Тема 11. ХЕШ-ФУНКЦІЯ

1. Односторонні функції і функції з лазівками

Стійкість асиметричної криптосистеми забезпечується за рахунок особливих властивостей шифрів – перетворення, яке являє собою так звану односторонню функцію з «лазівкою». Обчислення значення такої функції (від відкритого тексту і відкритого ключа) повинно бути нескладним. У той же час, її звернення повинно бути обчислювально нереалізованим без знання секретної інформації, «лазівки», пов'язаної з секретним ключем.

Строго кажучи, не доведено, що односторонні функції існують. Однак визнано, що деякі перетворення мають властивості, близькі до властивостей односторонніх функцій. Вони широко використовуються в діючих системах криптографічного захисту інформації (ще один аргумент на користь актуальності проблеми надійності криптосистем.)

Функція $f(x) = a^x \bmod p$, при великих значеннях x і $ord_n a$, веде себе як одностороння. Обернена функція (дискретний логарифм) обчислювально нереалізовувана і завдання дискретного логарифмування є алгоритмічної проблемою.

Аналогічними властивостями володіє і степенева функція виду $g(x) = x^e \bmod n$, де $n = pq$. Для зворотної дії цієї функції досить вирішувати завдання розкладання числа n на множники, однак це завдання також є алгоритмічної проблемою.

Яким чином проблема безпечного розповсюдження ключів вирішується за допомогою асиметричних криптосистем?

Для цього кожен бажаючий передати ключ для симетричної криптосистеми своєму абоненту, перезашифровує його ключем e цього абонента (вважається, що асиметрична система створена заздалегідь і відкритий ключ опублікований). Результат шифрування передається по відкритому каналу.

Одностороння функція гарантує безпеку, тому що розшифрувати повідомлення можна лише знаючи ключ d, а його знає лише потрібний абонент.

Загальновідомо, що даний механізм все одно не є безпечним. Справа ускладнюється настільки, що на практиці виявилося необхідним вводити в глобальному масштабі систему так званих центрів сертифікації відкритих ключів.

Центр сертифікації грає роль довіреної особи, яка гарантує, що повідомлення, зашифроване даними відкритим ключем, зможе розшифрувати тільки абонент, для якого це повідомлення призначалося.

При подальшому розвитку, ідея використання односторонніх функцій в криптографії дозволила вирішити ряд проблем, пов'язаних із захистом інформації. [14]

2. Загальні поняття хеш-функцій

Часто виникають ситуації, коли одержувач повинен вміти довести достовірність повідомлення зовнішній особі. Щоб мати таку можливість, до передавальних повідомлень мають бути приписані так звані цифрові сигнатури.

Цифрова сигнатура – це рядок символів, який залежить як від індентифікатора відправника, так і від змісту повідомлення.

Використання цифрової сигнатури передбачає застосування деяких функцій шифрування:

$$S=H(k,T),$$

де S – сигнатура, k – ключ, T – вихідний текст. Функція $H(k,T)$ – хеш – функція.

Хешування – перетворення вхідного масиву даних будь-якої довжини у вихідний бітовий рядок фіксованої довжини. Такі перетворення також називаються *хеш-функціями* або *функціями згортки*, а їх результати називають *хешем, хеш-кодом* або *дайджестом повідомлення.*

По іншому, *алгоритм хешування* – це послідовність математичних перетворень, в результаті яких з деякої двійкової послідовності змінної

довжини отримується унікальна двійкова послідовність фіксованої довжини. Функція, що реалізовує даний алгоритм називається *хеш-функцією.*

Хешування застосовується для порівняння даних: якщо в двох масивах хеш-коди різні, то масиви гарантовано відрізняються; якщо однакові – масиви, швидше всього, однакові. В загальному випадку однозначної відповідності між вихідними даними та хеш-кодом немає, тому що кількість значень хеш-функцій менше, ніж варіантів вхідного масиву; існує множина масивів, що дають однакові хеш-коди – так звані *колізії*. Ймовірність виникнення колізій грає важливу роль в оцінці якості хеш-функцій.

Існує багато алгоритмів хешування з різними характеристиками (розрядність, обчислювальна складність, криптостійкість і т.д.). Вибір тої чи іншої хеш-функції визначається специфікою вирішуваної задачі. Найпростішим прикладом хеш-функції є контрольна сума.

Для того, щоб перетворення даних можна було назвати хеш-функцією, воно повинно одночасно задовольняти наступні умови:

- обробляти послідовність різної довжини;
- видавати у якості результату бітову послідовність фіксованої довжини;
- достатньо просто обчислюватися при будь-якій вхідній інформації;
- володіти властивістю незворотності, коли для отриманого результату неможливо отримати вхідне значення;
- бути однозначним (тобто для різних вхідних послідовностей не повинно бути побудовано однакові хеш-образи).

З криптографічної точки зору, останні дві властивості хеш-функції найбільш важливі, так як забезпечують неспівпадання хешованих форм, наприклад паролів різних користувачів і ускладнюють задачу розкриття цих паролів. [11, 12, 13, 14]

3. Вимоги до хеш-функцій

При практичному використанні хеш-функцій повинні виконуватися такі вимоги:

- алгоритм повинен володіти високою швидкістю обробки інформації (це особливо актуально для банківських операцій, де необхідна особлива оперативність обробки інформації);
- хеш-функція повинна бути стійкою до атаки методом “грубої сили”;
- програмна реалізація хеш-функції повинна бути оптимізована під використання на сучасній апаратно-програмній базі.

Ці вимоги повинен задовольняти як сам алгоритм вироблення хеш-функції, так і хеш-функція.

В сучасних умовах алгоритмічне підвищення швидкості вироблення хеш-значення може бути досягнуто за рахунок застосування простого перетворення, яке переводить одне повідомлення в інше за допомогою елементарної операції, наприклад видалення будь-якого блоку повідомлення. Подібними перетвореннями можна також описати залежність між двома повідомленнями, що практично не відрізняються одне від одного. Даний тип повідомлення дуже часто зустрічається у банківських справах, наприклад з ціллю заповнення бланків платіжних доручень. Звідси слідує, що для збільшення швидкості обробки необхідно, щоб алгоритм вироблення хеш-значення включав у себе також алгоритм обчислення хеш-значення одного повідомлення із хеш-значення другого повідомлення, яке отримується із початкового з допомогою елементарного перетворення.

4. Криптографічні хеш-функції. Поділ хеш-функцій

Серед багатьох існуючих хеш-функцій прийнято виділяти криптографічно стійкі, які застосовуються в криптографії. Для того, щоб хеш-функція H вважалася криптографічно стійкою, вона повинна задовольняти три основні вимоги, на яких основана більшість застосувань хеш-функції у криптографії:

- *незворотність:* для заданого значення хеш-функції M повинно бути обчислювально неможливо знайти блок даних X, для якого $H(X)=M$.

- *стійкість до колізій першого роду:* для заданого повідомлення M повинно бути обчислювально неможливо підібрати інше повідомлення *N*, для якого *H*(*N*) = *H*(*M*).
- *стійкість до колізій другого роду:* повинно бути обчислювально неможливо підібрати пару повідомлень *(M, M')*, які мають однаковий хеш-код.

Дані вимоги не є незалежними:

- зворотна функція нестійка до колізій першого і другого роду;
- функція, нестійка до колізій першого роду, нестійка до колізій другого роду; зворотне не вірно.

Слід відмітити, що не доведено існування незворотних хеш-функцій, для яких обчислення якого-небудь прообразу заданого значення хеш-функції теоретично неможливо. Зазвичай знаходження зворотного значення є лиш обчислювально складною задачею.

Для криптографічних хеш-функцій також важливо, щоб при найменшій зміні аргемента значення функції змінювалося якнайбільше. Значення хешу не повинно давати витоку інформації навіть про окремі біти аргументу. Ця вимога є основою криптостійкості алгоритмів хешування.

Основний принцип проектування хеш-функцій полягає в тому, що її значення повинні створювати лавинний ефект. Іншими словами, невелика зміна в аргументі хеш-функції повинна дуже впливати на її значення. Це забезпечує додаткову стійкість.

Щоб значення хеш-функції, що застосовується в системах з низьким рівнем стійкості, були вільні від повторень, їх довжина повинна бути приблизно рівна 128 бітам, але краще надавати перевагу значенню в 160 біт.

Хеш-функції	
Хеш-функції загального призначення	Adler-32 • CRC • FNV • Murmur2 • PJW-32 • TTH • Jenkins hash
Криптографічні хеш-функції	HAVAL • Keccak • LM-хеш • MD2 •

	MD4 • MD5 • MD6 • N-Hash • RIPEMD-128 • RIPEMD-160 • RIPEMD-256 • RIPEMD-320 • SHA-1 • SHA-2 • LanMan • NT LanMan • MySQL 3.23 • MySQL SHA1, Cisco PIX • Skein • Snefru • Tiger • Whirlpool • ГОСТ Р 34.11-94

Крім зазначеного поділу хеш-функцій, що у таблиці, розрізняють також ключові функції хешування та без ключові.

За внутрішнім перетворенням, що використовується у хеш-функції, їх поділяють на:

- функції, що використовують бітові логічні перетворення;
- функції, що використовують блочні симетричні шифри;
- функції, що використовують перетворення в групах, полях, кільцях з цілочисловим або поліноміальним базисом;
- функції, що використовують матричні перетворення. [11, 12, 13, 14]

5. Застосування хеш-функцій. Способи взлому

З хешуванням ми зустрічаємося на кожному кроці: при роботі з браузером (список web-посилань), текстовим редактором чи перекладачем (словники), мовами скриптів (Perl, PHP, Python та ін.), компілятором (таблиця символів).

Хеш-функції також використовуються у деяких структурах даних – хеш-таблицях, фільтрах Блума і декартових деревах. Вимоги до хеш-функцій у цьому випадку інші:

- хороша змішуваність даних;
- швидкий алгоритм обчислення.

Перевірка даних

В загальному випадку це застосування можна описати як перевірка деякої інформації на ідентичність оригіналу, без використання оригіналу. Для звірення

використовується хеш-значення інформації, що перевіряється. Розрізняють два основні напрями цього застосування:

Перевірка даних на помилки

- наприклад, контрольна сума може бути передана по каналу зв'язку разом з основним текстом. На пункті прийому, контрольна сума може бути розрахована заново і її можна порівняти з переданим значенням. Якщо буде виявлено розходження, це означає, що при передачі виникли спотворення і можна запросити повторно;
- бітовим аналогом хешування в даному випадку може слугувати спосіб, коли при переїздах в пам'яті тримають кількість місць багажу. Тоді для перевірки не потрібно згадувати про кожен чемодан, а досить їх порахувати. Співпадання буде означати, що жоден чемодан не загубився. Тобто, кількість місць багажу є його хеш-кодом.

Пришвидшення пошуку даних

- наприклад, при записі текстових полів у базі даних може розраховуватися їх хеш-код і дані можуть поміщатися в розділ, що відповідає цьому хеш-коду. Тоді при пошуку даних потрібно буде спочатку обчислити хеш-код тексту і зразу стане відомо, в якому розділі їх потрібно шукати, тобто, шукати не по всій базі, а тільки по одному розділу (помітно пришвидшує пошук);
- бітовим аналогом хешування в даному випадку може бути розміщення слів у словнику по алфавіту. Перша буква слова є його хеш-кодом, і при пошуку ми переглядаємо не весь словник, а тільки потрібну букву.

Також хеш-функції використовуються при обробці інформації для створення цифрового підпису.

Оскільки, застосування хеш-функцій набуло розмаху, то почали з'являтися способи взлому хеш-кодів. Єдиним відомим вразливим їх місцем є колізії, тому пошуком колізій і займаються програми для взлому хеш-кодів.

Перший варіант – це простий перебір, при якому виконується проходження по всіх можливих варіантах. Але для складних варіантів це займає дуже багато часу, тому такий спосіб вирішили допрацювати.

Тепер для розкриття паролів, перетворених за допомогою хеш-функції, використовують райдужні таблиці.

Райдужна таблиця (від англ. rainbow table) – спеціальний варіант таблиць пошуку (lookup table), що використовує механізм зменшення пам'яті, яка займається, за рахунок збільшення часу пошуку. Але таблиці можуть взломувати тільки ту хеш-функцію, для якої вони створювались.

Вперше цей метод був використаний у програмі Ophcrack для взлому хешів LanMan, що використовуються у Microsoft Windows. Пізніше була розроблена більш вдосконалена програма RainbowCrack, яка може взломувати хеші за деякими алгоритмами (LanMan, NT LanMan, MD5, MD4, MD2, SHA1, MySQL 3.23, MySQL SHA1, Cisco PIX, RIPEMD-160 та ін.). Необхідність у великих таблицях для взлому складних хешів привела до появи проектів розподіленого обчислення радужних таблиць та online-взломів хешів, оскільки обчислення таблиць такого розміру на одній машині не можливе. Сучасні програми містять порядку 1 Тб таблиць для кожного алгоритму.

На даний момент йде розробка програми The UDC, яка дозволяє будувати таблиці не по наборі символів, а по наборі словників, що дозволяє розкривати більш довгі паролі, ніж RainbowCrac.

Для захисту від райдужних таблиць використовують метод, який робить їх неефективними. Для цього використовують хеш-функції, які включають сіль (salt). Тому для розкриття пароля, взломщику необхідні таблиці для всіх значень солі. Таким чином, збільшується довжина і складність пароля та частково попереджається його розкриття. [11, 12, 13, 14]

6. Генератори превдовипадкових чисел

Призначені для отримання числових послідовностей у яких розподіл вибірок елементів поводить себе як аналогічні вибірки із сукупності з рівноймовірним і незалежним розподілом ймовірностей. Такі послідовності отримуються за допомогою математичних алгоритмів зі скінченним числом параметрів. Тому не кожний спосіб вибору елементів числової послідовності дає сукупність чисел з бажаними характеристиками. При побудові генераторів

псевдовипадкових чисел висуваються такі необхідні вимоги до вибору елементів для яких статистичні властивості відповідають властивостям рівноймовірності і незалежності. Тобто програмний генератор псевдовипадкових чисел має задовольняти наступним вимогам:

- період гами має бути досить великим;
- гама має бути практично непередбачуваною;
- гама має бути відтворюваною

Період гами – це та кількість псевдовипадкових чисел у послідовності, після якої вони починають повторюватися. Чим більший період гами, тим для довших відкритих текстів її можна застосовувати. Чим менший період гами, тим легше передбачувати числа в ній і зламувати ключі та шифри. Довжина періоду гами залежить від вибраного алгоритму генерування послідовності псевдовипадкових чисел.

Непередбачуваність гами означає неможливість передбачити наступне число гами, навіть тоді, коли відомі тип генератора і попередній фрагмент гами. Загальних методів визначення рівня непередбачуваності гами не існує. Але вважається, що рівень непередбачуваності буде достатнім, якщо гама має дуже великий період. Крім того, різні комбінації сусідніх чисел гами мають бути рівномірно розташовані вздовж неї.

Відтворюваність гами означає наявність можливості отримати ту ж саму послідовність псевдовипадкових чисел. Це потрібно для того, щоб можна було дешифрувати криптограму, отриману шляхом застосування гами шифру. Для цього застосовують такі програмні генератори псевдовипадкових чисел, в яких та чи інша їх послідовність залежить від, так званого, ключа генератора. Це означає, що для різних значень ключа генератора мають генеруватись різні послідовності. Але при одному й тому ж значені ключа має генеруватись одна й таж сама послідовність. [8]

В криптографії застосовуються так званні *криптографічно стійкі датчики (КСД)* – так називають генератори що використовують секретні параметри. Для таких генераторів потрібна властивість непедбачуваності

(відрізок вихідної послідовності відносно великої довжини не може бути продовжено як вперед так і назад без знака ключа.

Приклади КСД:

1. Генератор рекомендований стандартом *ANSI X9.17*, що використовується частково при виконанні платіжних операцій.

2. В українському стандарті на цифровий підпис *ДСТУ 4145-2002* генератор випадкових двійкових послідовностей побудований за схемою ПСЧ *ANSI X9.17* з використанням криптоалгоритму *ГОСТ 28147-89*.

3. Генератор BBS *(Algorithm Blum-Blum-Shub)* – генератор псевдовипадкових чисел, запропонований в 1986 році Ленор Блюмом, Мануелом Блюмом, Майклом Шубом. Праметром генератора є просте число $n=pq$, де співмножники великі псевдовипадкові прості числа обнакового розміру, при чому кожний співмножник порівнюваний з $3 \bmod 4$.