У цьому випадку $(n,k)_p$ є ортогональним масивом сили t і утворює t-суворо універсальний клас геш-функцій.

Відомо декілька конструкцій t-зв'язних масивів. Кодово-теоретична конструкція представлена теоремою.

Теорема 4.27.

Нехай $(n,k)_p$ масив B із зміщенням ε і $(N, N-k, t+1)$ – лінійний код. Тоді існує $(n,N)_p$ – масив, котрий є t-зв'язним і ε-зміщеним.

Результуючий масив будується шляхом множення матриці B і перевірочної матриці коду H. Важливе співвідношення між зміщенням і залежністю встановлено в роботі [98].

Теорема 4.28.

Якщо масив є t-зв'язним і ε-зміщеним, він є також і t-зв'язний і ε'-залежний, причому, $\varepsilon' < \varepsilon$.

Фундаментальне значення цієї теореми полягає в тому, що вона визначає можливість застосування слабозміщених масивів у схемах імітозахисту.

Основні визначення для універсальних кодів імітозахисту можна представити в термінології слабозміщених і майже незалежних масивів [25].

$(n,k)_q$ – масив є (ASU_2), якщо його стовпці мають зміщення 0 і для різних стовпців C, C' і будь-яких записів e, e', при рівномірному виборі i-го рядку, умови ймовірності $\Pr(c_i = e \mid c'_i = e') \le \varepsilon$.

Вибір рядку масиву визначається значенням ключа, стовпець – станом джерела даних і значення запису є MAC кодом.

$(N,m)_p$ – масив є (δ, t) – майже суворо універсальним $(\delta - ASU_t)$, якщо для будь-якого набору $U = U_0 \bigcup \{u\}$ із t-стовпців і кожного запису $a' \in F_p^{t-1}, x \in F_p$ відношення частот $v_{U_0}(a')$ і $v_U(a', x)$ задовольняє умові:

$$\left| v_U(a', x) / v_{U_0}(a') \right| \le \delta.$$

Зв'язок між майже незалежними масивами і $\delta - ASU_t$ кодами досить очевидний і був встановлений у роботі [56].

Однією із основних властивостей t-зв'язності є те, що t-зв'язність дозволяє враховувати залежність між геш-кодами і кореляційними зв'язками по ключам геш-функцій.

Багатократне каскадне гешування з АГ кодами.

Багатократне гешування H_m, що визначається виразом (4.1), розповсюджується на універсальне RSh_q, HCh_q і на $Ch_q(M)$. Далі подані такі основні визначення.

Нехай F_q – кінцеве поле, M – повідомлення. Алгоритм обчислення геш-коду в конструкції із t-кратним RSh_q визначається виразом:

$$(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M) = RSh_{q1}(M) \,||\, RSh_{q2}(M) \,||\, ... \,||\, RSh_{qt}(M). \qquad (4.10)$$

Структурна схема $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ конструкції наведена на рис. 4.16.

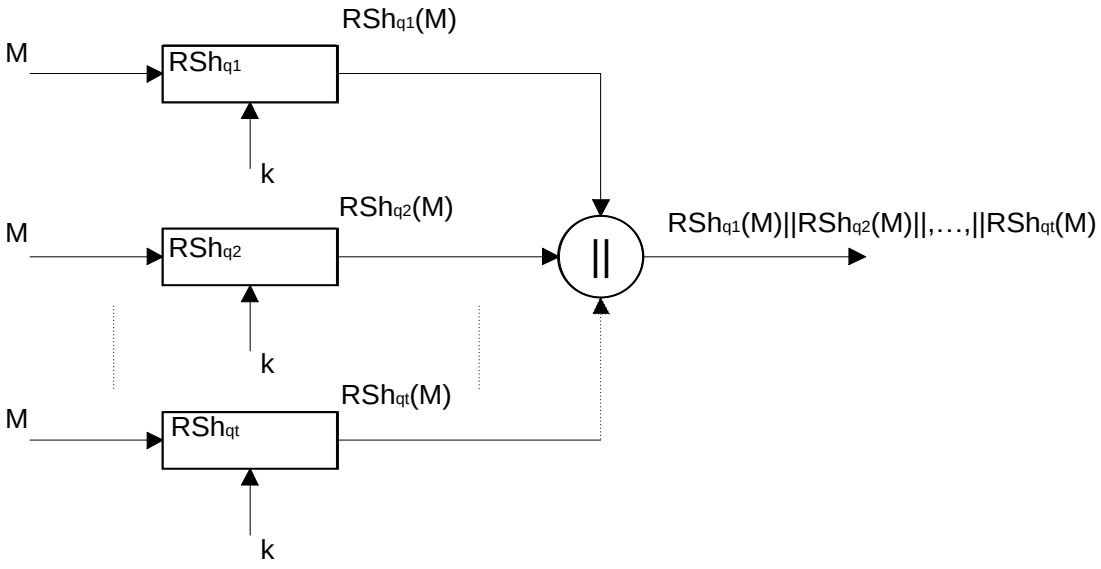


Рис. 4.16. **Структурна схема** $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$**конструкції**

Оцінки для імовірності колізії схеми із багатократним гешуванням визначаються універсальними властивостями $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ конструкції. Справедлива така теорема.

Теорема 4.29 [68]. Клас геш-функцій $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ є t-зв'язним $\varepsilon - U$ універсальним, де $\varepsilon = \frac{k-1}{q}$.

Нехай M і M′ різні повідомлення довільної довжини.

Для M і M′ є геш-коди $RSh_{q1}(M) \| RSh_{q2}(M) \| ... \| RSh_{qt}(M)$ і $RSh_{q1}(M') \| RSh_{q2}(M') \| ... \| RSh_{qt}(M')$, відповідно.

Потрібно розглянути один геш-код із загального рядка. За властивості універсальності RSh_{qi} є:

$$\Pr(RSh_{qi}(M) = RSh_{qi}(M')) \le \varepsilon = (k-1)/q. \qquad (4.11)$$

З іншого боку $\Pr(RSh_{qi}(M) = RSh_{qi}(M')) \le \varepsilon = \left|\frac{v_U(a)}{n}\right|$. За визначенням, масив геш-кодів U є t-зв'язним, якщо для будь-якого набору із $s \le t$ стовпців і кожного вектора α, що визначаєгеш-код, частота появи в стовпцях значення α задовольняють умові:

$$\left|\frac{v_U(a)}{n} - \frac{1}{p^s}\right| \le \varepsilon. \qquad (4.12)$$

Імовірність $\Pr(RSh_{qi}(M) = RSh_{qi}(M'))$ відповідає випадку $s = 1$, і дозволяє отримати потрібну нерівність.

Для оцінки імовірності колізії при багатократному RSh_q потрібно довести таке твердження.

Для t-зв'язного $\varepsilon - U$ універсального класу при рівноімовірному виборі геш-функції імовірність колізії $\Pr[h(M_1) = h(M_2)] \le \varepsilon^t$, де $\varepsilon = \frac{k-1}{q}$.

Дійсно, геш-код $RSh_{q1}(M) \| RSh_{q2}(M) \| ... \| RSh_{qt}(M)$ представляє конкатенацію геш-результатів RSh_{qi}, i = 1, ..., t тоді для кожного з них існує найбільше εN функцій $h_i \in RSh_q$ таких, що $RSh_{qi}(M) = RSh_{qi}(M')$ при $M \ne M'$. Оскільки h_i вибираються незалежно і рівноймовірно із множини

RSh_{qi}, тоді для повного геш-коду $RSh_{q1}(M) \| RSh_{q2}(M) \| ... \| RSh_{qt}(M)$ існує найбільше $(\varepsilon N)^t$ геш-функцій таких, що $RSh_{q1}(M) \| RSh_{q2}(M) \| ... \| RSh_{qt}(M)$ і $RSh_{q1}(M') \| RSh_{q2}(M') \| ... \| RSh_{qt}(M')$. Звідси витікає потрібний результат. На рис 4.17 наведені графіки залежностей імовірності колізії для схеми гешування $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ при різних q, t.

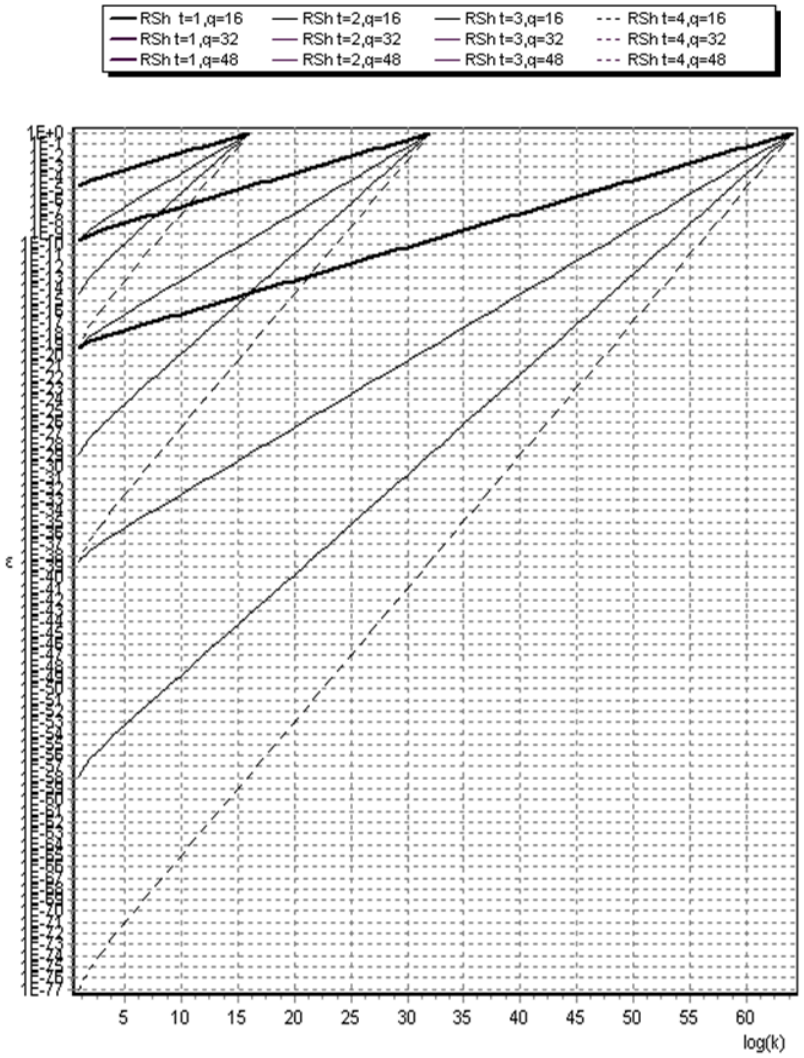


Рис. 4.17. **Графіки залежності імовірності колізії для багатократної конструкції $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ від об'єму гешованих даних при різному значенні поля обчислень**

Де імовірності колізії для схеми гешування $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ при різних q, t. Аналіз графіків показує, що багатократне гешування ефективно знижує імовірність колізії, а збільшення параметру поля обчислень збільшує розмір гешованих даних і фіксує на заданому рівні імовірність колізії. Застосування багатократного гешування приводить до

t-кратного збільшення складності обчислень $N_t = tN_1$, де N_1 – складність обчислень для універсального гешування. Для багатократного RSh_{qi} справедливе таке тривіальне твердження.

Нехай F_q – кінцеве поле, M – повідомлення. Складність обчислення геш-коду в конструкції з t-кратним гешуванням $(RSh_{q1},RSh_{q2},...,RSh_{qt})(M)$ визначається виразом:

$$N_1 = tk. \qquad (4.13)$$

У табл. 4.10 приведені зведені результати по імовірності колізії і складності обчислень для $(RSh_{q1},RSh_{q2},...,RSh_{qt})(M)$ гешування при оптимальному виборі параметрів схеми.

Визначення для багатократного гешування із використанням HCh_q схеми має такий вигляд.

Нехай $q, q = p^2$ – розширене кінцеве поле, M – повідомлення. Алгоритм обчислення геш-коду в конструкції із t-кратним HCh_q визначається виразом:

$$(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M) = HCh_{q1}(M) \| HCh_{q2}(M) \| ... \| HCh_{qt}(M), \qquad (4.14)$$

де $HCh_{qi}(M)$ – гешування на незалежно вибраному ключі.

Структурна схема $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ конструкції наведена на рис. 4.18.

Як і для багатократного RSh_q, геш-функція $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ є t-зв'язаною $\varepsilon - U$ універсальною. Доведення цього твердження очевидно і подібне доведенню для $(RSh_{q1},RSh_{q2},...,RSh_{qt})(M)$ схеми.

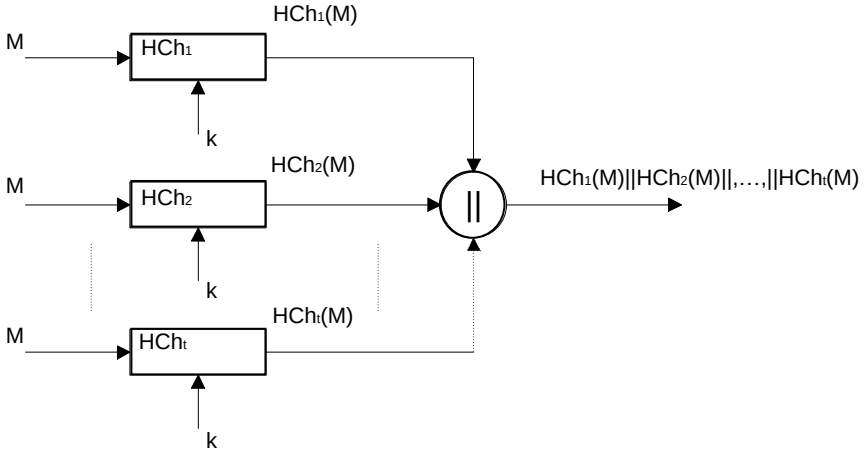


Рис. 4.18. **Структурна схема багатократного гешуванняз використанням** HCh_q

На рис. 4.19 представлені графіки залежностей імовірності колізії для схеми гешування $(HCh_{q1}, HCh_{q2}, \ldots, HCh_{qt})(M)$ при різних q, t.

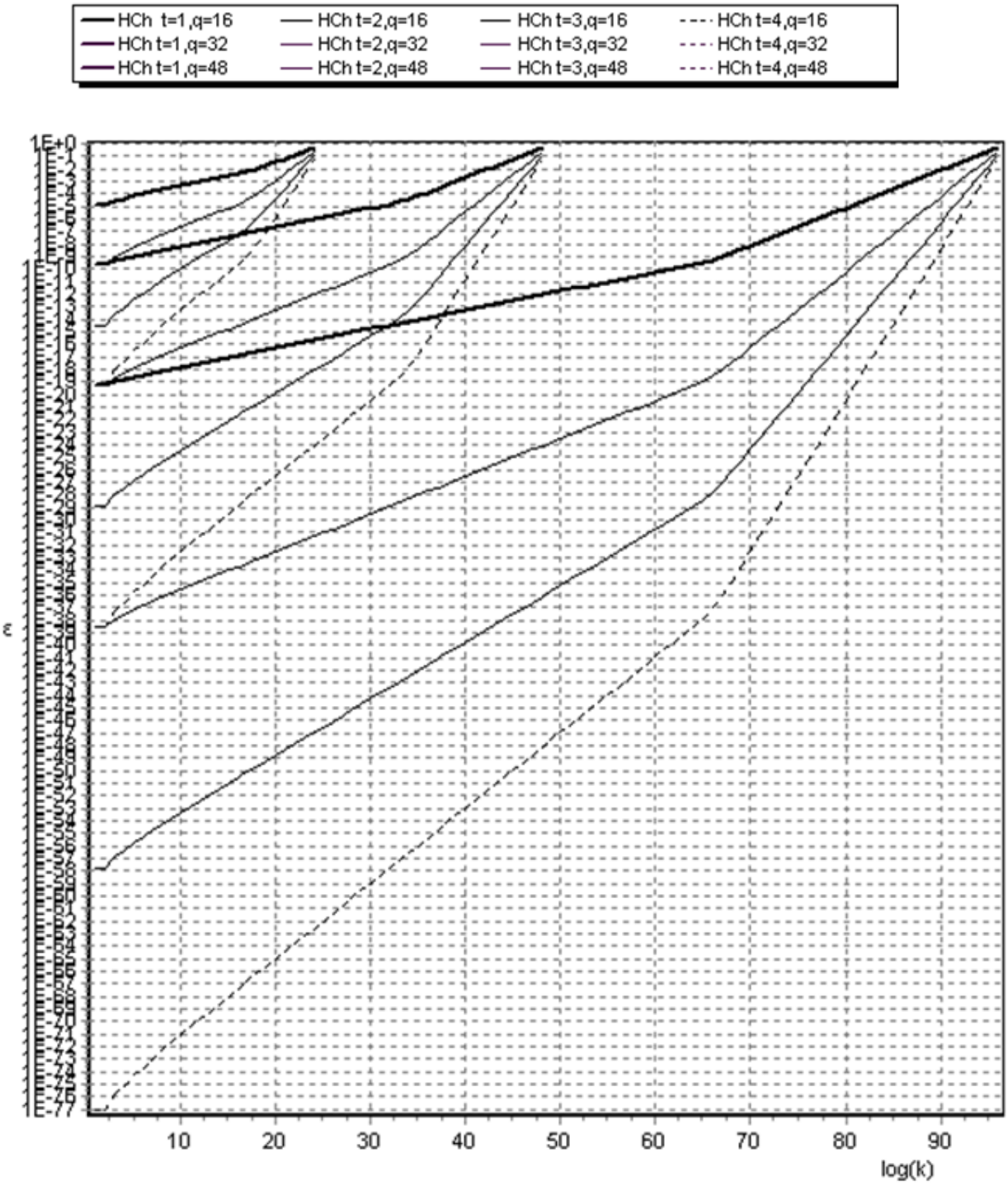


Рис. 4.19. **Графіки залежності ймовірності колізії і для багатократної каскадної конструкції $(HCh_{q1}, HCh_{q2}, \ldots, HCh_{qt})(M)$ від об'єму даних, що гешуються, при різному числі каскадів**

Де імовірності колізії для схеми гешування $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ при різних q, t розраховується імовірність колізії $Pr[h(M_1) = h(M_2)] \le \varepsilon^t$.

Також справедливим є і те, що для t – зв'язаного $\varepsilon - U$ універсального класу $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ при рівноймовірному виборі геш-функції, імовірність колізії $Pr_{h\in H}[h(M_1) = h(M_2)] \le \varepsilon^t$, де ε – це імовірність колізії при однократному HCh_q гешуванні.

Аналіз графіків показує, що $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ схема при обчисленнях у полі $q = 2^{32}$ підтримує імовірність колізії на рівні 10^{-19} (2^{-64}) для даних усіх практичних довжин. Для підтримки імовірності колізії $10^{-38}10^{-38}$ (2^{-128}), необхідно використовувати обчислення в полі розмірності $q = 2^{48}$. У табл. 4.10 наведені зведені імовірності колізії і оцінці складності обчислень для різних варіантів використання $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ схеми.

У графах таблиці, де оцінюється складність обчислень (кількість операцій на байт), показані відносні добавки, котрі визначають другий прохід обчислень за схемою Горнера, відповідно до прискореного алгоритму гешування із кодами Ерміта.

Із аналізу даних, наведених у табл. 4.10 витікає, що багатократне гешування із НС забезпечує значення ймовірність колізії, що вимагається, для даних у всьому практичному діапазоні довжин при полі обчислень $GF(2^{32})$. Деяке зростання складності обчислень з'являється при імовірності колізії 10^{-38} і довжині даних більше 2^{20} байт. У цьому випадку необхідно використовувати 6-кратне гешування.

Аналогічно багатократному гешуванню $(RSh_{q1}, RSh_{q2},..., RSh_{qt})(M)$ і $(HCh_{q1}, HCh_{q2},..., HCh_{qt})(M)$ багатократне гешування H_m для каскадної схеми $Ch_q(M)$ визначається таким чином.

Нехай F_q, $q = p^2$ – розширене кінцеве поле, M – повідомлення. Алгоритм обчислення геш-коду в конструкції із t-кратним каскадним гешуванням визначається виразом:

$$(Ch_{q1}, Ch_{q2},..., Ch_{qt})(M) = Ch_{q1}(M) \,||\, Ch_{q2}(M) \,||\, ... \,||\, Ch_{qt}(M), \qquad (4.15)$$

де $Ch_{qi}(M)$ – каскадне гешування з незалежно обраним ключем.

Таблиця 4.10

Складність обчислень для універсальних схем гешування

Розмір блока L (байт)	Ймовірність колізії	Параметри $(AGh_{q1}, AGh_{q2}, ..., AGh_{qt})(M)$ схеми											
		$RSh_q(M)$						$RSh_q(M)$					
		t			Складність обчислень (кількість операцій на байт)			t			Складність обчислень (кількість операцій на байт)		
2	3				2^{16}	2^{32}	2^{48}				2^{16}	2^{32}	2^{48}
$2^6 \div 2^7$	10^{-3}	1	1	1	0.5	0.25	0.167	1	1	1	0.625	$0.25+2^{-4}$	$0.167+2^{-4}$
	10^{-6}	2	1	1	1.0	0.25	0.167	2	1	1	1.25	$0.25+2^{-4}$	$0.167+2^{-4}$
	10^{-9}	3	2	1	1.5	0.5	0.167	3	1	1	1.875	$0.25+2^{-4}$	$0.167+2^{-4}$
	10^{-19}	6	3	2	3.0	0.75	0.33	5	2	2	3.125	0.625	$0,334+2^{-4}$
	10^{-38}	-	6	3		1.5	0.5	-	3	3		0.875	$0,5+2^{-4}$
2^{14}	10^{-3}	3	1	1		0.25	0.167	1	1	1	$0.5+2^{-7}$	$0.25+2^{-8}$	$0.167+2^{-8}$
	10^{-6}	-	1	1		0.25	0.167	2	1	1	$1.0+2^{-6}$	$0.25+2^{-8}$	$0.167+2^{-8}$
	10^{-9}		2	1		0.5	0.167	3	1	1	$1.5+2^{-6}$	$0.25+2^{-8}$	$0.167+2^{-8}$
	10^{-19}		4	2		1.0	0.33	7	3	2	$3.5+2^{-4}$	$0.75+2^{-7}$	$0,334+2^{-7}$
	10^{-38}		7	4		1.75	1,169		5	3		$1.25+2^{-6}$	$0,5+2^{-6}$
$2^{20} \div 2^{27}$	10^{-3}	-	2	1		0.5	0.167		1	1		$0.25+2^{-13}$	$0.167+2^{-12}$
	10^{-6}		3	1		0.75	0.167		1	1		$0.25+2^{-13}$	$0.167+2^{-12}$
	10^{-9}		4	2		1.0	0.33		2	1		$0.5+2^{-12}$	$0.167+2^{-12}$
	10^{-19}			3			0.5		3	2		$0.75+2^{-12}$	$0,334+2^{-11}$
	10^{-38}			6			1.0		6	4		$1.5+2^{-11}$	$0,668+2^{-10}$
$2^{30} \div 2^{37}$	10^{-3}			1			0.167		1	1		$0.25+2^{-16}$	$0.167+2^{-15}$
	10^{-6}			1			0.167		2	1		$0.5+2^{-15}$	$0.167+2^{-15}$
	10^{-9}			2			0.33		2	1		$0.5+2^{-15}$	$0.167+2^{-15}$
	10^{-19}			5			0,835		4	2		$1.0+2^{-14}$	$0,334+2^{-14}$
	10^{-38}									4			$0,668+2^{-13}$

Усі отримані раніше результати також справедливі для $(Ch_{q1}, Ch_{q2}, ..., Ch_{qt})(M)$. Так, клас геш-функцій $(Ch_{q1}, Ch_{q2}, ..., Ch_{qt})(M)$ є t-зв'язаним $\varepsilon - U$ універсальним. Доведення цього твердження очевидно і подібне до доведення теореми $(RSh_{q1}, RSh_{q2}, ..., RSh_{qt})(M)$ схеми.

При рівноймовірному виборі геш-функції імовірність колізії для t-зв'язного $\varepsilon - U$ універсального класу $(Ch_{q1}, Ch_{q2},..., Ch_{qt})(M)$ дорівнює $Pr_{h\in H}[h(M_1) = h(M_2)] \le \varepsilon^t$, де ε імовірність колізії при однократному каскадному гешуванні.

Основні результати з оцінки імовірності колізії при $(Ch_{q1}, Ch_{q2},..., Ch_{qt})(M)$ на довжинах більших ніж $2^6 - 2^7$ слів по 32 біти практично не відрізняється від $(Ch_{q1}, Ch_{q2},..., Ch_{qt})(M)$, як це витікає із результатів розрахунків, які зображені у вигляді графіку залежності імовірності колізії і для багатократної каскадної конструкції на рис. 4.19. Порівняльний аналіз даних, наведених у табл. 4.10 показує, що HCh_q програє за обчислювальною складністю RSh_q.

Застосування каскадної схеми гешування на основі АГ кодів для побудови універсальних класів геш-функцій забезпечує якнайкращі співвідношення між розміром геш-значень, довжини даних і ймовірністю колізії при обчисленнях у кінцевих полях, а також мінімізує витрати за ключовими даними і витратами на обчислення.

Застосування каскадної схеми-гешування на основі АГ кодів є універсальним класом геш-функцій, що забезпечує найкращі співвідношення між розміром геш-значень, довжин даних, і ймовірністю колізії при обчисленнях у кінцевих полях, а також мінімізує витрати за ключовими даними і витратами на обчислення. Запропонована Ch_q конструкція на основі RS і HC об'єднує переваги RSh_q і HCh_q схем. На малих довжинах даних виконується тільки RSh_q гешування із якнайменшою складністю, що є добре пристосованим для даних малої довжини. При великих довжинах даних здійснюється перехід до HCh_q гешування, практично з тією ж відносною швидкістю гешування, що і для RSh_q схеми без підвищення ймовірності колізії.

Технологія багатократного гешування на різних ключах ефективно знижує імовірність колізії, а каскадна схема Ch_q, що використовується при цьому, збільшує розмір гешованих даних і фіксує на заданому рівні імовірність колізії.

Багатократне гешування RSh_q і HCh_q забезпечує необхідні значення імовірності колізії для даних у всьому практичному діапазоні довжин при полі обчислень $GF(2^{32})$. Деяке зростання складності обчис-

лень з'являється при ймовірності колізії 10^{-38} і довжині даних більше 2^{30} байт. У цьому випадку необхідно використовувати 6-ти кратне гешування. Порівняння HCh_q гешування зі схемою на основі використання RS показує, що HCh_q програє по складності на малих довжинах у 2 – 3 рази.

Схеми RSh_q, HCh_q і Ch_q мають перевагу в швидкості обчислень. Порівняно з надшвидкісним алгоритмом MD4, виграш складає майже 2 рази, а порівняно з практичним MD5 – майже у 3 рази. Разом з тим схеми RSh_q, HCh_q і Ch_q є більш гнучкими в практичних реалізаціях, оскільки за рахунок багатократного каскадного гешування забезпечують істотно більш широкий діапазон застосувань по довжинах геш-кодів, імовірності колізії, обсяг гешованих даних і доказової секретності.

Модель каскадного формування MAC із використанням модулярних перетворень. В основі моделі лежить багатошарова схема універсального гешування з використанням на останньому етапі, модулярних перетворень.

Властивості багатошарової (композиційної) конструкції найкраще пояснити за допомогою мови відображень [64; 68]. Нехай X, Y, U є множинами з n, m, u елементів, $n < m < u$. H_1 є множиною функцій f_1, що здійснює відображення $X \to U$, а H_2 – множна функцій f_2 здійснюючих відображення $U \to Y$. Тоді $H = H_2 \times H_1$ є множиною функцій f, що є композицією $f = f_1 \times f_2$.

Характеристики багатошарової конструкції наведені результатом такої теореми [86; 87].

Теорема 4.27.

Композиція з універсального класу геш-функцій $\varepsilon_1 - U(N_1, n, u)$ і строго універсального класу геш-функцій $\varepsilon_2 - SU(N_2, u, m)$ є строго універсальним класом з параметрами $\varepsilon - SU(N_1 N_2, n, m)$, де $\varepsilon = \varepsilon_1 + \varepsilon_2 - \varepsilon_1 \varepsilon_2$.

Таким чином, використовуючи композицію алгоритмів формування кодів автентифікації, еквівалентних алгоритмам обчислення універсальних і строго універсальних класів геш-функцій одержується багатошарову схему формування MAC. Властивості сформованого таким спосо-

бом коду контролю цілісності й автентичності даних, будуть задовольняти властивостям строго універсального класу геш-функцій.

У методі формування кодів контролю цілісності й автентичності даних перші шари перетворення пропонується реалізувати традиційними для алгоритму UMAC високошвидкісними але криптографічно слабкими схемами універсального гешування, останній шар пропонується реалізувати з використанням розробленої безпечної (криптографічно сильної) схеми строго універсального гешування на основі модулярних перетворень. Формально пропонована схема каскадного формування кодів контроля цілісності та автентичності даних наведена на рис. 4.24.

Основна частина інформаційних даних, обробляється першими шарами універсального гешування. Сформований у результаті такого перетворення геш-код на останньому, заключному етапі, обробляється криптографічно сильною функцією строго універсального гешування на основі модулярних перетворень.

Таким чином, в основі схеми формування МАС із використанням модулярних перетворень лежить використання: на перших шарах – високошвидкісних методів універсального гешування (Nh-гешування, поліноміальне гешування, гешування Картера – Вергмана); на останньому шарі – безпечного строго універсального гешування на основі модулярних перетворень (з використанням циклових функцій).

Пропонований розв’язок дозволяє забезпечити високі колізійні властивості сформованих кодів контролю цілісності й автентичності даних, зберегти властивості універсального гешування й забезпечити високі показники безпеки на рівні сучасних засобів криптографічного захисту доказової стійкості. Застосування багатошарової конструкції дозволяє також суттєво знизити обчислювальні витрати на формування кодів контролю цілісності й автентичності більших масивів даних.

У табл. 4.11 наведено порівняння обчислювальної складності деяких функцій гешування. Дані по швидкодії для пропонованої схеми МАС із модулярними перетвореннями наведені для мінімального рівня стійкості (потужність множини ключових даних блокового симетричного шифру дорівнює 280) і достатнього рівня стійкості (для модулярних перетворень еквівалентна довжина ключа блокового симетричного шифру дорівнює 128 бітам). Довжина сформованого при цьому МАС дорівнює 80 і 128 бітам відповідно.

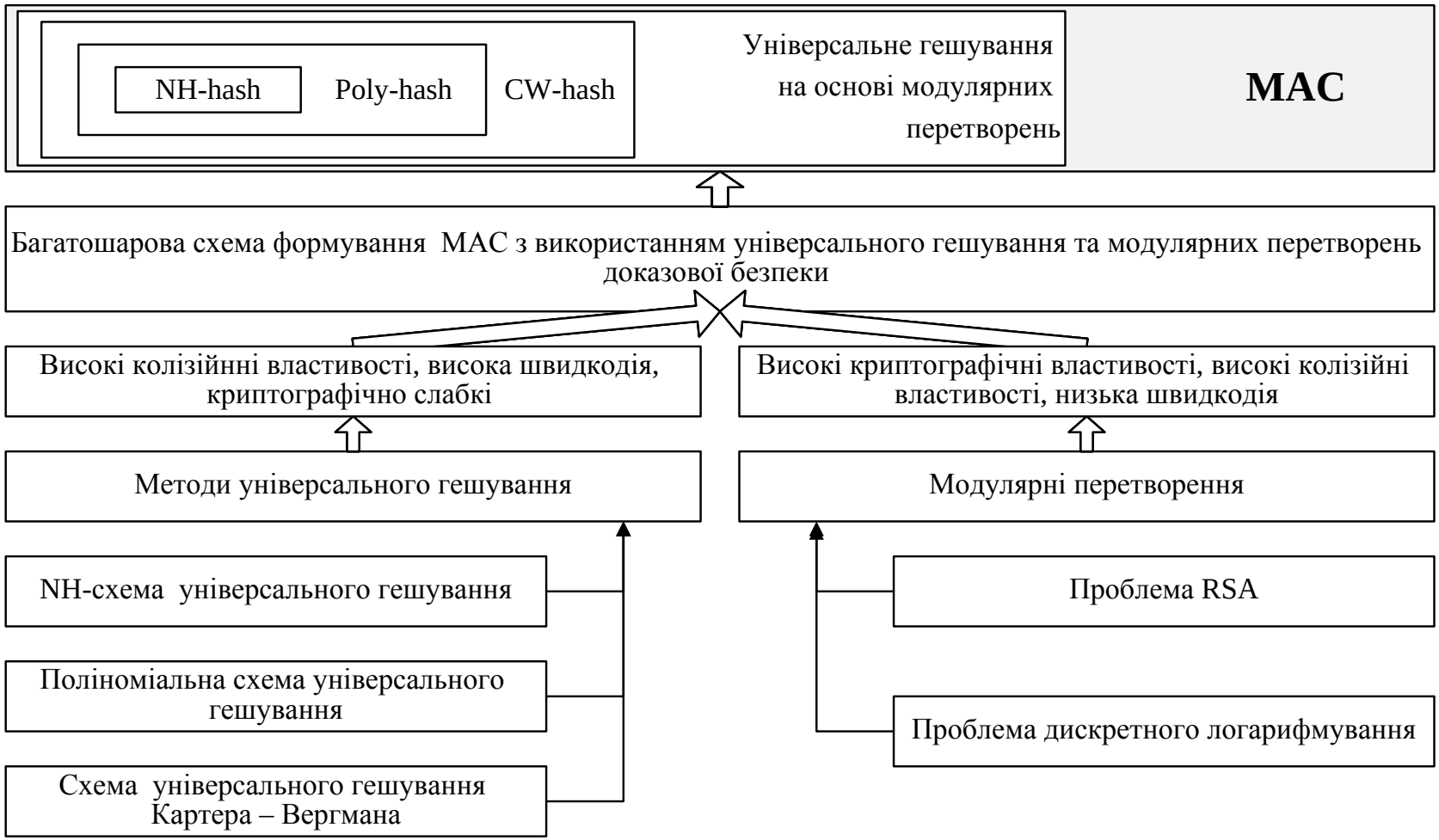


Рис. 4.24. **Схема каскадного формування кодів контроля цілісності та автентичності даних з використанням модулярних перетворень**

Таблиця 4.11

Оцінка складності формування MAC різними схемами

Алгоритм	Довжина вхідних даних, байт					
	2048	4096	8192	16384	32768	65536
HMAC-MD5 (128 біт)	9	9	9	9	9	9
HMAC-RIPE-MD (160 біт)	27	27	27	27	27	27
HMAC-SHA-1 (160 біт)	25	25	25	25	25	25
HMAC-SHA-2 (512біт)	84	84	84	84	84	84
CBC Mac-rijndael (128 біт)	26	26	26	26	26	26
CBC MAC-DES (64 біта)	62	62	62	62	62	62
Пропонована схема MAC із модулярними перетвореннями (80 біт)	38	22	14	10	8	7
Пропонована схема MAC із модулярними перетвореннями (128 біт)	294	150	78	42	24	15

Для всіх функцій, наведених у табл. 4.11 (крім запропонованих, з використанням модулярних перетворень) питома складність формування кодів контролю цілісності й автентичності даних не залежить від обсягу оброблюваних даних.

Для моделі з використанням модулярних перетворень питома складність із зростанням довжини оброблюваних даних знижується. Так для високого рівня стійкості (еквівалентна довжина ключа блокового симетричного шифру дорівнює 128 бітам) вже для блоків даних з 32768 байт порівняно з відомими й застосовуваними в протоколах мережної безпеки алгоритмами формування MAC. Для мінімального рівня стійкості (потужність множини ключових даних блокового симетричного шифру дорівнює 2^{80}) пропонована схема каскадного формування кодів контролю, цілісність й автентичність даних з використанням модулярних перетворень уже для пакетів даних з 2048 байт та практично не поступається по швидкодії застосовуваним на сьогоднішній день алгоритмам формування MAC у протоколах мережної безпеки, у тому числі в протоколах IPSEC.

Таким чином, отримані результати досліджень показують, що розроблена схема формування кодів контролю цілісності й автентичності даних з використанням модулярних перетворень дозволяє забезпечити високі колізійні властивості безпечного гешування. Крім того, за рахунок багатошарової конструкції обчислення геш-коду вдається суттєво скоротити обчислювальну складність гешування й підвищити, таким чином, швидкість обробки інформаційних повідомлень.

Розділ 5. Результати міжнародних криптографічних конкурсів в галузі гешування

Розглянуто алгоритми-переможці міжнародних крипто-графічних конкурсів NESSIE і SHA-3 в галузі гешування інформації. Проаналізовано властивості алгоритмів відповідно до вимог, які висуваються до сучасних алгоритмів формування геш-коду.

5.1. Результати конкурсу NESSIE в галузі гешування інформації

В Європі в рамках проекту NESSIE (New European Schemes for Signature, Integrity, and Encryption) проходив конкурс на розробку європейських криптографічних стандартів. Як відомо, у 2001 р. у США за результатами конкурсу, організованого Національним інститутом стандартизації та технологій США, був прийнятий новий стандарт блочного шифрування AES-Rijndael. Даний алгоритм був визнаний після дворічних досліджень і відкритого обговорення в рамках проекту AES (Advanced Encryption Standard) в якості кращого і в даний час прийнятий в якості федерального стандарту США (FIPS 197) [124].

Проект AES як концептуальний підхід до розробки якісних криптографічних примітивів показав досить високу ефективність у широкому сенсі. Саме тому європейське співтовариство пішло по цьому шляху для розробки нових європейських стандартів [103].

Одними з криптографічних примітивів, що висувалися на конкурс NESSIE були алгоритми формування MAC-кодів та геш-функцій.

Головні критерії відбору які використовувались на конкурсі:

безпека – найважливіший критерій, тому що безпека алгоритмів шифрування – головне призначення цих алгоритмів. Процес оцінки безпеки враховував і вплив подій поза проектом NESSIE (таких як нові напади або методи аналізу);

ринкові вимоги пов’язані з потребою в алгоритмі, його зручності і простоті використання, можливості міжнародного використання;

продуктивність алгоритму шифрування на певному обладнанні. Для програмного забезпечення розглядали 8-бітові процесори (як у недорогих платіжних картках з вбудованим мікропроцесором), 32-бітові процесори (наприклад, старе сімейство Pentium), сучасні 64-бітові процесори;

Гнучкість алгоритму.

Зараз в Україні інтенсивно впроваджуються різні системи електронного документообігу. Обов'язковим їх реквізитом є наявність ЕЦП, при виробленні якого в свою чергу використовується функція гешування. З цією метою використовується міждержавний стандарт ГОСТ 34.311-95 [21]. Розглядаються питання гармонізації та використання функцій гешування, що визначені в ISO/IEC 10118 [107], перш за все, SHA-2. Але фундаментальним вирішенням проблеми появи в Україні ефективного алгоритму або алгоритмів гешування є врахування як теоретичних, так і практичних результатів конкурсу NIST SHA-3 Competition.

На даний момент існує велика кількість криптографічних сервісів, які потребують різних ступенів захисту і відповідно різних довжин геш-значень, що передбачено, наприклад, у стандарті SHA-2 [104]. Крім того, важливо мати можливість реалізації геш-функції на різних апаратних платформах, у тому числі з обмеженим об'ємом пам'яті та низькою розрядністю процесора (наприклад, смарт-картка). На рівні стандарту важливо, щоб один алгоритм поєднував у собі всі перераховані функціональні можливості. Наявність кількох стандартизованих алгоритмів ускладнює розробку інформаційно-телекомунікаційних систем і приводить до конфліктів між системами різних розробників.

Використовуючи досвід проведених та перспективних міжнародних конкурсів на нові криптографічні стандарти (NIST SHA-3 Competition, NESSIE) необхідно визначити мінімальний перелік вимог, якому має відповідати геш-функція. Перелік вимог умовно поділяється на дві частини: вимоги до стійкості та вимоги до функціональності алгоритму. Вимоги до стійкості мають відповідати простому правилу – не повинно існувати жодної атаки на алгоритм складністю менше ніж атака "груба сила". Наступні критерії є ключовими для забезпечення захищеності, про що свідчать матеріали міжнародних конкурсів на нові криптографічні стандарти [104; 124]:

складність знаходження колізії $2^{n/2}$;

складність відновлення прообразу 2^n;

складність знаходження другого прообразу не менше 2^n;

стійкість до атак length extension;

стійкість до усічених колізій;

стійкість до атак мультиколлізій;

відсутність атак розпізнавання для генераторів псевдовипадкових послідовностей, що використовують HMAC, побудованих на базі геш-функції зі складністю, меншою ніж знаходження другого прообразу і кількістю запитів до генератора не менше $2^{n/2}$;

надійність математичної бази.

Вимоги до функціональності алгоритму гешування формуються зважаючи на ймовірну галузь застосування. Перш за все необхідно забезпечити сумісність нового стандарту з вже діючими на території України стандартами ЕЦП, а також передбачити можливість суміщення з рядом міжнародних стандартів та перспективних стандартів, що можуть бути прийняті в межах України.

Функціональність нового стандарту гешування має відповідати такому переліку вимог:

довжина виробленого геш-значення;

максимальна швидкодія;

максимальна кількість процесорів, що може бути ефективно використана для паралельних обчислень;

мінімальні вимоги до обчислювальних ресурсів;

можливість реалізації алгоритму на різноманітних програмних, програмно-апаратних та апаратних платформах;

простота архітектури алгоритму.

Новий національний стандарт, який має бути розроблений, планується використовувати у широкому колі криптографічних додатків, які висувають до нього ряд специфічних вимог технічного характеру. Галузь застосування нового алгоритму охоплює:

використання виробленого геш-значення у якості інструмента контролю цілісності інформації при передачі, зберіганні та розповсюдженні. Найбільш відомими прикладами є протоколи встановлення та розповсюдження ключів, механізми надання послуги неспростовності, асиметричні шифри, системи електронного цифрового підпису з додатком або відновленням повідомлення та ін.;

вироблення кодів автентифікації повідомлень за технологією HMAC;

використання геш-функції у якості генератора псевдовипадкових послідовностей.

Наведений перелік є нормальною міжнародною практикою використання геш-функцій як криптографічних алгоритмів для вирішення задач криптографічного захисту інформації.

У фінальному звіті міжнародного криптографічного конкурсу NESSIE відзначена функція гешування Whirlpool (на основі власного вбудованого блочно-симетричного шифру) і відібрані чотири кращі алгоритми формування MAC-кодів: UMAC, Two-Track-MAC, CBC-MAC, HMAC [124].

Серед основних недоліків алгоритмів є погане розсіювання та використання для генерації ключа алгоритму DES (AES). Щодо використання алгоритмів DES та AES для отримання стійкої ключової послідовності, то це накладає обмеження на швидкість гешування самого алгоритму MAC-коду [26; 79].

У табл. 5.1. наведено аналіз тестування швидкості роботи алгоритмів гешування.

Таблиця 5.1

Аналіз тестування швидкості роботи алгоритмів гешування

Функція гешування	Кількість циклів	Мова реалізації	Швидкість роботи на Celeron 600 MHz	Швидкість роботи на Pentium III 1000 MHz
Whirlpool	10	C	28,013 Мбіт/с	46,961 Мбіт/с
SHA-2 (512)	80	C	41,159 Мбіт/с	68,701 Мбіт/с
SHA-2 (256)	64	C	81,308 Мбіт/с	135,557 Мбіт/с
ГОСТ 34311-95	-	C+Assembler	49,408 Мбіт/с	83,056 Мбіт/с
HAVAL	96(128, 160)	C	337,842 Мбіт/с	564,809 Мбіт/с
SHA-1	80	C, Assembler	206,285 Мбіт/с 361,581 Мбіт/с	344,433 Мбіт/с 605,558 Мбіт/с
RIPEMD-160	160	C	147,465 Мбіт/с	246,568 Мбіт/с
MD5	64	C	278,715 Мбіт/с	574,635 Мбіт/с
MD4	48	C	344,086 Мбіт/с	467,793 Мбіт/с
UMAC	-	C C+Assembler	989,371 Мбіт/с 3518,900 Мбіт/с	1648,953 Мбіт/с 5885,057 Мбіт/с
Rijndael CBC–MAC	14	C	139,376 Мбіт/с	231,255 Мбіт/с
ГОСТ 28147-89 (OFB)	16	C+Assembler	189,559 Мбіт/с	315,270 Мбіт/с

Як видно з аналізу табл. 5.1 Two-Track-MAC поступається за швидкістю тільки UMAC алгоритму, приблизно в 3 – 7 разів, але має в 2,5 рази більший розмір MAC коду.

В табл. 5.2 наведено можливі значення довжин ключа і геш-коду для різних алгоритмів сімейства MAC.

Таблиця 5.2.

Довжина ключа k і довжина геш-коду n для MAC-кодів

Алгоритм	k (ключ)	n (геш-код)
UMAC	128	64
TTMAC	160	≤160
EMAC-AES	128, 192, 256	≤128
RMAC-AES	128, 192, 256	≤128
HMAC-SHA-1	≤512	≤160

Алгоритм EMAC заснований на AES CBC-MAC вважається стандартом для вхідних у цю категорію шифрів. Схема використовує AES (Rijndael) блоковий симетричний шифр і генерує величину MAC до 128 біт. RMAC становить випадковий варіант схеми EMAC, що забезпечує кращу протидію атаці на основі внутрішніх помилок. Відмінність – у вихідному перетворенні (шифрування виробляється на ключі, отриманому порозрядним доповненням K_2 і R. Two-Track-MAC заснований на геш-функції RIPEMD-160 з модифікаціями. Алгоритм працює, шляхом ітераційного використання функції стиску [83; 84].

У табл. 5.3 наведено основні результати оцінки швидкодії Two-Track-MAC алгоритму для різних операційних платформ. Швидкість обчислень визначається кількістю циклів процесора, затрачуваних на один байт оброблюваного повідомлення.

Аналіз табл. 5.3 показує, що схеми ключового гешування UMAC на основі поліноміальних функцій дозволяють одержати саму високу швидкість гешування, крім цього вони є переможцем Міжнародного європейського криптографічного конкурсу NESSIE.

Таблиця 5.3

Швидкодія MAC алгоритмів

Алгоритм	Довжина MAC-коду (біт)	Довжина ключа (біт)	Тип ПЕВМ				
			Pentium2	PIII/Linux	Pentium4	Xeon	AMD
TTMAC	160	160	21	21	40	37	21
UMAC-16	64	128	6.1	6.0	6.2	6.1	6.2
UMAC-32	64	128	2.5	2.9	6.7	6.6	1.9
HMAC-Whirlpool	512	512	86	72	98	103	100
HMAC-MD4	128	512	4.7	4.7	6.4	6.4	4.7
HMAC-MD5	128	512	7.2	7.3	9.4	9.4	7.4
HMAC-RIPE-MD	160	512	23	18	27	26	21
HMAC-SHA-0	160	512	16	15	23	23	13
HMAC-SHA-1	160	512	16	15	25	24	12
HMAC-SHA-2	256 384 512	512	40 84 84	39 84 84	40 124 124	39 132 132	33 72 72
HMAC-Tiger	192	512	24	21	28	26	20
CBC MAC-Rijndael (EMAC)	128	128	24	26	26	27	31
CBC MAC-DES	64	56	62	61	72	69	54
CBC MAC-Shacal	512	160	31	31	67	74	29

Проаналізувавши характеристики MDC-кодів і MAC-кодів можна зробити висновок, що MAC-коди є більш стійкими і на відміну від MDC-кодів не потребують додаткових алгоритмів для шифрування повідомлення. Тому більш перспективним є дослідження ключових геш-функцій.

5.1.1 Колізійно-стійкі функції гешування Whirpool та SHA-256, SHA-384, SHA-512

Whirlpool

Whirlpool – це стійка до збігів (колізій) геш-функція, представлена в NESSIE Пауло Баррето й Вінсентом Райменом. Конструкція Whirlpool базується на основі блокового шифру, який використовується в так званому "Міагучі-Прінель" – режиму, що гешує. Цей блоковий шифр подібний за структурою із шифром AES, але працює з більшими блоками 512 біт [103].

Короткий опис примітива.

Whirlpool відображає повідомлення M, що складається з довільної кількості біт, в 512-бітне геш-значення Whirlpool(M). Алгоритм базується на функції компресії, яка використовується ітеративно на блоках повідомлення у 512 біт. Ця функція базується на спеціально призначеному 512-бітному блоковому шифрі, що лежить в основі та використовує 512-бітний ключ.

Спочатку повідомлення розширюється до необхідної довжини й ділиться на блоки повідомлення $M_0 \ldots M_{t-1}$, де кожний блок $M_i (0 \le i \le t-1)$ складається з 512 біт. Початкове значення визначене як рядок, що складається з 512 біт-нулів: $M_0 = 0$.

Компресійна функція Whirlpool перетворює вхідне значення H_i і блок повідомлення Mi у вихідне значення H_{i+1}. Ця функція залежить від зашифрування із внутрішнім блоковим шифром W, де M_i формує відкритий текст, і H_i служить як ключ. Далі, отриманий шифртекст складається по модулю 2 з M_i і H_i (усі ці рядки по 512 біт). Тобто, $H_{i+1} = W_{H_i}(M_i) \oplus M_i \oplus H_i$. У такому використанні компресійна функція H_{i+1} формує вхід спільно з таким блоком повідомлення M_{i+1}. Результат цього – в такому описі алгоритму Whirlpool.

Розширення повідомлення M и розподіл його на блоки повідомлення $M_0 \ldots M_{t-1}$

Визначення початкового значення $M_0 = 0$

використання функції компресії для обчислення геш-значення.

Встановлення геш-значення Whirlpool (M) = H_t.

Надійність і продуктивність.

Надійність Whirlpool може бути доведена, базуючись на припущенні, що певні ідеальні (бажані) властивості справедливі для блокового шифру W, що лежить в основі. Цей блоковий шифр дуже схожий з AES. Не існує відомих більш швидких, ніж перебір, атак на Whirlpool, і алгоритм має високий рівень надійності завдяки довгому геш-значенню (512 біт).

Продуктивність Whirlpool залежить від продуктивності лежачого в основі блокового шифру, який досить ефективний. Однак він потребує перерахування процедури формування ключів для цього шифру для кожного використання функції компресії – це для кожного блоку повідомлення з 512 біт. Структура алгоритму не орієнтована на якусь певну платформу, але можуть бути зроблені різні оптимізації для різних платформ. Спеціальна структура компонентів, особливо S-блоку, що використовується, дозволяє виконувати ефективні апаратні реалізації.

SHA-256, SHA-384 and SHA-512

У 1993 р. НІСТ видав FIPS-180 [463], який представив стандартну геш-функцію, названу SHA. У FIPS-180-1 [465] від 1995 р. SHA був поліпшений (додана ротація ліворуч на 1 бітову позицію) і поліпшена версія була названа SHA-1. Обидві версії мають розмір дайджесту 160 біт. Тільки в роботі Саарінєна була наведена слайд-атака на SHA-1.

Оскільки індустрія захисту інформації вимагала більших запасів надійності, у серпні 2002 р. НІСТ опублікував ще три геш-функції: SHA-256, SHA-384 і SHA-512. FIPS-180-2 [123] визначає ці три нові геш-функції разом з SHA-1 як стандартні геш-функції для використання урядом США. Три нові функції мають більший дайджест-розмір – 256 біт для SHA-256, 384 біт для SHA-384 і 512 біт для SHA-512.

Опис примітива.

SHA-256 базується на діленні повідомлення на блоки по 512 біт кожний. Кожний блок потім уводиться в компресійну функцію, яка розширює 512-бітний блок до 2048 біт, які використовуються для перетворення поточного значення стану в 256 біт у нове значення стану. Початковий 256-бітний стан завантажується у вісім 32-бітних слів. Потім протягом 64 раундів ці слова впливають один на одного, використовуючи розширений блок повідомлення й різні функції. Ці 64 раунди можуть також бути використані в режимі блокового шифру (див. Шакал-2, розд. 4 у частині B (блокові шифри) NESSIE портфоліо).

SHA-384 і SHA-512 базуються на діленні повідомлення на блоки по 1024 біт кожний. Кожний блок вводиться у функцію компресії й використовується для зміни стану в 512 біт протягом 80 раундів функції компресії.

І SHA-384, і SHA-512 мають подібну структуру до використаної в SHA-256, але розмір слова подвоєний і функції трохи відрізняються. Єдиною відмінністю між SHA-384 і SHA-512 є початковий стан і той факт, що в SHA-384 вихід – це стан, усічене до 384 біт.

Стійкість і продуктивність.

Не оприлюднено ніяких атак проти геш-функцій SHA-256, SHA-384 і SHA-512.

Дані з продуктивності для цих трьох геш-функцій показують, що вони дуже ефективні. Програмна реалізація для NESSIE досяглася швидкостей у 30 – 70 циклів (тактів) на байт для SHA-256, і 16 – 190 циклів (тактів) для SHA-384 і SHA-512. Простота операцій, що використовуються у всіх трьох представлених алгоритмах, вселяє впевненість, що апаратна реалізація буде мати невелику кількість вентилів. Більше того, SHA-256 може бути використана разом з алгоритмом зашифрування Шакал-2 для збереження (зменшення кількості) вентилів або розміру реалізації.

SHA-384 і SHA-512 підігнані для оточення, де існують 64-бітні регістри (або більше), і вони оперують із 64-бітними словами.

Опис.

Усі три геш-функції мають ту саму базову конструкцію. Кожна функція додає біти до повідомлення, щоб його довжина була кратною розміру блоку, з якими працює компресійна функція (1024 біт для SHA-384 і SHA-512 і 512 біт для SHA-256) [123].

Нехай l буде довжина повідомлення M, яке потрібно гешувати. Слід додати 1, за якою іде k нульових біт, де k – це найменше ненегативне значення, для якого $l + 1 + k \equiv 448 \bmod 512$ для SHA-256 і $l + 1 + k \equiv 896 \bmod 1024$ для SHA-384 і SHA-512. Для досягнення кратності довжини блоку, з яким працює геш-функція, потім додається довжина повідомлення ($\bmod 2^{64}$ для SHA-256 або $\bmod 2^{128}$ для SHA-384 і SHA-512).

Як тільки цей крок виконаний, ділиться повідомлення M на блоки однакового розміру (1024 біт для SHA-384 і SHA-512 і 512 біт для SHA-256). Позначається ця декомпозиція як $M = m_1 m_2 \ldots m_n$.

Тепер той самий процес виконується для якої-небудь із трьох геш-функцій (константи й точний опис функцій можуть відрізнятися):

Початкове значення восьми слів завантажується у 8 слів: A, B, C, D, E, F,G і H.

Для i від 1 до n виконати:

-Set AA = A, BB = B, CC = C, DD = D, EE = E, FF = F, GG = G and HH = H.

-for t = 0 to 63 (79 for SHA-512 and SHA-384) do:

$$T_1 = H + \sum_1(E) + C_h(E,F,G) + K_t + W_t$$

$$T_2 = \sum_0(A) + M_{aj}(A,B,C).$$

H = G

G = F

F = E

$E = D + T_1$

D = C

C = B

B = A

$A = T_1 + T_2$

-set A = A+AA, B = B+BB, C = C+CC, D = D+DD, E = E+EE, F = F+FF, G = G+GG, and H = H+HH.

Продукувати геш-значення A|B|C|D|E|F|G|H для SHA-256 і SHA-512, і A|B|C|D|E|F для SHA-384.

Додавання виконано по модулю 2^{32} для SHA-256 і 2^{64} для SHA-384 і SHA-512.

K_t – це раундові константи, що додаються на кожному раунді компресійної функції. W_t – це множини 64 або 80 слів, зроблених із блоку повідомлення, який обробляється. $C_h(x,y,z)$ і $M_{aj}(x,y,z)$ – це нелінійні функції, використовувані в компресійній функції. $\sum_0$ і $\sum_1$ – це дві лінійні функції, які різні для SHA-256 і для SHA-384 і SHA-512.

Кожний блок M_i ділиться на 16 слів $m_i^0, m_i^1, \ldots m_i^{15}$, потім розраховуються W_t: $W_t = \begin{cases} m_i^t, 0 \le t \le 15 \\ s_1(W_{t-2}) + (W_{t-7}) + s_0(W_{t-15}), 16 \le t \le 63 (or 79) \end{cases}$.

Функції s_0 і s_1 є лінійні функції, але вони різні для SHA-256 і для SHA-384 і SHA-512.

Функції, використані в SHA-256.

SHA-256 використовує 6 функцій: $C_h(x,y,z)$ і $M_{aj}(x,y,z)$ – нелінійні функції з 96-бітним входом і 32-бітним виходом, і 4 лінійні функції з 32-бітним входом і 32-бітним виходом: $\sum_0, \sum_1, s_0, s_1$.

Спочатку визначаються 4 лінійні операції:

$$\sum_0(X)=X>>>_2 \oplus X>>>_{13} \oplus X>>>_{22}$$
$$\sum_1(X)=X>>>_6 \oplus X>>>_{11} \oplus X>>>_{25}$$
$$\sigma_0(X)=X>>>_7 \oplus X>>>_{18} \oplus X>>>_3$$
$$\sigma_1(X)=X>>>_{17} \oplus X>>>_{19} \oplus X>>>_{10}$$

Операція C_h приймає три 32-бітних слова X, Y, Z. Операція є функцією побітового вибору, тобто якщо і-й біт X установлений, то і-й біт виходу є і-й біт Y. З іншого боку, і-й біт виходу є і-й біт Z. Ця операція може бути легко реалізована, як: $C_h(X,Y,Z)=(X \wedge Y) \oplus (\overline{X} \wedge Z)$:

$$C_h(X,Y,Z)=(X \wedge Y) \oplus (\overline{X} \wedge Z).$$

Операція M_{aj} також приймає три 32-бітних слова X, Y, Z. Операція є функцією побітової мажоритарності, тобто якщо більшість і-х бітів X, Y і Z установлені, тоді і-й біт виходу встановлений, і навпаки. Ця операція може бути легко реалізована, як: $M_{aj}(X,Y,Z)=(X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$.

Функції, використані в SHA-384 і SHA-512.

SHA-384 і SHA-512 використовують 6 функцій $C_h(x,y,z)$ і $M_{aj}(x,y,z)$ нелінійні функції з 192-бітним входом і 64-бітним виходом, і 4 лінійні функції з 64-бітним входом і 64-бітним виходом: $\sum_0, \sum_1, s_0, s_1$.

Спочатку визначаються 4 лінійні операції:

$$\sum_0(X)=X>>>_{28} \oplus X>>>_{34} \oplus X>>>_{39}$$
$$\sum_1(X)=X>>>_{14} \oplus X>>>_{18} \oplus X>>>_{41}$$
$$\sigma_0(X)=X>>>_1 \oplus X>>>_8 \oplus X>>>_7$$
$$\sigma_1(X)=X>>>_{19} \oplus X>>>_{19} \oplus X>>>_6$$

Операція C_h приймає три 64-бітних слова X, Y, Z. Операція є функцією побітового вибору, тобто якщо і-й біт X установлений, те і-й біт виходу є і-й біт Y. З іншого боку, і-й біт виходу є і-й біт Z. Ця операція може бути легко реалізована, як: $C_h(X,Y,Z)=(X \wedge Y) \oplus (\overline{X} \wedge Z)$.

Операція M_{aj} також приймає три 64-бітних слова X, Y, Z. Операція є функцією побітової мажоритарності, тобто якщо більшість і-х біт X, Y і Z установлені, тоді і-й біт виходу встановлений, і навпаки. Ця операція може бути легко реалізована, як: $M_{aj}(X,Y,Z)=(X \wedge Y) \oplus (X \wedge Z) \oplus (Y \wedge Z)$.

5.1.2. Коди автентифікації повідомлень UMAC, TTMAC, EMAC, HMAC

Основними задачами проекту NESSIE є відбір кращих десяти криптографічних примітивів. У цей набір входять алгоритми блокового і потокового шифрування, генератори випадкових чисел, схеми швидкої автентифікації даних (MAC), геш-функції і алгоритми цифрового підпису. В якості основних критеріїв відбору претендентів обрані реальна безпека, продуктивність, гнучкість і вимоги ринку.

Перевагою MAC є те, що він дозволяє одночасно отримати і перевірити інформацію за допомогою того ж секретного ключа. Це означає, що відправник і одержувач повідомлення повинні домовитися про ключ до початку повідомлення, як у випадку з симетричним шифруванням.

Код автентифікації повідомлення (MAC) є коротким фрагментом інформації, який використовується для перевірки достовірності повідомлення. MAC (Message Authentication Code) – код перевірки повідомлення, який використовує функцію відображення і надає дані у вигляді значень фіксованого розміру, а потім – гешує саме повідомлення [79].

MAC алгоритми можуть бути виготовлені з інших криптографічних примітивів, таких, як криптографічні геш-функції (як у випадку з UMAC), або для блокових алгоритмів шифрування (OMAC, CBC-MAC і PMAC).

Для вивчення основних характеристик MAC кодів і їх порівняльної оцінки в науково-дослідному проекті NESSIE (IST-1999-12324) були розглянуті основні практичні алгоритми MAC кодів і відібрані чотири кращі [124].

Two-Track-MAC

TTMAC (також відомий як Two-Track-MAC) має найвищий рівень безпеки MAC примітивів розглянутих у NESSIE. Дизайн TTMAC заснований на геш-функції RIPEMD-160 (з невеликими змінами). Безпека може бути доведена в припущенні, що основна функція стиснення

є псевдовипадковою. TTMAC має конкретне виконання переваги: вона особливо ефективна в разі коротких повідомлень, а також має оптимальну спритність ключів.

Особливість Two-Track-MAC полягає в шифруванні повідомлення по двох незалежних шляхах (на рис. 5.2 позначені як L і R). Такий підхід дозволяє збільшити розмір внутрішнього стану. У результаті чого отримується більше можливих значень внутрішньої функції шифрування. Це дозволяє ускладнити атаки, засновані на переборі всіляких значень.

У порівнянні з MDx-MAC, який так само заснований на RIPEMD-160 Two-Track-MAC набагато ефективніше для коротких повідомлень (512 або 1024 біт), і також ефективніше на довгих повідомленнях.

Інша важлива перевага TTMAC – можливість швидкої зміни ключа шифрування. Це дозволяє збільшити стійкість системи, без зниження швидкості. При досить частій зміні ключа зловмиснику не вдасться зібрати великої кількості пар повідомлення – MAC-код, що сильно знижує імовірність підбору ключа або MAC-коду.

Проект двухтрекового MAC засновано на геш-функції RIPEMD-160. Спочатку повідомлення, що автентифікується, заповнюється 1-бітами, і потім 0-бітами, поки довжина не досягнеться $448 \bmod 512$. Потім двійкове представлення довжини вихідного повідомлення ($\bmod 2^{64}$) додається так, що довжина повідомлення стає кратною 512. Кожний 512-бітовий блок розділено на набори шістнадцяти 32-бітових слів, $W_0, \ldots, W_{15}$.

Секретний ключ становить набір з 5-ти 32-бітових слів, $K_0, \ldots, K_4$. Алгоритм працює на основі повторення функції стиску таким способом.

Два набори з п’яти 32-бітових слів $X_0, \ldots, X_4$ і $Y_0, \ldots, Y_4$ і набір 16 слів повідомлення є входом для двох різних функцій f_L і f_R, що генерують на виході п’ять 32-бітових слова кожна:

$$A_0, \ldots, A_4 = f_L(X_0, \ldots, X_4, W_0, \ldots, W_{15})$$
$$B_0, \ldots, B_4 = f_R(Y_0, \ldots, Y_4, W_0, \ldots, W_{15}).$$

Ці дві функції ідентичні використаним у RIPEMD-160 (деталі наведені нижче). Потім обчислюється два нові набори з п’яти слів кожний, віднімаючи вхідні повідомлення з результатів попереднього кроку:

$$C_i = A_i - X_i \bmod 2^{32},\ 0 \le i \le 4$$
$$D_i = B_i - Y_i \bmod 2^{32},\ 0 \le i \le 4.$$

Для завершення функції стиску, десять слів C_i і D_i змішані у двох лінійних перетвореннях g_L і g_R:

$$E_0, \ldots, E_4 = g_L(C_0, \ldots, C_4, D_0, \ldots, D_4)$$
$$F_0, \ldots, F_4 = g_R(C_0, \ldots, C_4, D_0, \ldots, D_4)$$

E_i і F_i разом з таким набором слів повідомлення, формують вхідні значення для такого проходу функції стиску. У першій ітерації п'ять секретних ключових слова K_i є входом як X_i так і Y_i, $0 \le i \le 4$.

В останній ітерації, де останні слово повідомлення є входом, f_L і f_R здійснюють свопінг простору. Після віднімання вхідних слів, замість виконання g_L і g_R наприкінці цієї ітерації обчислюється:

$$E_i = C_i - D_i \bmod 2^{32},\ 0 \le i \le 4.$$

Результати цього вирахування формують вихід двохтрекового MAC. Додаткове вихідне перетворення визначене для обчислення більш коротких значень MAC.

Функції f_L і f_R.

Функції f_L і f_R, які відомі як лівий і правий слід функції стиску, ідентичні функціям, використаним у функції стиску RIPEMD-160. Вони складаються з 80 послідовних кроків. Спершу визначаються константи й функції.

Аддитивні константи:

$$k_i = 00000000_x,\quad k_i' = 50a28be6_x,\quad 0 \le i \le 15,$$
$$k_i = 5a827999_x,\quad k_i' = 5c4dd124_x,\quad 16 \le i \le 31,$$
$$k_i = 6ed9eba1_x,\quad k_i' = 6d703ef3_x,\quad 32 \le i \le 47,$$
$$k_i = 8f1bbcdc_x,\quad k_i' = 7a6d76e9_x,\quad 48 \le i \le 63,$$
$$k_i = a953fd4e_x,\quad k_i' = 00000000_x,\quad 64 \le i \le 79.$$

Нелінійні функції на бітовому рівні:

$$f_i(x,y,z) = x \oplus y \oplus z,\qquad 0 \le i \le 15,$$
$$f_i(x,y,z) = (x\&y) \mid (\bar{x}\&z),\quad 16 \le i \le 31,$$
$$f_i(x,y,z) = (x\&\bar{y}) \oplus z,\qquad 32 \le i \le 47,$$
$$f_i(x,y,z) = (x\&z) \mid (y\&\bar{z}),\quad 48 \le i \le 63,$$
$$f_i(x,y,z) = x \oplus (y \mid \bar{z})\ z,\qquad 64 \le i \le 79.$$

Вибір слова повідомлення:

$r[i] = i,\ 0 \le i \le 15$

$r[i] = 7,4,13,1,10,6,15,3,12,0,9,5,2,14,11,8 \quad 16 \le i \le 31$

$r[i] = 3,10,14,4,9,15,8,1,2,7,0,6,13,11,5,12 \quad 32 \le i \le 47$

$r[i] = 1,9,11,10,0,8,12,4,13,3,7,15,14,5,6,2 \quad 48 \le i \le 63$

$r[i] = 4,0,5,9,7,12,2,10,14,1,3,8,11,6,15,13 \quad 64 \le i \le 79$

$r'[i] = 5,14,7,0,9,2,11,4,13,6,15,8,1,10,3,12 \quad 0 \le i \le 15$

$r'[i] = 6,11,3,7,0,13,5,10,14,15,8,12,4,9,1,2 \quad 16 \le i \le 31$

$r'[i] = 15,5,1,3,7,14,6,9,11,8,12,2,10,0,4,13 \quad 32 \le i \le 47$

$r'[i] = 8,6,4,1,3,11,15,0,5,12,2,13,9,7,10,14 \quad 48 \le i \le 63$

$r'[i] = 12,15,10,4,1,5,8,7,6,2,13,14,0,3,9,11 \quad 64 \le i \le 79$

Константи обертання:

$s[i] = 11,14,15,12,5,8,7,9,11,13,14,15,6,7,9,8, \quad 0 \le i \le 15$

$s[i] = 7,6,8,13,11,9,7,15,7,12,15,9,11,7,13,12, \quad 16 \le i \le 31$

$s[i] = 11,13,6,7,14,9,13,15,14,8,13,6,5,12,7,5, \quad 32 \le i \le 47$

$s[i] = 11,12,14,15,14,15,9,8,9,14,5,6,8,6,5,12, \quad 48 \le i \le 63$

$s[i] = 9,15,5,11,6,8,13,12,5,12,13,14,11,8,5,6, \quad 64 \le i \le 79$

$s'[i] = 8,9,9,11,13,15,15,5,7,7,8,11,14,14,12,6, \quad 0 \le i \le 15$

$s'[i] = 9,13,15,7,12,8,9,11,7,7,12,7,6,15,13,11, \quad 16 \le i \le 31$

$s'[i] = 9,7,15,11,8,6,6,14,12,13,5,14,13,13,7,5, \quad 32 \le i \le 47$

$s'[i] = 15,5,8,11,14,14,6,14,6,9,12,9,12,5,15,8, \quad 48 \le i \le 63$

$s'[i] = 8,5,12,9,12,5,14,6,8,13,6,5,15,13,11,11, \quad 64 \le i \le 79$

Нехай дані вихідні значення a_0, b_0, c_0, d_0, e_0. Функція f_L (лівий слід функції стиску), складається з таких кроків для $0 \le l \le 79$ (додавання по mod 2^{32}):

$$a_{i+1} = e_i$$

$$b_{i+1} = (a_i + f_i(b_i,c_i,d_i) + W_{r[i]} + k_i)^{<<s[i]} + e_i\ ,$$

$$c_{i+1} = b_i$$

$$d_{i+1} = c_i^{<<10}$$

$$e_{i+1} = d_i$$

Аналогічно, коли дані вихідні значення a_0, b_0, c_0, d_0, e_0, функція f_R (правий слід функції стиску), складається з таких кроків для для $0 \le l \le 79$ (операції є mod 2^{32}):

$$a_{i+1} = e_i$$
$$b_{i+1} = (a_i + f_{79-i}(b_i,c_i,d_i) + W_{r'[i]} + k_i^{'})^{<<s'[i]} + e_i ,$$
$$c_{i+1} = b_i$$
$$d_{i+1} = c_i^{<<10}$$
$$e_{i+1} = d_i$$

Функції g_L і g_R.

Лінійні перетворення g_L і g_R використовуються для змішування вихідних значень двох слідів функції стиску. Для входів C_0, …, C_4 і D_0, …, D_4, функція g_L обчислює п'ять слів E_0, …, E_4 таким способом (операції є mod 232):

$$E_0 = (C_1 + C_4) – D_3,$$
$$E_1 = C_2 – D_4,$$
$$E_2 = C_3 – D_0,$$
$$E_3 = C_4 – D_1,$$
$$E_4 = C_0 – D_2$$

Для входів C_0, …, C_4 і D_0, …, D_4, функція gr обчислює слово F_0, …, F_4 у таким способом (операції є модулем 2^{32}):

$$F_0 = C_3 - D_4,$$
$$F_1 = (C_4 + C_2) – D_0,$$
$$F_2 = C_0 – D_1,$$
$$F_3 = C_1 – D_2,$$
$$F_4 = C_2 – D_3.$$

Аналіз безпеки.

Безпека двохтрекового MAC може бути доведена на основі припущення, про те, що основна функція стиску псевдодовільна. Ця функція дуже схожа на функцію стиску, використану RIPEMD-160 [78], що і є добре вивченим примітивом, у якому не виявлені ніякі вразливості.

Функції f_L і f_R складаються з 80 ітерацій кроків і використовують різні бітові операції І, АБО, XOR і побітове додавання. Кроки функції також

включають бітові обертання й додавання mod 232, при 80-кратному повторенні цього, здається досить складним простежити невідомі ключові біти. Ніякі вразливості у двохтрековому MAC не виявлені.

Необхідно відзначити, що єдиною ділянкою, де використовується ключовий матеріал, є початкове значення. 160-бітовий вихід алгоритму – різниця між двома 160-бітовими значеннями, і знання цієї різниці все ще дає 160 бітів невизначеності щодо двох оброблених величин. Інакше кажучи, міркування на базі здогадів про значення цих величин і зворотні розрахунки в пошуках вхідного значення, тобто ключа, ненабагато ефективніше, ніж спроба безпосереднього вгадування ключа [26].

Великий розмір внутрішнього стану (320 біт) у двохтрековом MAC надає алгоритму високий рівень безпеки проти атаки, заснованої на внутрішніх помилках. Якщо позначити вихідну довжину m (значення m перебувають між 32 і 160 бітами із кроком 32), складність загальної атаки в цьому примітиві визначається таким способом:

близько 2^{159} обчислень MAC і 160/m відомих пар текст-MAC необхідні для повного ключового пошуку.

Імовірність вгадування величини MAC становить 2-m.

Атака, заснована на внутрішній помилці вимагає близько 2^{160} відомих пар текст-MAC і близько 2^{320-m} обраних текстів.

Схема роботи алгоритму Two-Track-MAC наведено на рис. 5.2.

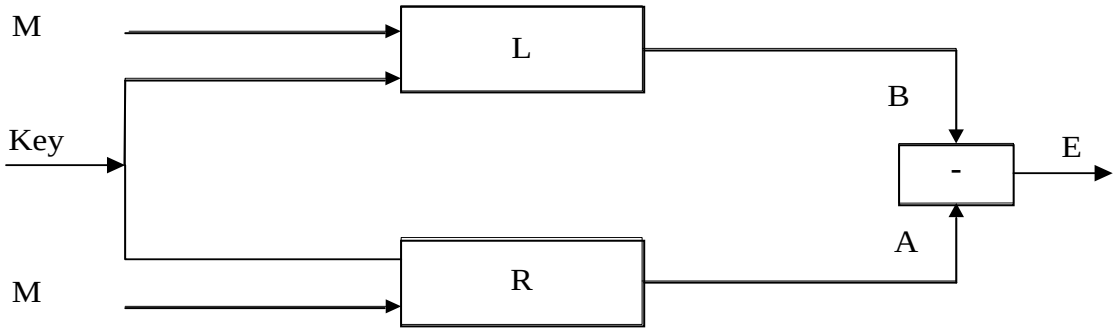


Рис. 5.2. **Схема роботи алгоритму Two-Track-MAC**

CBC-MAC (ISO/IEC 9797-1), до яких відносяться EMAC та RMAC.

Блоковий шифр E є функцією $E: K_E \times \{0,1\}^n \to \{0,1\}^n$, де кожна E (K, •) = Ek (•) є перестановкою на $\{0,1\}^n$, K_E – це множина можливих ключів, а n – довжина блоку.

CBC MAC найбільш простий і добре відомий алгоритм, щоб зробити MAC з блочного шифру E. Нехай M = M [1] ◦ M [2] ◦ … ◦ M [m] буде рядком повідомлення, де | M [1] | = | M [2] | = … = | M [m] | = n. Тоді CBC_K (M), CBC MAC з M з ключами K визначається як Y [m], де Y [i] = = E_K (M [i]⊕ Y[i -1])

для i = 1,..., m і Y [0] = 0^n.

В існуючих реалізаціях MAC-алгоритмів CBC MAC Беллейр, Килиан, і Рогвей запропонували EMAC.

Алгоритм EMAC

EMAC (також відомий як DMAC) має таку перевагу, що дозволяє повторне використання існуючих блокових шифрів реалізації (у CBC-режимі з додатковим шифруванням у вигляді перетворення на виході). Безпека може бути доведена в припущенні, що основний блоковий шифр є псевдовипадковим. Продуктивність і швидкість формування ключів є розумними (EMAC кращий для коротких повідомлень, оскільки довжина блоку менше в порівнянні зі схемами, заснованими на геш-функції). NESSIE рекомендує використовувати цю конструкцію з 128-бітним блоковим шифром, включеним у NESSIE [124]. На рис. 5.3 наведена схема роботи алгоритму CBC-MAC.

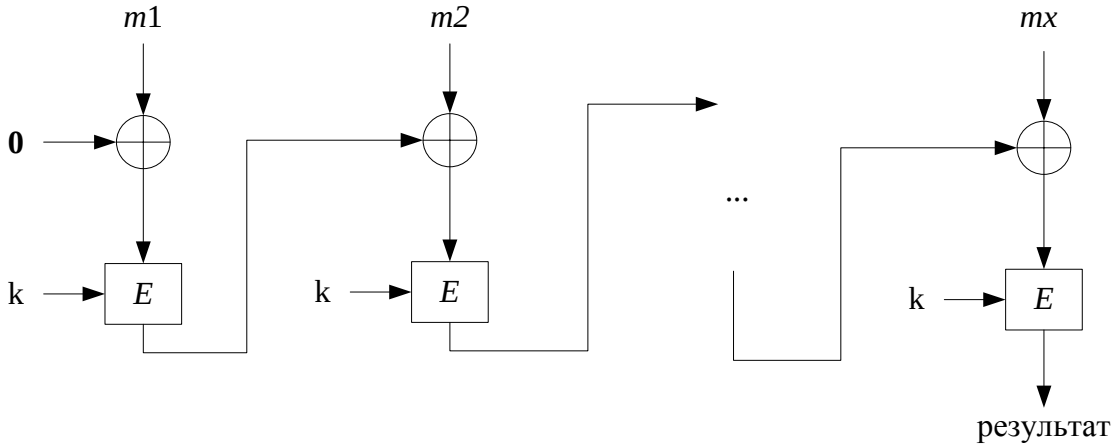


Рис. 5.3. **Схема роботи алгоритму CBC-MAC**

Обчислення EMAC для секретного ключа K і повідомлення X – поділеного (після заповнення) на 128-бітові блоки X_1, …, X_t – оброблених таким способом (тут EK позначає шифрування з використанням 128-бітового блокового шифру AES на ключі K):

1. Обчисліть $H_1 = EK_1(X_1)$.

2. Для i = 2, …, t: обчисліть $H_1 = E_{K1}(X_i⊕ H_{i-1})$.

3. Обчисліть вихідне перетворення: $H_{out} = EK_2(H_t)$. Ключ K_2 може бути похідною від K_1, отриманою в ході такої процедури:

$$K_1 = K_1 \oplus f_0 f_0 \ldots f_{0r}).$$

4. Для того, щоб одержати величину m-біт MAC, виберіть розташовані ліворуч m біти H_{out}.

RMAC становить випадковий варіант цієї схеми, що забезпечує кращу протидію атаці на основі внутрішніх помилок. Єдина відмінність – у вихідному перетворенні, де можна кодувати на ключі, отриманому порозрядовим доповненням K_2 і R:

$$H_{out} = E_{K2 \oplus R}(H_t).$$

Для RMAC, ключі K_1 і K_2 можуть бути незалежними або похідними від одного головного ключа стандартним чином. R – значення, довжиною r біт, й повинне бути заповнене 0-бітами якщо воно менше, ніж K_2. П'ять різних наборів параметрів були визначені для розмірів m і r.

Аналіз безпеки.

Безпека EMAC може бути доведена на базі припущення, що основний блоковий шифр – псевдовипадковий [103], у такому випадку Rijndael, який було проаналізовано, протягом тривалого часу протягом і після процесу AES, реалізує вимоги безпеки.

Необхідно також відзначити, що без додаткового шифрування в остаточному підсумку (або у випадку, якщо K_2 повинен бути обраний рівним K_1), проста (адаптивно обраний текст) підробка буде можлива для схеми CBC-MAC (через атаку підробки EXOR). Якщо K_1 і K_2 повинні бути обрані незалежно, рівень безпеки проти атаки відновлення ключа буде менш імовірним, ніж передбачається за розміром алгоритмів формування ключа (через атаку "розділяй і пануй"). Внутрішня помилка утворює підроблене значення, що вимагає близько 2^{64} відомих пар текст-MAC і 1 обраний текст при вихідній довжині 128 біт. Більша кількість обраних текстів була б потрібна при виключенні виходу MAC, наприклад, при виборі 64 біт, розташованих ліворуч атаці потрібно близько 2^{64} відомих пар і 2^{64} обраних текстів. З іншого боку, це збільшує імовірність успішної атаки вгадування величини MAC.

Основна перевага випадкового варіанта RMAC – те, що надається краща протидія атаці, заснованій на внутрішніх помилках. Наприклад, якщо обрані параметри m = 128 і r = 128, така атака вимагає близько 2^{128} відомих пар і 1 обраний текст. З іншого боку, RMAC необхідні міцні основи для доказу безпеки шифру; наприклад, основний блоковий шифр повинен бути безпечним проти атаки зчепленого ключа. Відомо, що для RMAC із двома незалежними ключами K_1 і K_2 очікується, що повний пошук ключів зажадає генерації 22^{k-1} MAC, де k – розмір одного ключа. Проте, це можна зробити значно швидше для параметрів m = 128 і r = 128: при атаці по обраному повідомленню з одним відомим повідомленням і одним обраним повідомленням K_2 може бути виявлено в 2^k операціях розшифрування, згодом K_1 може бути виявлено протягом приблизно того ж часу. Альтернативна атака на RMAC (m = 128, r = 128) вимагає 2^{123} обраних текстів (при часі виконання 2^{124}) є серйозною атакою на RMAC, що використовує трихключовий Triple-des в основі блокового шифру замість AES (у визначених випадках проведення цих атак має складність близько 2^{56} операцій і імовірність успіху 2^{-16}).

Алгоритм HMAC (ISO/IEC 9797-1)

HMAC має таку перевагу, що дозволяє повторне використання існуючих реалізацій геш-функції. Безпека може бути доведена на таких припущеннях: основна геш-функція стійка до колізій з невідомим початковим значенням; функція стиснення забезпечена ключами на початкове значення безпечного MAC примітиву (для повідомлень з одного блоку); функція стиснення є слабкою псевдовипадковою функцією. Ці припущення слабкіші, ніж припущення, необхідні для TTMAC і EMAC. Продуктивність і спритність ключів є розумними. NESSIE рекомендує використовувати цю конструкцію з колізіонно-стійкою геш-функцією, включеної в NESSIE [118; 119; 124].

Формування MAC-кодів за допомогою HMAC:

$$HMAC(K, M) = h\big((K \oplus opad) \,||\, h(K \oplus ipad) \,||\, M\big),$$

де h – геш-функція;

K – секретний ключ, доповнений нулями до розміру блоку;

M – повідомлення для ідентифікації;

II – конкатенація;

opad – 0x5c5c..5c (довжина дорівнює розміру блока);

ipad – 0x3636..36 (довжина дорівнює розміру блока).

Схема роботи алгоритму HMAC наведено на рис. 5.4.

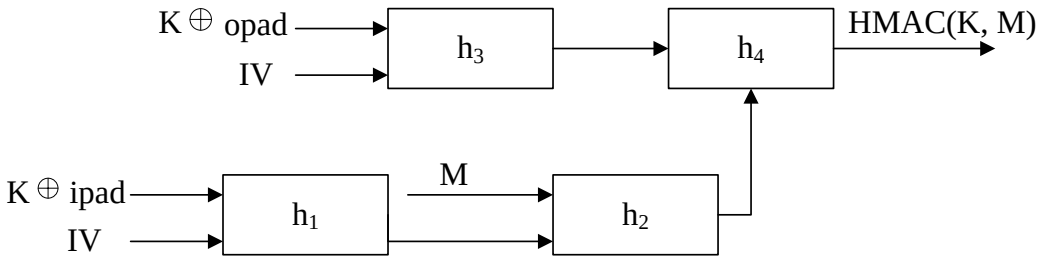


Рис. 5.4. **Схема роботи алгоритму HMAC**

Обчислення MAC для секретного ключа K і повідомлення X відбувається таким чином (тут h позначає геш із геш-функцією SHA-1):

1. Обчисліть ключову величину K’ довжиною 512 біт. Припустимо, що K має довжину l біт. Якщо l = 512, встановлюється K’ = K; якщо l < 512, одержується K’, шляхом додавання 512-l нульових біт до K; якщо l > 512, одержується K’, шляхом обчислення геш h(K) (160 біт довжиною) і додаванням 352 нульових біт до цієї величини геша.

2. EXOR K’ з 512-бітовою константою ipad і додається до повідомлення X: (K’⊕ipad) || X.

3. Обчисліть геш рядок із кроку 2: h((K’⊕ipad) || X).

4. EXOR K’ з 512-бітовою константою opad і додайте 160-бітовий результат із кроку 3: (K’⊕opad) || h((K’⊕ipad) || X).

5. Обчисліть геш рядка із кроку 4: h((K’⊕opad) || h((K’⊕ipad) || X)).

6. Щоб одержати величину m-біт MAC, виберіть m біт результату кроку 5, розташовані ліворуч.

Рядок ipad визначений як 64-кратна конкатенація шістнадцяткової величини "36", і рядок opad – як 64-кратна конкатенація шістнадцяткової величини "5c".

Аналіз безпеки.

Беллейр (Bellare) [36] дає теоретичну основу HMAC, що зв’язує безпеку схеми MAC з безпекою основної геш-функції, і в цьому випадку SHA-1 виступає як досконало вивчений примітив, у якому не знайдені вразливості.

А саме, доведено, що HMAC безпечний, якщо враховувати такі припущення, що f – функція стиску, яка повторюється у геш-функції для кожного 512-бітового блоку:

геш-функція є постійною помилкою при секретній початковій величині;

функція стиску f по ключу з початковою величиною становить досить міцний алгоритм MAC (означає, що вихід важко передбачити);

величини f(K'⊕ipad) і f(K'⊕opad) не відрізняються від дійсно довільних величин. Це означає, що функція стиску f є "слабкою" псевдо-випадковою функцією ("слабкою" оскільки криптоаналітик не має прямого доступу до K').

Слід відзначити також, що якщо конструкція HMAC використовується двома незалежними ключами (замість використання K_1 = K'⊕ipad і K_2 = K'⊕opad), рівень безпеки проти атаки відновлення ключа буде менше, ніж дає розмір ключа алгоритму (через атаку "розділяй і пануй"). Внутрішня помилка утворює підроблене значення для HMAC, засноване на SHA-1, якому необхідно близько 2^{80} відомих пар текст-MAC і 1 обраний текст при вихідній довжині 160 біт. Більша кількість обраних текстів була б потрібна при урізаному виході MAC, наприклад, при виборі 80 біт, які розташовані ліворуч, атака потребує близько 2^{80} відомих пар і 2^{80} обраних текстів. З іншого боку, це збільшує імовірність успіху для атаки вгадування величини MAC.

UMAC: розробка корпорації Intel (США), Університету з штату Невада в Рено (США), Науково-дослідної лабораторії IBM (США), Technion (Ізраїль) і Університету з Каліфорнії в Девісі (США);

Код автентифікації повідомлення UMAC заснований на сімействах універсальних геш-функцій і забезпечує перевірку безпеки в тому розумінні, що є доказові межі помилки геш частини обчислення MAC, так що безпека в підсумку залежить від шифрувального примітива, використаного для кодування залишку. Примітив, установлений у специфікаціях AES (Rijndael) блокового шифру, який аналізувався довгий час протягом і після процесу AES, реалізує вимоги до безпеки. Існує також додаткова перевага, яка полягає в тому, що шифрування виконується на невеликому залишку.

Ніяких вразливостей не було виявлено в ході доведення безпеки UMAC. Для прошарку, що використовує універсальне сімейство геш-функцій NH, перший доказ безпеки свідчить про те, що NH є майже 2^{-w} – універсальним (це означає, що імовірність зіткнення – не більше 2^{-w}) для рядків рівної довжини, для слів довжиною w біт. Це відповідає використанню NH на одному блоці повідомлення встановленої довжини. Другий доказ безпеки дозволяє поширювати цей результат на NH, що працює з будь-якою парою рядків подібно до UMAC. Причина того, що імовірність помилки після прошарку геша NH у схемі Uhash16 становить 2^{-15}, а не 2^{-16}, у тому, що використовується знакова версія NH. Перший

варіант доведення безпеки показує, що знакова версія NH 2^{-w+1} – майже універсальна.

Uhash16 і Uhash32 мають два додаткових прошарка, що використовують RP і універсальні сімейства геш-функцій IP; вони повторюють тришарову схему, відповідно, два й чотири рази. Доведено, що сімейство геша Uhash16 4 має 4 рівні ($2^{-15} + 2^{-18} + 2^{-28}$) і є майже універсальним, і що сімейство геша Uhash32 має 2 рівні ($2^{-31} + 2^{-33}$) і також майже універсальним.

На рис. 5.5 наведено схему роботи алгоритмів UMAC16/32.

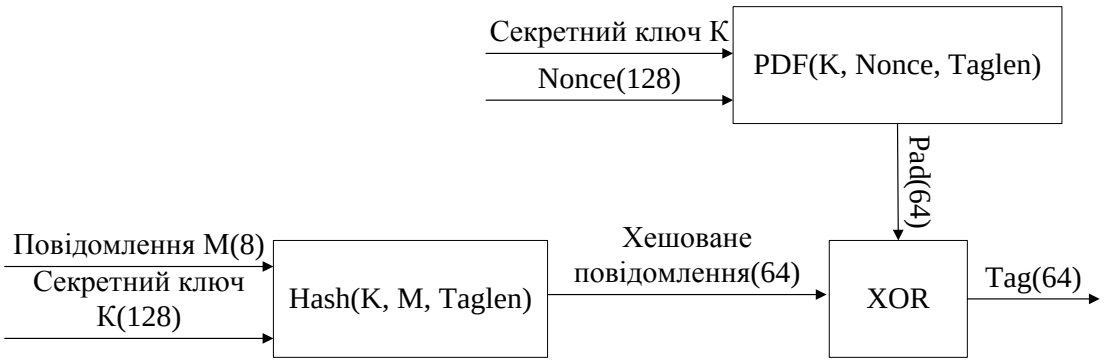


Рис. 5.5. **Схема роботи алгоритму UMAC16 / 32**

Алгоритм дозволяє забезпечити більш високу швидкість (особливо для довгих повідомлень) і підтверджувану безпеку ціною більшої складності.

5.2. Результати конкурсу SHA-3

Основні вимоги, висунуті Національним інститутом стандартів і технологій США (NIST) до алгоритмів-конкурсантів припускають створення класу геш-функцій потенційно стійких до атак, спрямованих на SHA-2, а також збереження або збільшення ефективності гешування в порівнянні з SHA-2 [123]. Алгоритм-переможець конкурсу SHA-3 повинен підтримувати розмір вихідного блоку 224, 256, 384 і 512 бітів. Використання дайджестів геш-кодів довжиною 160-біт не допускаються через можливість знаходження колізій атаками грубої сили (повного перебору всіх варіантів). При проведенні конкурсу зберігаються ті ж вимоги, що і до попередніх геш-функцій: максимальний розмір вхідного значення, розмір вихідного значення, колізійна стійкість, стійкість до знаходження прообразу і другого прообразу, потоковий режим обчислення "за один прохід" [123].

Алгоритми функцій обчислення для різного розміру блоків повинні бути ідентичні і мати мінімум відмінностей в реалізації. Використання абсолютно різних наборів алгоритмів для отримання чотирьох фіксованих значень довжини виходу не допускається [123].

Крім цього, висуваються вимоги щодо вдосконалення існуючих алгоритмів роботи геш-функцій: можливо включення опції рандомізованого гешування, покращене розпаралелювання, оптимальна робота на множині сучасних платформ (включаючи як 64-бітові процесори, так і 8-бітові процесори, а також смарт-карти) [123].

У геш-функції класу SHA-3 можуть бути вбудовані процедури погодження для обчислення кодів автентифікації повідомлень (HMAC): наприклад, передача в якості одного з вхідних параметрів довжини блоку вхідного повідомлення, якщо вона заздалегідь точно відома. Значення всіх вбудованих параметрів і констант повинні бути сконструйовані таким чином, щоб не дати можливості вбудовування лазівок і відмичок з боку розробників, для чого мають бути приведені відповідні докази та процедури перевірки [123].

Проведений аналіз специфічних вимог до стійкості показав, що основною вимогою є забезпечення теоретичних меж стійкості (в реальності вони можуть бути трохи менше) на розмір вихідної блоку в n біт і становитиме:

у конструкціях кодів автентифікації повідомлень (HMAC) і псевдовипадкових функцій (PRF) стійкість за кількістю запитів повинна бути не менше 2 (n/2) біт проти атак-розпізнювань;

стійкість рандомізованого гешування проти атаки знаходження H (M_1, r_1) = H (M_2, r_2) – не менше n біт, за умови, що значення рандомізатора r_1 не контролюється атакуючим.

Стандартні та додаткові параметри стійкості:

стійкість до знаходження колізій – не менше n / 2 біт;

стійкість до знаходження прообразу – n біт;

стійкість до знаходження другого прообразу – n – k біт для будь-якого повідомлення, коротше 2^k бітів;

стійкість до атак на зміну довжини повідомлення;

підмножина геш-функцій (наприклад, отримане усіканням числа бітів виходу) розміром m бітів повинно зберігати властивості n-бітної множини в перерахунку на m, з урахуванням статистичних (не відрізняються від випадкових) відхилень. Не повинно існувати атаки на швидке знаходження малостійких підмножин геш-функції з вихідної множини n;

стійкість до нових типів атак, наприклад, на основі мультиколізій.

Національним інститутом стандартів і технологій США 29 березня 2012 року був проведений третій раунд конкурсу SHA-3, у табл. 5.5 наведені основні характеристики алгоритмів-претендентів.

Таблиця 5.5

Характеристики алгоритмів гешування кандидатів 3 раунду конкурсу

Назва алгоритму	Платформа	Стійкість алгоритму	Швидкодія (Мбіт/с)
BLAKE	8, 32,64	є достатньо стійким	44,37
Gröstl		Нестійкий до атак "напіввільний початок"	7,98
JH	8, 32, 64	низька алгебраїчна степінь у висновках функції стиснення	10,6
Keccak	32, 64	кількість раундів, необхідне для захисту від атак – 18	8,05
Skein	8, 32, 64	Захищений від атак – підбору подовжених повідомлень і псевдоколізій	39,23

Аналіз табл. 5.5 показує, що основна увага розробників алгоритмів-конкурсантів була спрямована на виконання основних вимог з продуктивності і можливості оптимальної роботи алгоритму при його реалізації на множині сучасних платформ. Разом з тим, керівники конкурсу при відборі алгоритмів-кандидатів у другий тур звернули увагу на стійкість даних алгоритмів до різних видів атак.

У 3 раунд були відібрані алгоритми гешування, в яких не була порушена будь-яким чином безпека. У деяких випадках, представлений варіант був занадто повільним чи вимагав занадто багато пам’яті, але NIST вважає, що в деяких випадках, параметри, що настроюються, можуть бути відкоректовані, щоб дати прийнятний рівень продуктивності без шкоди для безпеки [123].

Попереднє вивчення алгоритмів-учасників показало, що кожен з них запозичив деякі принципи побудови у алгоритму AES для забезпечення стійкості геш-кодів [103].

Таким чином, головною вимогою до алгоритмів-конкурсантів керівники NIST висунули безпеку, що і стало основним критерієм відбору кандидатів у 3 раунд. Тільки 5 алгоритмів з 14 відібраних після другого

раунду (всього на конкурс було подано 51 алгоритм) змогли продемонструвати необхідний рівень захисту від криптографічних атак. До основних критеріїв відбору також відноситься продуктивність і можливість роботи на великому числі платформ, що також дозволило керівникам конкурсу "відсіяти" кілька кандидатів (алгоритми Vortex, LUX і т. д.).

5.2.1. BLAKE (Jean-PhilippeAumasson)

BLAKE Алгоритм стиснення, функція якого заснована на використанні ключової підстановки в конструкції Davies-Meyer. Ключова підстановка заснована на внутрішній організації потокового шифру ChaCha. Отримав свою не лінійність з накладенням модульного складання і операцій XOR. Сама інноваційна частина BLAKE – своя ключова перестановка [67].

Структурна схема алгоритму наведено на рис. 5.5.

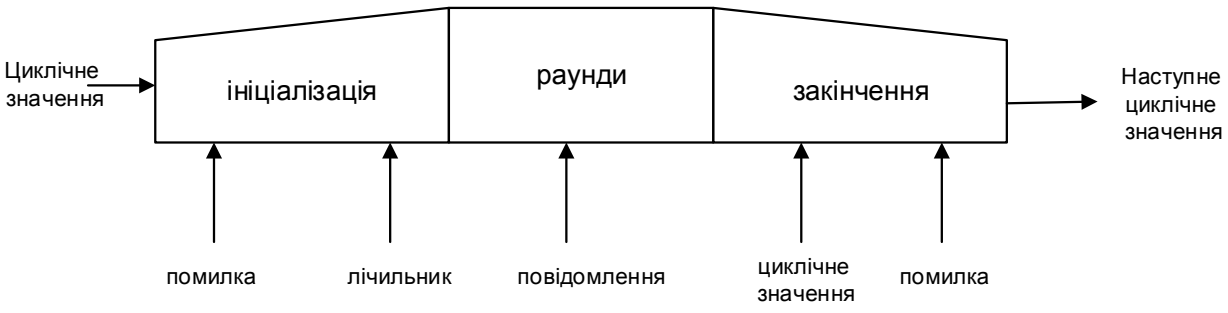


Рис. 5.5. **Структурна схема алгоритму BLAKE**

Геш-функція BLAKE-32 працює з 32-бітними словами й повертає 32-байтовое значення геша. Цей відрізок визначає BLAKE-32, виходячи з постійних параметрів власної функції стиску, а потім переходить до його ітеративного режиму.

Константи.

BLAKE-32 запускає гешування від того ж початкового значення, що й SHA-256:

IV_0 = 6A09E667 IV_1 = BB67AE85

IV_2 = 3C6EF372 IV_3 = A54FF53A

IV_4 = 510E527F IV_5 = 9B05688C

IV_6 = 1F83D9AB IV_7 = 5BE0CD19

BLAKE-32 використовує 16 постійних значень:

c_0 = 243F6A88 c_1 = 85A308D3

c_2 = 13198A2E c_3 = 03707344

c_4 = A4093822 c_5 = 299F31D0

c_6 = 082EFA98 c_7 = EC4E6C89

c_8 = 452821E6 c_9 = 38D01377

c_{10} = BE5466CF c_{11} = 34E90C6C

c_{12} = C0AC29B7 c_{13} = C97C50DD

c_{14} = 3F84D5B5 c_{15} = B5470917

Десять перестановок {0, . . ., 15}, що використовуються всіма функціями BLAKE, наведено в табл. 5.6.

Функція стиску.

Функція стиску BLAKE-32 приймає на вході чотири значення:

• ланцюгове значення h = h_0, . . ., h_7;

• блок повідомлення = m_0, . . .,m_{15};

• сіль s = s_0, . . ., s_3;

• лічильник t = t_0, t_1.

Таблиця 5.6

Перестановки {0, . . ., 15} використовувані функціями BLAKE

σ_0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
σ_1	14	10	4	8	9	15	13	6	1	12	0	2	11	7	5	3
σ_2	11	8	12	0	5	2	15	13	10	14	3	6	7	1	9	4
σ_3	7	9	3	1	13	12	11	14	2	6	5	10	4	0	15	8
σ_4	9	0	5	7	2	4	10	15	14	1	11	12	6	8	3	13
σ_5	2	12	6	10	0	11	8	3	4	13	7	5	15	14	1	9
σ_6	12	5	1	15	14	13	4	10	0	7	6	3	9	2	8	11
σ_7	13	11	7	14	12	1	3	9	5	0	15	4	8	6	2	10
σ_8	6	15	14	9	11	3	0	8	12	2	13	7	1	4	10	5
σ_9	10	2	8	4	7	6	1	5	15	11	9	14	3	12	13	0

Ці чотири входи становлять у підсумку 30 слів (тобто, 120 байт = 960 біт). На виході функції – нове ланцюгове значення h' = h'_0, . . . , h'_7 з восьми слів (тобто 32 байта = 256 біт). Записується стиск h, m, s, t в h', як h′ = compress (h, m, s, t).

Ініціалізація.

Стан 16-слова $v_0, \ldots, v_{15}$ ініціалізовано так, що різні входи виробляють різні початкові стани. Стан представляється як матриця розміром 4 × 4, і заповнюється так, як зазначено нижче:

$$\begin{pmatrix} \nu_0 & \nu_1 & \nu_2 & \nu_3 \\ \nu_4 & \nu_5 & \nu_6 & \nu_7 \\ \nu_8 & \nu_9 & \nu_{10} & \nu_{11} \\ \nu_{12} & \nu_{13} & \nu_{14} & \nu_{15} \end{pmatrix} \leftarrow \begin{pmatrix} h_0 & h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 & h_7 \\ s_0 \oplus c_0 & s_1 \oplus c_1 & s_2 \oplus c_2 & s_3 \oplus c_3 \\ t_0 \oplus c_4 & t_0 \oplus c_5 & t_1 \oplus c_6 & t_1 \oplus c_7 \end{pmatrix}$$

Циклічна функція.

Як тільки стан ν проініціалізовано, функція стиску виконує ітерації серіями по 10 раундів. Раунд – це перетворення значення ν, яке обчислюється:

$G_0(v_0, v_4, v_8, v_{12})$ $G_1(v_1, v_5, v_9, v_{13})$ $G_2(v_2, v_6, v_{10}, v_{14})$ $G_3(v_3, v_7, v_{11}, v_{15})$
$G_4(v_0, v_5, v_{10}, v_{15})$ $G_5(v_1, v_6, v_{11}, v_{12})$ $G_6(v_2, v_7, v_8, v_{13})$ $G_7(v_3, v_4, v_9, v_{14})$

де на раунді r, $G_i(a, b, c, d)$ множини:

$$a \leftarrow a + b + (m_{\sigma_r(2i)} \oplus c_{\sigma_r(2i+1)})$$
$$d \leftarrow (d \oplus a) <<< 16$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) <<< 12$$
$$a \leftarrow a + b + (m_{\sigma_r(2i+1)} \oplus c_{\sigma_r(2i)})$$
$$d \leftarrow (d \oplus a) <<< 8$$
$$c \leftarrow c + d$$
$$b \leftarrow (b \oplus c) <<< 7$$

Перші чотири виклики $G_0,\ldots,G_3$ можуть бути розраховані паралельно, тому що кожний з них поновлює окремий стовпець матриці. Потрібно звернутися до процедури обчислення $G_0,\ldots,G_3$ крок-стовпець.

Так само, останні чотири виклики $G_4,\ldots,G_7$ поновлюють певні діагоналі й таким способом можуть бути синхронізовані так, що потрібно звернутися до діагонального кроку.

На рис. 5.6 і 5.7 наведено Gi, крок-стовпець і діагональний крок.

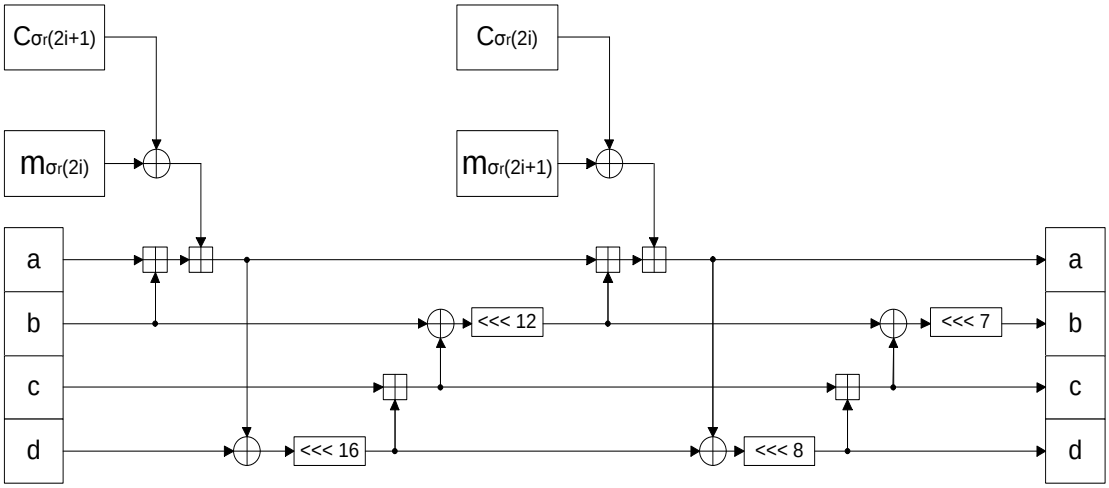


Рис. 5.6. **Функція Gi**

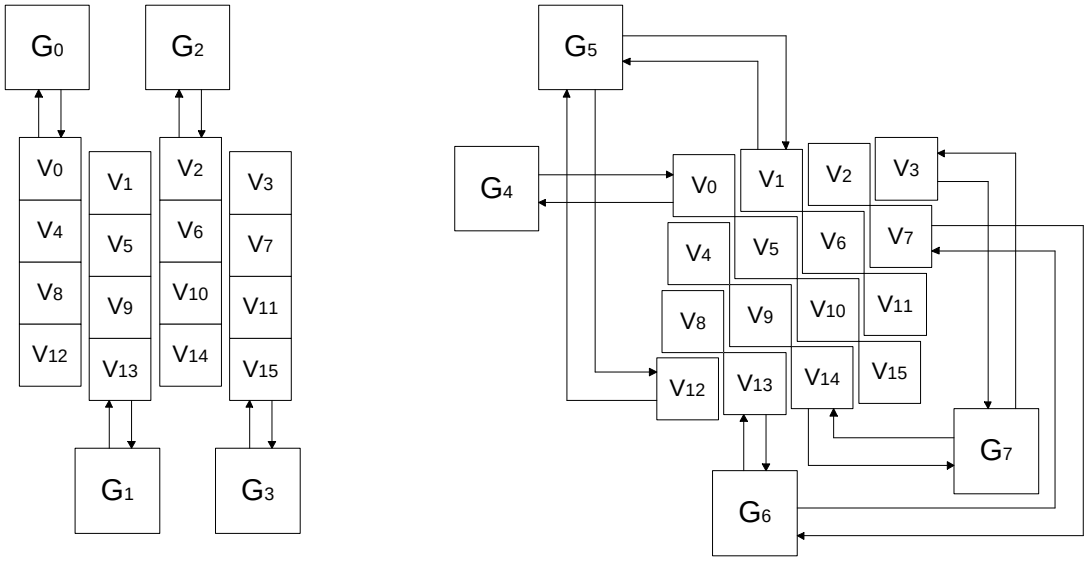


Рис. 5.7. **Крок-стовпець і діагональний крок**

Завершення.

Після послідовності циклів нове ланцюгове значення $h'_0, \ldots, h'_7$ отримується із установленого $v_0,\ldots,v_{15}$ із входом початкового ланцюгового значення $h_0,\ldots,h_7$ і сіллю $s_0,\ldots,s_3$:

$$h'_0 \leftarrow h_0 \oplus s_0 \oplus \nu_0 \oplus \nu_8$$
$$h'_1 \leftarrow h_1 \oplus s_1 \oplus \nu_1 \oplus \nu_9$$
$$h'_2 \leftarrow h_2 \oplus s_2 \oplus \nu_2 \oplus \nu_{10}$$
$$h'_3 \leftarrow h_3 \oplus s_3 \oplus \nu_3 \oplus \nu_{11}$$
$$h'_4 \leftarrow h_4 \oplus s_0 \oplus \nu_4 \oplus \nu_{12}$$
$$h'_5 \leftarrow h_5 \oplus s_1 \oplus \nu_5 \oplus \nu_{13}$$
$$h'_6 \leftarrow h_6 \oplus s_2 \oplus \nu_6 \oplus \nu_{14}$$
$$h'_7 \leftarrow h_7 \oplus s_3 \oplus \nu_7 \oplus \nu_{15}$$

Гешування повідомлення.

Процедура для гешування повідомлення m розрядної довжини $\ell < 264$ притаманна циклічним геш-функціям, повідомлення спочатку *доповнюється* (при додаванні BLAKE використовує правило подібне до правила HAIFA), потім оброблений блок приймається функцією стиску).

Доповнення.

Спочатку повідомлення розширюють таким чином, що його довжина збігається з 447 по модулю 512. Довжина збільшується, додаючи в кінець достатню кількість 0 біт. Збільшення як мінімум на один біт і як максимум на 512. Потім приєднується одиничний біт, за ним іде 64-бітне невизначене багатобайтовое представлення ℓ. Доповнення може бути подано, як: $m \leftarrow m \,||\, 1000...0001\langle l \rangle_{64}$.

Ця процедура гарантує, що бітова довжина повідомлення, що доповнюється, – кратна 512.

Гешування.

Щоб приступити до гешування, доповнене повідомлення розбивається на блоки по 16-слів $m_0,\ldots,m_{n-1}$. Нехай ℓ_i кількість біт у повідомленні $m_0,\ldots,m_i$, тобто, крім бітів, доданих при доповненні. Наприклад, якщо оригінальне (не доповнене) повідомлення 600-біт у довжину, то доповнене повідомлення буде мати два блоки, і $\ell^0 = 512$, $\ell^1 = 600$. Приватний випадок відбувається, коли останній блок не містить жодного біта оригінального повідомлення; наприклад, 1020-розрядне повідомлення приводить до доповненого повідомлення із трьома блоками (які містять відповідно 512, 508, і 0 біт повідомлення), і встановлюється $\ell^0 = 512$, $\ell^1 = 1020$, $\ell^2 = 0$. Загальне правило: якщо останній блок не містить жодного біта оригінального повідомлення, тоді лічильник зводиться до нуля; це гарантує, що, якщо $i \neq j$, то $\ell_i \neq \ell_j$.

Сіль *s* вибирається користувачем, і встановлюється нульовою, якщо ніякої солі не потрібно (тобто, $s_0 = s_1 = s_2 = s_3 = 0$). Гешування доповненого повідомлення m проводиться, як зазначено:

$$h^0 \leftarrow IV$$
$$for\, i = 0,\ldots,N-1$$
$$h^{i+1} \leftarrow compress(h^i, m^i, s, l^i)$$
$$return\, h^N$$

Процедура гешування m з BLAKE-32 записується BLAKE-32(m, s)= = h_n, де m (не доповнене) повідомлення, і s – сіль. Запис BLAKE-32(m) означає гешування m коли ніяка сіль не використовується (тобто, s = 0).

5.2.2.Grostl (LarsRamkildKnudsen)

Grostl – колекція геш-функцій, здатних повертати короткі виклади повідомлення будь-якої кількості байтів від 1 до 64, тобто від 8 до 512 біт у 8 бітних кроках. Різновиди, що повертають n біт, називаються Grostl-n. Це включає розміри короткого викладу повідомлення 224, 256, 384, і 512 біт [123].

Конструкція геш-функції.

Геш-функції Grostl повторюють функцію стиску f таким чином. Повідомлення M доповнене й розділене на блоки повідомлення по ℓ біт m_1, . . . ,m_t, і кожний блок повідомлення обробляється послідовно. Початкове ℓ-бітне значення h_0 = iv визначене, і згодом блоки повідомлень mi обробляються як: $h_i \leftarrow (h_{i-1}, m_i)$, for i = 1,…,t.

Отже, в картах два входи по ℓ біт кожний і вихід з ℓ біт. Перший вхід називається послідовним входом, а другий вхід названий блоком повідомлення. Різновиди Grostl повертають аж до 256 біт, ℓ визначено як 512. Для більших різновидів, ℓ – 1024.

Після того, як останній блок повідомлення був оброблений, вихід геш-функції H(M) обчислюється, як: H(M) = Ω (ht),
де Ω – вихідне перетворення. Вихідний розмір Ω – n біт, і відзначається, що n < ℓ . Структуру геш-функції наведено на рис 5.9.

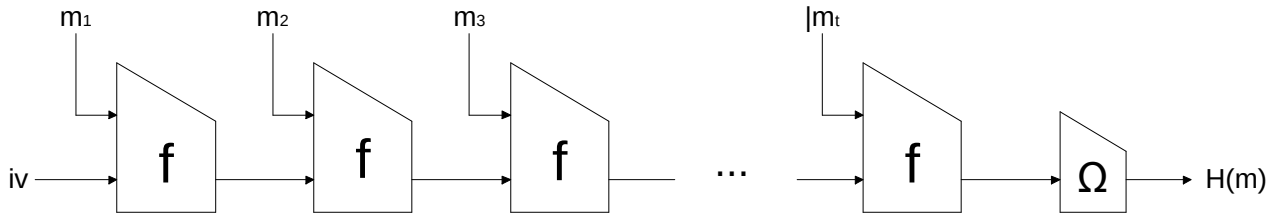


Рис 5.9. **Геш-функція Grostl**

Конструкція функції стиску.

Функція стиску заснована на двох основних ℓ-бітних перестановках – P і Q. Які визначені таким чином: f(h,m) = P(h⊕m)⊕Q(m)⊕h.

На рис. 5.10 наведено конструкцію функції стиску f.

Вихідне перетворення.

Нехай truncn(x) є операцією, яка відкидає всі майже кінцеві n біти x. Потім вихідне перетворення (рис. 5.11) визначається, як:

$$\Omega(x) = \text{truncn}(P(x) \oplus x).$$

Тобто, вихідне перетворення Ω розраховується як P(x) ⊕x, а потім виключає вихід, повертаючи тільки останні n біт.

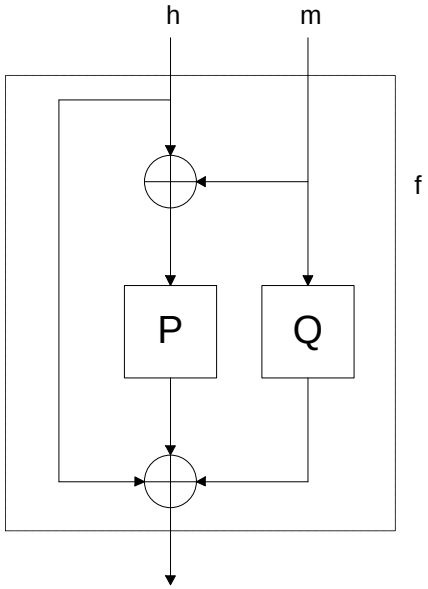


Рис. 5.10. **Функція стиску *f*. *P* і *Q* – ℓ-бітні перестановки**

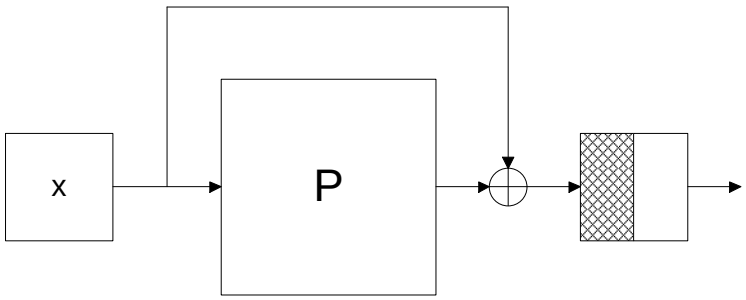


Рис. 5.11. **Вихідне перетворення Ω**

Структура P і Q.

Функція стиску *f* включає два варіанти – один використаний для коротких викладів повідомлення, а другий використаний для довгих викладів повідомлення.

Кожний варіант використовує свою власну пару перестановок *P* і *Q*. Отже, у підсумку, визначається чотири перестановки. Перестановки будуть призначені приписками 512 або 1024, щораз, коли необхідно буде їх розрізнити.

Задум P і Q був оснований на блоковому алгоритмі шифру Rijndael. Це означає, що їх ціль складається з множини циклів R, які складаються з множини перетворень. З того часу, коли *P* і *Q* – значно більше, ніж установлений 128-бітний розмір Rijndael, більшість перетворень перевизначена.

У Grostl, у цілому для кожної перестановки визначено чотири перетворення. Це:

Addroundconstant;

Subbytes;

Shiftbytes;

Mixbytes.

Коли необхідна відмінність, третє перетворення Shiftbytes називають Shiftbyteswide при використанні більших перестановок P_{1024} і Q_{1024}. Усі інші перетворення можуть бути описані таким же способом для всіх чотирьох перестановок.

Цикл R складається із чотирьох кругових перетворень доданих у порядку, наведеному на рис. 5.12.

$$R = MixBytes \circ ShiftBytes \circ SubBytes \circ AddRoundconstant.$$

Усі раунди ідуть за цим визначенням. Слід позначити r як кількість раундів.

Перетворення діють у стані, який представляється як матриця байтів A (8 біт кожний). Для коротких варіантів, матриця має 8 рядків і 8 стовпців, а для великих варіантів, матриця має 8 рядків і 16 стовпців. Потрібно позначити v кількістю стовпців.

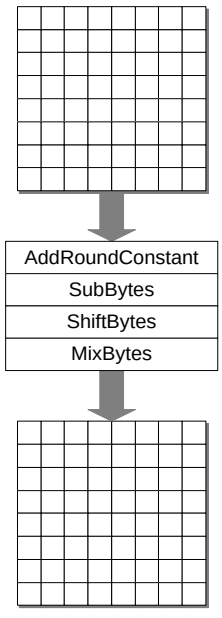


Рис. 5.12. **Один раунд Grostl перестановки *P* і *Q***

Проектування від байтової послідовності до встановленої матриці й навпаки.

Оскільки Grostl оперує байтами, це в основному невизначені байти. Однак необхідно конкретизувати як байтова послідовність проектується на матрицю *A*, і навпаки. Цей розподіл зроблений аналогічним чином як у Rijndael. Отже, 64-байтова послідовність 00 01 02...3f відображається як матриця 8 × 8.

$$\begin{bmatrix} 00 & 08 & 10 & 18 & 20 & 28 & 30 & 38 \\ 01 & 09 & 11 & 19 & 21 & 29 & 31 & 39 \\ 02 & 0a & 12 & 1a & 22 & 2a & 32 & 3a \\ 03 & 0b & 13 & 1b & 23 & 2b & 33 & 3b \\ 04 & 0c & 14 & 1c & 24 & 2c & 34 & 3c \\ 05 & 0d & 15 & 1d & 25 & 2d & 35 & 3d \\ 06 & 0e & 16 & 1e & 26 & 2e & 36 & 3e \\ 07 & 0f & 17 & 1f & 27 & 2f & 37 & 3f \end{bmatrix}$$

Для матриці 8 × 16, цей метод розширюється природним шляхом. Розподіл від матриці до байтової послідовності – просто зворотна операція. Далі не згадуватиметься цей розподіл явно.

Addroundconstant.

Перетворення Addroundconstant додає навколо залежну константу до встановленої матриці A. Доповненням визначається "Виключнє або" (XOR). P і Q мають різні округлі константи, які є єдиною відмінністю між двома перестановками.

Округлі константи можуть бути наведені як матриці того ж розміру, що й установлена матриця. Усі округлі байти констант є нульовими, крім єдиної позиції. Округлі константи використовуються для P_{512} і P_{1024} – в основному однакові: перші 8 стовпців не відрізняються в обох перестановках, а останні 8 стовпців кругових констант використовуються в P_{1024}, утримуючі тільки байти, що мають значення 00. Теж саме для Q_{512} і Q_{1024}. Байт у крайньому верхньому лівому куті округлої константи в раунді i_p має значення i; всі інші позиції в округлій постійної матриці мають значення 00. У Q, байт у крайньому нижньому лівому куті має значення i⊕ff, а всі байти мають значення 00. Кількість раундів зменшується по модулю 2^{56}, за необхідністю. Щоб бути точним,

перетворення Addroundconstant (що наведено на рис. 5.13) у раунді і модифікує стан A, як:

$$A \leftarrow A \oplus 3[i],$$

де C[i] – округла константа, що використовується в i-му раунді.

Округлі константи CP[i] і CQ[i] використовувані в i-му раунді P і Q відповідно.

$$C_P[i] = \begin{bmatrix} i & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \end{bmatrix} \text{ and } \quad C_Q[i] = \begin{bmatrix} 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ 00 & 00 & ... & 00 \\ i \oplus ff & 00 & ... & 00 \end{bmatrix}$$

Рис. 5.13. **Addroundconstant для перестановок *P* і *Q***

Subbytes.

Перетворення Subbytes замінює кожний байт у матриці станів іншою величиною, яка береться з s-box S. Цей s-box такий же як і в Rijndael. Отже, якщо $a_{i,j}$ – елемент у рядку *i* і стовпці *j* A, тоді Subbytes виконує таке перетворення:

$$ai_{,j} \leftarrow S(ai_{,j}),\ 0 \le i < 8,\ 0 \le j < v.$$

Shiftbytes і Shiftbyteswide.

Shiftbytes і Shiftbyteswide циклічно переміщує байти в межах рядка ліворуч на кількість позицій. Нехай $\sigma = [\sigma_0, \sigma_1, \ldots, \sigma_7]$ буде списком чітких цілих чисел у діапазоні від 0 до v – 1.

Потім, Shiftbytes переміщає всі байти в рядку і матриці станів σі позицій ліворуч, що згортаються наскільки необхідно. Вектор σ визначений як σ = [0, 1, 2, 3, 4, 5, 6, 7] в Shiftbytes, і σ = [0, 1, 2, 3, 4, 5, 6, 11] у Shiftbyteswide.

Mixbytes.

У перетворенні Mixbytes, кожний стовпець у матриці перетворюється незалежно. Щоб описати це перетворення, нам спочатку потрібно ввести обмежене поле F_{256}. Це обмежене поле визначається таким же чином, як у Rijndael, через що не піддається перетворенню поліном $x^8 \oplus x^4 \oplus x^3 \oplus x \oplus 1$ через F_2.

Байти матриці станів A можна представити як елементи F_{256}, тобто як поліноміальні степені, як мінімум, 7 з коефіцієнтами в {0, 1}. Молодший розряд кожного байта визначає коефіцієнт x^0 і т. п. [7].

Mixbytes множить кожний стовпець A на константу матриці B 8 × 8 у F_{256}. Отже, у цілому матриця перетворення може бути написана через множення матриць:

$$A \leftarrow B \times A.$$

Матриця B – циркулянт, який означає, що кожний рядок дорівнює рядку вище обертової праворуч на одну позицію.

Отже, можна записати B = circ (02, 02, 03, 04, 05, 03, 05, 07).

Матриця B визначена, як:

$$B = \begin{bmatrix} 02 & 02 & 03 & 04 & 05 & 03 & 05 & 07 \\ 07 & 02 & 02 & 03 & 04 & 05 & 03 & 05 \\ 05 & 07 & 02 & 02 & 03 & 04 & 05 & 03 \\ 03 & 05 & 07 & 02 & 02 & 03 & 04 & 05 \\ 05 & 03 & 05 & 07 & 02 & 02 & 03 & 04 \\ 04 & 05 & 03 & 05 & 07 & 02 & 02 & 03 \\ 03 & 04 & 05 & 03 & 05 & 07 & 02 & 02 \\ 02 & 03 & 04 & 05 & 03 & 05 & 07 & 02 \end{bmatrix}$$

Кількість раундів r є регульованим параметром безпеки. Рекомендовано такі величини r для чотирьох перестановок.

Перестановки	Розмір викладу	Рекомендоване значення *r*
P_{512} і Q_{512}	8 – 256	10
P_{1024} і Q_{1024}	264 – 512	14

Початкові значення.

Початкове значення ivngrostl – n – ℓ-бітне представлення n.

У табл. 5.7 наведено початкові значення, які необхідні для вихідних розмірів 224, 256, 384, і 512 біт.

Таблиця 5.7

Початкові значення

n	ivn
224	00...00 00 e0
256	00...00 01 00
384	00...00 01 80
512	00...00 02 00

Доповнення.

Довжина кожного блоку повідомлення – ℓ. Щоб бути здатним подіяти на входи змінної довжини, функція доповнення pad визначена. Ця функція доповнення приймає рядок x довжиною N біт і повертає доповнений рядок x* = pad(x) довжиною, яка є кратною ℓ.

Функція, що доповнює, робить таке. Спочатку, вона додає "1" біт до x. Потім, вона додає w = – N-65 modℓ "0"біт, і нарешті, вона додає 64-бітне представлення (N + w + 65) / ℓ. Це число є цілим через вибір w, і представляє кількість блоків повідомлення в кінцевому, доповненому повідомленні. Оскільки це повинно бути доступним для кодування кількості блоків повідомлення, в доповненому повідомленні в межах 64 біт, максимальна довжина повідомлення – 65 біт до 2^{64-1} блоків повідомлення. Для коротких варіантів, максимальна довжина повідомлення в бітах – $512\cdot(2^{64-1}) - 65 = 2^{73} - 577$, і для більш довгих варіантів це – $1024\cdot(2^{64-1}) - 65 = 2^{74} - 1089$.

Для початку, повідомлення, яке повинне бути перекручене Grostl, доповнюється використовуючи функцію доповнення pad. Потім геш-функція повторює функцію стиску $f: \{0, 1\}^{\ell} \times \{0, 1\}^{\ell} \rightarrow \{0, 1\}^{\ell}$, яка заснована на двох перестановках P і Q. Якщо вихідний розмір n геш-функції становить як мінімум 256 біт – встановлюється ℓ = 512. Для більш довгих варіантів – встановлюється ℓ = 1024. Отже, $\ell \geq 2^n$ для всіх випадків. Початкове значення Grostl-n – ℓ-бітне представлення n. В остаточному підсумку, вихід останнього виклику f обробляється вихідним перетворенням Ω, яке скорочує розмір висновку з ℓ до n бітам.

5.2.3. JH (HongjunWu)

JH – використовує нову конструкцію, що дещо нагадує конструкцію "sponge" для вбудовування алгоритму геш у блок-підстановку. Блок-підстановка – комбінація з двох 4-бітових S-боксу з низкою лінійних операцій змішування і розрядних підстановок. Уся нелінійність у цьому дизайні отримана з S-боксів [103]. Структурну схему алгоритму наведено на рис. 5.14.

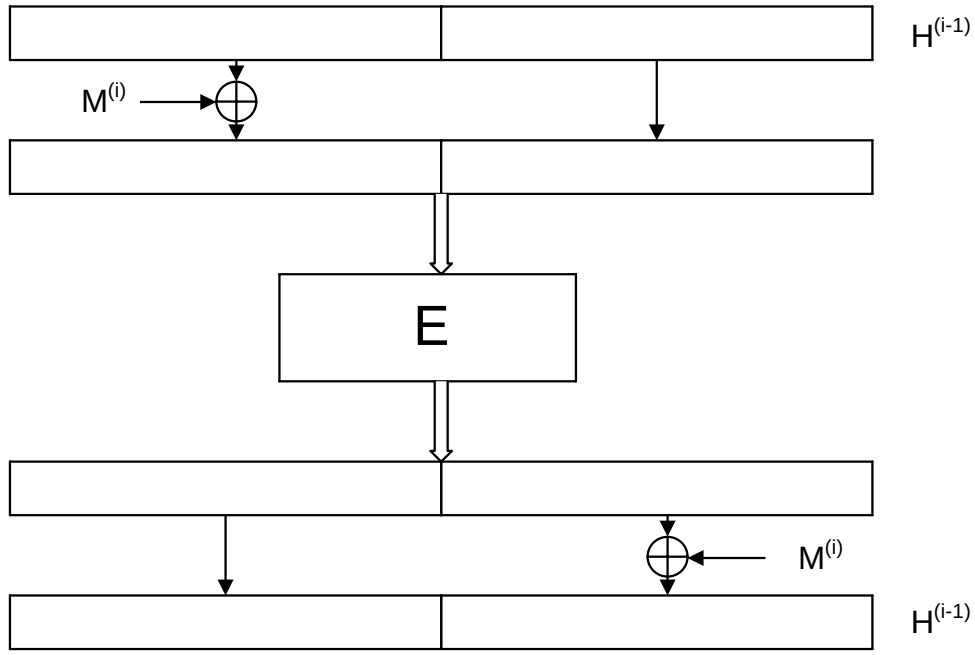


Рис. 5.14. **Структурна схема алгоритму JH**

5.2.4. Keccak (JoanDaemen)

Keccak – використовує конструкцію "sponge" і блок-підстановку. Підстановка може бути реалізована на основі 5-бітових S-блоків або на комбінації лінійної і нелінійної операціях змішування.

Структурну схему алгоритму наведено на рис. 5.15.

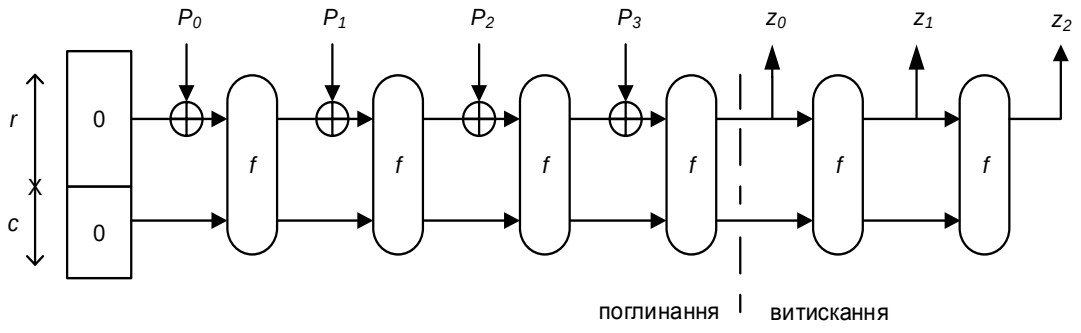


Рис. 5.15. **Структурна схема алгоритму Keccak**

5.2.5. Skein

Skein – сімейство геш-функцій із трьома різними внутрішньо-встановленими розмірами: 256, 512, і 1024 біта.

Skein-512. Воно може безпечно використовуватися для всіх поточних додатків, що гешують, і повинно залишитися безпечним для найближчого майбутнього.

Skein-1024. Має двічі внутрішньовстановлений розмір – Skein-512, який задовольняє вимогам безпеки. Навіть якщо якій-небудь майбутній атаці вдалося зламати Skein-512, то імовірно, що Skein-1024 залишиться безпечним. Skein-1024 може також працювати в два рази швидше, ніж Skein-512 у визначених апаратних реалізаціях.

Skein-256 може реалізовуватися, використовуючи близько 100-байтів у RAM.

Нова ідея Skein у тому, щоб побудувати геш-функцію поза налагодженим блоковим шифром. Використання налагодженого блокового шифру дозволяє Skein гешувати дані конфігурації разом із вхідним текстом у кожному блоці, і зробити будь-яке середовище функції стиску унікальним. Ця властивість безпосередньо адресує множину атак геш-функцій, і суттєво поліпшує гнучкість Skein.

Точніше кажучи Skein створений із трьох нових компонентів:

Threefish. Threefish це модифікований блоковий шифр у ядрі Skein, певний 256-, 512-, і 1024-бітним розміром блоку.

Unique Block Iteration (унікальний блок ітерацій, UBI). UBI – режим, що підключається, який використовує Threefish щоб побудувати функцію стиску яка перетворює довільний вхідний розмір у фіксований вихідний.

Optional argument system (додаткова система аргументу). Вона дозволяє Skein підтримувати ряд додаткових характеристик, не вимагаючи ніяких перевантажень при виконанні й додатків, які не використовують характеристики. Основний Threefish алгоритм описує багаторічні знання проектування блокового шифру й аналізу. UBI доведено безпечний і може використовуватися з будь-якими налагодженими шифрами. Додаткова система аргументу дозволяє Skein пристосовуватися для різних цілей. Ці три компоненти незалежні й можуть використовуватися окремо, але їх комбінація забезпечує реальні переваги.

Блоковий шифр Threefish.

Threefish є більшим, модифікованим блоковим шифром [66]. Він визначений для трьох різних розмірів блоку: 256 біт, 512 біт і 1024 біт.

Ключ такого ж розміру як і блок, а модифіковане значення – 128 біт для будь-якого розміру блоку. Основний проектний принцип Threefish – те, що більша кількість простих раундів більш безпечніша, ніж меньша кількість раундів. Threefish використовує тільки три математичні операції – виключаюче або (XOR), доповнення, і постійні обертання на 64-бітні слова – дуже швидкі на сучасних 64-бітних процесорах.

Гешування Skein.

Skein побудований на численних викликах UBI. На рис. 5.21 наведено Skein як просту геш-функцію. Починаючи з ланцюгового значення 0, існує три виклики UBI: по кожному на конфігураційний блок, повідомлення (до $2^{96 - 1}$ байт у довжину), і вихідне перетворення [123].

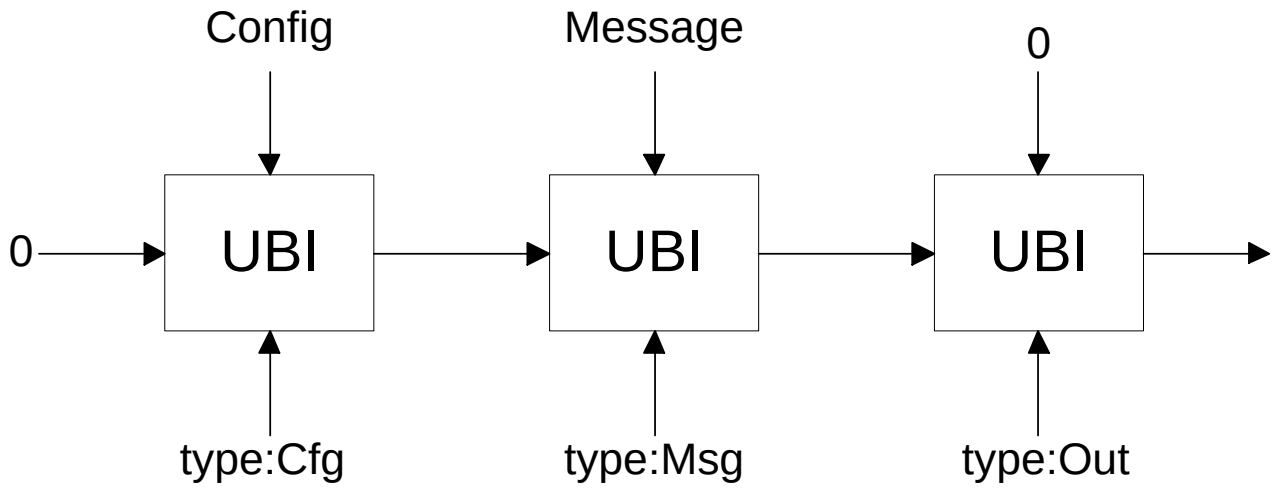


Рис. 5.21. **Режим нормального гешування Skein**

32-байтовий рядок конфігурації шифрує бажану вихідну довжину й деякі параметри, щоб підтримувати дерево гешування. Якщо Skein використовується як стандартна геш-функція – з фіксованим вихідним розміром і без дерева гешування або MAC-ключа – результатом обчислення конфігураційного блоку UBI є константа для всіх повідомлень і може бути попередньо розрахована як IV.

Вихідні перетворення потрібні для досягнення гешування необхідної довільності. Це також дозволяє Skein породжувати виходи будь-якого розміру аж до 2^{64} біт. Якщо одного вихідного блоку недостатньо, вихідні перетворення запускаються кілька раз, як наведено на рис. 5.22.

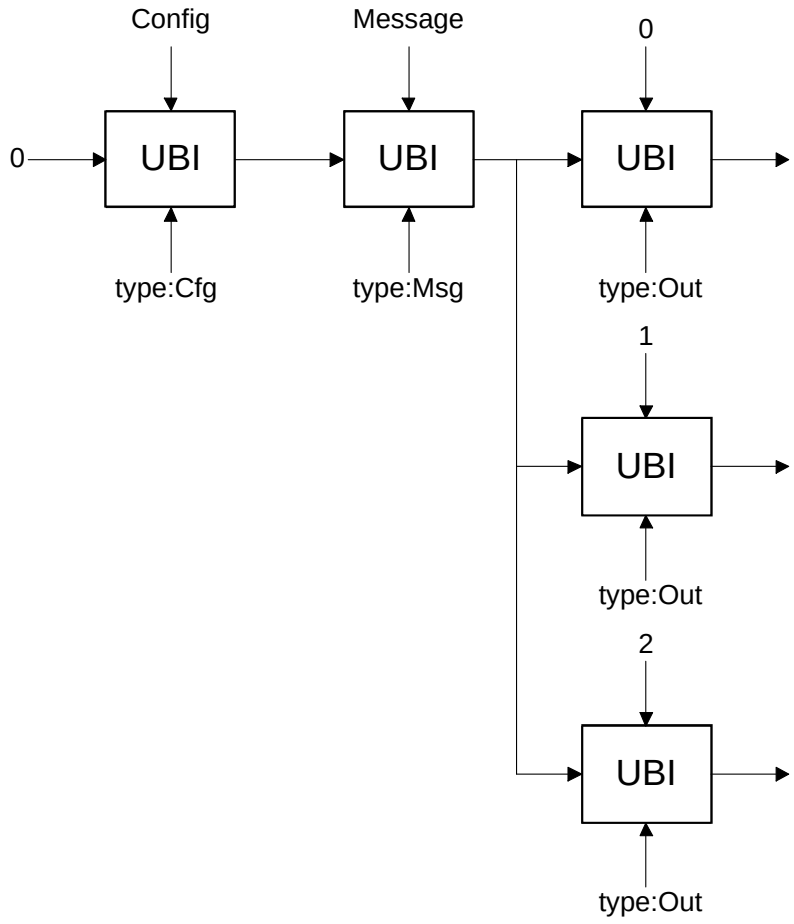


Рис. 5.22. **Skein з більшим вихідним розміром**

З’єднані входи з усіма вихідними перетвореннями однакові, поле даних складається з 8-байтового лічильника. По суті, використовується Threefish у режиму лічильника. Створення більших виходів більш зручно, але звичайно безпека Skein обмежена внутрішньо встановленим розміром.

Додаткові аргументи.

Для того, щоб збільшити гнучкість Skein, кілька додаткових входів можуть бути дозволені, у якості необхідних. Усі ці опції управляються додатками реального часу.

Ключ (додатковий). Ключ який перетворює Skein у MAC або KDF функцію. Ключ завжди обробляється в першу чергу, щоб підтримувати деякі з доказів безпеки.

Конфігурація (необхідна).

Персоналізація (додаткова). Рядок, який можуть використовувати додатки для створення різних функцій для різного використання.

Відкритий ключ (додатковий). Відкритий ключ використовується при гешуванні повідомлення для його підпису. Це зв'язує геш-підпису й відкритий ключ. Таким чином, ця характеристика перевіряє, що одне й теж повідомлення генерує різний геш для різних відкритих ключів.

Ключ ідентифікатор висновка (додатковий). Використовується для добутку ключів. Для того, щоб одержати ключ заготовлюють головний ключ як ключ для входу, й ідентифікатор запитуваного похідного ключа.

Поточний час (додатковий). Поточне значення часу використовується в потокових шифрах і рандомізованому гешуванні.

Повідомлення (додаткове). Нормальне вхідне повідомлення для геш-функції.

Вихід (необхідний). Вихідне перетворення.

Обчислення Skein складаються з обробки цих опцій у порядку, що використовується UBI. Кожний вхід має різний "тип" значення для модифікації, який гарантує, що входи не взаємозамінні.

Жодне з них не впливає на виконання й складність основної геш-функції жодною мірою; інші реалізації можуть вибирати яку опцію реалізувати, а яку проігнорувати.

Очевидно, Skein може бути розширений з іншими необов'язковими аргументами. Вони можуть бути додані в будь-який час, навіть, коли функція вже була стандартизована, тому що додавання нових необов'язкових аргументів є оборотно-сумісним.

Skein-MAC

Стандартний спосіб використовувати геш-функцію для автентифікації – використовуючи конструкцію HMAC [6; 85]. Skein звичайно ж може бути використаний з HMAC, але це вимагає як мінімум двох геш-обчислень для кожної автентифікації, що є неефективним для коротких повідомлень. Перетворення Skein у MAC наведено на рис.5.23.

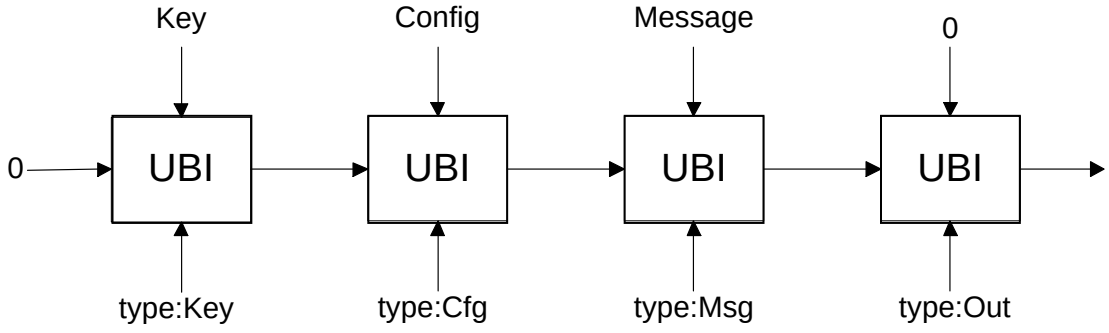


Рис. 5.23. **Skein-MAC**

Замість обробки блоку конфігурації, яка починається з нуля, починається обробка ключа з нуля, а потім вже блоку конфігурації. Або дивлячись з іншого боку, Skein-гешування просто Skein-MAC з нульовим ключем. І подібно тому, як вихід Skein конфігураційного блоку попередньо обчислюється для наданого ключа. З того часу, як найбільш загальні шляхи були використані MAC для автентифікації численних повідомлень із єдиним ключем – це значно збільшує виконання для коротких повідомлень.

Дерево гешування.

При гешуванні дуже великої кількості даних, лінійна структура класичної лінійної геш-функції стає обмеженою – це перешкоджає багатоядерному ЦП використовувати кілька ядер одночасно. Також загальне використання геш-функцій повинне перевірити цілісність великої кількості даних. З лінійної геш-функцією, всі дані повинні бути перевірені в деякий час. Це може бути дуже неефективним, тому що часто бажано перевіряти цілісність тільки незначної частини даних.

Дерево гешування вирішує ці проблеми. Замість гешування даних як одного великого рядка, вони розділяються на частини. Кожна частина гешується й результуючий геш розглядається як нове повідомлення. Ця процедура може застосовуватися рекурсивно, поки результат не стане єдиним значенням геша.

Skein включає режим гешування деревом для підтримки додатків цього типу. Оскільки різні додатки мають різні вимоги, ці три параметри для додатка можуть обиратися для конкретного використання додатка: вузловий розмір листа, treefan-out і максимальна висота дерева.

Повна специфікація Skein.

Рядок.

Рядок X – послідовність із нуля або більшої кількості значень, кожний з якого має тип X. Наприклад: рядок байтів – послідовність із нуля або більше байтів. Записуються рядки, як відділені комою списки, і, звичайно, нумеруються елементи, починаючи з нуля; наприклад, рядок t з 7 значень запишеться:

$$t = t_0, t_1, ..., t_6.$$

Оператор конкатенації || означає конкатенацію рядків. Використовується 0^n, щоб позначити рядок з n нулів, де тип нулів (біти або байти) буде зрозумілий від контексту.

Порядок біт і байт.

Порядок біт і байт – загальне джерело плутанини в шифрувальних алгоритмах. Отже, Skein завжди використовує найменшу значущу байтове перше значення.

Основний універсальний тип даних у сучасному ЦП – рядок байтів. Кожний байт має значення в діапазоні 0..255. Байт також часто розглядається як послідовність із 8 біт b_7, b_6,..., b_0, де кожний b_i – 0 або 1 і байтовое значення b надане:

$$b := \sum_{i=0}^{7} b_i \cdot 2^i .$$

Значення b_i часто йменується як "біт i" b.

Рядок біт запам'ятовується як рядок байтів. Для змагання геш-функції, NIST специфікує конкретний розподіл від рядка біт до рядка байтів. Кожна група з 8 біт закодована в байті, перший біт вступає в 7 біт байта, що випливає на 6 біт і т. д. Якщо довжина рядка біт – не кратна 8, останній байт використовується тільки частково, з нижчими бітовими позиціями, що йдуть невикористаними [123].

Щоб перетворити послідовності байтів до цілого числа, використовується найменше значуще байтове перше значення. Нехай b_0, ..,b_{n-1} буде рядок з n байт.

Визначається:

$$\text{ToInt}(b_0, ...,b_{n-1}) := \sum_{i=0}^{n-1} b_i \cdot 256^i .$$

Зворотне перетворення забезпечене функцією ToBytes:

$$\text{ToBytes}(v,n) := b_0, ...,b_{n-1}, \text{ where } b_i := \left\lfloor \frac{v}{256^i} \right\rfloor \bmod 256 .$$

Ця функція застосовна тільки тоді, коли $0 \le v < 256^n$, тому що байти повністю кодують значення v.

Часто відбуваються перетворення між рядком з 8n байт і рядком з n 64-бітних слів і назад. Нехай $b_0,\ldots,b_{8n-1}$ байти. Визначаються:

$\text{BytesToWords}(b_0,\ldots,b_{8n-1}) := w_0,\ldots, w_{n-1}$ where $w_i := \text{ToInt}(b_{8i},b_{8i+1},\ldots b_{8i+7})$.

Зворотне перетворення проводитися:

$\text{WordsToBytes}(w_0,\ldots,w_{n-1}) := \text{ToBytes}(w_0,8)||\text{ToBytes}(w_1,8)||\ldots||\text{ToBytes}(w_{n-1},8)$.

Повний опис Threefish.

Threefish – модифікований блоковий шифр з 256, 512, 1024-бітними розмірами блоків. Вхід модифікації становить завжди 128-біт.

Функція шифрування E(K; T; P) ухвалює такі параметри:

K – блоковий шифро-ключ; рядок з 32, 64, або 128 байт (256, 512, або 1024 біт).

T – щипок, рядок з 16 байт (128 біт).

P – звичайний текст, рядок байт довжиною, рівною ключу.

Threefish діє повністю на беззнакових 64-бітних словах (тобто, значення в діапазоні $0..2^{64-1}$). Усі входи перетворюються в рядки 64-бітних слів. Нехай N_w є кількістю слів у ключі (а також у звичайному тексті). Ключ K представляється як ключові слова (k_0, k_1, . . ., kn_{w-1}), модифікація T представляється як слова (t_0, t_1), і звичайний текст P, як (p_0, p_1, . . . , pn_{w-1}): $k_0,\ldots,k_{Nw-1}$:=BytesToWords(K), t_0,t_1 :=BytesToWords(T), $p_0,\ldots,p_{Nw-1}$:=BytesToWords(P).

Кількість раундів – N_r – функція блокового розміру. Її значеня наведено у табл. 5.8.

Таблиця 5.8

Кількість раундів для різних блокових розмірів

Block/Key Size	# Words N_w	# Rounds N_r
256	4	72
512	8	72
1024	16	80

Ключовий список (описаний далі) перетворює ключ і модифікує в послідовності з Nr/4+1 підключей, кожний з яких складається з N_w слів. Визначаються слова підключа s як ($ks_{,0}$, . . ., $ks_{,Nw-1}$).

Нехай $v_{d,i}$ значення i-th слова зашифроване після d раундів. Потрібно почати з:

$$v_{0,i} := p_i \text{ for } i=0,\ldots,N_{w-1},$$

а потім ухвалити N_r раундів пронумерованих d = 0, . . .,N_{r-1}.

Для кожного раунду, додається підключ, якщо залишок від розподілу d на 4 рівний 0. Для i = 0, . . .,N_{w-1}, одержується:

$$e_{d,i} := \begin{cases} (v_{d.i} + k_{d/4,i}) \bmod 2^{64} & \text{if } d \bmod 4 = 0 \\ v_{d.i} & \text{otherwise} \end{cases}$$

Змішування й перестановка слів визначаються:

$$(f_{d,2j}, f_{d,2j+1}) := MIX_{d,j}(e_{d,2j}, f_{d,2j+1}) \quad \text{for } j = 0,\ldots,N_{w/2}-1$$
$$v_{d+,I} := f_{d,\pi(i)} \quad \text{for } i = 0,\ldots,N_w-1,$$

$f_{d,I}$ значення – результати MIX функції (визначеної далі); і вихід з перестановки слова, що і є виходом раунду.

Зашифрований текст C наведений:

$$c_i := (v_{Nr,i}+k_{Nr/4,i}) \bmod 2^{64} \quad \text{For } i = 0,\ldots,N_w-1$$
$$C := \text{WordsToBytes}(c_0,\ldots,c_{Nw-1}).$$

MIX-функції.

$MIX_{d,j}$ функція має два слова на вході (x_0, x_1) і формує два слова на виході (y_0, y_1), використовуючи такі відносини:

$$y_0 := (x_0 + x_1) \bmod 2^{64}$$
$$y_1 := (x_1 <<< R_{(d \bmod 8),\, j}) \oplus y_0,$$

де <<< – оператор обертання ліворуч. Значення константи $R_{d,j}$ наведено в табл. 5.9.

Таблиця 5.9

Константи обертання $R_{d,j}$ для кожного N_w

N_w		4		8				16							
j		0	1	0	1	2	3	0	1	2	3	4	5	6	7
d=	0	14	16	46	36	19	37	24	13	8	47	8	17	22	37
	1	52	57	33	27	14	42	38	19	10	55	49	18	23	52
	2	23	40	17	49	36	39	33	4	51	13	34	41	59	17
	3	5	37	44	9	54	56	5	20	48	41	47	28	16	25
	4	25	33	39	30	34	24	41	9	37	31	12	47	44	30
	5	46	12	13	50	10	17	16	34	56	51	4	53	42	41
	6	58	22	25	29	39	43	31	44	47	46	19	42	44	25
	7	32	32	8	35	56	22	9	48	35	52	23	31	37	20

Список ключів.

Список ключів запускає визначення двох додаткових слів k_{Nw} і t_2:

$$k_{Nw} := \lfloor 2^{64}/3 \rfloor \oplus \bigoplus_{i=0}^{N_w-1} k_i$$
$$t_2 := t_0 \oplus t_1.$$

Константа $[2^{64}/3]$ гарантує, що розширюваний ключ клавіша не може містити всі нулі. Список ключів визначений як:

$$k_{s,i} := k_{(s+i) \bmod (Nw+1)} \quad \text{for } i = 0,\ldots,N_w-4$$
$$k_{s,i} := k_{(s+i) \bmod (Nw+1)} + t_s \bmod 3 \quad \text{for } i = N_w-3$$
$$k_{s,i} := k_{(s+i) \bmod (Nw+1)} + t_{(s+1)} \bmod 3 \quad \text{for } i = N_w-2$$
$$k_{s,i} := k_{(s+i) \bmod (Nw+1)} + s \quad \text{for } i = N_w-1$$

де доповнення кратні 2^{64}.

Розшифрування.

Операція розшифрування в Threefish – очевидна протилежність операції шифрування. Підключі використовуються у зворотному порядку й кожний раунд складається із застосування зворотної перестановки, що супроводжується зворотними MIX-функціями.

Повний опис UBI.

Послідовний режим UBI заснований на модифікованому блоковому шифрі із блоковим розміром і розміром ключа N_b байт, а розмір модифікації 16 байт. Функція UBI (G, M, T_s) має входи:

G – початкове значення N_b байт.

M – рядок повідомлення довільної бітової довжини до 2^{998} біт, зашифрована в рядок байт.

T_s – 128-бітне ціле число, яке є початковим значенням для модифікації.

UBI обробляє повідомлення в блоках, використовуючи унікальне значення модифікації для кожного блоку. Поля модифікації наведено в табл. 5.10. Щоб уникнути багатьох різних параметрів слід розглядати модифікацію як єдине 128-бітне значення.

Таблиця 5.10

Поля в значенні модифікації

Найменування	Біти	Опис
Position	0 – 95	Кількість оброблених у рядку байт (включаючи цей блок)
reserved	96 – 111	Резервується для майбутнього використання, повинен бути нульовим
Treelevel	112 – 118	Рівень у дереві геш, нуль для обчислень поза деревом
Bitpad	119	Установлюється, якщо цей блок містить останній байт входу, чиєю довжиною не були інтегральний ряд байт. 0 – інакше
Type	120 – 125	Тип поля (конфігурація, повідомлення, вихід і т. д.)
First	126	Установлюється для першого блоку UBI стиску
Final	127	Установлюється для останнього блоку UBI стиску

Це спрощує пояснення, але й накладає деякі обмеження на значення які може мати T_s. Bitpad, First і Final поля повинні бути нульовими; поле Position повинне мати таке значення, що сума поля Position і довжина M у байтах не повинна перевищувати 2^{96-1}.

Якщо кількість біт у даних M – кратна 8, визначається B := 0 і M’ := M. Якщо кількість біт у M – не кратна 8, останній байт тільки

частково використаний. Найбільш значущі бітові позиції останнього байта містять дані. Потрібно доповнити останній байт, встановлюючи найбільш значущій невикористаний біт в 1, а інші невикористані біти (якщо є) у нуль. Варто визначити B := 1, і нехай M’ буде M із прикладеним бітовим доповненням.

Нехай N_M кількість байт в M′. Вхід обмежено до $N_M < 2^{96}$. Слід доповнити M'_p нульовими байтами поки довжина не стане кратною розміру блоку, гарантуючи, що одержиться як мінімум один цілий блок:

$$p := \begin{cases} N_b & \text{if } N_M = 0 \\ (-N_M) \bmod N_b & \text{otherwise} \end{cases}$$

$$M'' := M' \| 0^p$$

Потрібно розбити M″ на k блоків повідомлень блокує $M_0,\ldots,M_{k-1}$, кожний з N_b байт.

Результат UBI обчислюється як:

$$H_0 := G$$

$H_{i+1} := E(H_i, \text{ToBytes}(T_s+\min(N_M,(i+1)N_b)+a_i 2^{126}+b_i(B2^{119}+2^{127}),16),M_i) \oplus M_i$, де $a_0 = b_{k-1} = 1$, і всі інші a_i і b_i значення становлять 0, E() – модифікований блоковий шифр, функції шифрування, а H_k – результат послідовного режиму UBI.

Значення модифікації для кожного блоку створює доповнення:

$$T_s+\min(N_M,(i+1)N_b)+a_i 2^{126}+b_i(B2^{119}+2^{127}).$$

Перший термін – T_s який визначає поля Treelevel і Type, і додатково забезпечує компенсацію для поля Position. $\min(N_M,(i + 1)N_b)$ змінює тільки поле Position. Для кожного блоку поле Position ця кількість оброблених байт, включаючи всі байти на поточному блоці плюс компенсація від Ts. Обмеження Ts гарантовано ніколи не потурбують поле Position щодо доповнення, яке могло б змінити інші поля. a_i 2^{126} установлює прапор First, але тільки для першого блоку UBI обчислень. $b_i(B2^{119}+2^{127})$ здійснює дві дії. Для будь-якого блоку, крім останнього, $b_i = 0$, тому що цей термін нічого не робить. В останньому блоці, встановлюється прапор Final (бітова позиція 127), і якщо жоден біт доповнення не був прикладений, тоді встановлюється прапор Bitpad (бітова позиція 119).

Висновки

Процес побудови та експлуатації комплексних систем захисту інформації вимагає використання найбільш ефективних механізмів забезпечення цілісності й автентичності інформації. Тому теорія універсального гешування, яка використовується для автентифікації в цих системах є актуальною.

Робота розкриває основні поняття теорії автентифікації та безумовної автентифікації ґрунтуючись на основних положеннях теорії гешування. Це дозволяє чітко простежити порядок формування автентифікаторів зі специфічними властивостями універсального гешування.

Викладені положення з теорії гешування мають безперечну наукову новизну, що підтверджується посиланнями на наукові праці та дисертаційні дослідження провідних спеціалістів у галузі побудови й експлуатації комплексних систем захисту інформації, дозволяють самостійно вивчити основні положення теорії автентичності на основі використання геш-функцій.

Моделі та методи універсального гешування інформації з високими показниками криптографічної стійкості, у тому числі ті, що володіють властивостями доказово стійких криптографічних систем, які розроблені на основі обчислювальних алгоритмів гешування.

У роботі докладно розглянуто найсучасніші схеми автентифікації повідомлень із використанням відомих алгоритмів із зазначенням їх переваг та недоліків, що дозволяє визначити шляхи їх подальшого вдосконалення залежно від сфери їх використання з урахуванням широкого поширення обчислювальних систем на 64-розрядних платформах та досягненням потрібної криптографічної стійкості схеми автентифікації повідомлень.

Використана література

1. Баранов О. А. Інформаційне право України: стан, проблеми, перспективи / О. А. Баранов. – К. : ВД "СофтПрес", 2005. – 316 с.
2. Берлекэмп Э. Алгебраическая теория кодирования / Э. Берлекэмп ; пер. с англ. – М. : Мир, 1971. – 478 с.
3. Блейхут Р. Теория и практика кодов, контролирующих ошибки / Р. Блейхут ; пер. с англ. – М. : Мир, 1986. – 576 с.
4. Вервейко В. Н. Функции хеширования: классификация, характеристика и сравнительный анализ / В. Н. Вервейко, А. И. Пушкарев, Т. В. Цепурит. – Х. : ХНУРЕ, 2002. – 68 с.
5. Влэдуц С. Г. Линейные коды и модулярные кривые / С. Г. Влэдуц, Ю. И. Манин // Современные проблемы математики. – М. : ВИНИТИ. – 1984. – Т. 25. – С. 209–257.
6. Гоппа В. Д. Коды на алгебраических кривых // Докл. АН СССР. – 1981. – Т. 259. – № 6. – С. 1289–1290.
7. Горбенко И. Д. Исследование аналитических и статистических свойств булевых функций криптоалгоритма Rijndael (FIPS 197) / И. Д. Горбенко, А. В. Потий, Ю. А. Избенко // Радиотехника. Всеукраинский межведомственный научно-технический сборник. – 2004. – № 126. – С. 132–138.
8. Горбенко И. Д. Анализ каналов уязвимости системы RSA // Безопасность информации / И. Д. Горбенко, В. И. Долгов, А. В. Потий. – Х. : ХНУРЕ, 1995. – № 2. – С. 22–26.
9. ГОСТ Р 34.10–2001. Государственный стандарт Российской Федерации. Информационная технология. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. – М. : Госстандарт России, 2001. – 24 с.
10. Грушо А. А. Анализ и синтез криптоалгоритмов: курс лекций / А. А. Грушо, Э. А. Применко, Е. Е. Тимонина. – М. : ЛОГОС, 2000. – 110 с.
11. Диффи У. Защищенность и имитостойкость // Введение в криптографию / У. Диффи, М. Хеллман. – СПб. : Питер, 1999. – 309 с.
12. Долгов В. И. О некоторых подходах к построению безусловно стойких кодов аутентификации коротких сообщений / В. И. Долгов, В. Н. Федорченко // Управление и связь. – Х. : ХНУРЕ, 1996. – № 4 – С. 47–51.
13. Домарев В. В. Защита информации и безопасность компьютерных систем. / В. В. Домарев. – К. : Издательство "ДиаСофт", 1999. – 480 с.
14. ДСТУ 4145–2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка. – К. : Держстандарт України, 2002. – 40 с.

15. Евсеев С. П. Механизмы обеспечения аутентичности банковских данных во внутриплатежных системах комерческого банка : збірник наукових статей ХНЕУ / С. П. Євсеєв, В. Е. Чевардин, С. А. Радковский. – Х. : ХНЕУ, 2008. – Вип. 6. – С. 40–44.

16. Зубов А. Ю. Криптографические методы защиты информации. Совершенные шифры : учебное пособие / А. Ю. Зубов. – М. : Гелиос АРВ, 2005. – 192 с.

17. Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях / М. А. Иванов. – М. : Кудиц-Образ, 2001. – 368 с.

18. Иванов М. А. Криптографические методы защиты информации в компьютерных системах и сетях / М. А. Иванов. – М. : Кудиц-Образ, 2001. – 368 с.

19. Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 2 : ГОСТ 7498-2-99. − [Действующий с 2000-01-01]. − М. : Госстандарт России, 1998. – 32 с.

20. Информационная технология. Криптографическая защита информации. Процедуры выработки и проверки электронной цифровой подписи на базе асимметричного криптографического алгоритма: ГОСТ Р 34.10-94. – [Действующий с 1995-01-01]. − М. : Госстандарт России, 1998. – 23 с.

21. Информационная технология. Криптографическая защита информации. Функция хеширования: ГОСТ 34.311-95. − [Действующий с 1998-04-16]. − К. : Госстандарт Украины, 1998. – 1 с.

22. Информационная технология. Криптографическая защита информации. Функция хеширования: ГОСТ Р 34.11-94. – [Действующий с 1994-05-23]. − М. : Госстандарт России, 1998. – 16 с.

23. Иохов А. Ю. Анализ основных требований к защите информации и определение аутентификации объектов в АСУВ // Збірник наукових праць ХВУ. – Х. : ХВУ, 2004. – Вип. № 13(50). – С. 48–55.

24. Иохов А. Ю. Композиционные схемы хеширования с аддитивными кодами / А. Ю. Иохов, Г. З. Халимов // VII Міжнародна науково-практична конференція. Безпека інформації в інформаційно-телекомунікаційних системах. – К. ,11–14 травня 2004 року. – С. 98.

25. Иохов А. Ю. Универсальные коды аутентификации с использованием t-связных массивов // Радиоэлектроника и молодежь в XXI веке. Материалы 8-го Международного молодежного форума. – Х. : ХНУРЭ, 2004. – Часть 1. – С. 80.

26. Иохов А. Ю. Анализ алгоритмической атаки на двухкаскадный МАС-алгоритм в системе имитозащиты АСУ войсками : збірник наукових праць ХУПС. – Х. : ХУПС, 2005. – Вип. 13(43). – С. 155–161.

27. Иохов А. Ю. Криптоанализ двухкаскадной схемы аутентификации с кодами Эрмита : збірник наукових праць ХУПС. – Х. : ХУПС, 2005. – Вип. 12(42). – С. 190–196.

28. Ковтун В. Ю. Методы и алгоритмы арифметических преобразований с уменьшенной вычислительной сложностью на алгебраических кривых для криптографических приложений : дис. на соис. кандидата технических наук: 05.13.21 / В. Ю. Ковтун. – Х., 2005. – 249 с.

29. Кузнецов А. А. Оценка свойств кривых Сузуки и их дополнений / А. А. Кузнецов // Інформаційно-керуючі системи на залізничному транспорті. – Х. : ХарДАЗТ. – № 6. – 2000. – С. 36–37.

30. Кузнецов О. О. Захист інформації та економічна безпека підприємства : монографія / О. О. Кузнецов, С. П. Євсеєв, С. В. Кавун. – Х. : Вид. ХНЕУ, 2008. – 360 с.

31. Мак-Вильямс Ф. Дж. Теория кодов, исправляющих ошибки / Ф. Дж. Мак-Вильямс, Н. Дж. А. Слоэн. – М. : Связь, 1979. – 744 с.

32. Молдовян А. А. Криптография : учебник для ВУЗов / А. А. Молдовян. – СПб. : Лань, 2000. – 224 с.

33. Осипян В. О. Криптография в задачах и упражнениях / В. О. Осипян, К. В. Осипян. – М. : Гелиос АРВ, 2004. – 144 с.

34. Основы информационной безопасности : учебное пособие для вузов / Е. Б. Белов, В. П. Лось, Р. В. Мещеряков и др. – М. : Горячая линия – Телеком, 2006. – 544 с.

35. Основы криптографии / А. П. Алферов, А. Ю. Зубов, А. С. Кузмин и др. – М. : "Гелиос Ассоциации Российских ВУЗов", 2001. – 480 с.

36. Потий А. В. Стандартизация и сертификация в сфере защиты информации. Стандарты механизмов безопасности : учебное пособие / А. В. Потий. – Х. : ХНУРЕ, 2002. – 80 с.

37. Ростовцев А. Г. Алгебраические основы криптографии / А. Г. Ростовцев. – СПб. : НПО "Мир и семья", 2000. – 354 с.

38. Ростовцев А. Г. Подпись и шифрование на эллиптической кривой: анализ безопасности и безопасная реализации. Проблемы информационной безопасности: компьютерные системы / А. Г. Ростовцев, Е. Б. Маховенко. – М. : МИФИ, 2003 – № 1 − С. 64–73.

39. Северинов А. В. Коды аутентификации сообщений на основе каскадного кодирования / А. В. Северинов, А. Ю. Иохов, В. П. Лисечко // теория и техника передачи, приема и обработки информации : тезисы докладов международной научной конференции. – Харьков-Туапсе, 7 – 10 октября 2003 г. – С. 523–524.

40. Сєвєрінов О. В. Алгоритм формування кодів перевірки дійсності на основі використання ортогональних масивів / О. В. Сєверінов, О. О. Кузнєцов, О. Ю. Іохов // ІІ наукова конференція молодих вчених : тези доповіді. Ч. 1. – ХВУ. 16–17 квітня 2003 р. – С. 95.

41. Столлингс В. Криптография и защита сетей: принципы и практика / В. Столлингс ; пер. с англ. – 2-е изд. – М. : ИД "Вильямс", 2001. – 672 с.

42. Термінологія в галузі захисту інформації в комп'ютерних системах від несанкціонованого доступу. НД СТЗІ 1.1-003-99. – [Чинний від 28.04.1999]. − К. : Держстандарт України, 1999. – 24 с.

43. Халимов Г. З. Аутентификация с применением Эрмитовых кодов / Г. З. Халимов, А. Ю. Иохов // Вестник ХПИ. – Х. : НТУ "ХПИ", 2005. – Вып. 9. – С. 26–32.

44. Халимов Г. З. Каскадное универсальное хеширование с использованием АГК кодов / Г. З. Халимов, А. Ю. Иохов // Восточно-европейский журнал передовых технологий. – Х., 2005. – Вып. 2/2(14). – С. 155–119.

45. Халимов Г. З. Криптоанализ прямой атаки на универсальное семейство геш-функций с алгебраическим кодированием / Г. З. Халимов, А. В. Северинов, А. Ю. Иохов // Збірник наукових праць ХВУ. – Х. : ХВУ, 2004. – Вып. 12(40). – С. 238–247.

46. Хорошко В. А. Методы и средства защиты информации / В. А. Хорошко, А. А. Чекатков. – К. : Юниор, 2003 – 504 с.

47. Чмора А. Л. Современная прикладная криптография / А. Л. Чмора. – М. : Гелиос АРВ, 2001. – 256 с.

48. Шипли Г. Основы безопасности ИТ // Сети и системы связи. / Г. Шипли.– М. : МИФИ, 2003 – № 4. – С. 78–82.

49. Шнаер Б. Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С / Б. Шнаер. – 2-е изд. – М. : ТРИУМФ, 2001. – 308 с.

50. ANSI X9.42-1998: Public Key Cryptography for the Financial Service Industry: Agreement of Symmetric Keys on Using Diffie-Hellman and MQV Algorithms. – 1998. – 93 p.

51. ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). – 1998. – 192 p.

52. ANSI X9.63-1999 Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography. – 1999. – 207 p.

53. Bierbrauer J. On families of hash function via geometric codes and concatenation / J. Bierbrauer, T. Johansson, G. Kabatianskii // Advances in Cryptology – CRYPTO 93. Lecture Notes in Computer Science. – 1994. – № 773. – Pp. 331–342.

54. Brassard J. Modern Criptolog / J. Brassard. – Berlin : Springer-Verlag, 1988. – 107 p.

55. Carter J. L. Universal classes of hash functions / J. L. Carter, M. N. Wegman // Computer and System Science. – 1979. – № 18. – Pp. 143–154.

56. Carter J. L. Universal classes of hash functions, J. Cornput / J. L. Carter, M. N. Wegman // System Sci. – 1979. – № 18. – Pp. 143–154.

57. CarterJ. L. N. Universal classes of hash functions / J. L. Carter, M. N. Wegman // J. ComputerandSystemSci. – 1979. – Pp. 143–154.

58.Cryptrec. Cryptrec liaison report to ISO/IEC 18033-2 and 18033-3. Technical report, ryptography Research and Evaluation Committees. – October, 2002 [Electronic resource]. – Access mode : https://www.cosic.esat.kuleuven.be/nessie/deliverables/D21-v2.pdf.

59. D. Aztec. 2-1.2 Alternatives to RSA (Lightweight Asymmetric Cryptography and Alternatives to RSA). Revision 1.2 / D. Aztec., R. Avanzi, G. Bockle, A. Breaken et al. // ECRYPT Research report IST-2002-507932. European Network of Excellence in Cryptology / R. Avanzied. – July, 27, 2005. Revised August, 17, 2005. –138 p.

60. D.VAM.1 Performance Benchmarks. Revision 1.1 / R. Avanzi, B. Chevallier-Mames etc. // ECRYPT Research report IST-2002-507932. European Network of Excellence in Cryptology / M. Joyeed. – 2005. – August, 3. – 87 p.

61. Diffi W. The First Ten Years of Public-Key Cryptography / W. Diffi / Computer Since. – 1988. – № 5. – P. 21.

62. Evolving of Boolean functions satisfying multiple criteria / Clark J.Jacob, S. Stepney, S. Maitra et al. // INDOCRYPT'02, LNCS. – 2002. – Vol. 2551. – Pp. 246–259.

63. Federal Criteria for Information Technology Security. – NIST, NSA, US Government, 1993.

64.Hoholdt T. An explicit construction of a sequence of codes attaining the Tsfasman-Vladut-Zink bound. The first steps, IEEE Trans. Info. Theory [Electronic resource]. – Access mode : http://www.win.tue.nl/~ruudp//paper/25.pdf.

65. Koblitz N. Hyperelliptic cryptosystems / N. Koblitz // Journal of cryptology. – 1989. – № 1. – Pp. 139–150.

66. Krawczyk H. LFSB-based Hashing and Authenticator / H. Krawczyk / Proceedings of CRYPTO Notes in Computer Science. – 1994. – № 80 – Pp. 129–139.

67. Maier W. Nonlinearity criteria for cryptographic functions / W. Maier, O. Staffelbach // Advances in Cryptology. – EUROCRYPT'89. – 1990. – Vol. 434. – Pp. 549–562.

68. Maitra S. Further constructions of resilient Boolean functions with very high nonlinearity / S. Maitra, E. Pasalic // Accepted in SETA. – May, 2001.

69. Matthews R. The use of genetic algorithms in cryptanalysts / R. Matthews // Cryptologia. –1993. – № 17(2). – Pp. 187–201.

70. McElliece R. J. A public-key cryptosystem based on algebraic coding theory / R. J. McElliece. – NY : Springer-Verlag, 1978. – 116 p.

71. Millan W. An effective genetic algorithm for finding highly nonlinear Boolean functions / W. Millan, A. Clark, E. Dawson // First International Conference on Information and Communications Security. – 1997. – № 1334. – Pp. 149–158.

72. Millan W. Boolean function design using hill climbing methods / W. Millan, A. Clark, E. Dawson // 4th Australasian Conference on Information, Security and Privacy. – 1999. – Num. 1587. – Pp. 1–11.

73. Millan W. Smart Hill Climbing Finds Better Boolean Functions / W. Millan, A. Clark, E. Dawson // Workshop on Selected Areas in Cryptography. – 1997. – P. 50.

74. Oppliger R. Contemporary Cryptography / R. Oppliger. – Artech House Publishers, 2005. – 510 p.

75. Pasalic E. Further Results on the Relation Between Nonlinearity and Resiliency for BF / E. Pasalic, T. Johansson // IEEE Trans. on Information Theory July 2002. – Vol. 48. – No. 7. – Pp. 1825–1834.

76. Pieprzyk J. P. On public-key cryptosystems built using polynomial rings / J. P. Pieprzyk // Advanced in Cryptography-Eurocrypt. – NY : Springer-Verlag, 1985. – 80 p.

77. Plackett R. L. The design of optimum multifactorial experiments / R. L. Plackett and J. P. Burman // Biometrika. – 1945. – № 33. – Pp. 305–325.

78. Preneel B. "New European Schemes for Signature, Integrity and Encryption" Final report of European project number IST-1999-12324, NESSIE / B. Preneel, A. Biryukov. –2004. – April. – Pp. 487–623

79. Preneel B. MDx-MAC and building fast MACs from hash functions / B. Preneel // Proceedings of Crypto'95 (D. Coppersmith, ed.). – № 963. Lecture Notes in Computer Science. – Berlin : Springer-Verlag, 1995. – 152 p.

80. Sarwate D. V. A note on universal classes of hash functions / D. V. Sarwate // Inform. Proc. Letters. – 1980. – № 10. – Pp. 41–45.

81. Seberry J. Hadamar Matrices, Bent Functions and Cryptography / J. Seberry, X. Zhang // Contemporary Design Theory : a Collection of Surveys / ed. J. H. Dinitz, D. R. Stinson. – 1995. – Chapter 11. – Pp. 431–559.

82. Shanks D. Class number, a theory of factorization and genera / D. Shanks // Proc. Symposium Pure Mathematics. – 1970. – Vol. 20. – Pp. 415–440.

83. Simmons G. J. An impersonation-proof identity verification scheme / G. J. Simmons // Computer Science. – 1988. – № 87. – Pp. 211–215.

84. Simmons G. J. Authentication theory/coding theory in Cryptology / G. J. Simmons / Computer Science. – 1985. – № 96. – Pp. 411–431.

85. Smid M. E. The Past and Future / M. E. Smid, D. K. Branstad / Computer Science. – 1988. – № 76. – Pp. 122–132.

86. Stinson D. R. Some constructions and bounds for authentication codes / D. R. Stinson // J. Cryptology. – 1988. – № 1. – Pp. 37–51.

87. Stinson D. R. The combinatorics of authentication and secrecy codes / D. R. Stinson // J. Cryptology. – 1990. – № 2. – Pp. 23–49.

88. UMAC: Fast and secure message authentication. // Advances in Cryptology 'CRYPTO '99 (1999) / J. Black, S. Halevi, H. Krawczyk at al. – Berlin : Springer-Verlag, 1999. – Pp. 216–233.

89. Use of a Genetic Algorithm in the Cryptanalysis of Simple Substitution Ciphers / R. Spillman, M. Janssen, B. Nelson, at al. // Cryptologia. – 1993. – № 17(1). – Pp. 31–44.

90. Wegman M. N. New hash functions and their use in authentication and set equality / M. N. Wegman, J. L. Carter / Computer and System Science. – 1981. – № 22. – Pp. 265–279.

91. Weimerskirch A. Generic arithmetic in software and its application to ECC ECC / A. Weimerskirch, D. Stebila, Sh. Chang Shantz // The 8th Australasian conference on information security and privacy (9–11, July, 2003). – ACISP`2003. – Australia : Wollongong, 2003. – 14 p.

92. Жуков А. Е. Криптоанализ по побочным каналам (Side Channel Attacks) [Электронный ресурс] / А. Е. Жуков. – Режим доступа : http://www.ruscrypto.ru/sources/publications.

93. Жуков А. Е. Криптография и клептография. Криптосистемы со встроенными лазейками [Электронный ресурс] / А. Е. Жуков. – Режим доступа : http://www.ruscrypto.ru/netcat_files/File/seminar.005.zip.

94. Программное средство "Библиотека функций криптографической защиты информации "Грифон-Л" [Электронный ресурс]. – Режим доступа : http://www.banksoft.com.ua/index.php?id=27.

95. Программное средство криптографической защиты информации "Грифон-Б" [Электронный ресурс]. – Режим доступа : http://www.banksoft.com.ua/index.php?id=28.

96. Ростовцев А. Г. Методы криптоанализа классических шифров [Электронный ресурс] / А. Г. Ростовцев. – Режим доступа : http://crypto.hotbox.ru/.

97. Украинский ресурс по безопасности [Электронный ресурс]. – Режим доступа : http://kiev-security.org.ua.

98. Цифровая подпись НОТАРИУС [Электронный ресурс]. – Режим доступа : http://www.lancrypto.com/products/.

99. Bierbrauer J. Authentication via algebraic-geometric codes [Electronic resource]. – Access mode : http://www.math.mtu.edu/~jbierbra/ potpap.ps.

100. Bierbrauer J. Universal hashing and geometric codes [Electronic resource]. – Access mode : http://www.math.mtu.edu/~jbierbra/ hashco1.ps.

101. Black J. Ciphers with arbitrary finite domains / J. Black // Proceedings of CT-RSA’02 (B. Preneel, ed.), no. 2271 in Lecture Notes in Computer Science, pp. 114–130, Springer-Verlag, 2002 [Electronic resource]. – Access mode : www.cs.ucdavis.edu/~rogaway/papers/subset.htm.

102. Brown D. R. L. Generic groups, collision resistance, and ECDSA [Electronic resource]. – Access mode : http://eprint.iacr.org/2002/026/2002.

103. Daemen J. AES Proposal: Rijndael, AES Algorithm Submission / J. Daemen, V. Rijmen. – September, 3, 1999 [Electronic resource]. – Access mode : http://www.docstoc.com/docs/14641406/AES-Implementation-and-Performance-Evaluation-on-8-bit-Microcontrollers.

104. Fast hashing on the Pentium // Advances in Cryptology ’CRYPTO ’96 (1996) / A. Bosselaers, R. Govaerts, J. Vandewalle ; vol. 1109 of Lecture Notes in Computer Science, Springer-Verlag, pp. 298–312 [Electronic resource]. – Access mode : http://www.esat.kuleuven.ac.be/ bosselae/fast.html.

105. IEEE 1363-2000 "Standard Specifications for Public-Key Cryptography". – August, 2000 [Electronic resource]. – Access mode : http://standards.ieee.org/catalog/olis/busarch.

106. IEEE P1363 working group "Standard Specifications for Public-Key Cryptography" [Electronic resource]. – Access mode : http://grouper.ieee.org/.

107. ISO/IEC 10181-1:1996 – Information technology – Open Systems Interconnection – Security frameworks for open systems: Overview [Electronic resource]. – Access mode : http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=18199.

108. ISO/IEC 10181-3:1996 – Information technology – Open Systems Interconnection – Security frameworks for open systems: Access control framework [Electronic resource]. – Access mode : http://www.iso.org/iso/iso_catalogue_detail.htm?csnumber=18199.

109. ISO/IEC 10181-5:1996 – Information technology – Open Systems Interconnection – Security frameworks for open systems: Confidentially frameworks [Electronic resource]. – Access mode : http://www.iso.org/iso/iso_catalogue_detail.htm?csnumber=18199.

110. ISO/IEC 14888-1:2008 – Information technology – Security techniques – Digital signatures with appendix – [Electronic resource]. – Access mode : http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=44226.

111. ISO/IEC 14888-2:2008 – Information technology – Security techniques – Digital signatures with appendix – Part 2: Integer factorization based mechanism [Electronic resource]. – Access mode : http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=44227.

112. ISO/IEC 14888-3:2006 – Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanism [Electronic resource]. – Access mode : http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=43656.

113. ISO/IEC 15408-1:1999 – Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model [Electronic resource]. – Access mode : http://www.iso.org/iso/catalogue_detail.htm?csnumber=27632.

114. ISO/IEC 15408-3:1999 – Information technology – Security techniques – Evaluation criteria for IT security – Part 3: Security assurance requirements [Electronic resource]. – Access mode : http://www.iso.org/iso/catalogue_detail.htm?csnumber=40614.

115. ISO/IEC 7498-2:1989 – Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 2: Security Architecture [Electronic resource]. – Access mode : http://www.iso.org/iso/catalogue_detail.htm?csnumber=14256.

116. ISO/IEC 9796-1:2010 – Information technology – Security techniques – Digital Signatures schemes giving message recovery – Part 1:Integer factorization based mechanisms [Electronic resource]. – Access mode : http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=54788.

117. ISO/IEC 9796-2:2010 – Information technology – Security techniques – Digital Signatures schemes giving message recovery – Part 2: Discrete logarithm based mechanisms [Electronic resource]. – Access mode : http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=42228.

118. ISO/IEC 9797-1:1999 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanism using a block cipher [Electronic resource]. – Access mode : http://www.standards.ru/print.aspx?control=27&id=3632410&print=yes.

119. ISO/IEC 9797-2:2002 – Information technology – Security techniques – Message Authentication Codes (MACs) – Part 2: Mechanism using a dedicated hash-function [Electronic resource]. – Access mode : http://www.iso.org/iso/catalogue_detail.htm?csnumber=51618.

120. Knudsen L. R. Block Ciphers – Analysis, Design, Applicatons : Ph.D. dissertation, Aarhus Unversity, Nov. 1994 [Elecrtronic resource]. – Access mode : http://www.selmer.uib.no/researchcourse2004/.

121. Lange T. Efficient arithmetic on genus 2 hyperelliptic curves over finite fields via explicit formulae / T. Lange // Cryptology ePrint Archive. – Report 2002/121. – 2002. – 13 p. [Electronic resource]. – Access mode : http://eprint.iacr.org.

122. Lange E. Inversion-free arithmetic on genus 2 hyperelliptic curves / E. Lange // Cryptology ePrint Archive. – Report 2002/147. – 2002. – 7 p. [Electronic resource]. – Access mode : http://eprint.iacr.org.

123. Lee A. NIST Special Publication 800-21. Guideline for Implementing Cryptography in the Federal Government, National Institute of Standards and Technology / A. Lee. – November, 1999 [Electronic resource]. – Access mode : http://csrc.nist.gov/publications/nistpubs/800-21-1/sp800-21-1_Dec2005.pdf.

124. NESSIE consortium "NESSIE Security report." Deliverable report D20 – NESSIE, 2002. – NES/DOC/ENS/WP5/D20 [Electronic resource]. – Access mode : http://www.cryptonessie.org/.

Зміст

НАУКОВЕ ВИДАННЯ

Євсеєв Сергій Петрович
Йохов Олександр Юрійович
Король Ольга Григорівна

ГЕШУВАННЯ ДАНИХ В ІНФОРМАЦІЙНИХ СИСТЕМАХ

Монографія

Відповідальний за випуск **Пономаренко В. С.**

Відповідальний редактор **Сєдова Л. М.**

Редактор **Пушкар І. П.**

Коректор **Мартовицька-Максимова В. А.**

План 2013 р. Поз. № 7-Н.
Підп. до друку Формат 60 х 90 1/16. Папір MultiCopy. Друк Riso.
Ум.-друк. арк. 19,5. Обл.-вид. арк. 24,38. Тираж прим. Зам. №

Видавець і виготівник – видавництво ХНЕУ, 61166, м. Харків, пр. Леніна, 9а

Свідоцтво про внесення до Державного реєстру суб'єктів видавничої справи ***Дк № 481 від 13.06.2001 р.***