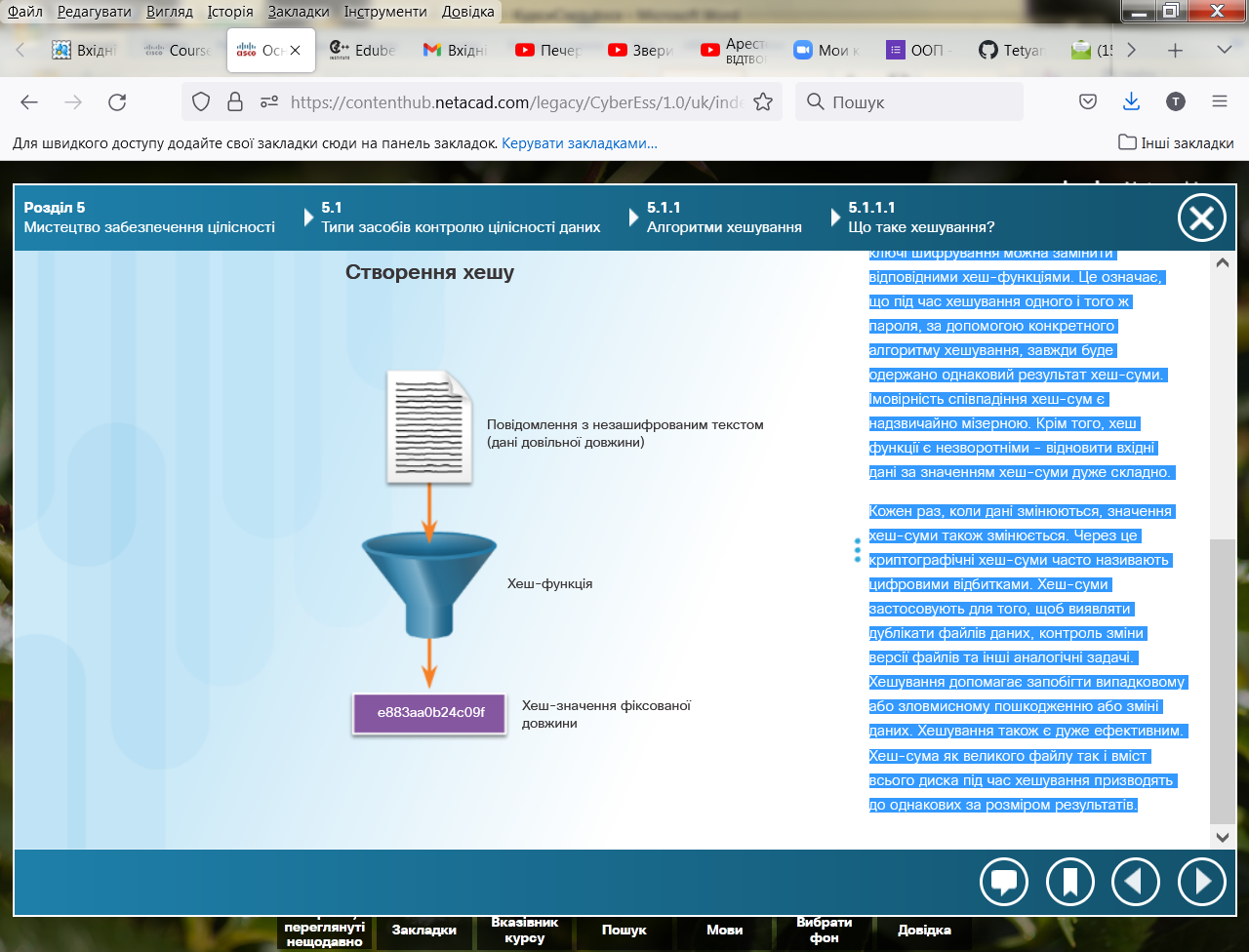
**Розділ 5. Мистецтво забезпечення цілісності даних**

Забезпечення цілісністі гарантує, що дані залишаються незмінними і заслуговують довіри весь свій життєвий цикл. Цілісність даних є важливим компонентом для проектування, реалізації та використання будь-якої системи, яка зберігає, обробляє або передає дані. Розділ починається з опису різноманітних засобів контролю цілісності даних, зокрема алгоритми хешування, додавання солі (salting) та аутентифікації повідомлень з використанням ключа та хеш-суми (HMAC). Далі розглядається використання цифрових підписів і сертифікатів за допомогою яких перевіряється автентичність повідомлень та документів. В кінці розділу обговорюється забезпечення цілісності баз даних. Наявність добре контрольованої і чітко структурованої системи цілісності даних підвищує стабільність, продуктивність і зручність в обслуговуванні систем баз даних.

**Що таке хешування?**

В багатьох випадках необхідний механізм, який гарантує, що дані залишаються незмінними під час їх зберігання або передаванні. Хешування - це інструмент, який забезпечує цілісність даних, використовуючи вхідні бінарні дані (повідомлення) і створюючи з них вихідні дані фіксованої довжини, які називають хеш-значенням або дайджестом, як показано на рисунку.

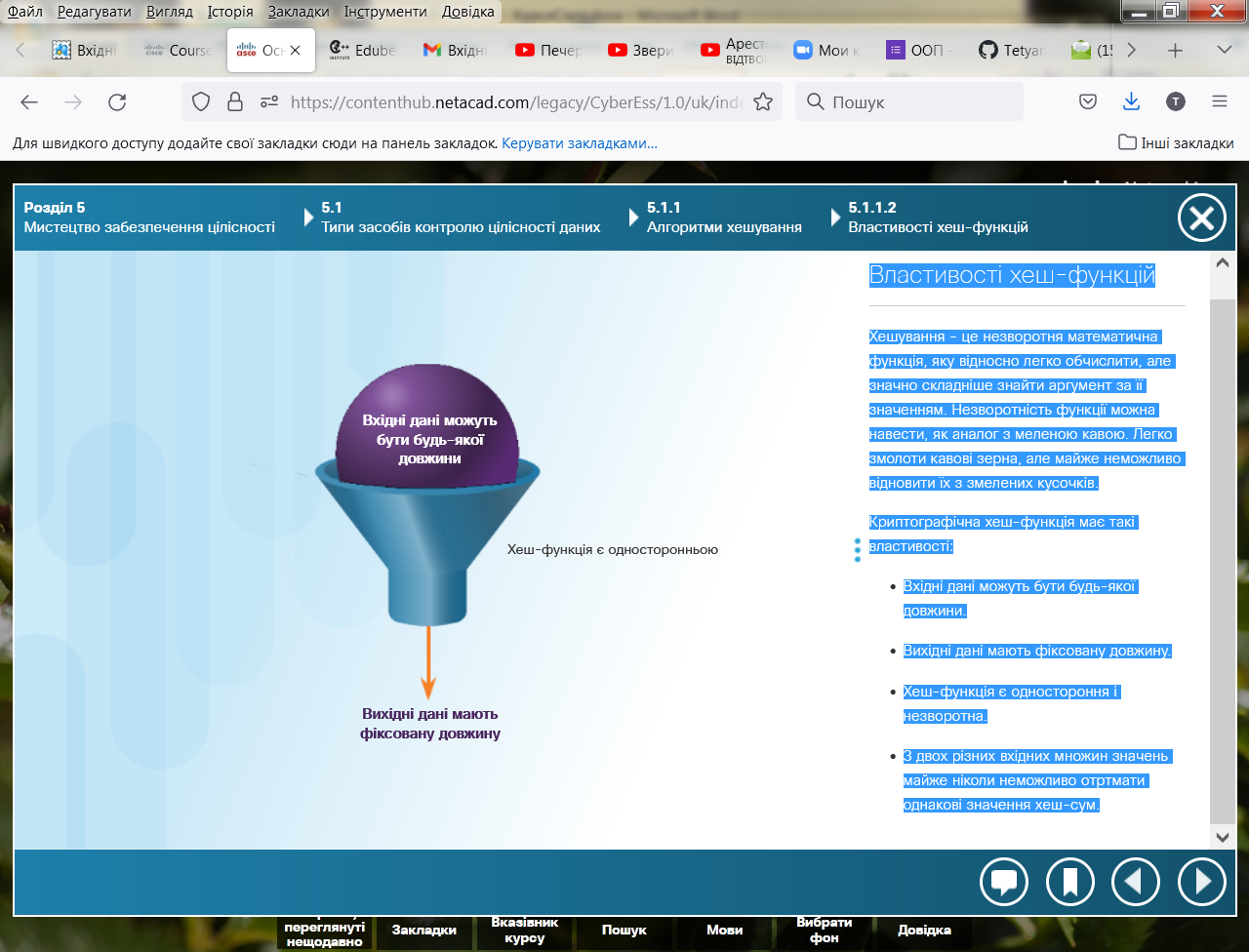
У системах хешування застосовуються криптографічно стійкі хеш-функції, які і забезпечують перевірку та гарантію цілісності даних. Вони також можуть застосовуватись для перевірки автентифікації. Незашифровані паролі та ключі шифрування можна замінити відповідними хеш-функціями. Це означає, що під час хешування одного і того ж пароля, за допомогою конкретного алгоритму хешування, завжди буде одержано однаковий результат хеш-суми. Імовірність співпадіння хеш-сум є надзвичайно мізерною. Крім того, хеш функції є незворотніми - відновити вхідні дані за значенням хеш-суми дуже складно.

Кожен раз, коли дані змінюються, значення хеш-суми також змінюється. Через це криптографічні хеш-суми часто називають цифровими відбитками. Хеш-суми застосовують для того, щоб виявляти дублікати файлів даних, контроль зміни версії файлів та інші аналогічні задачі. Хешування допомагає запобігти випадковому або зловмисному пошкодженню або зміні даних. Хешування також є дуже ефективним. Хеш-сума як великого файлу так і вміст всього диска під час хешування призводять до однакових за розміром результатів.

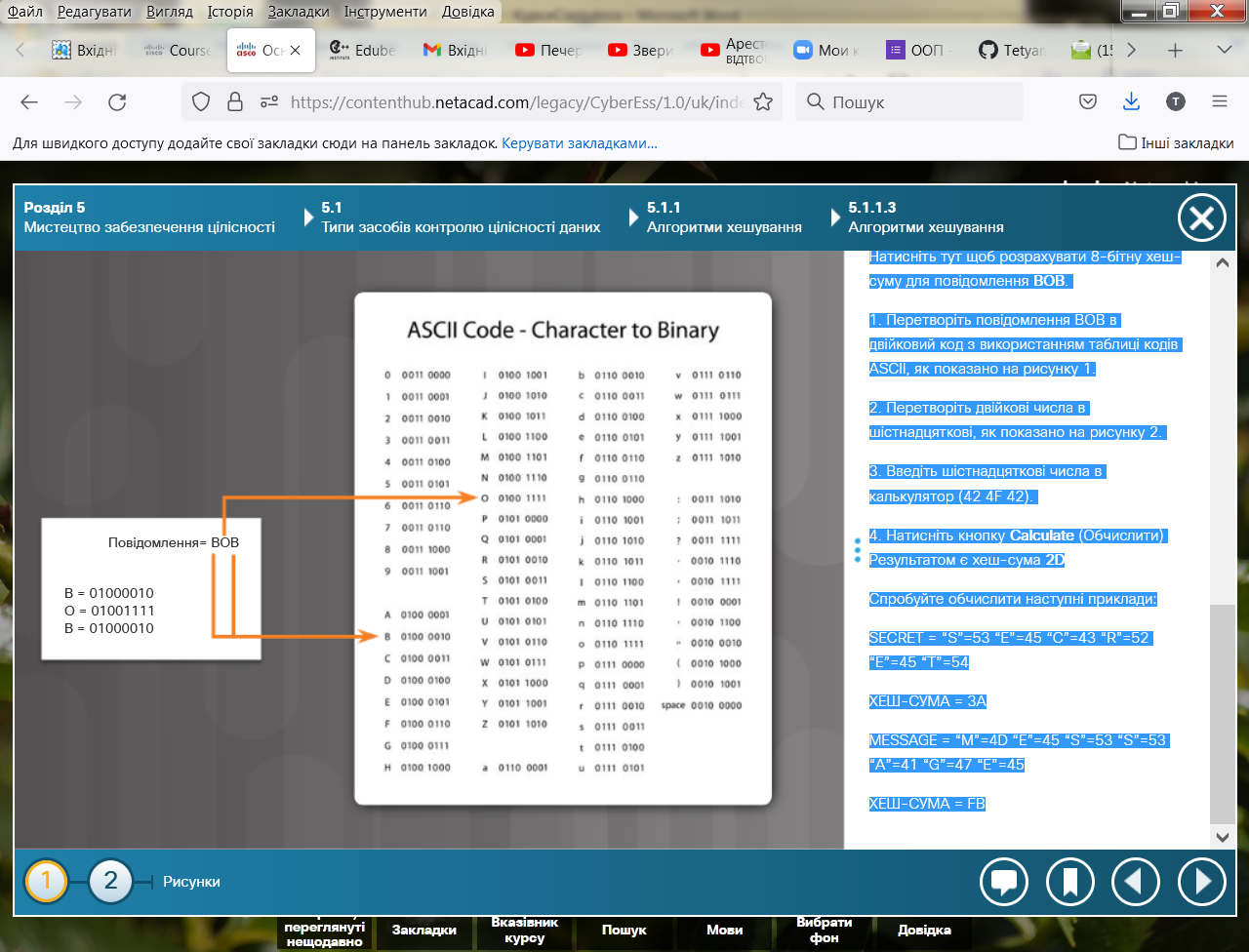
**Властивості хеш-функцій**

Хешування - це незворотня математична функція, яку відносно легко обчислити, але значно складніше знайти аргумент за її значенням. Незворотність функції можна навести, як аналог з меленою кавою. Легко змолоти кавові зерна, але майже неможливо відновити їх з змелених кусочків.

Криптографічна хеш-функція має такі властивості:

* Вхідні дані можуть бути будь-якої довжини.
* Вихідні дані мають фіксовану довжину.
* Хеш-функція є одностороння і незворотна.
* З двох різних вхідних множин значень майже ніколи неможливо отртмати однакові значення хеш-сум.

# Алгоритми хешування

Хеш-функції дозволяють визначити, що дані змінились через помилку користувача або через збій в процесі передавання. Наприклад, відправник може переконатися, що повідомлення дійшло до одержувача без змін. Для цього передавальний пристрій обчислює хеш-суму повідомлення (відбиток фіксованої довжини).

**Простий алгоритм хешування (8-бітна контрольна сума)**

8-бітна контрольна сума є одним з перших алгоритмів хешування. Це є найпростіша форма хеш-функції. Під час розрахунку хеш-суми за цим алгоритмом 8-розрядна контрольна сума обчислює хеш, перетворюючи повідомлення в двійкові числа, а потім організовуючи рядок двійкових чисел в 8-бітові набори. Далі 8-бітові значення сумуються. Останній крок - перетворити результат у доповняльний код. Перетворення у доповняльний код відбувається наступним чином: розряди двійкового числа інвертуються, після чого до до числа додається одиниця. Це означає, що нулі перетворюється в одиниці, а одиниці перетворюються в нулі. Останній крок - додати 1. Таким чином отримується 8-бітна хеш-сума.

Натисніть [тут](http://easyonlineconverter.com/converters/checksum_converter.html) щоб розрахувати 8-бітну хеш-суму для повідомлення **BOB**.

1. Перетворіть повідомлення BOB в двійковий код з використанням таблиці кодів ASCII, як показано на рисунку 1.

2. Перетворіть двійкові числа в шістнадцяткові, як показано на рисунку 2.

3. Введіть шістнадцяткові числа в калькулятор (42 4F 42).

4. Натисніть кнопку **Calculate** (Обчислити) Результатом є хеш-сума **2D**

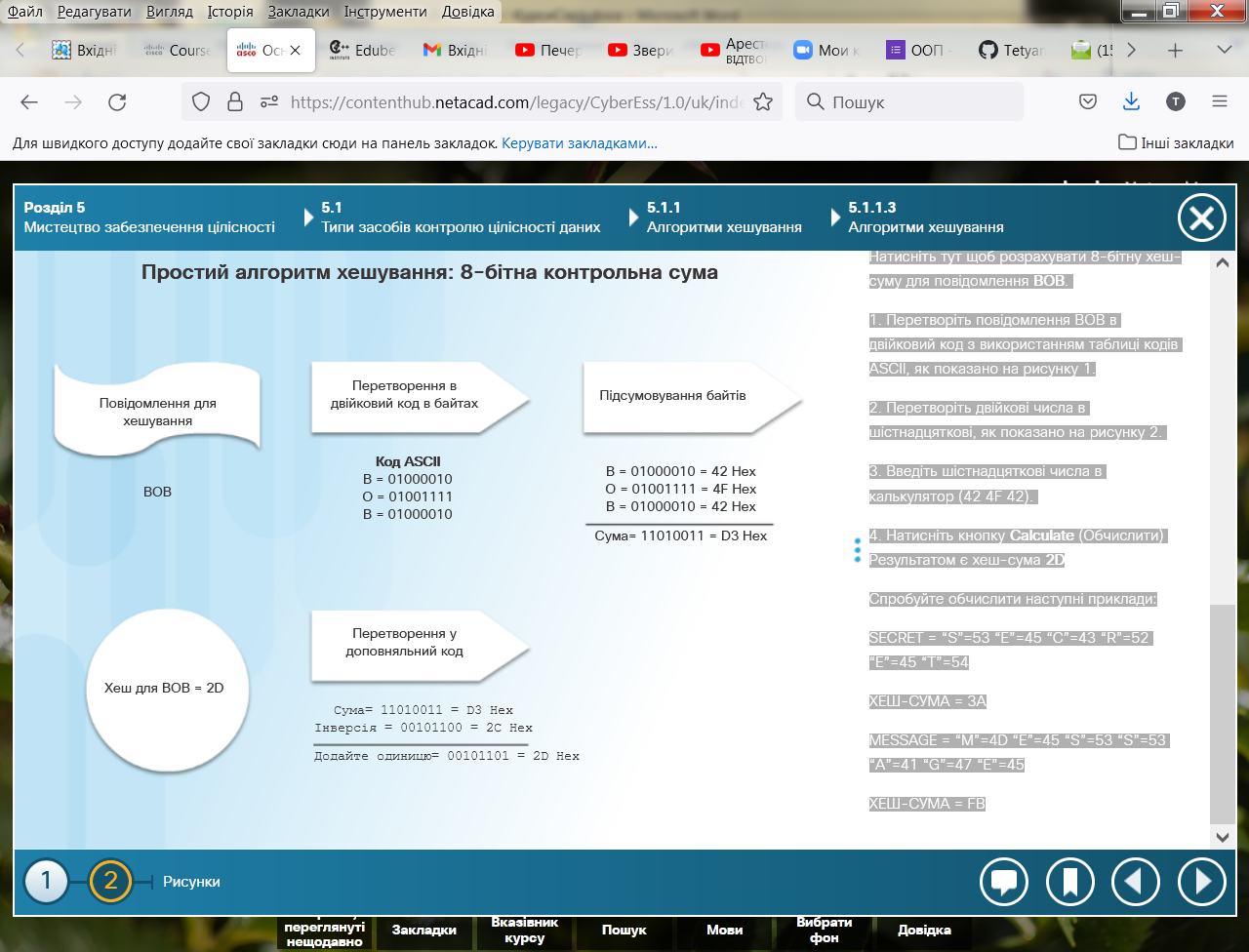
Спробуйте обчислити наступні приклади:

SECRET = “S”=53 “E”=45 “C”=43 “R”=52 “E”=45 “T”=54

ХЕШ-СУМА = 3A

MESSAGE = “M”=4D “E”=45 “S”=53 “S”=53 “A”=41 “G”=47 “E”=45

ХЕШ-СУМА = FB



# Сучасні алгоритми хешування

На сьогоднішній день широко застосовується велика кількість сучасних алгоритмів хешування. Двома найпопулярнішими є MD5 і SHA.

**Алгоритм Message Digest 5 (MD5)**

Рон Рівест розробив алгоритм хешування MD5 і цей алгоритм використовується в Інтернеті для певного кола задач. MD5 - це незворотня функція, яка дозволяє легко обчислити хеш із заданих вхідних даних, але робить складним обчислення вхідних даних при наявності лише значення хешу.

За допомогою механізму перевірки цілісності можна переконатись, що дані та інформація не зазнали змін на шляху від відправника до одержувача. Впевненість у цілісності даних є важливою, коли користувач завантажує файл з Інтернету або під час криміналістичного дослідження цифрових носіїв.

Щоб перевірити цілісність всіх образів IOS, Cisco надає контрольні хеш-суми MD5 і SHA на веб-сайті програмного забезпечення для завантаження. Користувач може порівняняти значення MD5-дайджесту на сайті з дайджестом MD5 образу IOS, встановленого на пристрої, як показано на рисунку. Якщо хеш-суми співпадають, користувач може бути впевненим в тому, що файл образу IOS було завантажено та встановлено у незміненому вигляді.

**Примітка**: Команда **verify /md5** результат виконання якої показано на рисунку, виходить за рамки цього курсу.

У сфері цифрової криміналістики хешування використовується для перевірки цифрових носіїв, що містять файли. Наприклад, експерт розраховує хеш-суму і створює побітну копію носія з файлами, формуючи таким чином цифровий клон. Потім експерт порівнює хеш-суму оригінального носія з хеш-сумою копії. Якщо значення хеш-сум співпадають, то копія є ідентична оригіналу. Повна ідентичність бітів копії бітам оригіналу доводить незмінність. Спираючись на цю незмінність, можна відповісти на наступні питання:

* Чи дійсно експерт має у своєму розпорядженні файли з оригінального носія?
* Чи можна стверджувати, що дані пошкоджені або змінені?
* Чи може експерт довести, що файли не пошкоджені?

Таким чином експерт можуть дослідити копії цифрових доказів не ризикуючи оригінальним носієм.

Читати опис колізійної атаки шкідливим ПЗ Flame.

# MD5 формує 128-бітове значення хеш-суми. У 2012 році була реалізована атака шкідливим ПЗ Flame, яке скомпрометувало безпеку MD5. Автори ПЗ Flame використовували колізію MD5 для підробки сертифіката підпису коду Windows. Натисніть [тут](https://blogs.technet.microsoft.com/srd/2012/06/06/flame-malware-collision-attack-explained/) щоб проХешування файлів і цифрових носіїв

**Алгоритм безпечного хешування (Secure Hash Algorithm, SHA)**

У Національному інституті стандартів і технологій США (NIST) був розроблений алгоритм SHA, який включено у стандарт безпечного хешування (Secure Hash Standard, SHS). NIST опублікував SHA-1 в 1994 році. Сімейство алгоритмів SHA-2 замінило SHA-1, додавши чотири додаткові хеш-функції для створення сімейства SHA:

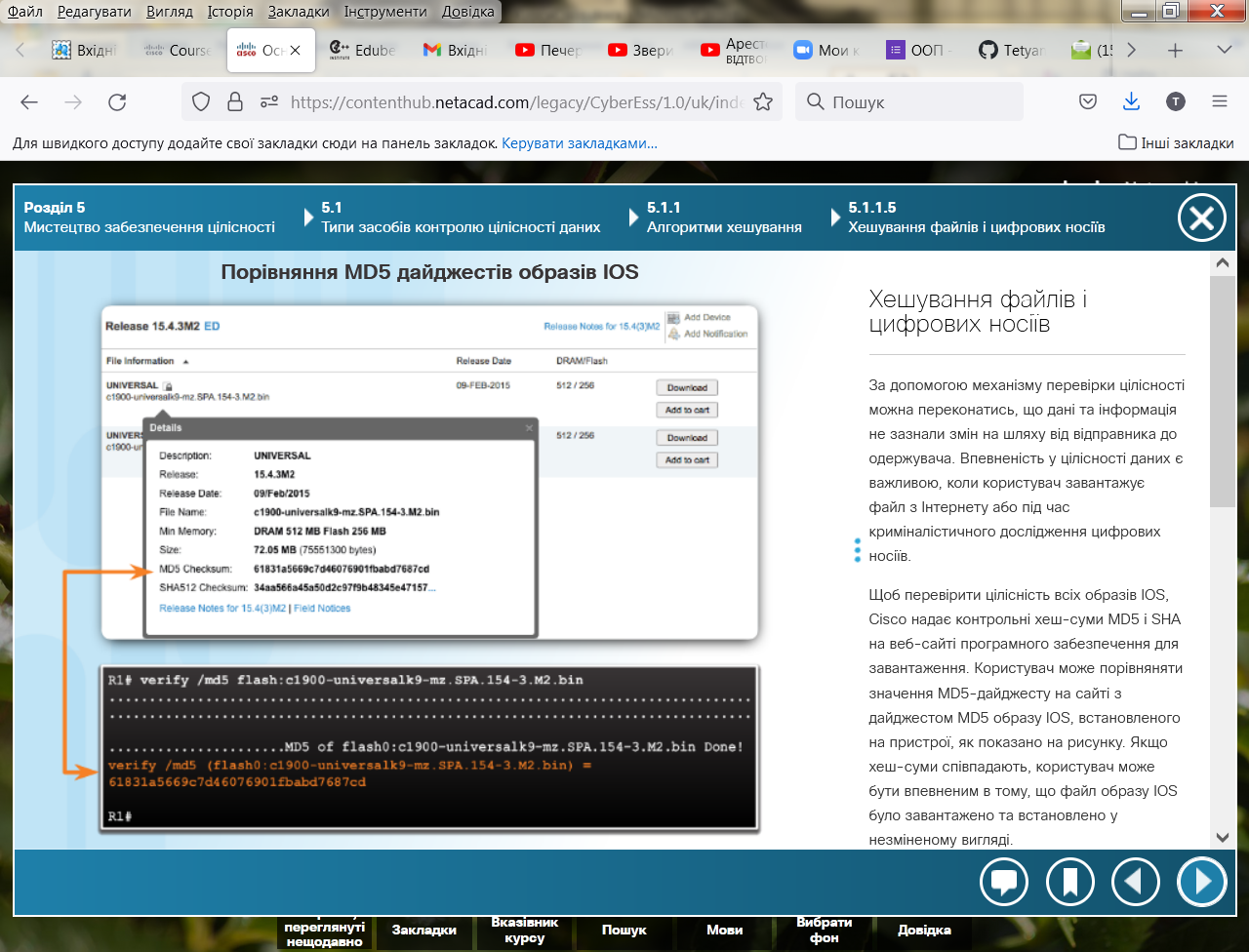
* SHA-224 (224 біт)
* SHA-256 (256 біт)
* SHA-384 (384 біт)
* SHA-512 (512 біт)

SHA-2 - більш стійкіший алгоритм і він все частіше застосовується замість MD5. SHA-256, SHA-384 і SHA-512 є алгоритмами наступного покоління.

**Хешування файлів і цифрових носіїв**

За допомогою механізму перевірки цілісності можна переконатись, що дані та інформація не зазнали змін на шляху від відправника до одержувача. Впевненість у цілісності даних є важливою, коли користувач завантажує файл з Інтернету або під час криміналістичного дослідження цифрових носіїв.

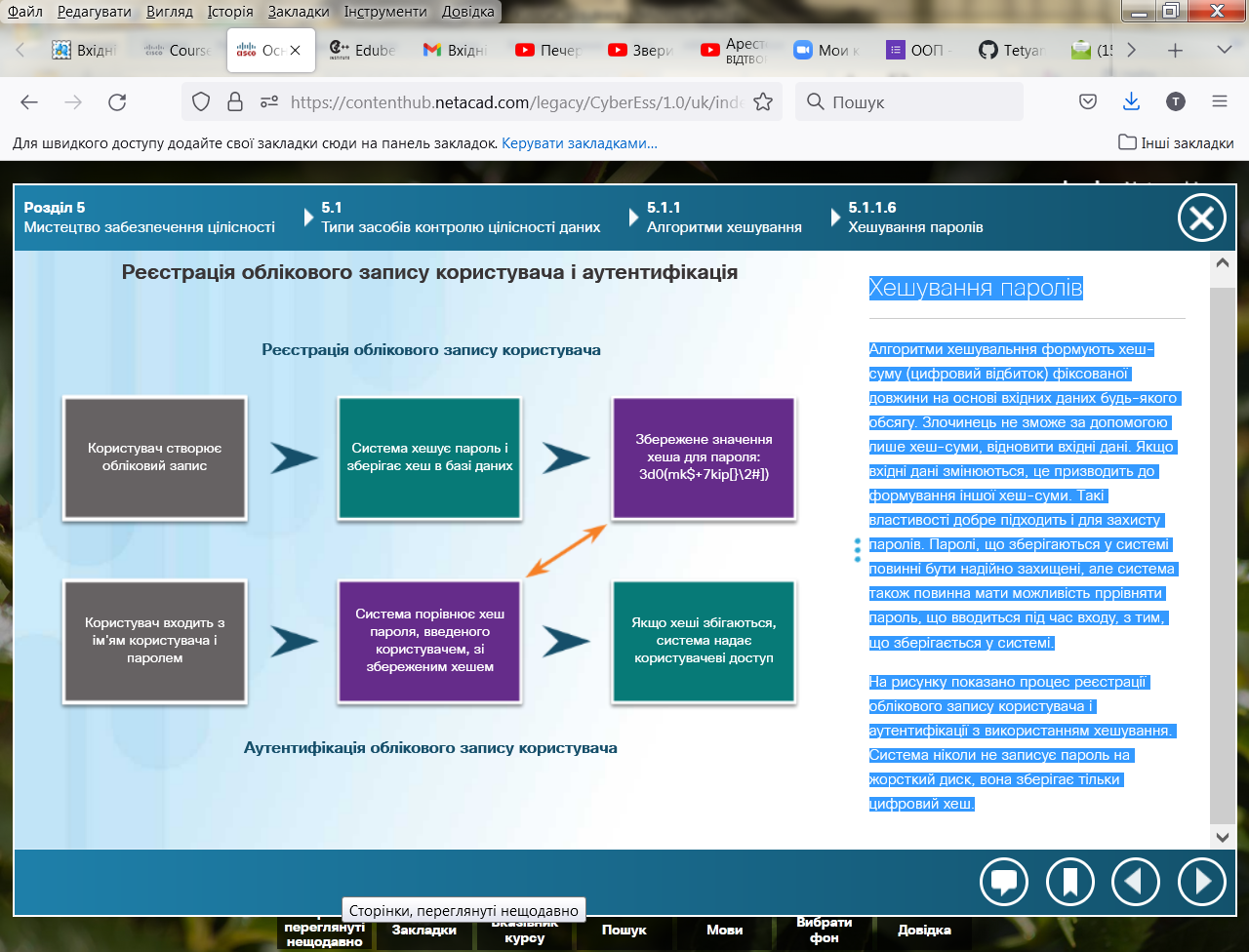
Щоб перевірити цілісність всіх образів IOS, Cisco надає контрольні хеш-суми MD5 і SHA на веб-сайті програмного забезпечення для завантаження. Користувач може порівняти значення MD5-дайджесту на сайті з дайджестом MD5 образу IOS, встановленого на пристрої, як показано на рисунку. Якщо хеш-суми співпадають, користувач може бути впевненим в тому, що файл образу IOS було завантажено та встановлено у незміненому вигляді.

**Примітка**: Команда **verify /md5** результат виконання якої показано на рисунку, виходить за рамки цього курсу.

У сфері цифрової криміналістики хешування використовується для перевірки цифрових носіїв, що містять файли. Наприклад, експерт розраховує хеш-суму і створює побітну копію носія з файлами, формуючи таким чином цифровий клон. Потім експерт порівнює хеш-суму оригінального носія з хеш-сумою копії. Якщо значення хеш-сум співпадають, то копія є ідентична оригіналу. Повна ідентичність бітів копії бітам оригіналу доводить незмінність. Спираючись на цю незмінність, можна відповісти на наступні питання:

* Чи дійсно експерт має у своєму розпорядженні файли з оригінального носія?
* Чи можна стверджувати, що дані пошкоджені або змінені?
* Чи може експерт довести, що файли не пошкоджені?

Таким чином експерти можуть дослідити копії цифрових доказів не ризикуючи оригінальним носієм.

**Хешування паролів**

Алгоритми хешувальння формують хеш-суму (цифровий відбиток) фіксованої довжини на основі вхідних даних будь-якого обсягу. Злочинець не зможе за допомогою лише хеш-суми, відновити вхідні дані. Якщо вхідні дані змінюються, це призводить до формування іншої хеш-суми. Такі властивості добре підходить і для захисту паролів. Паролі, що зберігаються у системі повинні бути надійно захищені, але система також повинна мати можливість порівняти пароль, що вводиться під час входу, з тим, що зберігається у системі.

На рисунку показано процес реєстрації облікового запису користувача і автентифікації з використанням хешування. Система ніколи не записує пароль на жорсткий диск, вона зберігає тільки цифровий хеш.

**Застосування**

Криптографічні хеш-функції слід використовувати в наступних цілях:

* Забезпечення підтвердження достовірності. У цьому випадку хеш-функція застосовується спільно з симетричним ключем автентифікації. Наприклад таким чином працюють протоколи IP Security (IPsec) або протоколи автентифікації маршрутизаторів.
* Автентифікацію шляхом генерації одноразових і односторонніх відповідей на запити протоколу автентифікації.
* Надання доказів перевірки цілісності повідомлень, зокрема, які використовуються в контрактах з цифровим підписом, і сертифікатах інфраструктури відкритого ключа (PKI), (такі сертифікати застосовуються, наприклад, для доступу до захищеного сайту з використанням браузера).

При виборі алгоритму хешування рекомендується використовувати, як мінімум, SHA-256 або більш стійкі алгоритми, оскільки вони є в даний час найбільш безпечними. Уникайте використання SHA-1 і MD5 через виявлені в них серйозні вразливості безпеки. У промислових мережах слід застосовувати SHA-256 або більш стійкі алгоритми.

Хешування може виявити випадкові зміни, але воно не може запобігти навмисній зміні даних. Результат хешування не містить унікальної інформації, за допомогою якої можна було б однозначно визначити відправника. Це означає, що будь-хто може обчислити значення хеш-функції для будь-яких даних, якщо у нього є доступ до алгоритму потрібної хеш-функції. Наприклад, зловмисник може перехопити повідомлення, що передається через мережу, змінити його, перерахувати значення хеш-функції і додати до нового(підробленого) повідомлення. Приймаючий пристрій перевірятиме тільки те значення хешу, яке додане до повідомлення. Тому хешування вразливе для атак типу «людина по середині» і не захищає переданих даних.

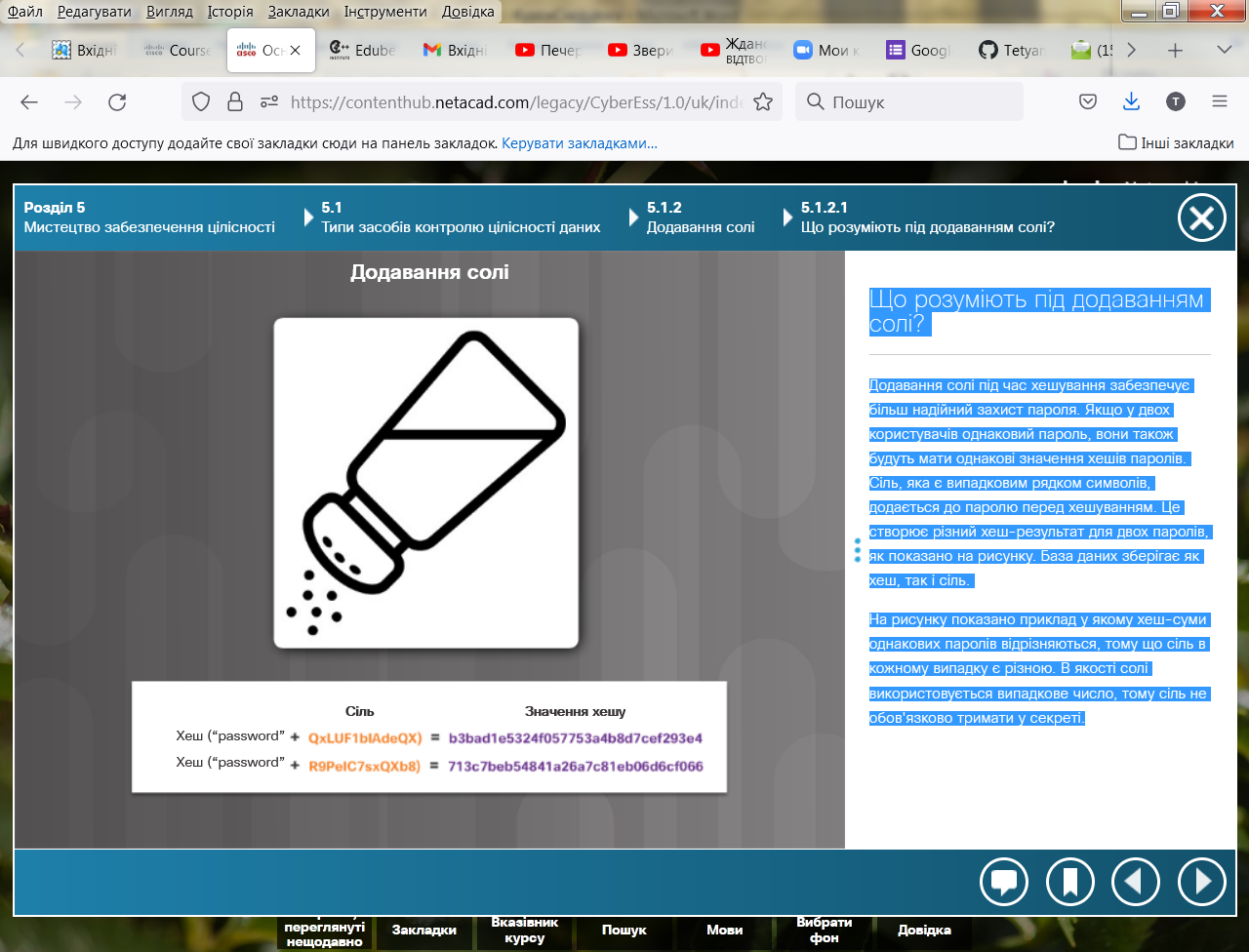
**Злам хешу**

Щоб одержати хеш-код, для входу в систему, зловмисник повинен вгадати пароль. Дві найбільш популярні атаки, що використовуються для зламування паролів, - це атаки перебору за допомогою словників та атака грубої сили (brute-force attacks).

Атака методом перебору за допомогою словника використовує файл, який містить розповсюджені слова, фрази і паролі. Файл має розраховані хеші. Така атака порівнює хеш-суми у файлі з значенням хешу пароля. Якщо хеш збігається, зловмисник дізнається групу потенційно правильних паролів.

Атака грубої сили намагається використовувати будь-яку можливу комбінацію символів для заданої довжини пароля. Такі атаки вимагають потужних обчислювальних ресурсів, але якщо наявний великий запас часу, то цей метод виявить пароль з сто відсотковою точністю. Тому паролі повинні бути досить довгими, щоб зробити час, необхідний для виконання атаки занадто довгим для успішності атаки. Хешування паролів ускладнює зловмисникам задачу вгадування паролів.

**Що розуміють під додаванням солі?**

Додавання солі під час хешування забезпечує більш надійний захист пароля. Якщо у двох користувачів однаковий пароль, вони також будуть мати однакові значення хешів паролів. Сіль, яка є випадковим рядком символів, додається до паролю перед хешуванням. Це створює різний хеш-результат для двох паролів, як показано на рисунку. База даних зберігає як хеш, так і сіль.

На рисунку показано приклад у якому хеш-суми однакових паролів відрізняються, тому що сіль в кожному випадку є різною. В якості солі використовується випадкове число, тому сіль не обов'язково тримати у секреті.

# Запобігання атакам

Додавання солі не дозволяє зловмисникові використовувати атаку перебору за словником, щоб спробувати вгадати пароль. Крім того, наявність солі також не дозволяє використовувати таблиці пошуку і таблиці веселки, щоб зламати хеш.

**Таблиці пошуку**

У таблицях пошуку зберігаються попередньо обчислені хеш-суми паролів, які взяті з відповідних словників, а також самі ці паролі. За допомогою таблиць пошуку можна перевіряти сотні хеш-сум в секунду. Натисніть [тут](https://crackstation.net/) щоб дізнатися, як швидко можна зламати хеш за допомогою таблиці пошуку.

**Реверсивні таблиці пошуку**

За допомогою таких таблиць кіберзлочинці реалізують атаку методом грубої сили або перебором за допомогою словника з перевіркою великої кількості хеш-сум без попередньо побудованої таблиці пошуку. Використовуючи викрадену базу даних облікових записів, зловмисник створює таблицю пошуку, яка містить список хеш-сум усіх правильних паролів та список користувачів. Зловмисник хешує кожен ймовірний пароль і за допомогою таблиці пошуку, отримує список користувачів, паролі яких співпали з ймовірними, як показано на рисунку. Така атака є ефективною, оскільки багато користувачів має однакові паролі.

**Веселкові таблиці**

Веселкові таблиці, у порівнянні зі звичайними таблицями пошуку, мають менший об'єм. Скорочення об'єму досягається шляхом зменшення швидкості зламу. Скоротивши об'єм таблиці, зловмисник може перевірити більшу кількість хеш-сум використовуючи однаковий об'єм пам'яті.

# 

# Реалізація механізму додавання солі

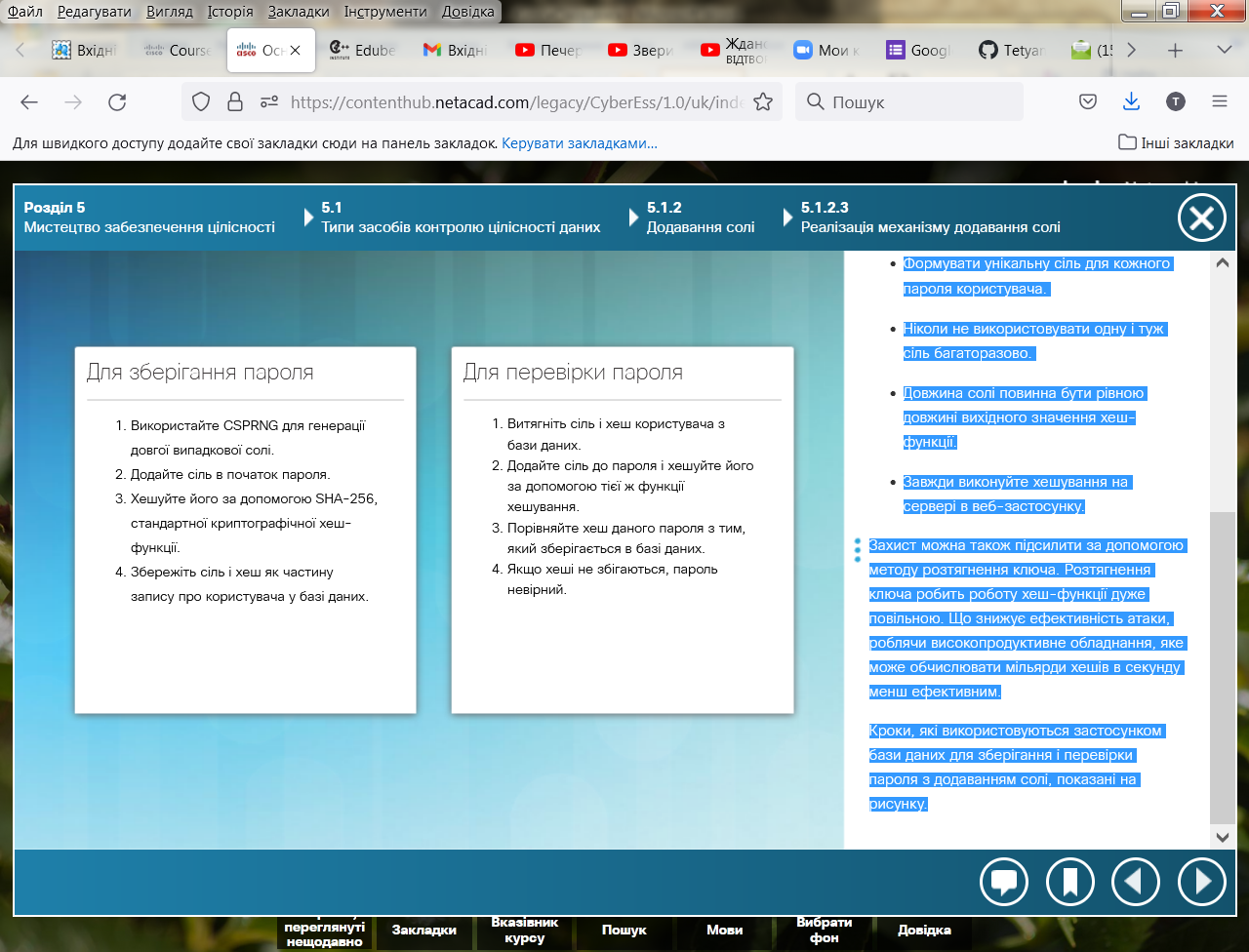
Криптографічно стійкий генератор псевдовипадкових чисел ( Cryptographically Secure Pseudo-Random Number Generator, CSPRNG) - кращий вибір для генерування солі. Такий генератор генерує випадкове число, яке має високий ступінь випадковості і абсолютно непередбачуване, тому воно забезпечує криптографічну стійкість.

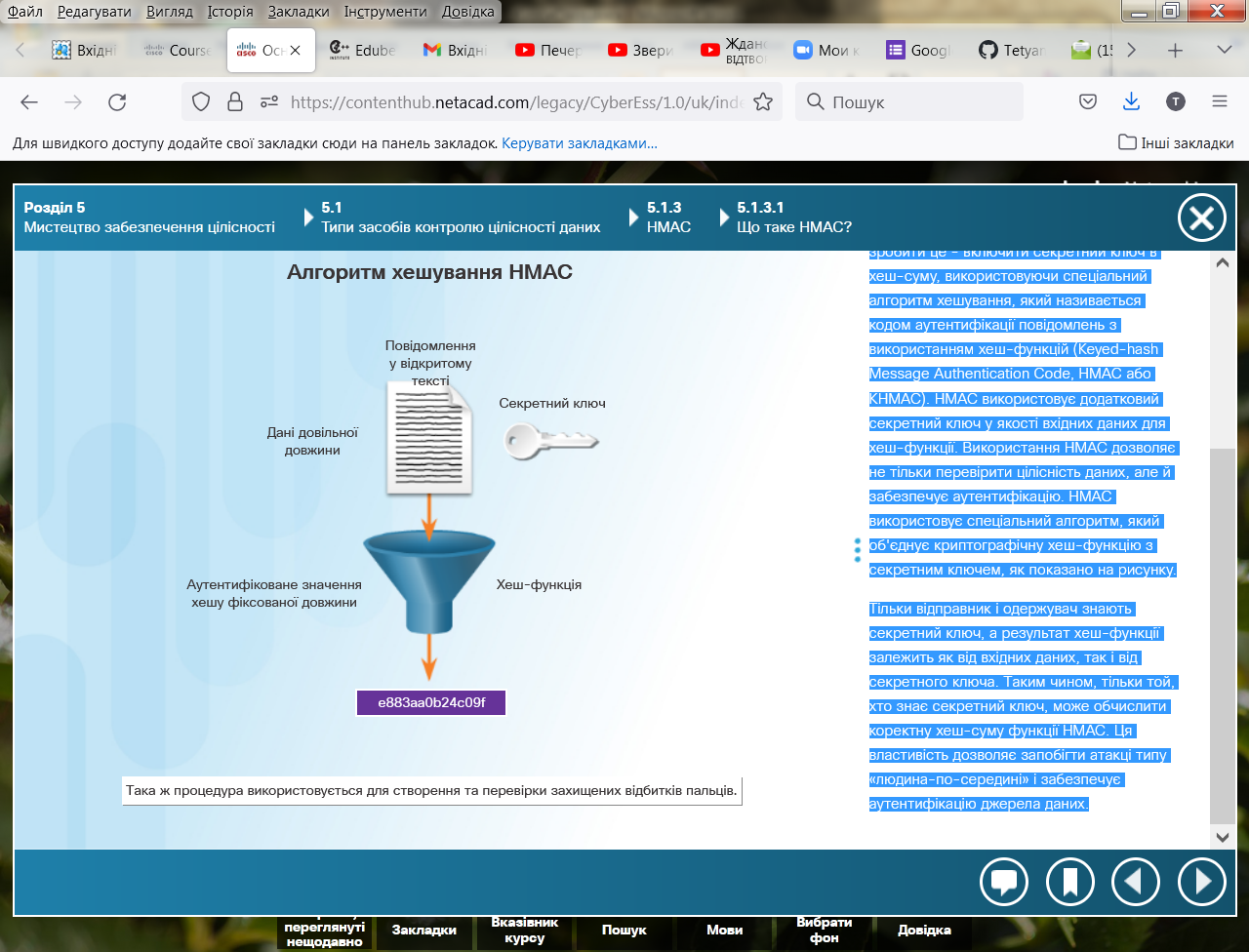
Впроваджуючи механізм додавання солі, необхідно дотримуватись наступних рекомендацій:

* Формувати унікальну сіль для кожного пароля користувача.
* Ніколи не використовувати одну і туж сіль багаторазово.
* Довжина солі повинна бути рівною довжині вихідного значення хеш-функції.
* Завжди виконуйте хешування на сервері в веб-застосунку.

Захист можна також підсилити за допомогою методу розтягнення ключа. Розтягнення ключа робить роботу хеш-функції дуже повільною. Що знижує ефективність атаки, роблячи високопродуктивне обладнання, яке може обчислювати мільярди хешів в секунду менш ефективним.

Кроки, які використовуються застосунком бази даних для зберігання і перевірки пароля з додаванням солі, показані на рисунку.





**Що таке HMAC?**

Наступний рівень захисту від атак методом грубої сили, або атаки з допомогою словника забезпечується поєднанням секретного ключа і хеш-суми. У такій схемі тільки той, хто знає хеш-суму, може перевірити пароль. Одним із способів зробити це - включити секретний ключ в хеш-суму, використовуючи спеціальний алгоритм хешування, який називається кодом автентифікації повідомлень з використанням хеш-функцій (Keyed-hash Message Authentication Code, HMAC або KHMAC). HMAC використовує додатковий секретний ключ у якості вхідних даних для хеш-функції. Використання HMAC дозволяє не тільки перевірити цілісність даних, але й забезпечує автентифікацію. HMAC використовує спеціальний алгоритм, який об'єднує криптографічну хеш-функцію з секретним ключем, як показано на рисунку.

Тільки відправник і одержувач знають секретний ключ, а результат хеш-функції залежить як від вхідних даних, так і від секретного ключа. Таким чином, тільки той, хто знає секретний ключ, може обчислити коректну хеш-суму функції HMAC. Ця властивість дозволяє запобігти атаці типу «людина-по-середині» і забезпечує автентифікацію джерела даних.

# Принцип дії механізму HMAC

Розглянемо приклад, коли відправник хоче гарантувати, що повідомлення залишаються незмінними під час передавання і хоче надати одержувачу можливість автентифікації джерела даних.

Як показано на рисунку 1, пристрій, що передає, вводить дані (наприклад, інформацію про оплату для Террі Сміта у $100 і секретний ключ) на вхід алгоритму хешування і обчислює відбиток, або хеш-суму HMAC фіксованої довжини. Одержувач отримує автентифікований відбиток, прикріплений до повідомлення.

На рисунку 2 приймаючий пристрій відділяє відбиток від повідомлення і використовує повідомлення у відкритому тексті у поєднанні з секретним ключем в якості вхідних даних для тієї ж функції хешування. Якщо приймаючий пристрій обчислює відбиток, рівний відправленому відбитку, повідомлення одержано в незмінному вигляді. Крім того, одержувач може бути впевненим у походженні повідомлення, тому що тільки відправнику відомий спільний секретний ключ. Функція HMAC підтвердила достовірність повідомлення.

# Використання HMAC

HMAC також можна використовувати для автентифікації Інтернет-користувачів. На багатьох веб-сервісах використовують базову автентифікацію, у якій ім'я користувача і пароль передаються у незашифрованому вигляді. Якщо ж автентифікація відбувається з використанням механізму HMAC, то користувач зобов'язаний передати ідентифікатор приватного ключа і хеш-суму НМАС. Сервер знаходить приватний ключ користувача та обчислює хеш-суму HMAC. Обчислена сервером хеш-сума повинна співпасти з хеш-сумою HMAC, яка була одержана від користувача.

За допомогою функції НМАС проводиться автентифікація джерела кожного з пакетів та перевірка цілісності даних в VPN-мережах, що базуються на протоколі IPsec.

Як показано на рисунку, продукти Cisco використовують хешування для автентифікації об'єктів, перевірці цілісності і достовірності даних:

* Маршрутизатори Cisco IOS під час роботи використовують схему аналогічну HMAC: в повідомлення про оновлення протоколу маршрутизації додається автентифікаційна інформація за допомогою секретного ключа.
* Шлюзи та клієнти IPsec перевіряють цілісність та достовірність пакетів використовує алгоритми хешування, зокрема MD5 і SHA-1 в режимі HMAC.
* Образи програмного забезпечення Cisco, які розміщені на сайті Cisco.com містять контрольну суму на основі MD5, щоб клієнти могли перевіряти цілісність завантажених образів.

**Примітка**: Термін "об'єкт" означає пристрій або систем всередині організації.

# 

# Що таке цифровий підпис?

Рукописні підписи і печатки на документах підтверджують авторство. Цифрові підписи можуть забезпечувати ті ж функції.

Внести зміни у незахищений цифровий документ дуже легко. Цифровий підпис дозволяє визначити, чи редагувався документ після його підпису користувачем. Цифровий підпис - це математичний метод, який використовується для перевірки достовірності та цілісності повідомлення, цифрового документа або програмного забезпечення.

У багатьох країнах цифрові підписи мають таку ж юридичну силу, як документ, підписаний вручну. Електронний підпис, що стоїть під електронним контрактом, договором або будь-яким іншим документом, який у паперовому вигляді вимагає рукописного підпису, має повну юридичну силу. Для вирішення задач, що пов'язані з юридичним захистом та регулюванням, передбачені та ведуться відповідні журнали подій, відстежують історію змін у електронному документі.

Цифровий підпис допомагає встановити справжність, цілісність і неможливість відмови (невідхильність). Цифрові підписи мають характерні властивості на яких побудовано механізм аутентифікації об'єкта і перевірки цілісності даних, як показано на рисунку.

Цифрові підписи є альтернативою механізму HMAC.

# 

# Неможливість відмови (невідхильність)

Під відхиленням розуміють відмову від авторства. Невідхильність - це спосіб гарантувати, що відправник повідомлення або документа не може відмовитись у надсиланні, а одержувач не може відмовитись у отриманні повідомлення або документа.

Наявність цифрового підпису гарантує, що відправник електронним способом підписав повідомлення або документ. Кожен з користувачів має унікальний цифровий підпис, тому той, хто створив підпис під документом, не зможе пізніше заперечувати свого авторства.

# Процес створення цифрового підпису

В основі цифрових підписів є асиметрична криптографія. Алгоритм відкритого ключа, такий як RSA, генерує два ключі: один закритий і інший відкритий. Ці ключі математично пов'язані між собою.

Аліса хоче відправити Бобу електронний лист, у якому міститься важлива інформація про вихід нового продукту на ринок. Боб повинен бути абсолютно впевненим, що повідомлення прийшло саме від користувача Аліса і доставлено в незмінному вигляді.

Аліса створює повідомлення та дайджест (хеш-суму) повідомлення. Вона шифрує дайджест повідомлення своїм закритим ключем, як показано на рисунку 1. Аліса пов'язує повідомлення, зашифрований дайджест повідомлення і відкритий ключ для створення підписаного цифровим підписом документа. Одержаний результат Аліса відправляє Бобу, як показано на рисунку 2.

Боб отримує повідомлення і читає його. Щоб переконатися, що повідомлення прийшло від Аліси, він створює дайджест повідомлення. Він бере зашифрований дайджест повідомлення, отриманий від Аліси, і розшифровує його за допомогою відкритого ключа Аліси. Боб порівнює дайджест повідомлення, отриманий від Аліси, з тим, який він створив. Якщо вони збігаються, Боб знає, що може довіряти отриманим даним і що ніхто не підробляв повідомлення, як показано на рисунку 3.

Натисніть [тут](https://www.youtube.com/watch?v=E5FEqGYLL0o) щоб переглянути відео, що пояснює процес створення цифрового сертифікату.



3

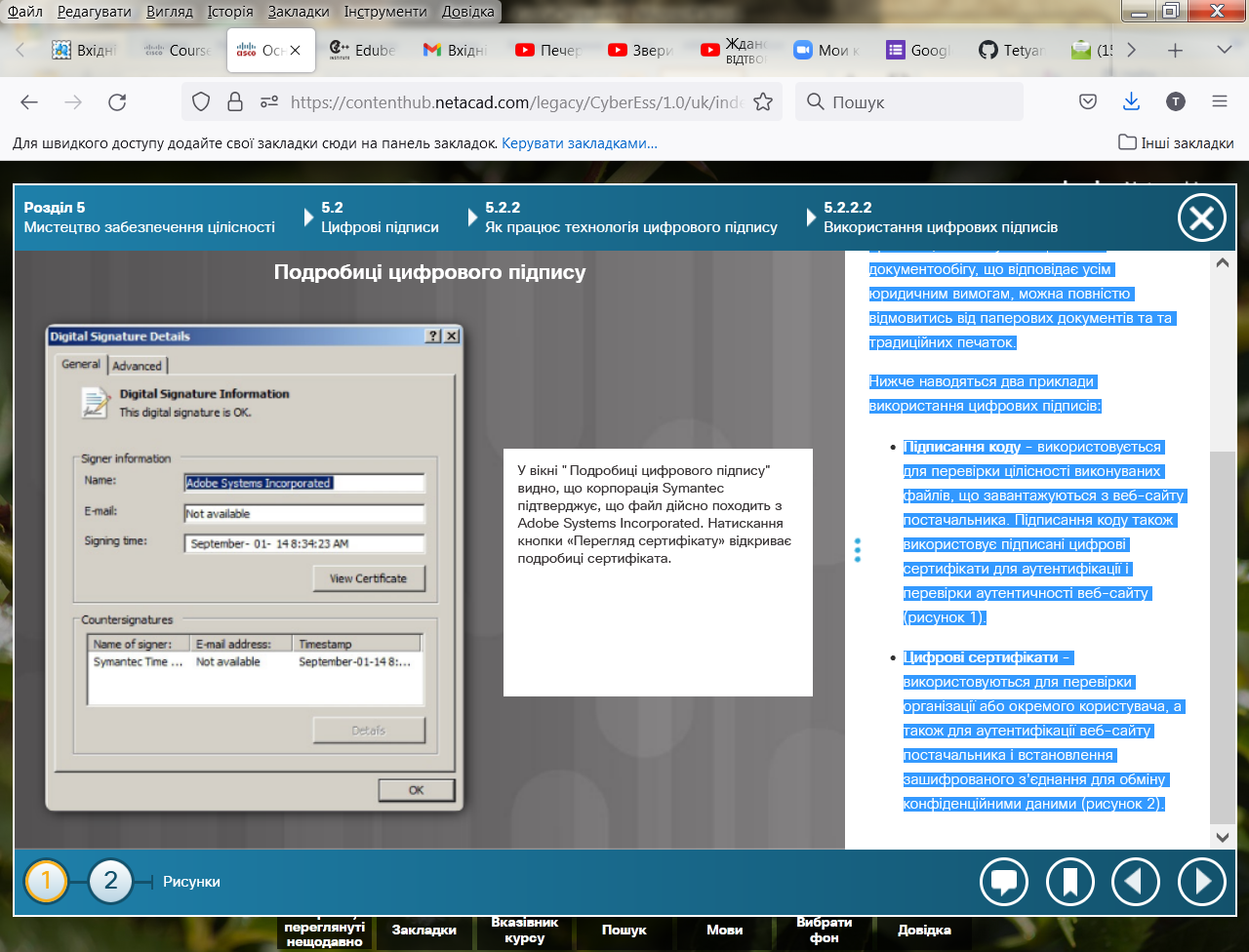
2

1

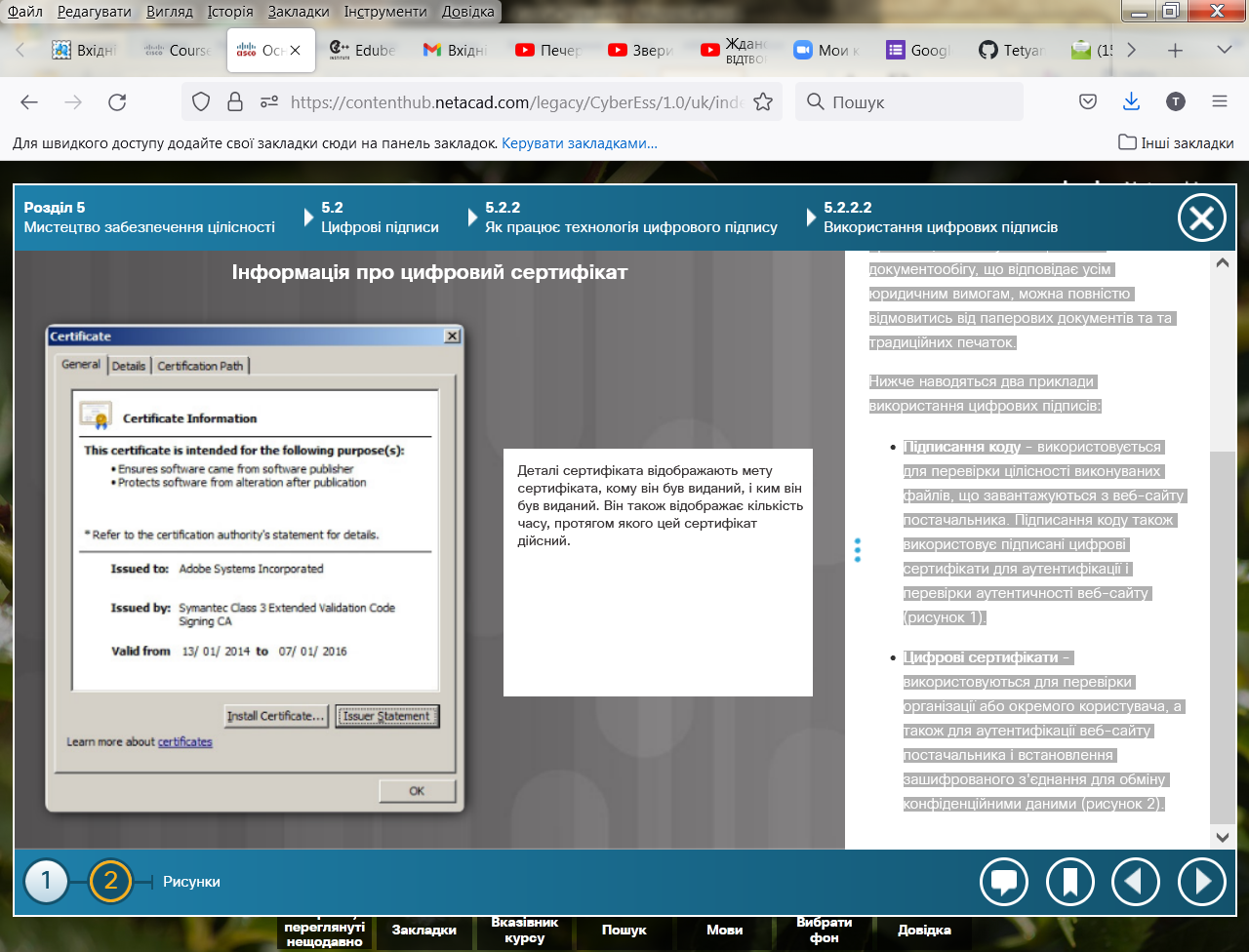
**Використання цифрових підписів**

Замість підписування всього документа можна підписати тільки його хеш-суму. Така схема підвищує ефективність та спрощує задачі пов'язані з перевіркою цілісніості та забезпечення сумісності. Впровадивши в організації систему електронного документообігу, що відповідає усім юридичним вимогам, можна повністю відмовитись від паперових документів. Нижче наводяться два приклади використання цифрових підписів:

* **Підписання коду** - використовується для перевірки цілісності виконуваних файлів, що завантажуються з веб-сайту постачальника. Підписання коду також використовує підписані цифрові сертифікати для автентифікації і перевірки автентичності веб-сайту (рисунок 1).
* **Цифрові сертифікати** - використовуються для перевірки організації або окремого користувача, а також для автентифікації веб-сайту постачальника і встановлення зашифрованого з'єднання для обміну конфіденційними даними (рисунок 2).



1



2

# Порівняння алгоритмів цифрового підпису

Найбульш поширеними є три алгоритми цифрового підпису - Digital Signature Algorithm (DSA), Rivest-Shamir-Adleman (RSA) і Elliptic Curve Digital Signature Algorithm (ECDSA). Всі три алгоритми містять механізми створення та перевірки цифрових підписів. Ці алгоритми функціонують на основі асиметричного шифрування з використанням відкритих ключів. Робота з цифровими підписами базується на двох операціях:

1. Генерація ключа

2. Перевірка ключа

Обидві операції вимагають шифрування і дешифрування ключів.

Алгоритм DSA базується на складності обчислень дискретних логарифмів. У багатьох країнах алгоритм DSA використовують у державному секторі для створення цифрових підписів. Можливості алгоритму DSA обмежуються формуванням та перевіркою цифрового підпису.

RSA є найбільш поширеним алгоритмом шифрування з відкритим ключем, з тих, які використовуються сьогодні. RSA створено в 1977 році. Названий на честь авторів: Рона Ривеста, Аді Шаміра і Леонарда Адлемана. В основі RSA лежить асиметричне шифрування. RSA можна використовувати не тільки для роботи з цифровими підписами, а також шифрування повідомлення.

DSA працює швидше, ніж RSA, в якості підпису для цифрового документа. Алгоритм RSA краще підходить для ситуацій, коли крім вимог підпису і перевірки електронних документів необхідне ще й шифрування повідомлень.

Як і більшість криптографічних систем, алгоритм RSA базується на двох математичних компонентах: модульній арифметиці і задачі факторизації великих цілих чисел. Натисніть [тут](https://www.youtube.com/watch?v=wXB-V_Keiu8) щоб дізнатися більше про те, як у RSA використовується модульна арифметика і факторизація великих цілих чисел.

ECDSA - це новий алгоритм цифрового підпису, який поступово замінює RSA. Його перевага полягає в тому, що він може використовувати набагато менші розміри ключа при такому ж рівні безпеки і вимагає значно менших об'ємів обчислень, ніж RSA.

# Лабораторна робота – Використання цифрових підписів

У цій лабораторній роботі ви виконаєте наступні завдання:

* Вивчити механізм, що лежить в основі технології цифрових підписів.
* Продемонструвати практичне використання цифрових підписів.
* Продемонструвати перевірку цифрового підпису.

[Лабораторна робота – Використання цифрових підписів](https://contenthub.netacad.com/legacy/CyberEss/1.0/uk/course/files/5.2.2.4%20Lab%20-%20Using%20Digital%20Signatures.pdf)