МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ І ІНФОРМАТИКИ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання курсової роботи з дисципліни

«Методи та засоби криптографічного захисту інформації»

за темою

«Криптографічне хешування. Методи та алгоритмічна підтримка»

(для студентів спеціальності 125 Кібербезпека

всіх форм навчання)

Покровськ – 2022

УДК 681.3.06

Методичні вказівки до виконання курсової роботи з дисципліни «Методи та засоби криптографічного захисту інформації» за темою: «Криптографічне хешування. Методи та алгоритмічна підтримка» (для студентів спеціальності 125 Кібербезпека всіх форм навчання) / укладач: проф. Башков Є.О. - Покровськ: ДонНТУ, 2022 р. - 27 с.

Методичні вказівки до виконання курсової роботи з дисципліни «Методи та засоби криптографічного захисту інформації» призначені для закріплення знань, отриманих при вивченні лекційного матеріалу студентами факультету комп'ютерних наук і технологій, що навчаються за спеціальністю 125 Кібербезпека. Наведено короткі теоретичні відомості та індивідуальні завдання за розділом методи та стандарти хешування.

Для розвитку і закріплення теоретичних знань і практичних навичок студентам пропонується розробити програмний засіб обчислення хеш дайджесту відкритого повідомлення. Реалізація індивідуального завдання передбачає залучення можливостей стандартного Python модуля HashLib та написання власного програмного коду.

Методичні вказівки і завдання до курсових робіт за дисципліною «Методи та засоби криптографічного захисту інформації» можуть бути корисними також і для студентів, що навчаються за спеціальностями 121 «Інженерія програмного забезпечення», 122 «Комп'ютерні науки», 123 «Комп'ютерна інженерія».

Укладач: Є.О. Башков, д.т.н., проф. каф. ПМІ

Рецензент: С.О. Ковальов, к.т.н., доц. каф. КІ

Відповідальний за випуск: О.А. Дмитрієва, д.т.н., проф., зав. каф. ПМІ

Затверджено навчально-методичним відділом ДВНЗ «ДонНТУ»,
протокол № від _____2022 р.

Розглянуто на засіданні кафедри прикладної математики та інформатики,
протокол № 1 від 23.01.2022 р.

З М І С Т

	ВСТУП ……	4
1.i	Елементи теорії криптографічного хешування……	5
2.	Модуль HashLib. Базові алгоритми криптографічного хешування ……	14
3.	Завдання до курсової роботи…..…….……..…….	19
4.	Порядок виконання роботи……	20
5.	Порядок контролю та прийому……	21
	СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ……...	22
i	Додаток А – Приклад Python скрипту ……..	24
	Додаток Б – Технічне завдання …….	27

ВСТУП

Курсова робота орієнтована на закріплення теоретичного матеріалу та придбання практичних навичок в застосуванні сучасних алгоритмів та програмних засобів криптографічного захисту інформації. Мета розробки полягає в оволодінні теоретичними аспектами обчислення хеш дайджестів, опануванні сучасними програмними модулями, що підтримують міжнародні стандартні хеш-алгоритми та практичним застосуванням хеш функція для створення програмних додатків розрахунку хеш дайджестів.

Курсова робота виконується на підставі навчального плану підготовки студентів за освітньо-кваліфікаційним рівнем «бакалавр» та «Технічного завдання до курсової роботи» за дисципліною «Методи та засоби криптографічного захисту інформації» за темою «Криптографічне хешування. Методи та алгоритмічна підтримка» для студентів, що отримують освіту в галузі знань 12 «Інформатика та обчислювальна техніка», спеціальності 125 Кібербезпека.

В рамках виконання курсової роботи студентам пропонується розробити програмний засіб обчислення хеш дайджесту відкритого повідомлення. Реалізація індивідуального завдання передбачає залучення можливостей стандартної Python бібліотеці HashLib та написання власного програмного скрипту.

Захист курсової роботи передбачає презентацію теоретичного матеріалу відносно визначеного індивідуальним завданням стандартного хеш алгоритму, результатів роботи розроблених програмних скриптів та висновків щодо криптостійкості отриманих хеш дайджестів.

1 ЕЛЕМЕНТИ ТЕОРІЇ КРИТОГРАФІЧНОГО ХЕШУВАННЯ

1.1. Загальні відомості.

В обміні будь-якою інформацією беруть участь (як мінімум) дві сторони Передаюча сторона (відправник), яка формує якесь повідомлення і приймаюча сторона (одержувач), яка повинна отримати це повідомлення. Причому дуже часто передача повідомлення від вдправника до одержувача повинна проходити **ТАЄМНО**, тобто ніяка третя сторона (порушник, перехоплювач) не повинна мати можливості:

а) перехопити та зрозуміти повідомлення (конфідеційність),

б) змінити авторстово повідомлення (аутентифікація),

в) спотворити повідомлення (цілістність).

Вирішення задачі підтримки таємності передачі інформації покладається на множину різноманітних методів та засобів захисту, серед яких суттєву роль мають криптографічні підходи.

Суть криптографічного захисту інформації полягає в наступному (рис. 1). Відправник інформації (в критографічній літературі ***Аліса***) має для передачі одержувачу (в критографічній літературі ***Боб***) відкрите пвідомлення $\boldsymbol{M}$ (message, plaintext). За допомогою спеціального пристрою (методу, агоритму, процедури, програми) ***Аліса*** перетвоює відките повідомлення у відповідне таємне (масковане, зашифроване) повідомлення $\boldsymbol{C}$ (chiphertext), яке і передає ***Бобу*** через дякий канал звязку. ***Боб***, в свою чергу, за допомогою зворотнього перворення відновлює (розшифровує) повідомлення і сприймає його. Звісно, в сучасному цифровому світі, повідмлення (як відкрите, так і зашифроване) розуміється як деяка послідовність двійкових бітів.

Процедура перетоврення відкритого повідомлення $\boldsymbol{M}$ в зашифроване $\boldsymbol{C}$ називається шифруванням, а пристрій (метод, алгоритм) що її виконую –

шифратор (encoder, енкодкр). Зворотне переторення називається розшиірування, пристрій (алгоритм) – розшифратор (decoder, декодер).

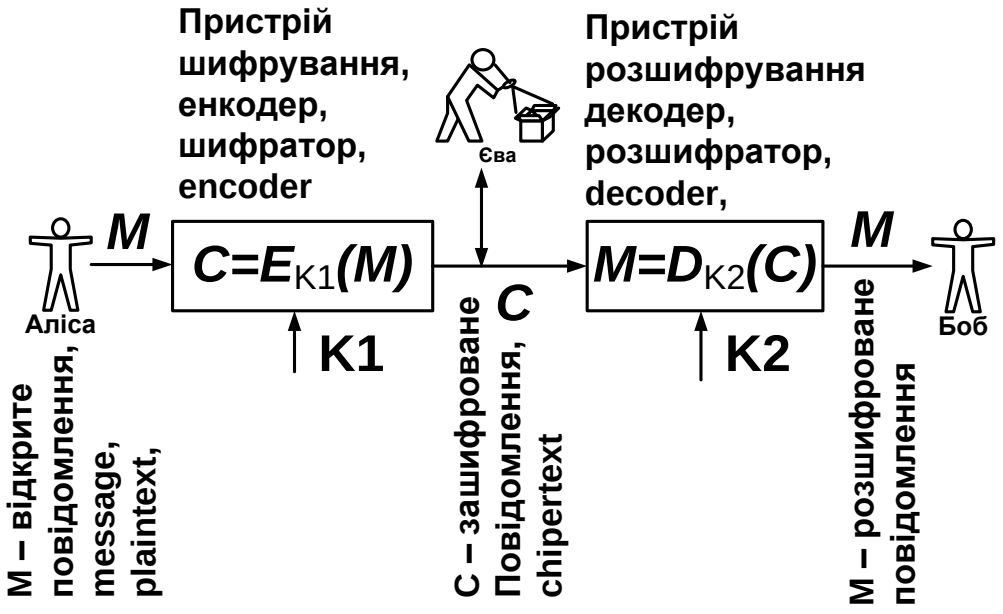

Рисунок 1 – Схема криптозахисту інформації

Математично, процедуру шифрування можна представити функцією

$$\boldsymbol{C} = \boldsymbol{E}(\boldsymbol{M}). \quad (1)$$

Аналогічно, процедуру розшифрування можно представити функцією

$$\boldsymbol{M} = \boldsymbol{D}(\boldsymbol{C}). \quad (2)$$

Умова відновлення вихідного повідомлення на стороні ***Боба***

$$\boldsymbol{M} = \boldsymbol{D}\big(\boldsymbol{E}(\boldsymbol{M})\big) \text{ или } \boldsymbol{D}(\diamond)^{-1} = \boldsymbol{E}(\diamond), \quad (2)$$

властивість зворотнісі функції шифрування.

За інформацією, яка предається, полює третя строна (порушник, перехоплювач, в термінології критографічної літератури ***Єва***). Передбачається, що в найгіршому випадку ***Єва*** знає алгоритм шифрування (і, відрповідно, розшифрування) та завжди має можливість отримати (перехопити) зашифроване повідомлення $\boldsymbol{C}$. З урахуванням цього, кріптографічну процедуру шифрування необхідно будувати так, щоб ***Єва*** не мала можливості ніяким чином дешифрувати повідомлення і занайти $\boldsymbol{M}$.

Сучасні методи критграфічного захисту базуються на використанні ключей.

Криптографічний ключ $\boldsymbol{K}$ це спеціальне повідомлення (послідовність двійкових бітів), яке обирається з великої множини можливих значень – спеціального простіру ключей). І шифруванні і розшифрування провадится з використанням ключей, тобто

$$\boldsymbol{C = E(M, K_1)}, \qquad \boldsymbol{M = D(M, K_2)}, \tag{1}$$

та, відповідно,

$$\boldsymbol{M = D(E(M, K_1), K_2)}. \tag{1}$$

При цьому, захист інформації від дешифрації ***Євою*** забезбечується секретністю одного та/або двох ключей $\boldsymbol{K}$.

1.2. Криптографічний хеш

Хешування (іноді «ґешування», англ. hashing) - перетворення по детермінованому алгоритму вхідного масиву даних довільної довжини у вихідний бітовий рядок фіксованої довжини. Таке перетворення також називаються хеш-функцією або функцією згортки. При цьому результат пертоврення називають хешом, хеш-кодом чи зведенням повідомлення, дайджестом (англ. message digest).

Хешування застосовується для побудови асоціативних масивів, пошуку дублікатів у серіях наборів даних, побудови досить унікальних ідентифікаторів для наборів даних, контрольне підсумовування з метою виявлення випадкових або навмисних помилок при зберіганні або передачі, для зберігання паролів у системах захисту (у цьому випадку доступ до області пам'яті, де знаходяться паролі, не дозволяє відновити сам пароль), при виробленні електронного підпису (на практиці часто підписується не саме повідомлення, а його хеш-образ).

Загальна ідея створеня хешу деякого повідомлення $\boldsymbol{M}$ з разрядністю $\boldsymbol{q}$ біт ілюструється рис.2.

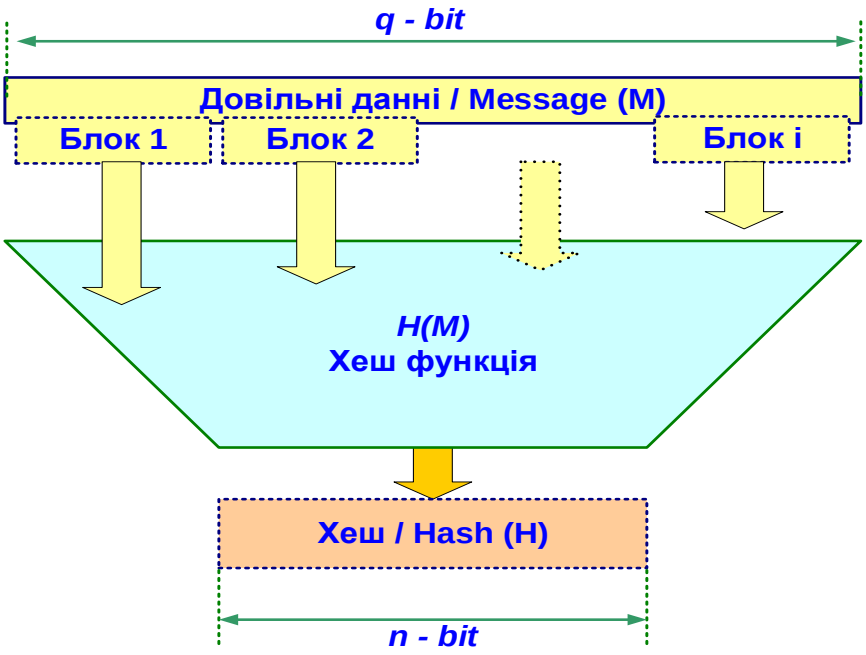

Рисунок 2 – Створення хешу

Повідомлення розбивається на блоки (різрядністю ***p*** біт), які послідовно подаються на вхід спеціально обраної хеш-функціїї ***H( ),*** що формує на своєму виході ***n*** - бітовий двійковий код ***H***. Цей код є хешем (дайджестом) повідомлення ***M***.

Наприклад, ***Аліса*** подає повідомлення ***M*** на вхід функції хешування та розраховує його хеш. Далі ***Аліса*** додає хеш до повідомлення та відправляє ***Бобу***. Він отримує ***M+H*** та пропускає ***M*** через функцію хешування ***H(M)***, тобто. розраховує власний хеш ***H****. Наразі ***Боб*** порівнює свій хеш ***H**** з хешом ***H***., отриманим від ***Аліси***, і, якщо ***H**** = ***H***, то повідомлення ***M*** не змінювалося.

Існує безліч алгоритмів хешування з різними властивостями (розрядність, обчислювальна складність, криптостійкість тощо). Вибір тієї чи іншої хеш-функції визначається специфікою завдання, що вирішується.

У загальному випадку однозначної відповідності між вихідними даними та хеш-кодом немає через те, що кількість значень хеш-функцій менше, ніж варіантів вхідного повідомлення. Так як ***n*** << ***q*** **,** існує безліч різних повідомлень, що дають однакові хеш-коди - так звані колізії. Імовірність виникнення колізій грає важливу роль оцінці якості хеш-функцій.

Криптографічна хеш-функція повинна відповідати трьома властивостями,

щоб вважатися безпечною. Це: стійкість до колізії, та стійкість до пошуку першого та другого прообразу.

Стійкість до колізії: неможливо знайти два різні входи, що виробляють хеш, аналогічний висновку.

Колізія відбувається, коли різні вхідні дані виробляють однаковий хеш. Таким чином, хеш-функція вважається стійкою до колізій доти, доки хтось не виявить колізію. Зверніть увагу, що колізії завжди будуть існувати для будь-якої з хеш-функцій у зв'язку з нескінченною кількістю вхідних даних та обмеженою кількістю висновків.

Таким чином, хеш-функція стійка до колізії, коли ймовірність її виявлення настільки мала, що для цього знадобляться мільйони років обчислень. З цієї причини, незважаючи на те, що не існує хеш-функцій без колізій, деякі з них настільки сильні, що можуть вважатися стійкими (наприклад, SHA-256).

Серед різних алгоритмів SHA групи SHA-0 та SHA-1 більше не є безпечними, оскільки в них було виявлено колізії. В даний час тільки групи SHA-2 і SHA-3 вважаються найбезпечнішими та стійкішими до колізій.

Стійкість до пошуку першого прообразу: відсутність способу або алгоритму зворотного відновлення хеш-функцію (перебування входу по заданому виходу).

Ця властивість тісно взаємопов'язана з концепцією односторонніх функцій. Хеш-функція вважається стійкою до пошуку першого прообразу, доки існує дуже низька ймовірність того, що хтось зможе знайти вхід, за допомогою якого можна буде згенерувати певний висновок.

Важливо, що ця властивість відрізняється від попереднього, оскільки зловмиснику потрібно вгадувати вхідні дані, спираючись на певний висновок. Такий вид колізії відбувається, коли хтось знаходить два різні входи, які виробляють той самий код на виході, не надаючи значення вхідним даним, які для цього використовувалися.

Властивість стійкості до пошуку першого прообразу є цінним для захисту даних, оскільки простий хеш повідомлення може довести його справжність без необхідності розголошення додаткової інформації. На практиці багато постачальників послуг і веб-додатків зберігають і використовують хеші, згенеровані з паролів, замість того, щоб користуватися ними в текстовому форматі.

Стійкість до пошуку другого прообразу: неможливо знайти будь-який другий вхід, який перетинався б з першим.

Для спрощення можна сказати, що цей вид стійкості знаходиться десь між двома іншими властивостями. Атака знаходження другого прообразу полягає в знаходженні певного входу, за допомогою якого можна згенерувати висновок, що спочатку утворювався за допомогою інших вхідних даних, які були відомі.

Іншими словами, атака знаходження другого прообразу включає виявлення колізії, але замість пошуку двох випадкових входів, які генерують один і той же хеш, атака націлена на пошук вхідних даних, за допомогою яких можна відтворити хеш, який спочатку був згенерований за допомогою іншого входу .

Отже, будь-яка хеш-функція, стійка до колізій, також стійка і до подібних атак, оскільки остання завжди має на увазі колізію. Тим не менш, все ще залишається можливість здійснення атаки знаходження першого прообразу на функцію стійку до колізій, оскільки це передбачає пошук одних вхідних даних за допомогою одного виводу.

1.3. Криптографічні хеш функції. Стандарт SHA-2

SHA-2 (Secure Hash Algorithm Version 2 — друга версія безпечного алгоритму хешування) — сімейство назва односторонніх хеш-функцій: SHA-224, SHA-256, SHA-384 і SHA-512.

Хеш-функції SHA-2 розроблені Агентством національної безпеки США

(ANB) та опубліковані Національним інститутом стандартів та технологій (NIST) у у серпні 2002 року. Хеш-функції сімейства SHA-2 побудовані на основі структури Меркла-Дамгора.

Вихідне повідомлення після доповнення розбивається на блоки, кожен блок – на 16 слів. Алгоритм пропускає кожен блок повідомлення через цикл із 64 або 80 ітераціями (раундами). На кожній ітерації 2 слова перетворюються, функцію перетворення задають інші слова. Результати обробки кожного блоку складаються, сума є значенням хеш-функції. Тим не менш, ініціалізація внутрішнього стану здійснюється результатом обробки попереднього блоку. Тому незалежно обробляти блоки та складати результати не можна.

Хеш-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, SHA-224	256/224	256 (8 × 32)	512	$2^{64} - 1$	32	64
SHA-512, SHA-384, SHA-512/256, SHA-512/224	512/384/256/224	512 (8 × 64)	1024	$2^{128} - 1$	64	80

1.4. Криптографічні хеш функції. Стандарт SHA-3

SHA-3 (інша назва Keccak) — алгоритм хешування змінної розрядності. В 2015 року алгоритм затверджено та опубліковано в якості стандарту. Алгоритм SHA-3 побудований за принципом криптографічної губки (дана структура криптографічних алгоритмів була запропонована авторами алгоритму Keccak).

Губка (spomge) – це спеціальна ітеративна конструкція для створення функції з довільною довжиною змінною на вході та довільною довжиною змінної

на виході на основі перетворень перестановки. Сутність губки поягає в двофазному віиконанні алгоритму, в якому дані на першій фазі «всмоктується» в губку (поглинання), а потім результат на другій фазі «видавлюється» до результату (стиснення). Губка — це проста ітерована конструкція для побудови функції F зі змінною довжиною вхідних даних і довільною вихідною довжиною на основі перетворення або перестановки f фіксованої довжини, що працює з фіксованим числом b бітів . Тут b – ширина стану, $b = r + c$, де r – "швидкість", розмір записуваної та читаної частини, а де c – "ємність", розмір частини, не порушеної введенням/виводом.

Спочатку всі b біт стану ініціалізуються в нуль. Вхідне повідомлення заповнюється і розрізається на блоки по r бітів.

• На фазі поглинання r-бітові блоки вхідних повідомлень перетворюються за допомогою операції XOR на перші r бітів стану, доповнюються бітами ємності та перемежуються із застосуванням функції f. Коли всі блоки повідомлень оброблені, конструкція губки перемикається на фазу стискання.

• У фазі стиснення перші r бітів стану повертаються як вихідні блоки, перемежовані із застосуванням функції f . Кількість вихідних блоків вибирається користувачем за бажанням.

Останні c біти стану ніколи не впливають безпосередньо на вхідні блоки і ніколи не виводяться під час фази стиснення.

Сімейство SHA-3 складається з чотирьох криптографічних хеш-функцій з фіксованим розміром хешу (SHA3-224, SHA3-256, SHA3-384, SHA3-512) і двох вихідних функцій (XOFs), що розширюються, з іменами SHAKE128 і SHAKE256, кожна з яких заснована на екземплярі алгоритмів Keccak. При цьому для SHA-3 обрано $b = 1600$.

Хеш-функція	Довжина дайджесту повідомлення (біт / байт)	Швидкість ***r*** = розмір блоку (біт)	Ємність ***c*** (біт)
SHA3-224	224 / 28	1152	448
SHA3-256	256 / 32	1088	512
SHA3-384	384 / 48	832	768
SHA3-512	512 / 64	576	1024
SHAKE 128 (d)	***d***	1344	256
SHAKE 256 (d)	***d***	1088	512

d – довільна довжина дайджесту.

2 МОДУЛЬ HASHLIB. БАЗОВІ АЛГОРИТМИ КРИТОГРАФІЧНОГО ХЕШУВАННЯ

Модуль реалізує загальний інтерфейс для алгоритмів безпечного хешування та дайджесту повідомлень. Надає FIPS алгоритми безпечного хешування SHA1, SHA224, SHA256, SHA384 та SHA512, а також алгоритм RSA MD5 (визначений в Інтернеті RFC 1321). Терміни «**безпечний хеш**» та «**дайджест повідомлення**» взаємозамінні.

Для кожного типу хешу існує один метод конструктора. Всі вони повертають хеш-об'єкт з однаковим простим інтерфейсом. Наприклад: можна скористатися конструктором sha256() для створення хеш-об'єкта SHA-256. Тепер можна заповнити цей об'єкт байтоподібними об'єктами (зазвичай bytes) за допомогою методу update(). У будь-який момент можна запросити у нього дайджест конкатенації даних, переданих йому досі, за допомогою методів digest() або hexdigest().

В модулі завжди присутні конструктори для наступних хеш-алгоритмів: sha1(), sha224(), sha256(), sha384(), sha512(), blake2b() та blake2s(). Алгоритм md5() також зазвичай доступний. Також можуть бути доступні додаткові алгоритми в залежності від бібліотеки OpenSSL. В версії модуля 3.6 додані алгоритми групи SHA3 (Keccak) та SHAKE: sha3_224(), sha3_256(), sha3_384(), sha3_512(), shake_128(), shake_256(). Також в цій версії додані blake2b() та blake2s().

2.1 Базові атрибути модуля.

- hashlib.**algorithms_guaranteed.** Повертає перелік імен хеш-алгоритмів, які гарантовано підтримуються модулем на всіх платформах.
- hashlib.**algorithms_available**. Повертає перелік імен хеш-алгоритмів, які доступні для поточного Python інтерпретатора.

2.2 Постійні атрибути хеш обєктів, що повертаються конструкторами:

- **hash.digest_size**. Розмір результуючого хешу в байтах.
- **hash.block_size**. Внутрішній розмір блоку алгоритму хешування у байтах.
- **hash.name.** Канонічне ім'я цього хешу, Використовується як параметр для **new()** для створення іншого хешу цього типу.

2.3 Методи хеш обєктів:

- **hash.update(**data**)**. Оновлює хеш-об'єкт байтовим об'єктом (data – послідовність байтів). Повторні виклики еквівалентні одному виклику з об'єднанням всіх аргументів. Тобто **m.update**(a); **m.update**(b) еквівалентно **m.update**(a+b) .
- **hash.digest()**. Повертає дайджест даних, переданих на даний момент методом **update**(). Це є об'єкт (типу digest_size) із розміром digest_size, який може містити байти у всьому діапазоні від 0 до 255.
- **hash.hexdigest()**. Як і метод **digest()** за винятком того, що дайджест повертається як рядковий об'єкт подвійної довжини, що містить лише шістнадцятирічні цифри. Типово використовується для безпечного обміну хешами електронною поштою або в інших небінарних середовищах.
- **hash.copy().** Повертає копію (клон) хеш-об'єкту. Це можна використовувати для ефективного обчислення дайджестів даних, що розділяють загальний початковий підрядок.

2.4 Особливості Shake()

Алгоритми shake_128() та shake_256() забезпечують дайджести змінної довжини. Таким чином, ці алгоритми потребують параметр digest_size, що визначає довжину дайджесту. Максимальна довжина дайджесту shake алгоритмами не обмежується.

- **shake.digest(**digest_size**)**. Повертає дайджест даних, переданих на даний момент методом **update**(). Це є об'єкт (типу digest_size) із розміром digest_size, який може містити байти у всьому діапазоні від 0 до 255.
- **shake.hexdigest(**digest_size**)**. Як і метод **digest()** за винятком того, що дайджест повертається як рядковий об'єкт подвійної довжини, що містить лише шістнадцятирічні цифри.

2.5 Створення хеш обєкту

Модуль **heshlib** надає два способи створення хеш обєкту обраного типу.

Перший спосіб

m = hashlib.hash. Цей вираз свіорює хеш обєкт **m** конструктором **hash**. Наприклад

my_hash = hashlib.sha1() створює хеш-об'єкт SHA-1 з імям **my_hash**.

Дргий спосіб – за допомогою загального конструктору new()

hashlib.new(*name*, [*data*,]*, *usedforsecurity=True*)

Конструктор **new** приймає назву ***name*** потрібного алгоритму хешування як свій перший параметр (рядок). Наприклад

my_new_hash = hashlib.new(`sha1`) також створює хеш-об'єкт SHA-1 з імям **my_new_hash**.

Конструктор **new** надає доступ до всіх перерахованих вище хешів, а також до будь-яких інших алгоритмів, які може запропонувати бібліотека OpenSSL, що використвується опеаційною системою. Рекомендовано, при можливості, віддавати перевагу першому спосібу.

2.5 Типові приклади

- Визначення імен хеш-алгоритмів, які гарантовано підтримуються **hashlib.**

1	**>>> import hashlib**
2	**>>> hashlib.algorithms_guaranteed**

3	{'sha512', 'blake2b', 'sha3_384', 'shake_128', 'sha1', 'sha224', 'sha3_224', 'sha256', 'shake_256', 'sha3_256', 'md5', 'blake2s', 'sha384', 'sha3_512'}

Надається перелік хеш-алгоримів модуля **hashlib**.

- Визначення імен хеш-алгоритмів, які підтримуються поточною платформою

1	**>>> import hashlib**
2	**>>> hashlib. algorithms_available**
3	{'sha1', 'sha512_224', 'sha512_256', 'md4', 'md5', 'sha512', 'blake2b', 'sha3_384', 'ripemd160', 'md5-sha1', 'sha384', 'sha3_512', 'whirlpool', 'mdc2', 'sha224', 'sha3_224', 'shake_256', 'sm3', 'shake_128', 'sha256', 'sha3_256', 'blake2s'}

Зверніть увагу, кількість достпних хеш-агоритмів збільшилась.

- Створення хеш обєкту та визначення дайджесту

1	**import hashlib**
2	***my_sha1*** **= hashlib.sha1()**
3	***my_sha1*****.update(b"*First message*")**
4	***first_digest*** **=** ***my_sha1*****.digest()**
5	**print(*first_digest*)**
6	b'\xff\xc1$\xfd\xf1\xd8\x0b&Z\xb0\x80\xb2\x08\x1e\xb1\x8a\xec\xa2\xd0\x8a'

Рядок 1. Імпорт бібліотеки **hashlib**

Рядок 2. Створення хеш обєкту SHA-1 з *імям* ***my_sha1***

Рядок 3. Визначає дайджест обєкту ***my_sha1*** для рядка "***First message***"

Рядок 4. Поточний дайджест привоюється змінній ***first_digest***

Рядок 5. Друк дайджесту

Рядок 6. Надрукований дайджест (бінарний).

- Визначення дайджесту в шістнадцятковій системі (продовженя попереднього прикладу)

7	***first_digest_hex*** **=** ***my_sha1*****.hexdigest()**
8	**print(*first_digest_hex*)**
9	ffc124fdf1d80b265ab080b2081eb18aeca2d08a

Рядок 7. Поточний дайджест в 16-вій системі привоюється змінній

***first_digest_hex*.**

Рядок 8. Друк дайджесту

Рядок 9. Надрукований дайджест в шістнадцятковій системі. Важливо: дайджести (6) та (9) це одне й теж у різних представленнях.

- Визначення парамерів дайджесту (продовженя попереднього прикладу)

10	**print(*my_sha1*.name)**
11	sha1
12	**print(*my_sha1*.digest_size)**
13	20
14	**print(*my_sha1*.block_size)**
15	64

Рядок 10. Визначення та друк назви хеш-алгоритму обєкту ***my_sha1.***

Рядок 12. Визначення та друк розміру дайджесту (в байтах).

Рядок 14. Визначення та друк розміру блоку, що використвує хеш-алгритм.

- Розширення тексту - оновлення хешу (продовженя попереднього прикладу)

16	***my_sha1*.update(b"SECOND_MESSAGE")**
17	***second_digest* = *my_sha1*.digest()** **print(*second_digest*)**
18	b'\x9az\xc5\xbe\x00\xd0\x08q\xbc\xdeq\xda\xdb\xe0[\xcfv\xec\xd7\x81'
19	***second_digest_hex* = *my_sha1*.hexdigest()** **print(*second_digest_hex*)**
20	9a7ac5be00d00871bcde71dadbe05bcf76ecd781

Рядок 16. Оновлення хеш-обєкту ***my_sha1.***

Рядки 17, 18. Визначення змінної та друк дайджесту в бінарній формі.

Рядки 19, 20. Визначення змінної та друк дайджесту в шістнадцятковій системі.

- Використання **hashlib** для текстів з кириліцею (unicode кодування)

1	**import hashlib**
2	***str*** = "ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ"
3	***my_sha1_ukr* = hashlib.sha1()**
4	***my_sha1_ukr*.update(*str*.encode())**
5	**print(*my_sha1_ukr*.digest())**

6	b'\xa0\x1aus{\x05a\xb08\xbdM\xf7\x04\xcfUw\xcf\x04\xf5\xee'
7	**print(*my_sha1_ukr*.hexdigest())**
8	a01a75737b0561b038bd4df704cf5577cf04f5ee

Рядок 1. Імпорт бібліотеки **hashlib**

Рядок 2. Візначення рідкової змінної ***str*** з україномовним текстом.

Рядок 3. Створення хеш обєкту SHA-1 з імям ***my_sha1_ukr***

Рядок 4. Визначає дайджест обєкту ***my_sha1_ukr*** для рядка ***str*** **.** Важлива примітка : метод encode() перекодує рядок до формату UTF-8 (фактично ACII)

Рядок 5. Друк дайджесту в бінарній формі

Рядок 7. Дрку дайджесту в шістнадцятковій системі.

Повний приклад формуівання хеш дайджестів для великорозмірних файлів надано в додатку А.

3 ЗАВДАННЯ ДО КУРСОВОЇ РОБОТИ

1. Надати опис **базової функції** хешування відповідно до варіанту індивідуального завдання.
2. Створити програмний додаток визначення дайджесту відкритого повідомлення довільного розміру за ***першим варіантом*** базової хеш функції.
3. Створити програмний додаток визначення дайджесту відкритого повідомлення довільного розміру за ***другим варіантом*** базової хеш функції.
4. Знайти дайджест за першим та другим варіантом базової функції для файлу **англомовного** довільного тексту, що має розмір не меньш як 10 Кбайт.
5. Знайти дайджест за першим та другим варіантом базової функції для файлу **україномовного** довільного тексту, що має розмір не меньш як

10 Кбайт.

Таблиця варіантів

	Базова функція хешування	Перший варіант базової хеш-функції	Другий варіант базової хеш-функції
1	**Sha 2**	**sha2_224**	**sha2_256**
2	**Sha 2**	**sha2_384**	**sha2_512**
3	**Sha 3**	**sha3_224**	**sha3_256**
4	**Sha 3**	**sha3_384**	**sha3_512**
5	**Sha 3**	**shake_128**	**shake_256**
6	**blake**	**blake2b**	**blake2s**

4 ПОРЯДОК ВИКОНАННЯ РОБОТИ

1. Ознайомитись з теоретичним матеріалом щодо криптографічних хеш функцій
2. Надати опис базової криптографічної хеш-функції відповідно варіанту.
3. Створити два текстових файли. Файл англійською мовою обсягом не менш 10 Кбайт та файл українською мовою обсягом не менш 10 Кбайт
4. Розробити скрипт на мові Python для розрахунку дайджесту для текстів англійською та українською мовами за першим варіантом базової хеш-функції відповідно Вашого завдання.
5. Розробити скрипт на мові Python для розрахунку дайджесту для текстів англійською та українською мовами за другим варіантом базової хеш-функції відповідно Вашого завдання.
6. Порівняти отримані хеш дайджести та надати висновок з криптостійкості отриманих хеш дайджестів.

5 ПОРЯДОК КОНТРОЛЮ Й ПРИЙОМУ

Курсова робота виконується 16 тижнів. Пояснювальна записка до роботи надається на перевірку викладачам не менш чим **за 3 робочі дні** до дати захисту.

Захист відбувається в присутності комісії в складі 2-3 осіб і включає:

а) доповідь, що відбиває всі етапи виконання курсової роботи;

б) презентацію роботи розробленого програмного модуля обчислення хеш дайджесту;

в) відповіді на запитання комісії.

Шкала оцінювання виконання курсового проекту

Теоретичне обґрунтування	Виконання програмного опису	Оформлення пояснювальної записки	Виступ з презентацією	Максимальна сума балів
40	**20**	**20**	**20**	**100**

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Базова

1. Остапов С.Е. Технології захисту інформації. Навчальний посібник для студентів вищих навчальних закладів. / С.Е. Остапов, С.П. Євсеєв, О.Г. Король. — Львів, «Новий світ 2000», 2020. — 678 с.
2. Бурячок В.Л. Інформаційний та кіберпростори: проблеми безпеки, методи та засоби боротьби. [Підручник]. / В.Л. Бурячок, Г.М. Гулак, В.Б. Голубко. — Львів, «Магнолія 2006», 2021. — 448 с.
3. Остапов С.Е. Технології захисту інформації. Навчальний посібник для студентів вищих навчальних закладів. / С.Е. Остапов, С.П. Євсеєв, О.Г. Король. — Харків, Вид. ХНЕУ, 2013. — 476 с.
4. Остапов С.Е. Основи криптографії. / С.Е. Остапов, Л.О. Валь. — Чернівці, Книги-XXI, 2008. — 544 с.
5. Криптографічна хеш-функція
   https://uk.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D1%96%D1%87%D0%BD%D0%B0_%D0%B3%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F
6. Стандарт SHA-2
   https://uk.wikipedia.org/wiki/SHA-2
7. Стандарт SHA-3
   https://uk.wikipedia.org/wiki/SHA-3
   https://en.wikipedia.org/wiki/SHA-3
8. Криптографічна сіль
   https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BB%D1%8C_(%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F)
9. Модуль Hashlib
   https://docs.python.org/3/library/hashlib.html

Додаткова

10. Рибальський О.В., Захист інформації в інформаційно-комунікаційних системах. Навчальний посібник для курсантів ВНЗ МВС України. /

О.В. Рибальський, В.Г. Хахановський, В.А. Кудінов, В.М. Смаглюк. — К.: Вид. Національної академії внутріш. справ, 2012. — с. 118.

11. Симетричне шифрування - це спосіб шифрування даних, при якому той самий ключ використовується і для кодування, і для відновлення інформації.

12. Шнайер Б. Прикладная криптография. Протоколи, алгоритмі, тексті на мові Си. — М.: Изд-во ТРИУМФІ, 2003. — с. 816.

Методична

1. Методичні вказівки і завдання до виконання практичних робіт по курсу «Емпіричні методи кібербезпеки» (для студентів, що навчаються за спеціальністю 125 «Кібербезпека» всіх форм навчання) / укладач: проф. Дмитрієва О.А. - Покровськ: ДонНТУ, 2020 р. – 80 с. http://ea.donntu.edu.ua:8080/jspui/handle/123456789/32378

ДОДАТОК А

ПРИКЛАД PYTHON СКРИПТУ

А.1 Текст Python скриту

```
"""
Created on 01/02/2022
@author: BASHKOV E.A.

КУРСОВА РОБОТА
ПРИКЛАД Python СКРИПТА

ДЕМОНСТРАЦІЯ ВИКОРИСТАННЯ ХЕШ АЛГОРИТМУ SHA-1
МОДУЛЬ

HASHLIB Secure hashes and message digests
https://docs.python.org/3/library/hashlib.html

"""

import hashlib

print('----------------------------')
print('\n ГАРАНТОВАНІ хеш-функції
HashLib\n',hashlib.algorithms_guaranteed)
print('\n Доступні хеш - функції \n', hashlib.algorithms_available)
print('----------------------------\n')


# Шлях до директорії з файлам
message_path = 'Ваш шлях до директорії з файлами'

# Text file name ENGLISH
plain_message_file = message_path + 'Ім'я Вашого файлу АНГЛ мова'

# Відкриття файлу
file_in = open(plain_message_file, "rb")
# Читання файлу
plain_message_text = file_in.read()
# Закриття файлу
file_in.close()
```

```
print('------------ ОБСЯГ АНГЛІЙСЬКОГО ТЕКСТУ ---------------')
print(len(plain_message_text))
print('-------------------------------------\n')


# SHA1
result_1 = hashlib.sha1()
result_1.update(plain_message_text)
print("CASE 1 ENGL  SHA1 HASH:---------------- ")
print('Дайджест бінарний ', result_1.digest())
print('Дайджест hex ', result_1.hexdigest())
print('Розмір хешу в байтах ', result_1.digest_size)
print('Розмір блоку хешу в байтах',result_1.block_size)
print ("\r")

# Text file name UKR
plain_message_file = message_path + Ім'я Вашого файлу УКР МОВА'
file_in = open(plain_message_file, "rb")
plain_message_text = file_in.read()
file_in.close()

print('------------ ОБСЯГ УКРАЇНСЬКОГО ТЕКСТУ ---------------')
print(len(plain_message_text))
print('-------------------------------------\n')

# SHA1
result_2 = hashlib.sha1()
result_2.update(plain_message_text)
print("CASE 2 UKR  SHA1 HASH:---------------- ")
print('Дайджест бінарний ', result_2.digest())
print('Дайджест hex ', result_2.hexdigest())
print('Розмір хешу в байтах ', result_2.digest_size)
print('Розмір блоку хешу в байтах',result_2.block_size)
print ("\r")
```

А.2 Результат роботи Python скриту

```
ГАРАНТОВАНІ хэш-функції HashLib
 {'blake2s', 'sha384', 'sha3_256', 'blake2b', 'shake_128', 'sha512', 'sha256',
'sha3_384', 'sha3_512', 'shake_256', 'sha1', 'md5', 'sha224', 'sha3_224'}

Доступні хеш - функції
 {'sha256', 'sha512_256', 'md4', 'blake2b', 'shake_128', 'ripemd160',
'sha3_512', 'sha1', 'sha3_384', 'blake2s', 'sha3_256', 'shake_256',
'sha512_224', 'sha224', 'mdc2', 'md5-sha1', 'sha384', 'sha512', 'md5',
'whirlpool', 'sm3', 'sha3_224'}


ОБСЯГ АНГЛІЙСЬКОГО ТЕКСТУ ----------
19324
-------------------------------------

CASE 1 ENGL  SHA1 HASH:----------
Дайджест бінарний
b'`\xf5{\x03\x99}M\x9c\xfas\xe2\xe9\x95!C!\xe8\x04\xd1p'
Дайджест hex  60f57b03997d4d9cfa73e2e995214321e804d170
Розмір хешу  в байтах  20
Розмір блоку хешу в байтах 64

---------- ОБСЯГ УКРАЇНСЬКОГО ТЕКСТУ ----------
11304
-------------------------------------

CASE 2 UKR  SHA1 HASH:----------
Дайджест бінарний
b'\xcd\x8d}^\x18\x1c>\xcd\xc8g\xdc\xa1\x1a\x1c~\x84\xa4\xc7`\xa7'
Дайджест hex  cd8d7d5e181c3ecdc867dca11a1c7e84a4c760a7
Розмір хешу в байтах  20
Розмір блоку хешу в байтах 64
```

ДОДАТОК Б

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
КАФЕДРА ПРИКЛАДНОЇ МАТЕМАТИКИ І ІНФОРМАТИКИ

ЗАТВЕРДЖУЮ
зав. кафедри ПМІ,
д.т.н., проф. Дмитрієва О.А.

_____

__ _____ 20__ р.

Т Е Х Н І Ч Н Е З А В Д А Н Н Я
до курсової роботи з дисципліни
«Методи та засоби криптографічного захисту інформації»

за темою
«**Криптографічне хешування. Методи та алгоритмічна підтримка**»

(для студентів спеціальності 125 Кібербезпека
всіх форм навчання)

Керівник:
д.т.н., проф. каф.
ПМІ Башков Є.О.

_____
_____20__ р.

Виконавець:
студент гр. _____

_____
__ _____ 20__ р.

Покровськ – 20__ р.

ВСТУП

Курсова робота виконується на підставі навчального плану підготовки студентів за освітньо-кваліфікаційним рівнем «бакалавр» та «Технічного завдання до курсової роботи» за дисципліною «Методи та засоби криптографічного захисту інформації» на тему: за темою «Криптографічне хешування. Методи та алгоритмічна підтримка» для студентів, що отримують освіту в галузі знань 12 «Інформатика та обчислювальна техніка», спеціальності 125 Кібербезпека.

ЗАВДАННЯ НА КУРСОВУ РОБОТУ

Загальне завдання на курсову роботу передбачає розробку програмного скрипту на мові Python, який реалізує визначені індивідуальним завданням стандартні хеш алгоритми обчислення дайджестів текстових повідомлень. Варіанти індивідуального завдання наведені в таблиці А.1.

Індивідуальне завдання:

1. Опанувати знаннями та надати опис [***базової функції***] хешування відповідно до варіанту індивідуального завдання.
2. Створити програмний додаток визначення дайджесту відкритого повідомлення довільного розміру за [***першим варіантом***] базової хеш функції.
3. Створити програмний додаток визначення дайджесту відкритого повідомлення довільного розміру за [***другим варіантом***] базової хеш функції.
4. Знайти дайджест за першим та другим варіантом базової функції для файлу **англомовного** довільного тексту, що має розмір не менш як 10 Кбайт.
5. Знайти дайджест за першим та другим варіантом базової функції для файлу **україномовного** довільного тексту, що має розмір не менш як 10 Кбайт.

Таблиця А.1.

Варіанти індивідуального завдання

	Базова функція хешування	**Перший варіант базової хеш-функції**	**Другий варіант базової хеш-функції**
1	**Sha 2**	**sha2_224**	**sha2_256**
2	**Sha 2**	**sha2_384**	**sha2_512**
3	**Sha 3**	**sha3_224**	**sha3_256**
4	**Sha 3**	**sha3_384**	**sha3_512**
5	**Sha 3**	**shake_128**	**shake_256**
6	**blake**	**blake2b**	**blake2s**

ЗМІСТ ТА ЕТАПИ РОЗРОБКИ

При виконанні курсової роботи студент повинен:

1. Ознайомитись з теоретичним матеріалом щодо криптографічних хеш функцій.
2. Надати опис базової криптографічної хеш-функції.
3. Створити два текстових файли. Файл англійською мовою обсягом не менш 10 Кбайт та файл українською мовою обсягом не менш 10 Кбайт
4. Розробити скрипт на мові Python для розрахунку дайджесту для текстів англійською та українською мовами за першим варіантом базової хеш-функції.
5. Розробити скрипт на мові Python для розрахунку дайджесту для текстів англійською та українською мовами за другим варіантом базової хеш-функції.
6. Порівняти отримані хеш дайджести та надати висновок з криптостійкості отриманих дайджестів.

Таблиця А.2.

Графік виконання курсового проекту

№	Найменування етапу	Строк виконання	
		тиждень	дата
1	Видача завдання на курсовий проект. З'ясування завдання.	1-2	
2	Опанування математичними співвідношеннями визначеної базової функції хешування.	3-5	
	Опанування бібліотекою HashLib	6	
3	Проектування Python скрипту для обчислення хеш дайджестів	7	
4	Розробка та тестування Python скриту	8 - 9	
6	Оформлення пояснювальної записки	10 - 11	
7	Оформлення презентації	12 - 13	
8	Захист курсового проекту	14 - 16	

Виконавець

студент гр. КІБ-_____

_____

«____»_____ 20__ р.