Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи та алгоритми хешування паролів на платформі .NET

(тема)

Виконав:

студент II курсу, групи СПм-21-1

Моруга Д.І.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Федорченко В.М.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп'ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Морузі Дмитру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи та алгоритми хешування паролів на платформі .NET

затверджена наказом по університету від " 07 " листопада 2022 р. № 1454 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2022 р.

3. Вхідні дані до роботи 1) рекомендації організації OWASP, щодо зберігання паролів;
2) документація програмної платформи .NET;
3) документація мови програмування C#;
4) інтегроване середовище розробки: Microsoft Visual Studio.

4. Перелік питань, що потрібно опрацювати у роботі
1) вбудовані у програмну платформу .NET алгоритми хешування;
2) реалізації актуальних алгоритмів хешування на програмній платформі .NET;
3) аналіз обраних алгоритмів;
4) проведення експериментальних досліджень;
5) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Хеш-функція; Загальний вигляд функції формування ключа; Вбудовані функції хешування; Блок схема роботи алгоритму MD5; Блок схема роботи алгоритму SHA-2; Блок схема алгоритму PBKDF2; Реалізації алгоритму Bcrypt у бібліотеках програмної платформи .NET; Реалізації алгоритму Scrypt у бібліотеках програмної платформи .NET; Реалізації алгоритму Argon2 у бібліотеках програмної платформи .NET; Обчислені хеш-значення та метрики обчислень; Середні швидкості хешування паролів для всіх наборів.

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз стану проблеми	07.11.2022-10.11.2022	
2	Аналіз методів хешування	11.11.2022-15.11.2022	
3	Аналіз вбудованих алгоритмів	16.11.2022-20.11.2022	
4	Аналіз сторонніх алгоритмів	21.11.2022-25.11.2022	
5	Проведення експерименту	26.11.2022-31.11.2022	
6	Оформлення матеріалів кваліфікаційної роботи	01.12.2022-05.12.2022	
7	Подання кваліфікаційної роботи керівникові	06.12.2022-07.12.2022	
	та її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	08.12.2022-12.12.2022	
	вання		

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Федорченко В.М.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 65 с., 21 рис., 2 табл., 3 дод., 25 джерел.

АЛГОРИТМИ ХЕШУВАННЯ, MD5, SHA256, BCRYPT, SCRYPT, ARGON2, .NET.

Метою кваліфікаційної роботи є порівняння і аналіз алгоритмів хешування вбудованих та доступних у бібліотеках програмної платформи .NET для захисту паролів, як основного аспекту при автентифікації користувачів. У результаті аналізу повинні бути визначені алгоритми, які мають найкращі характеристики швидкості хешування та часу необхідного для подолання захисту.

У ході виконання кваліфікаційної роботи було проведено аналіз і порівняння алгоритмів хешування доступних на програмній платформі .NET. За допомогою обраних алгоритмів був виконаний експеримент для порівняння їх характеристик.

ABSTRACT

Master’s thesis: 65 pages, 21 figures, 2 tables, 3 appendices, 25 sources.

HASHING ALGORITHMS, MD5, SHA256, BCRYPT, SCRYPT, ARGON2, .NET.

The major goal of this thesis is to compare and analyze hashing algorithms built-in and available in libraries of the .NET Software Platform for password protection, as the main aspect of user authentication. As a result of the analysis, should be identified algorithms that have the best characteristics of the hashing speed and the time required to overcome the protection.

During the qualification work, the analysis and comparison of hashing algorithms available on the .NET Software Platform was conducted. Using the selected algorithms, an experiment will be performed to compare their characteristics.

The result of this work is to establish the best hashing algorithm, considering the hashing speed and protection provided.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASIC – інтегральна схема для специфічного застосування (англ., Application-specific integrated circuit)

FPGA – програмована користувачем вентильна матриця (англ., Field-Programmable Gate Array)

GPU – графічний процесор (англ., Graphics processing unit)

KDF – функція формування ключа (англ., Key Derivation Function)

SHA – алгоритми безпечного хешування (англ., Secure Hash Algorithm)

АХ – алгоритм хешування

КС – комп’ютерна система

КХФ – криптографічна хеш-функція

ПП – програмна платформа

ХФ – хеш-функція

ВСТУП

Підприємства надають великого значення даним, які вони збирають про своїх клієнтів. За допомогою цієї інформації компанії можуть розсилати цільову рекламу, прогнозувати тенденції продажів і покращувати свої продукти. Для багатьох користувачів така інформація є конфіденційною, тому її захист є важливою характеристикою багатьох систем, ненадійний захист призводить до недовіри та підозрілості до багатьох підприємств.

Веб-застосунки є найпопулярнішим засобом для надання послуг та збору інформації, для багатьох установ. Через велику розповсюдженість вони є привабливою ціллю для зловмисників. Це погіршується появою державних сервісів, “ДІЯ” та інших, які обробляють важливу конфіденціальну інформацію. Програмна платформа (ПП) .NET є популярним рішенням для розробки веб-застосунків. Це викликано тим, що у неї сильна галузева підтримка від Microsoft, у неї є якісні інструменти для розробників, і вона використовує кілька мов програмування.

Переважна більшість комп’ютерних систем (КС) для забезпечення безпеки даних використовує паролі [1]. Пароль являє собою запам’ятований користувачем секрет [3], що складається з декількох символів, що друкуються. Безпека паролів забезпечується за допомогою хешування та інших технік.

Алгоритм хешування (АХ) – це математична функція, яка спотворює дані і робить їх нечитабельними. АХ – це односторонні програми, тому вхідні дані не можуть бути розшифровані будь-ким іншим. Хешування завжди захищає вхідні дані, тому, навіть якщо хтось отримає доступ до сховища, де зберігаються хеш-значення, вхідні дані залишаться захищеними. Хешування також застосовується, для цифрових підписів та при індексуванні даних.

Хеш функціям (ХФ) присутні також і недоліки, через них для захисту

паролів у наш час більш активно використовують функції формування ключа (KDF), деякі з них мають в своїй основі ХФ або алгоритми шифрування.

В ПП .NET є вбудовані АХ такі, як MD5, SHA-1, SHA-2 і функція формування ключа (KDF) PBKDF2, яка використовується ASP.NET Core Identity для хешування за замовченням. Ці АХ були створені більш ніж 20 років тому для іншого рівня та характеру загроз.

Існують також більш сучасні альтернативні KDF які використовуються для хешування паролів, а саме Bcrypt, Scrypt та Argon2, ці АХ обіцяють захист від графічних процесорів (GPU), інтегральних схем для специфічного застосування (ASIC) та програмованих користувачем вентильних матриць (FPGA). Ці KDF не є вбудованими у ПП .NET, але їх реалізації можна отримати зі сторонніх бібліотек.

Через те що захист паролів є одним із основних аспектів захисту конфіденціальної інформації користувача дослідження цього механізму на ПП .NET є актуальною задачею.

Метою даної випускної роботи є аналіз і порівняння АХ, а саме ХФ та KDF, які можна використовувати на ПП .NET для ефективного захисту паролів.

Практична значимість роботи полягає в встановлені найбільш ефективного засобу захисту інформації при однофакторній автентифікації.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи хешування

КС використовуються в нашій повсякденній діяльності, завдяки ним користувачі отримують доступ до багатьох послуг. Однофакторна автентифікація, що складається з імені користувача та пароля, є найпоширенішим вибором для автентифікації користувачів в Інтернеті. Однак зловмисники використовують неоптимальні методи управління паролями, які дозволяють розкрити облікові дані користувачів, завдаючи шкоди як користувачам, так і постачальникам. У більшості таких випадків дані користувачів зберігалися у відкритому вигляді або просто оброблялися криптографічною хеш-функцією (КХФ). Такими ХФ є MD5 та SHA.

Класичні КХФ мають велику швидкість обчислення хеш-значення, але з часу їх створення обчислювальна потужність КС значно зросла, що не тільки зробило їх вразливим перед атаками перебором, але й дозволило знайти в них колізії. Також з’явилися загрози від GPU, ASIC та FPGA.

Методи хешування паролів застосовуються для зміцнення інформації, пов’язаної з користувачем. Саме тому у сучасних КС для отримання хеш-значення паролів використовуються KDF. Стандартизованою KDF в даний час є PBKDF2, тоді як інші широко використовувані схеми включають Bcrypt, Scrypt та Argon2.

1.2 Характеристики функцій та алгоритмів хешування

Функції хешування відіграють важливу роль в криптографії, оскільки вони володіють важливими характеристиками, які допомагають в аутентифікації даних і забезпечують безпеку конфіденційних даних, наприклад паролів. Такі ХФ відомі, як криптографічні і їх характеристики

включають в себе наступне [21]:

- необоротність;
- детермінізм;
- стійкість до зіткнень;
- лавинний ефект;
- швидкість.

Необоротність АХ. АХ є односторонніми функціями – тобто не можливо обчислити вхідні дані, використовуючи хеш-значення. Це означає, можна легко перетворити вхідні дані в хеш, але не можливо отримати вхідні дані з його хеш-значення, до тих пір, поки АХ не порушений. Це важливо, оскільки хеші використовуються для зберігання паролів на загальнодоступних серверах. Оскільки хешування незворотне, зловмисники не зможуть відновити пароль з хешу, навіть якщо вони отримають в свої руки базу даних з хеш-значеннями паролів. Саме тому паролі ніколи не зберігаються у відкритому вигляді.

Детермінізм АХ. Вихідна довжина всіх результатів АХ повинна бути однаковою, незалежно від довжини вхідних даних. Це зручно для виділення місця для хеш-значення в структурі даних, форматі файлу або полі мережевого протоколу, оскільки відомо, якої довжини хеш-значення. Це також допомагає запобігти зловмисникам знати, наскільки великим було вхідне значення, оскільки всі вихідні хеш-значення, незалежно від того, наскільки довгим або коротким є вхідне значення, мають фіксовану довжину і не змінюються.

Стійкість до зіткнень або колізії АХ. При хешуванні вважається, що колізія сталася, коли в результаті хешування різних вхідних даних були отримані ідентичні хеш-значення. Якщо про колізію стане відомо зловмисникам, то за допомогою неї буде можливо обійти захист КС. Це відомо як атака з хеш-колізією.

Ще одна проблема – райдужні таблиці. Зловмисники можуть створювати величезну кількість попередньо обчислених комбінацій вхідних

даних та їх хеш-значень. Ця таблиця дозволить швидко шукати вхідні дані.

Ось чому всі АХ повинні бути стійкі до колізій. Одним із способів зменшити колізії при хешуванні паролів і знизити ризик атак на райдужні таблиці є використання соління.

Лавинний ефект АХ. Особливістю хеш функцій є те що, навіть найменша зміна вхідних даних призводить до значної зміни вихідного хеш-значення. Це гарантує, що ніхто не зможе розшифрувати вихідний текст.

Швидкість АХ. Багато сфер використання хешування потребують від алгоритмів високої швидкості обчислення хеш-значення. Однак не всі функції хешування повинні бути швидкими. Деякі функціональні можливості вимагають, щоб функції хешування були повільними. Це необхідно аби зловмисникам було складніше використовувати метод перебору, або райдужних таблиць для отримання вхідних даних.

1.3 Хеш функції

ХФ були введені в криптологію наприкінці сімдесятих років як інструмент для захисту автентичності інформації [4]. КХФ – це чітко визначена процедура, яка приймає блок даних і генерує бітовий рядок фіксованої довжини, відомий як хеш-значення або дайджест [5].

Алгоритм ХФ розроблений так, щоб бути односторонньою функцією, яку неможливо інвертувати. Проте в останні роки кілька АХ були скомпрометовані. Це сталося, наприклад, з MD5 – широко відомою ХФ, розробленою як КХФ. Загальну схему ХФ наведено нижче (рисунок 1.1).

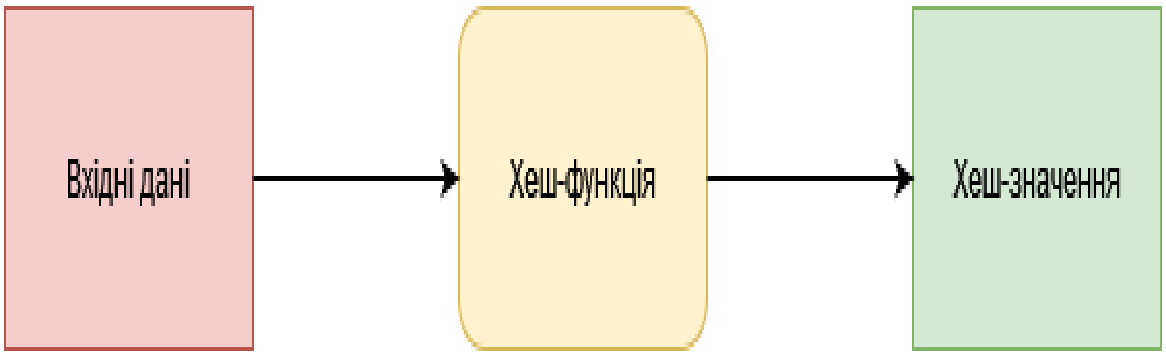

Рисунок 1.1 – Хеш-функція

1.4 Функції формування ключа

KDF використовується для генерації одного або декількох криптографічних ключів із закритого вхідного значення [6]. KDF повертають байти, які підходять для криптографічних операцій, з паролів або інших джерел даних з використанням псевдовипадкової функції. Різні KDF підходять для різних завдань, таких як [22]:

- отримання або розтягування криптографічного ключа;
- хешування паролів.

У криптографії методи розтягування ключів використовуються для того, щоб зробити потенційно слабкий ключ, зазвичай пароль або кодову фразу, більш захищеним від атаки методом перебору за рахунок збільшення ресурсів, часу і можливо простору, необхідних для перевірки кожного можливого ключа. Паролі або парольні фрази, створені людьми, часто бувають досить короткими або передбачуваними, що дозволяє легко зламувати паролі, а розтягування ключів призначене для того, щоб ускладнити такі атаки, ускладнюючи базовий етап спроби вибору єдиного пароля-кандидата. Розтягування ключа також підвищує безпеку в деяких реальних додатках, де довжина ключа була обмежена, за рахунок імітації більшої довжини ключа з точки зору зловмисника, який застосовує метод перебору.

При зберіганні паролів бажано використовувати алгоритм, що вимагає великих обчислювальних витрат. Користувачам потрібно буде обчислити його лише один раз, в той час як зловмисникам потрібно робити це мільярди разів. Ідеальне сховище паролів з KDF буде вимогливим як до обчислювальних ресурсів, так і до ресурсів пам'яті [25].

Різні KDF мають різні додаткові параметри, такі як коефіцієнт роботи та поля контекстної інформації.

Як мінімум KDF для хешування паролів мають наступні параметри:

- вхідні дані;
- соль;
- кількість ітерацій або фактор роботи.

KDF з регульованими робочими коефіцієнтами використовуються для зберігання паролів. KDF краще простої ХФ навіть з використанням солі, оскільки коефіцієнт роботи може бути обраний таким чином, щоб зробити дорогим вичерпний пошук в просторі ймовірних паролів, загальний вигляд KDF наведено нижче (рисунок 1.2).

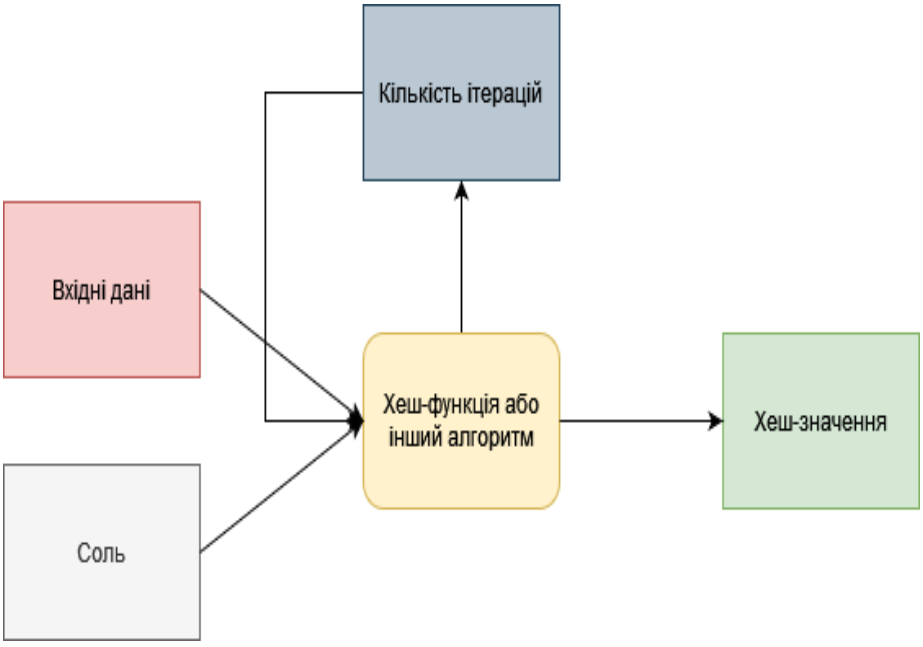

Рисунок 1.2 – Загальний вигляд функції формування ключа

1.5 Хешування за допомогою функцій формування ключа

Отже для ПП .NET є актуальною задачею встановити найбільш оптимальний алгоритм, для захисту паролів. Для цього будуть проаналізовані вбудовані алгоритми а саме ХФ MD5, SHA та KDF PDKDF2. Серед алгоритмів які не вбудовані у ПП.NET було обрано наступні KDF:

- Bcrypt;
- Scrypt;
- Argon2.

Дані АХ було обрано згідгно рекомендаціям організації OWASP, щодо захисту паролів [7].

Показники на основі який буде виконаний аналіз:

- мінімальна, максимальна та середня швидкість обчислення хеш значення;
- час необхідний на отримання вхідного значення методом повного перебору.

2 ВБУДОВАНІ АЛГОРИТМИ ХЕШУВАННЯ У ПРОГРАМНУ ПЛАТФОРМУ .NET

2.1 Вбудовані хеш-функції

Функції хешування представлені у вигляді класів в просторі імен System.Security.Cryptography ПП .NET:

- MD5;
- SHA1;
- SHA256;
- SHA384;
- SHA512.

Ці абстрактні класи є похідними від класу HashAlgorithm, детальна схема наведена нижче [23] (рисунок 2.1).

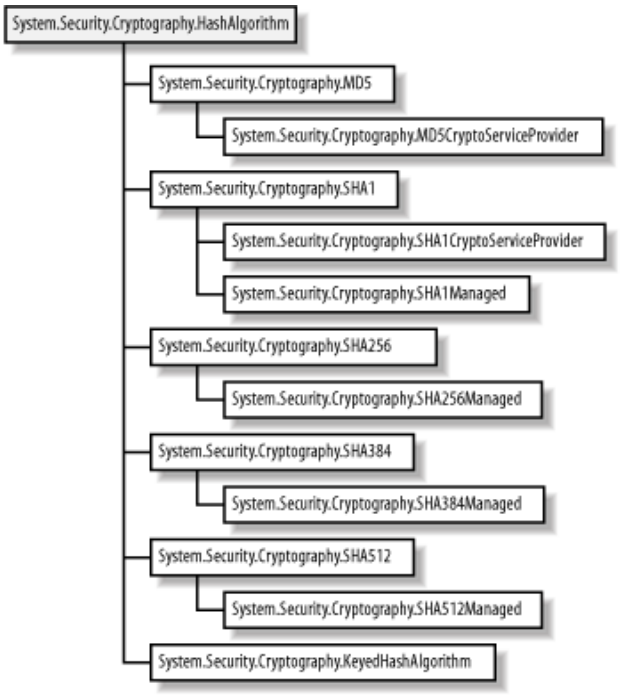

Рисунок 2.1 – Вбудовані функції хешування

Для обчислення хеш-значення за допомогою цих функцій необхідно виконати наступні кроки:

- створити екземпляр хеш-алгоритму. Можливо вибрати з MD5, SHA1, SHA256, SHA384 і SHA512;

- викликати метод ComputeHash, передавши масив байтів. Масив байтів може бути отриманий з будь-яких необроблених даних;

- метод ComputeHash після успішного виконання поверне масив байтів, який представлятиме хеш-значення.

Після успішного виконання ComputeHash масив байтів хеш-значення можна перетворити на інший тип для більш зручного зберігання.

2.1.1 Хеш-функція MD5

Алгоритм MD5 – це криптографічно порушена, але все ще широко використовувана ХФ, що видає 128-бітове хеш-значення. Хоча MD5 спочатку був розроблений для використання в якості КХФ, було виявлено, що він страждає від колізій, що є неприпустимо для КХФ. Його все ще можна використовувати в якості контрольної суми для перевірки цілісності даних, але тільки на випадок ненавмисного пошкодження. Він залишається придатним для інших некриптографічних цілей, наприклад, для визначення розділу для конкретного ключа в секціонованій базі даних, і може бути актуальним для цієї задачі через нижчі обчислювальні вимоги, ніж інші АХ.

MD5 обробляє повідомлення змінної довжини в вихідне значення фіксованої довжини в 128 біт. Вона приймає вхідні повідомлення змінної довжини і хешує їх у вихідні дані фіксованої довжини. MD5 працює з 512-бітними блоками повідомлень, розділеними на 32-бітні слова, і створює дайджест повідомлення з 128 біт [8].

Основний алгоритм MD5 працює з 128-бітовим значенням, розділеним на чотири 32-бітних слова, що позначаються A, B, C і D. Вони ініціалізуються певними фіксованими константами. Потім основний

алгоритм використовує кожен 512-бітний блок повідомлень по черзі для зміни значення. Обробка блоку повідомлень складається з чотирьох аналогічних етапів, званих раундами; кожен раунд складається з 16 аналогічних операцій, заснованих на нелінійній функції F, модульному складанні і повороті вліво. Нижче наведена одна операція в рамках раунду (рисунок 2.2). Є чотири можливі функції; в кожному раунді використовується інша функція.

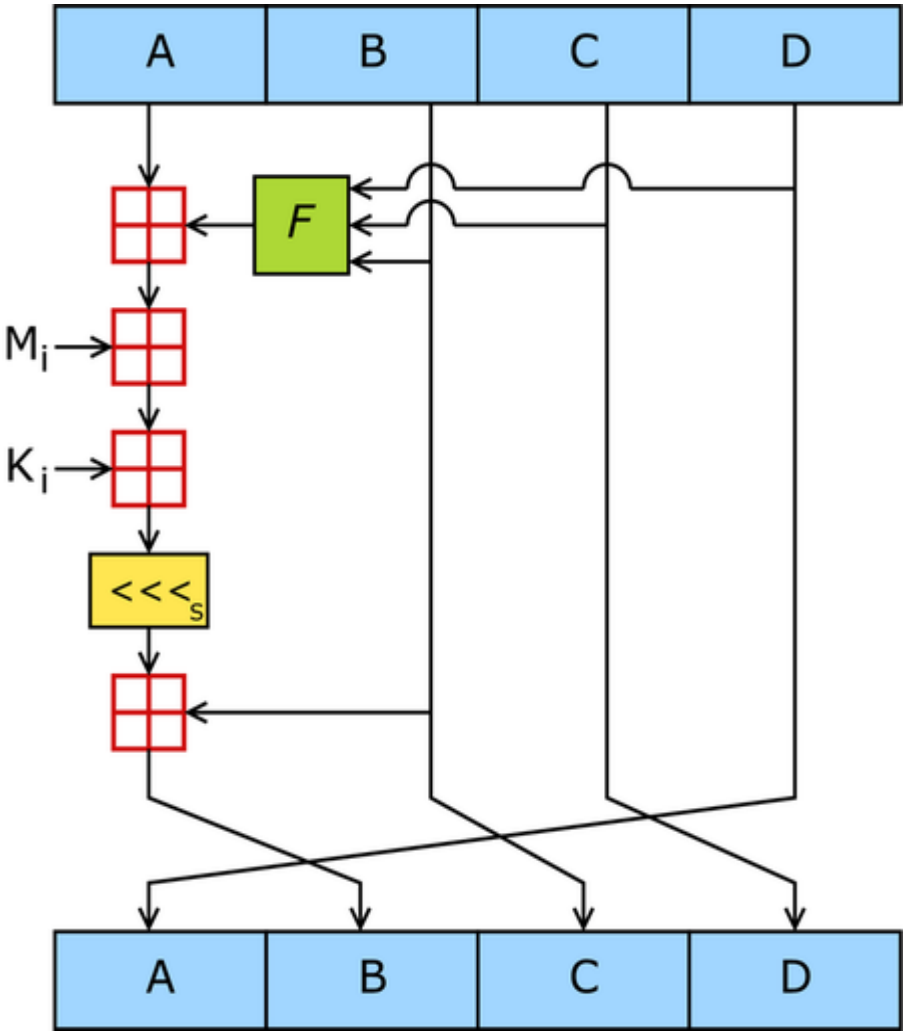

Рисунок 2.2 – Блок схема роботи алгоритму MD5

Безпека ХФ MD5 серйозно порушена. Існує колізійна атака, яка може виявити колізії протягом декількох секунд на КС з процесором x86 з частотою 2,6 ГГц [9]. Крім того, існує також колізійна атака з обраним

префіксом, яка може викликати колізію для двох входів із зазначеними префіксами протягом декількох секунд, використовуючи готові обчислювальне обладнання. Здатності знаходити колізії в значній мірі сприяло використання готових графічних процесорів.

Ці хеш-атаки і колізійні атаки були продемонстровані громадськості в різних ситуаціях. Аналіз відкритих джерел показав, що MD5 все ще досить широко використовується, в першу чергу дослідниками безпеки та антивірусними компаніями [10].

2.1.2 Алгоритми хешування SHA

Алгоритми безпечного хешування (SHA) є сімейством з шести КХФ SHA-0, SHA-1, SHA-224, SHA-256, SHA-384 і SHA-512 [11, 12]. Дане сімейство алгоритмів опубліковано Національним інститутом стандартів і технологій (NIST) в якості Федерального стандарту обробки інформації США (FIPS).

Дані стандарти включають:

- SHA-0, що застосовується до оригінальної версії 160-бітної ХФ, опублікованої в 1993 році під назвою SHA. Він був відкликаний незабаром після публікації через нерозкритий істотний недолік і замінений переглянутою версією SHA-1;
- SHA-1, 160-бітна ХФ, яка нагадує більш ранній алгоритм MD5. Вона було розроблена агентством національної безпеки, як частина алгоритму цифрового підпису. У SHA-1 були виявлені криптографічні недоліки, і стандарт не був схвалений для більшості криптографічних застосувань після 2010 року;
- SHA-2, сімейство двох подібних ХФ з різними розмірами блоків, відомих як SHA-256 та SHA-512. Вони відрізняються розміром слова, SHA-256 використовує 32-розрядні слова, тоді як SHA-512 використовує 64-розрядні слова. Існують також усічені версії кожного стандарту, відомі як

SHA-224, SHA-384, SHA-512/224 і SHA-512/256. Вони також були розроблені АНБ;

- SHA-3, ХФ, раніше називалася Keccak, обрана в 2012 році після відкритого конкурсу серед розробників, що не відносяться до АНБ. Він підтримує ті ж довжини хешу, що і SHA-2, і його внутрішня структура значно відрізняється від решти сімейства SHA.

Функція SHA-3 не є реалізованою у ПП .NET, а використання SHA-1 для захисту паролів є нерекомендованим, тому далі розглядається виключно сімейство ХФ SHA-2.

SHA-2 являє собою набір КХФ, воно складається з шести ХФ з довжиною хеш-значень, рівною 224, 256, 384 або 512 Бітам:

- SHA-224;
- SHA-256;
- SHA-384;
- SHA-512;
- SHA-512/224;
- SHA-512/256.

SHA-256 і SHA-512 – це ХФ, які обчислюються з використанням восьми 32-бітних і 64-бітних слів відповідно. Вони використовують різні величини зсуву і адитивні константи, але в іншому їх структури практично ідентичні, відрізняючись тільки кількістю раундів. SHA-224 і SHA-384 є усіченими версіями SHA-256 і SHA-512, обчисленими з різними початковими значеннями. SHA-512/224 і SHA-512/256 є скороченими версіями SHA-512, але початкові значення генеруються за допомогою методу, описаного в федеральних стандартах обробки інформації.

ХФ сімейства SHA-2 побудовані на основі структури Меркла-Дамгора.

Вхідне повідомлення після додавання ділиться на блоки, кожен блок на 16 слів. Кожен блок повідомлення пропускається алгоритмом через цикл з 64 або 80 ітераціями. Результати обробки кожного блоку складаються, сума є значенням ХФ. Проте, ініціалізація внутрішнього стану проводиться

результатом обробки попереднього блоку. Тому незалежно обробляти блоки і складати результати не можна. Схема раунду наведена нижче (рисунок 2.3).

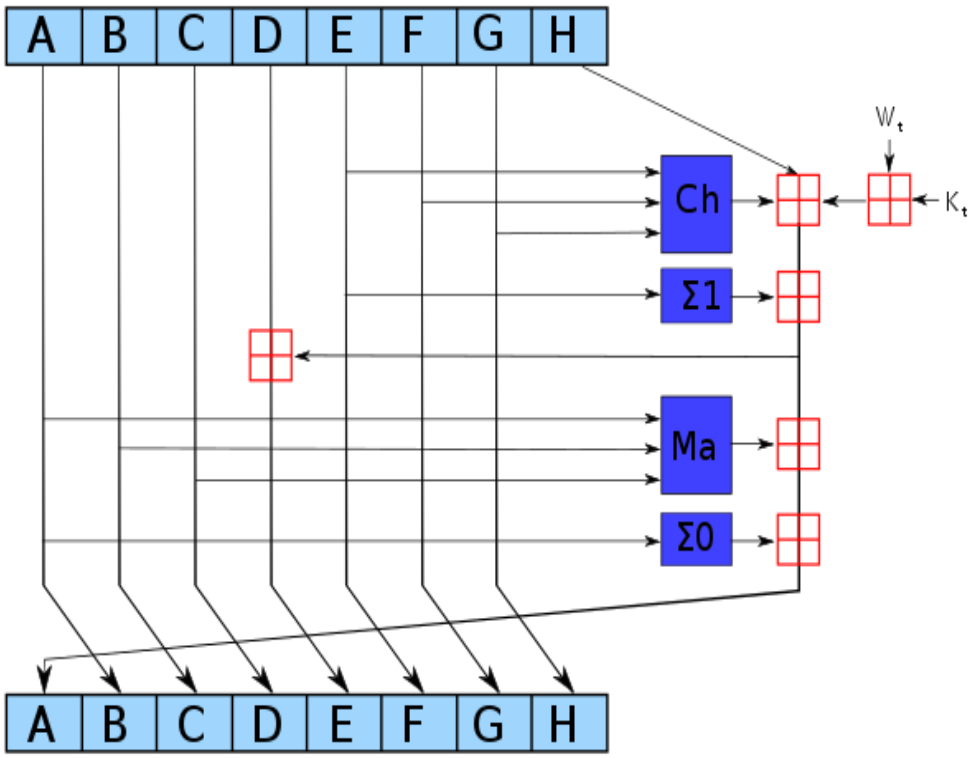

Рисунок 2.3 – Блок схема роботи алгоритму SHA-2

2.2 Вбудовані функції формування ключа

Функції формування ключа представлені у вигляді класів в просторі імен System.Security.Cryptography ПП .NET:

- PasswordDeriveBytes;
- Rfc2898DeriveBytes.

Rfc2898DeriveBytes є реалізацію AX PDKDF2 та використовується за замовченням у застосунках з використання ASP.NET. Імплементація алгоритму, залежно від оточення де розгорнуто застосунок, обирається за допомогою класу обгортки KeyDerivation.Pbkdf2 з простору імен Microsoft.AspNetCore.Cryptography.KeyDerivation.

2.3 Функція формування ключа PBKDF2

PBKDF2 – це проста функція отримання криптографічного ключа, яка стійка до атак за словником і атакам за допомогою райдужних таблиць. Вона заснований на багаторазовому ітеративному виводі HMAC з деяким доповненням. PBKDF2 є частиною серії криптографічних стандартів з відкритим ключем RSA (PKCS # 5 версії 2.0). PBKDF2 також є частиною специфікації робочої групи з розробки Інтернету RFC2898 [13].

Код автентифікації повідомлень на основі хешу (HMAC) – це метод криптографічної автентифікації, який використовує хеш-функцію та секретний ключ [24].

За допомогою нього можна виконувати аутентифікації і переконатися в правильності і достовірності даних за допомогою загальних секретів, на відміну від підходів, що використовують підписи і асиметричну криптографію.

HMAC – це код аутентифікації повідомлення на основі хешу, код, обчислений з використанням КХФ.

PBKDF2 використовує HMAC коди на основі SHA.

PBKDF2 приймає наступні параметри і видає згенерований ключ:

- пароль;
- сіль;
- кількість ітерацій;
- ХФ яку використовувати, зазвичай SHA-1, SHA-256 або SHA-512;
- довжина згенерованого ключа.

Обчислення за PBKDF2 виконується наступним чином, хешуються сіль і відкритий текст для отримання першого хеш-значення, потім в циклі той же алгоритм використовується для обчислення хешу від відкритого тексту і результату попередньої ітерації, після чого повертається результат застосування операції XOR по всіх обчислених хеш-значеннях, схема наведена нижче (рисунок 2.4).

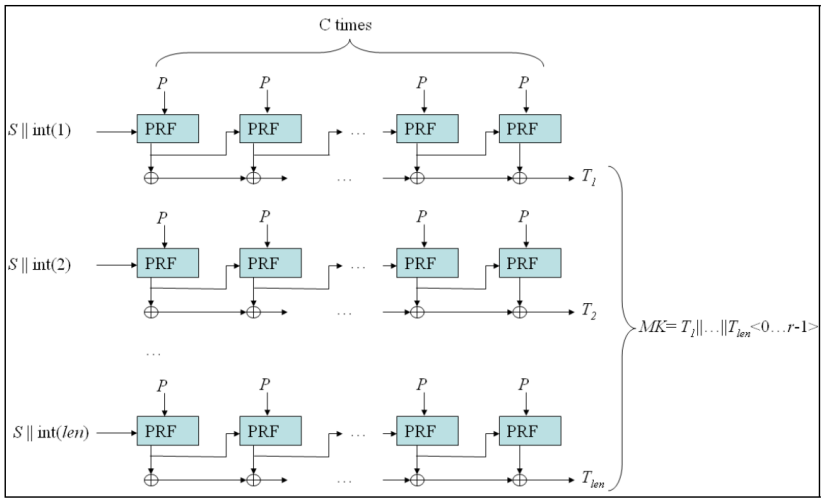

Рисунок 2.4 – Блок схема алгоритму PBKDF2

2.4 Аналіз вбудованих алгоритмів

В ПП .NET вбудовані АХ паролів, які забезпечували достатній захист, у час свого створення. Але с того часу вимогу до засобів захисту паролів зросли.

ХФ MD5 була зламана, але досі використовується великою кількістю КС. ХФ родини SHA можуть забезпечити захист від зловмисників з обмеженими потужностями, але за замовченням є вразливими до атак з використанням райдужних таблиць. Алгоритм PBKDF2, який є стандартним засобом хешування паролів у .NET, як і інші класичні ХФ є вразливим до атак з використанням GPU, ASIC та FPGA, які не біли розповсюджені у час їх створення. Для захисту від цих загроз пропонуються більш сучасні алгоритми, реалізації яких не вбудовані до ПП.NET.

3 ФУНКЦІЇ ФОРМУВАННЯ КЛЮЧА ДОСТУПНІ НА ПРОГРАМНІЙ ПЛАТФОРМІ .NET

3.1 Актуальні функції формування ключа

Не дивлячись на те що функція ключа PBKDF2 має переваги перед MD5, та SHA вона має і недолік, вона не стійка до атак перебором з використанням GPU та ASIC, оскільки PBKDF2 використовує відносно невеликий обсяг оперативної пам’яті і може бути обчислена на GPU або ASIC.

Сучасні функції формування ключа, такі як Bcrypt, Scrypt і Argon2, розроблені таким чином, щоб бути стійкими до атак за словником, атак з використанням GPU та ASIC. Ці функції формують ключ з пароля і вимагають великого обсягу пам’яті, що не дозволяє виконувати швидкі паралельні обчислення на GPU або ASIC.

Такі алгоритми, як Bcrypt, Scrypt і Argon2, вважаються більш безпечними функціями формування ключа, через те що для свого обчислення вони потребують:

- сіль;
- велику кількість ітерації ітерацій;
- багато ресурсів процесора;
- велику кількість оперативної пам’яті.

Це дуже ускладнює розробку обладнання для значного прискорення злому паролів. Сучасні комп’ютерні обчислення більшим чином обмежені швидкодією пам’яті, тому доступ до пам’яті є вузьким місцем обчислень. Більш швидкий доступ до оперативної пам’яті прискорить обчислення.

Коли для отримання ключа із заданого пароля використовується багато ресурсів процесора і велика кількість оперативної пам’яті, злом паролів відбувається повільно і неефективно, навіть при використанні дуже хорошого

апаратного і програмного забезпечення для злому паролів. Мета сучасних функцій формування ключа полягає в тому, щоб зробити практично неможливим виконання атаки методом перебору.

3.1.1 Функція формування ключа Bcrypt

Bcrypt – це адаптивна ХФ, заснована на криптографічному алгоритмі симетричного блочного шифру Blowfish. Вона використовує ключовий фактор, який регулює вартість хешування, що є найбільш помітною особливістю функції Bcrypt. Це надає можливість збільшити вартість – час і обчислювальні витрати хешування в майбутньому, коли КС стануть більш потужними.

Bcrypt використовує 128-бітну сіль і шифрує 192-бітове магічне значення. Алгоритм Bcrypt шифрує вхідний пароль, згідно встановленого ключового фактору з використанням Blowfish. Він використовує переваги дорогого налаштування ключа в eksblowfish [14].

Bcrypt був створений у той же час, що і PBKDF2, використання у своїй основі шифру Blowfish робить Bcrypt повільнішим і більш захищеним ніж PBKDF2 від атак перебором, також Bcrypt потребує більше пам’яті ніж мають сучасні обчислювальні блоки GPU. Але Bcrypt залишається вразливим при використанні ASIC та FPGA. Одним із його недоліків є обмеження на розмір вхідних даних, 72 байти. Використання довших паролів потребуватиме попереднього хешування іншим алгоритмом.

У бібліотеках ПП .NET присутня велика кількість реалізацій даного алгоритму, вони мають значну кількість завантажень і більшість була оновлена у 2022 році (рисунок 3.1).

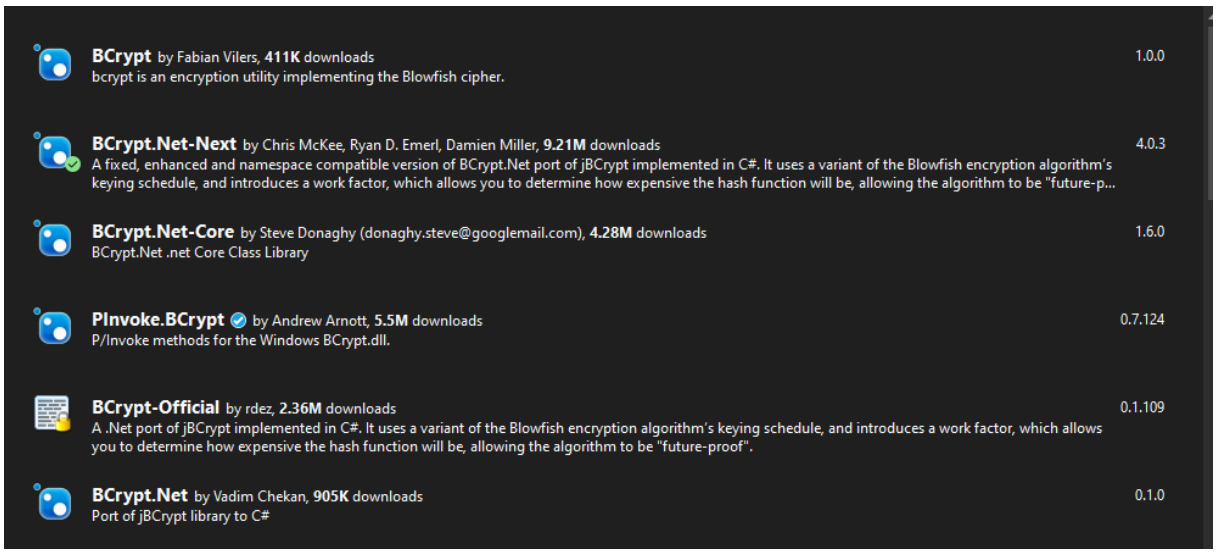

Рисунок 3.1 – Реалізації алгоритму Bcrypt у бібліотеках програмної платформи .NET

3.1.2 Функція формування ключа Scrypt

Scrypt – це надійна криптографічна функція формування ключа Алгоритм Scrypt описаний в інтернет-стандарті RFC 7914. Вона вимагає багато пам’яті і призначена для запобігання атак з використанням GPU, ASIC і FPGA, високоефективного обладнання для злому паролів.

Алгоритм Scrypt приймає кілька вхідних параметрів і видає похідний ключ в якості вихідних даних.

Параметри Scrypt наступні:

- кількість ітерацій, впливає на використання пам’яті і процесора;
- розмір блоку, впливає на використання пам’яті і процесора;
- коефіцієнт паралелізму, потоки виконуються паралельно, впливає на обсяг пам’яті і завантаження процесора;
- вхідний пароль, рекомендується мінімальна довжина 8-10 символів;
- сіль, згенеровані випадкові байти, мінімум 64 біта, рекомендується 128 біт;
- довжина сформованого ключа, скільки байт згенерувати в якості

вихідних даних.

Доступ до пам’яті в Scrypt здійснюється в строго залежному порядку на кожному кроці, тому швидкість доступу до пам’яті – це вузьке місце алгоритму.

При правильному налаштуванні Scrypt вважається високо захищеною функцією формування ключа, тому її можна використовувати як алгоритм хешування пароля загального призначення для, наприклад, шифрування паролів гаманців, файлів або застосунків. Scrypt популярний в схемах перевірки працездатності криптовалют в першу чергу Litecoin також Tenebrix та Dogecoin, і надихнув дизайн одного з переможців конкурсу з хешування паролів Argon2d [15]. З іншого боку великі вимоги до пам’яті роблять цей алгоритм непопулярним для захисту паролів у більшості застосунків з обмеженням на ресурси.

У даного алгоритму незначна кількість реалізацій на ПП .NET, вони мають невелику кількість завантажень і більшість була оновлена більш ніж 5 років тому, що створює ризики при використанні цього алгоритму (рисунок 3.2).

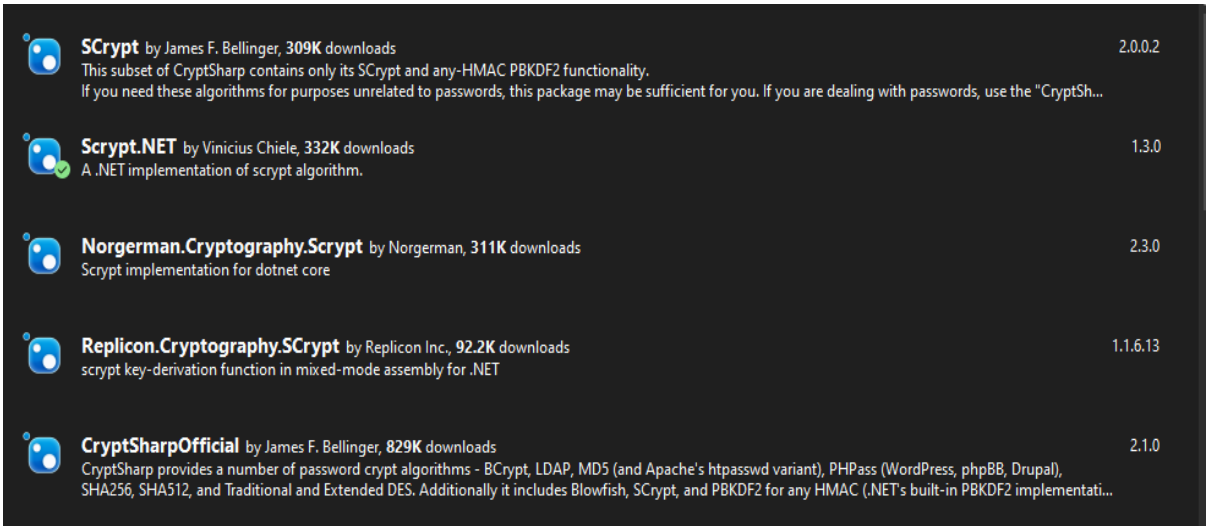

Рисунок 3.2 – Реалізації алгоритму Scrypt у бібліотеках програмної платформи .NET

3.1.3 Функція формування ключа Argon2

Argon2 – це сучасна функція формування захищеного ключа, стійка до ASIC і GPU. Вона має кращу стійкість до злому паролів, при правильному налаштуванні, ніж PBKDF2, Bcrypt і Scrypt, для аналогічних параметрів конфігурації використання процесора і оперативної пам’яті.

Argon2 був розроблений з наступною основною метою – максимізувати вартість вичерпного пошуку на архітектурах, відмінних від x86, так що перехід навіть на виділені ASIC не дасть істотної переваги в порівнянні з виконанням вичерпного пошуку на комп’ютері [16].

Функція Argon2 має кілька варіантів:

- Argon2d, забезпечує сильний захист при використанні GPU, але має потенційні атаки по побічних каналах (можливі в дуже особливих ситуаціях);
- Argon2i, забезпечує менший захист при використанні GPU, але не має атак по побічних каналах;
- Argon2id, рекомендована, бо поєднує в собі Argon2d і Argon2i.

У даного алгоритму незначна кількість реалізацій на ПП .NET, що можна пояснити новизною алгоритму, але є реалізації зі значною кількістю завантажень і оновленнями у 2022 році (рисунок 3.3).

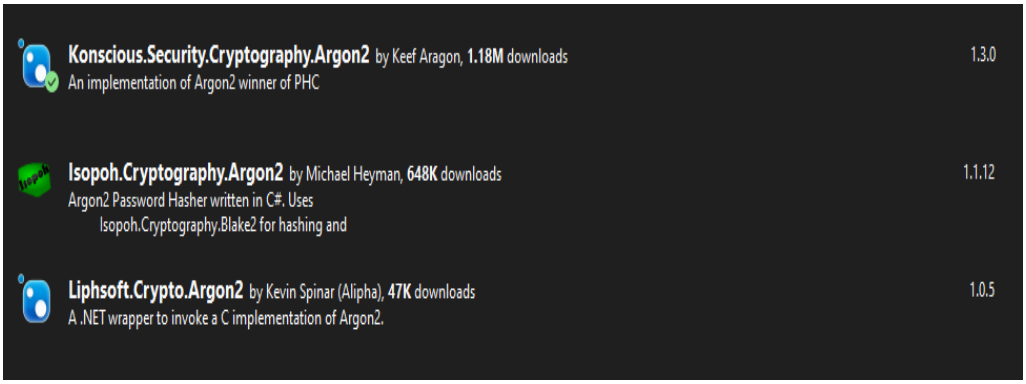

Рисунок 3.3 – Реалізації алгоритму Argon2 у бібліотеках програмної платформи .NET

3.2 Безпека реалізацій сторонніх функцій хешування

Безпеку і відсутність помилок у вбудованих хеш функціях, гарантується корпорацією Microsoft, яка підтримує ПП .NET. АХ, які розглядаються у цьому розділі не вбудовані до програмної платформи .NET і їх реалізації необхідно отримати з бібліотек ПП.

За безпеку бібліотек відповідає виключно її розробник, і якщо бібліотека не є діже популярною то оновлення такої бібліотек не будуть регулярними, що виключає можливість швидкого виправлення вразливостей. Статистично серед вразливостей бібліотек ПП .NET вразливості високого ступеня серйозності становлять 70,7% від загального числа і їх виправлення розробниками є вкрай важливим [2].

Серед алгоритмів, які розглядаються сумніви у своїй безпеці викликають реалізації алгоритму Scrypt.

3.3 Аналіз альтернативних алгоритмів

Альтернативні АХ обіцяють більш надійний захист від атак перебором ніж PBKDF2, яка є стандартним засобом захисту ПП .NET. Сучасні алгоритми також розраховані на захист від атак з використання GPU, ASIC та FPGA, але цей захист потребує більшого часу на обчислення та ресурсів КС.

4 АНАЛІЗ АЛГОРИТМІВ ХЕШУВАННЯ НА ПРОГРАМНІЙ ПЛАТФОРМІ .NET

4.1 Порівняння швидкості хешування

Для аналізу швидкості було проведено замірювання, середнього, мінімального та максимального часу, витраченого на обчислювання хеш-значення за допомогою досліджуваних алгоритмів. Хеш-значення було обчислено для декількох наборів паролів:

- набір 1, 500 найгірших паролів;
- набір 2, 10000 найбільш розповсюджених паролів;
- набір 3, 20 згенерованих паролів з великою ентропією включать маленькі та великі літери, цифри та інші символи, довжина 8 символів;
- набір 4, 20 згенерованих паролів з великою ентропією включать маленькі та великі літери, цифри та інші символи, довжина 12 символів;
- набір 5, 20 згенерованих паролів з великою ентропією включать маленькі та великі літери, цифри та інші символи, довжина 15 символів.

Час обчислення функцій формування ключа залежить від параметрів функції. Актуальні параметри були отримані з ресурсів організації OWASP, відкритий проект з безпеки веб застосунків.

Для функцій формування ключа були обрані наступні параметри:

- PBKDF2 (за замовленням), відповідають конфігурації у ASP.NET Core Identity. Число ітерацій 10000, розмір солі 128 біт, розмір хешу 256 біт;
- PBKDF2 (рекомендовані), змінені згідно до рекомендацій. Число ітерацій 310000, розмір солі 128 біт, розмір хешу 256 біт;
- Bcrypt, параметри залишені без змін бо встановлений робочий фактор за замовченням 11, перевищує рекомендований 10;
- Scrypt (ступінь паралелізації 1), ступінь паралелізації 1, розмір блоку 8, розмір пам’яті 64 Міб;

- Scrypt (ступінь паралелізації 4), ступінь паралелізації 4, розмір блоку 8, розмір пам’яті 16 Міб;

- Argon2 (1 ітерація), ступінь паралелізації 1, мінімальне число ітерацій 1, мінімальний розмір пам’яті 37 Міб;

- Argon2 (2 ітерації), ступінь паралелізації 1, мінімальне число ітерацій 2, мінімальний розмір пам’яті 15 Міб.

Хешування та замірювання виконувались на КС з центральним процесором AMD Ryzen 7 6800H, та графічним процесором NVIDIA GeForce RTX 3050 Ti Laptop GPU. Операційна система КС Windows 11, версія ПП .NET 6. Для замірювання метрик використовуваличсь засоби діагности ПП .NET та Visual Studio 2022.

4.1.1 Результати хешування

В результаті хешування наборів були отримані файли з хеш-значеннями та метриками обчислень. Для демонстрації наведено результати для AX Argon2 (рисунок 4.1).

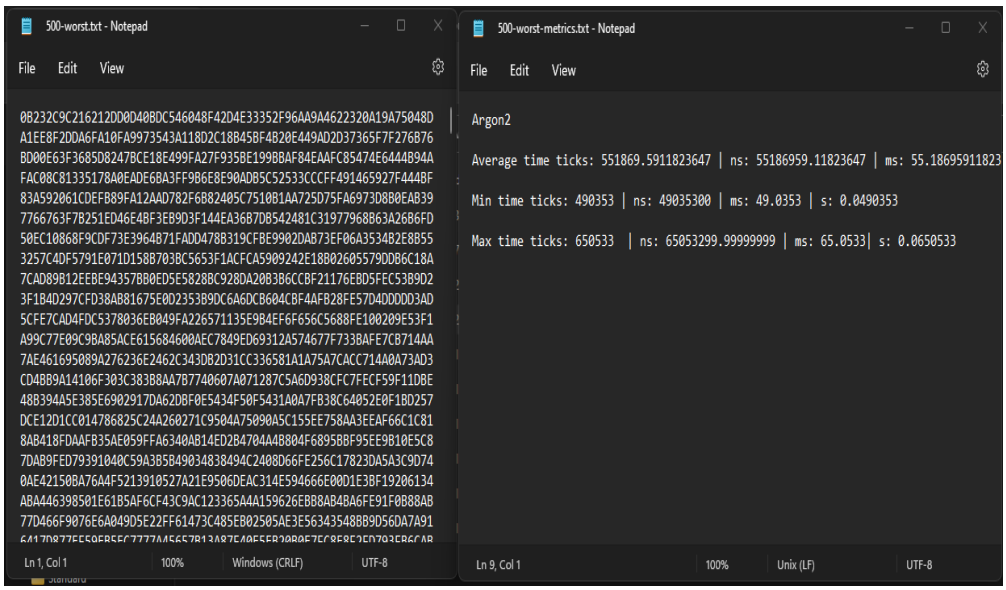

Рисунок 4.1 – Обчислені хеш-значення та метрики обчислень

Отримані хеш-значення були використані для аналізу захищеності, а метрики дозволили порівняти швидкість.

4.1.2 Час хешування

Для всіх наборів паролів були виконані заміри середнього, мінімального та максимального часу обчислення хеш-значення. Аналіз проводився по середньому значенні результатів для усіх наборів, графік для відносного порівняння швидкості АХ наведено нижче (рисунок 4.2).

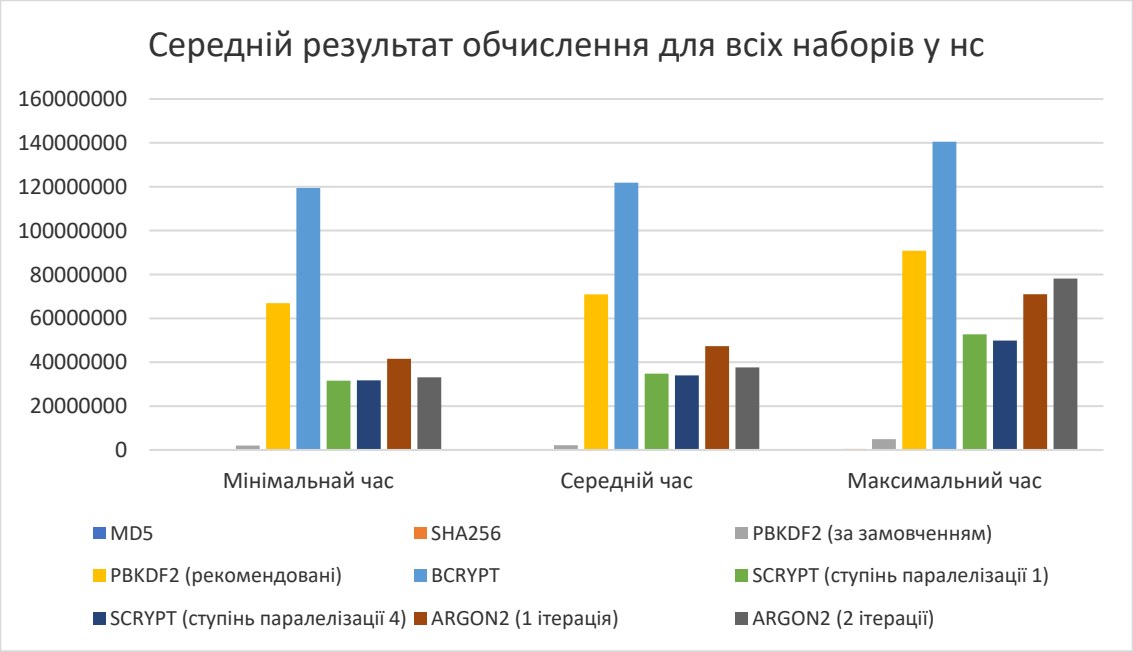

Рисунок 4.2 – Середні швидкості хешування паролів для всіх наборів

Детальне порівння результатів обчислення хеш-значення для кожного АХ, та його встановлених параметрів надає середній час, мінімальний та максиматьний час у наносекуднах нижче (таблиця 4.1).

Із результатів вимірювань можна побачити, що класичні ХФ та PBKDF2 з параметрами за замовченням є дуже швидкими. Інші алгоритми витрачають більше часу, але цього ще достатньо для застосування в веб

застосунках. Bcrypt має найменшу швидкість, інші функції формування ключа витрачають приблизно однаковий час для обчислення хеш-значення

Таблиця 4.1 – Метрики обчислення хеш-значень у наносекундах

	Мінімальний час	Середній час	Максимальний час
MD5	320	14042.2	395240
SHA256	180	1123.2	532380
PBKDF2 (за замовченням)	2001580	2178747.4	4974460
PBKDF2 (рекомендовані)	66961080	70902501	90793120
BCRYPT	119422860	121843986.8	140523459.6
SCRYPT (ступінь паралелізації 1)	31593020	34807792.4	52724700
SCRYPT (ступінь паралелізації 4)	31740859.8	34016152.6	49895700
ARGON2 (1 ітерація)	41610720	47300997.6	70995859.8
ARGON2 (2 ітерації)	33106679.8	37634215.2	78174880

4.2 Порівняння захищеності

Для аналізу захищеності алгоритмів, були виконані атаки на набори паролів. Атаки використовували метод повного перебору, для наборів з широко поширеними паролями також була виконана атака зі словником, був використаний словник Rockyou, цей словник містить 32 мільйони паролів, які були отримані при атаці на однойменну компанію.

Для виконання атаки була використана КС з центральним процесором

AMD Ryzen 7 6800H, та графічним процесором NVIDIA GeForce RTX 3050 Ti Laptop GPU. Інструмент для атак Hashcat.

Для алгоритмів, які мають достатній захист від наявної КС або на які не вдалося виконати атаки за допомогою інструменту Hashcat. Було виконано аналіз за допомогою матеріалів досліджень незалежних організацій

4.2.1 Захищеність MD5

Як було сказано вище алгоритм MD5 на даний час не вважається придатним для хешування паролів перш за все через наявність колізій.

Виконаємо атаки за словником. Для найгірших 500 паролів вдалось отримати за 2 секунди 497 паролів або 99.60%. Для найпоширеніших 10000 паролів вдалось отримати за 31 секунду 9471 паролів або 94.71%.

Найгірші 500 паролів атака методом повного перебору (рисунок 4.3):

- за 1 секунду було отримано 358 паролів або 71.74%;
- за 21 секунду було отримано 454 паролів або 90.98%;
- за 3 хвилини 16 секунд було отримано 498 паролів або 99.80%;
- отримання усіх паролів тривало 8 хвилин 3 секунди.

```
Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 0 (MD5)
Hash.Target......: D:\Diploma\HashComparison\Hashes\MD5\500-worst.txt
Time.Started.....: Tue Nov 22 13:49:16 2022 (8 mins, 3 secs)
Time.Estimated...: Tue Nov 22 13:57:19 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.......: ?1?2?2?2?2?2?3 [8]
Guess.Charset....: -1 ?l?d?u, -2 ?l?d, -3 ?l?d*!$@_, -4 Undefined
Guess.Queue......: 8/15 (53.33%)
Speed.#1.........:  7092.6 MH/s (11.60ms) @ Accel:32 Loops:512 Thr:256 Vec:1
Speed.#2.........:   606.3 MH/s (9.90ms) @ Accel:16 Loops:1024 Thr:64 Vec:1
Speed.#*.........:  7698.9 MH/s
Recovered........: 499/499 (100.00%) Digests (total), 499/499 (100.00%) Digests (new)
Progress.........: 3645599580160/5533380698112 (65.88%)
Rejected.........: 0/3645599580160 (0.00%)
Restore.Point....: 45355008/68864256 (65.86%)
Restore.Sub.#1...: Salt:0 Amplifier:2048-2560 Iteration:0-512
Restore.Sub.#2...: Salt:0 Amplifier:52224-53248 Iteration:0-1024
Candidate.Engine.: Device Generator
Candidates.#1....: 1za5e23h -> fer68vo9
Candidates.#2....: eb3e688f -> Fxhbjbek
Hardware.Mon.#1..: Temp: 79c Util: 99% Core:1880MHz Mem:6108MHz Bus:8
Hardware.Mon.#2..: Temp:  0c Fan:  0% Util:  0% Core: 533MHz Mem:2400MHz Bus:16
```

Рисунок 4.3 – Найгірші 500 паролів метод повного перебору

Найпоширеніші 10000 паролів атака методом повного перебору:

- за 4 секунди було отримано 1078 паролів або 10.78%;
- за 10 секунд було отримано 5495 паролів або 54.95%;
- за 27 секунд було отримано 7451 паролів або 74.51%;
- за 2 хвилини 36 секунд було отримано 8977 паролів або 89.77%;
- за 4 хвилини 19 секунд було отримано 9198 паролів або 91.98%;
- за 9 хвилини 34 секунд було отримано 9326 паролів або 93.26% максимальна швидкість 7892 мега хеші в секунду, далі почався перебір паролів довжиною 9 символів, що для обладнання даного рівня є непродуктивним;
- за 2 години було отримано 9931 пароль або 99.31%, прогнозований час перебору інших комбінацій з 9 символів 6 годин 14 хвилин, атаку було припинено.

З наведених вище даних видно що алгоритм MD5 є вразливий до атак перебором з КС загального призначення, збільшення довжини паролю збільшує час для його отримання, але для більш потужних систем це не є проблемою.

Для підтвердження результатів для паролів з великою довжиною використані результати дослідження HiveSystems [19]. Результати дослідження алгоритму MD5 від HiveSystems наведені нижче (рисунок 4.4).

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	Instantly	Instantly	Instantly
5	Instantly	Instantly	Instantly	Instantly	Instantly
6	Instantly	Instantly	Instantly	1 sec	5 secs
7	Instantly	Instantly	25 secs	1 min	6 mins
8	Instantly	5 secs	22 mins	1 hour	8 hours
9	Instantly	2 mins	19 hours	3 days	3 weeks
10	Instantly	58 mins	1 month	7 months	5 years
11	2 secs	1 day	5 years	41 years	400 years
12	25 secs	3 weeks	300 years	2k years	34k years
13	4 mins	1 year	16k years	100k years	2m years
14	41 mins	51 years	800k years	9m years	200m years
15	6 hours	1k years	43m years	600m years	15bn years
16	2 days	34k years	2bn years	37bn years	1tn years
17	4 weeks	800k years	100bn years	2tn years	93tn years
18	9 months	23m years	61tm years	100tn years	7qd years

Рисунок 4.4 – Час отримання паролю захищеного алгоритмом MD5 з використанням RTX 2080

Кількість хеш-значень, які обчислює RTX 2080 за 1 секунду 37085000000, для поточного найбільш потужного графічного процесора RTX 3090, це значення 69379700000. Також в MD5 відсутній захист від атак з використання райдужних таблиць. Саме тому цей алгоритм не рекомендований для захисту паролів, проте деякі застосунки все ще використовують його (рисунок 4.5).

Рисунок 4.5 – Веб застосунок з алгоритмом MD5

4.2.2 Захищеність SHA256

Алгоритм SHA256 все ще вважається придатним для захисту паролів, проте він не має захисту від атак за словником та райдужних таблиць.

Виконаємо атаки за словником. Для найгірших 500 паролів вдалось отримати за 3 секунди 497 паролів або 99.60%. Для найпоширеніших 10000 паролів вдалось отримати за 30 секунд 9471 паролів або 94.71% (рисунок 4.6).

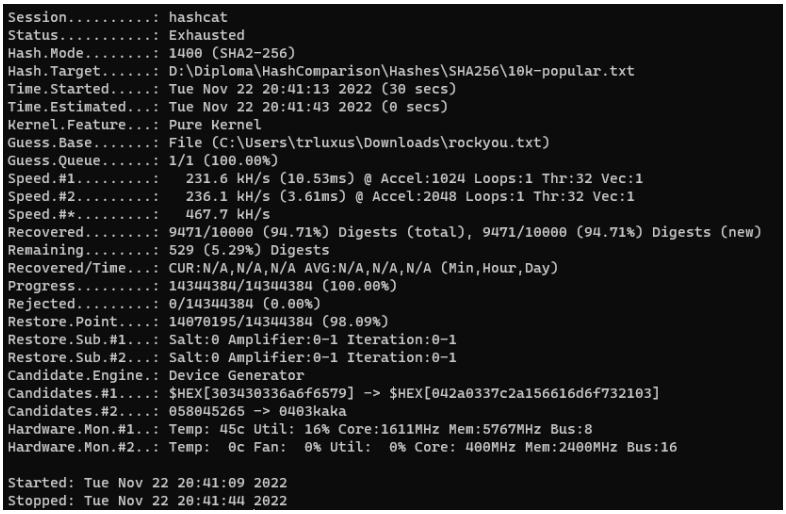

Рисунок 4.6 – Найпоширеніші 10000 паролів атака за словником

Найгірші 500 паролів атака методом повного перебору (рисунок 4.7):

- за 2 секунду було отримано 358 паролів або 71.74%, це паролі довжиною до 6 символів;
- за 1 хвилину було отримано 454 паролів або 90.98%, це паролі довжиною до 7 символів;
- отримання усіх паролів тривало 26 хвилин 28 секунд.

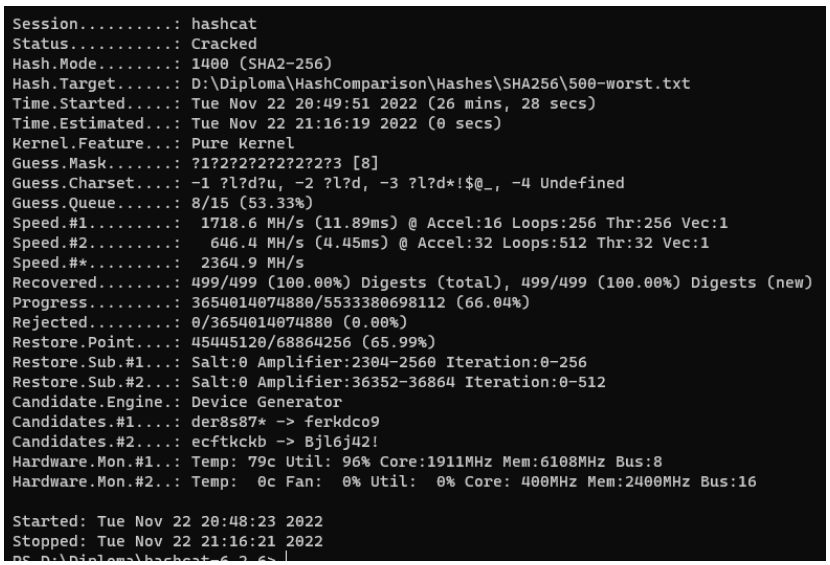

Рисунок 4.7 – Найгірші 500 паролів атака методом повного перебору

Найпоширеніші 10000 паролів атака методом повного перебору (рисунок 4.8):

- за 11 секунд було отримано 5852 паролів або 58.52%, це паролі довжиною до 6 символів;
- за 1 хвилину 2 секунди було отримано 7904 паролів або 79.04%, це паролі довжиною до 7 символів;
- за 40 хвилини 34 секунд було отримано 9834 паролів або 98.34%, це паролі довжино 8 символів;
- за 9 хвилини 34 секунд було отримано 9326 паролів або 93.26%, далі почався перебір паролів довжиною 9 символів, що для обладнання даного рівня є непродуктивним;
- прогнозований час перебору інших комбінацій з 9 символів 29 годин, атаку було припинено.

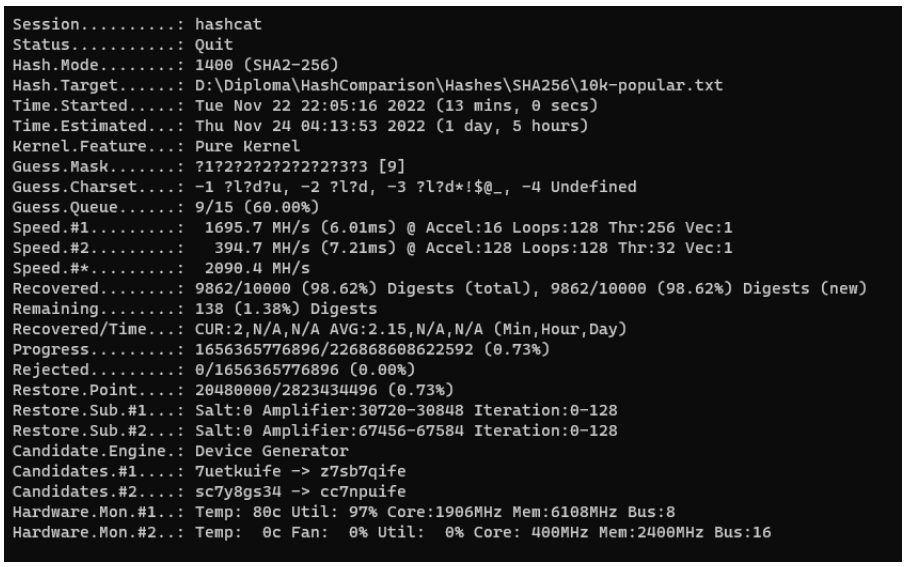

Рисунок 4.8 – Найпоширеніші 10000 паролів атака методом повного перебору

SHA256 залишається надійною КХФ, проте відсутність захисту від атак за словником та райдужних таблиць вимагають самостійного додавання солі. Використання SHA256 є можливим, але не рекомендуються бо існують алгоритми основані на цій ХФ, які потребують солі та числа ітерацій, для сповільнення алгоритму.

4.2.3 Захищеність PBKDF2

Алгоритм PBKDF2 є функцію формування ключа, він потребує солі та числа ітерацій, в своїй основі вин використовує ХФ родини SHA.

Серед застосунків, які використовують PBKDF2 на основі SHA256, рішення для зберігання паролів LastPass, 1Password і Bitwarden. Інструмент Hashcat підтримує хеші PBKDF2 у специфічному форматі, тому для аналізу були використані дані з відкритих джерел. Нижче наведені результати атаки методом повного перебору з використанням графічного процесору RTX 3090 від HiveSystems [19] (рисунок 4.9).

Number of Characters	Numbers Only	Lowercase Letters	Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters	Numbers, Upper and Lowercase Letters, Symbols
4	Instantly	Instantly	2 secs	4 secs	4 secs
5	Instantly	3 secs	2 mins	4 mins	12 mins
6	Instantly	1 min	1 hour	4 hours	15 hours
7	3 secs	35 mins	3 days	2 weeks	2 months
8	26 secs	15 hours	5 months	2 years	10 years
9	4 mins	2 weeks	23 years	100 years	800 years
10	44 mins	1 year	1k years	7k years	61k years
11	7 hours	31 years	63k years	435k years	5m years
12	3 days	800 years	3m years	27m years	363m years
13	1 month	21k years	170m years	2bn years	28bn years
14	10 months	539k years	9bn years	104bn years	2tn years
15	8 years	14m years	460bn years	6tn years	166tn years
16	84 years	365m years	24tn years	399 tn years	13 qdn years
17	800 years	9bn years	1qdn years	25 qdn years	983 qdn years
18	8k years	246bn years	65 qdn years	2 qntn yrs	76qntn years

Рисунок 4.9 – Час отримання паролю захищеного алгоритмом PBKDF2 SHA256 з використанням RTX 3090

Можна побачити, що паролі зі значною довжиною та високою ентропією, є досить захищеними, недоліком PBKDF2 є лише можливість посилення захисту шляхом збільшення ітерацій.

4.2.4 Захищеність Bcrypt

Bcrypt є одним із найповільніших алгоритмів, що повинно ускладнити атаки методом повного перебору (рисунок 4.10).

```
Session..........: hashcat
Status...........: Running
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target......: D:\Diploma\HashComparison\Hashes\BCrypt\500-worst.txt
Time.Started.....: Wed Nov 23 20:17:50 2022 (6 mins, 21 secs)
Time.Estimated...: Mon Mar 06 20:23:00 2023 (102 days, 23 hours)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (C:\Users\trluxus\Downloads\rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:      424 H/s (8.75ms) @ Accel:1 Loops:16 Thr:24 Vec:1
Speed.#3.........:      145 H/s (10.25ms) @ Accel:2 Loops:16 Thr:16 Vec:1
Speed.#*.........:      569 H/s
Recovered........: 146/499 (29.26%) Digests (total), 146/499 (29.26%) Digests (new), 146/499 (29.26%) Salts
Progress.........: 225408/7157847616 (0.00%)
Rejected.........: 0/225408 (0.00%)
Restore.Point....: 0/14344384 (0.00%)
Restore.Sub.#1...: Salt:336 Amplifier:0-1 Iteration:560-576
Restore.Sub.#3...: Salt:334 Amplifier:0-1 Iteration:1328-1344
Candidate.Engine.: Device Generator
Candidates.#1....: alyssa -> kelly
Candidates.#3....: 123456 -> november
Hardware.Mon.#1..: Temp: 67c Util: 98% Core:1927MHz Mem:5989MHz Bus:8
Hardware.Mon.#3..: Temp:  0c Fan:  0% Util: 98% Core:2200MHz Mem:2400MHz Bus:16
```

Рисунок 4.10 – Найгірші 500 паролів атака за словником

Навіть під час виконання атаки за словником за 6 хвилин, було отримано лише 146 паролів або 29.26%. Що показує неможливість атаками методом повного перебору на даній КС.

Для дослідження захисту від більш потужної КС та FPGA було вивчено дослідження ScatteredSecrets [20]. Нижче для Bcrypt наведені обчислення хешів у секунду для центрального процесору AMD EPYC 7401P та графічного процесору Nvida RTX-2080Ti (рисунок 4.11).

Work factor	CPU: hashes per second	GPU: hashes per second
05	25,200	28,000
07	6,300	7,000
08	3,150	3,500
10	788	875
11	394	438
12	197	219
14	98	109

Рисунок 4.11 – Обчислення хеш-значень Bcrypt

Низька швидкість обчислень на графічному процесорі пов’язана з тим, що Bcrypt потребує понад 4 кілобайти пам’яті для максимальної швидкості, а RTX-2080Ti має лише 1 кілобайт. Отже поточного захисту завдяки пам’яті від графічних процесорів достатньо, але цей параметр є незмінним і з часом можуть з’явитися пристрої для яких це не буде перешкодою.

Інструмент аудиту безпеки паролів і відновлення паролів John The Ripper підтримує плати FPGA. Вони достатньо ефективні, щоб запускати 124 оптимізовані ядра Bcrypt на FPGA. Це призводить до високої швидкості хешування Bcrypt, вище ніж швидкість хешування останнього покоління графічних процесорів високого класу (рисунок 4.12).

Work factor	FPGAs: hashes per second	GPU: hashes per second
05	120k	28k

Рисунок 4.12 – Обчислення хеш-значень Bcrypt з використанням FPGA

Отже Bcrypt є захищеним алгоритмом від сучасних загроз, але не можливість конфігурації захисту завдяки пам’яті створює ризики при використанні ASIC та FPGA.

4.2.5 Захищеність Scrypt

Scrypt є подовженням ідей Bcrypt і додає можливість конфігурування обсягу необхідної пам’яті. Але на ПП .NET відсутні популярні реалізації, які можна рекомендувати до використання у реальних застосунках. Згідно досліджень вартість обчислення пароля з використанням Scrypt у 2^5 разів дорожча за Bcrypt та 2^8 разів дорожча за PBKDF2 [17].

4.2.6 Захищеність Argon2

Argon2 алгоритм, який переміг на конкурс із хешування паролів 2015 року, метою конкурсу бу вибір однієї чи кількох функцій хешування паролів, які можна визнати рекомендованим стандартом. Згідно дослідження компанії RedHat для зламу восьми символьної парольної фрази, яка використовується для розблокування зашифрованого тому приблизно за дві секунди на Raspberry PI, може знадобитися до 1085 графічних процесорів NVIDIA Tesla P100, що коштує близько 120 мільйонів доларів [18].

4.3 Аналіз наданого захисту

Нижче наведено від яких саме загроз кожен з алгоритмів надає захист (таблиця 4.2).

Можна побачити, що Argon2 та Scrypt надають захист від найбільшої кількості загроз, проте Scrypt не має реалізації, яку можна було б рекомендувати, як безпечну. Bcrypt не має можливості конфігурування

необхідної пам’яті, але використаного об’єму достатньо для захисту від атак з використанням GPU. PBKDF2 надає захист від атак методом повного перебору перебором та використання райдужних таблиць, що є мінімально необхідним захистом. MD5 та SHA256 не забезпечують захист від сучасних загроз.

Таблиця 4.2 – Аналіз захисту

	Атака з райдужною таблицею (наявність солі)	Атака перебором (сповільнення функції)	Захист від GPU	Захист від FPGA/ASIC	Безпека реалізації
MD5	Ні	Ні	Ні	Ні	Так
SHA256	Ні	Ні	Ні	Ні	Так
PBKDF2	Так	Так	Ні	Ні	Так
Bcrypt	Так	Так	Так	Ні	Так
Scrypt	Так	Так	Так	Так	Ні
Argon2	Так	Так	Так	Так	Так

ВИСНОВКИ

Захист паролів на ПП .NET, яка розповсюджена у корпоративних та веб застосунках, є важливим питанням. Алгоритми хешування вбудовані у ПП відрізняються високою швидкодією і достатнім захистом від загроз, які існували на момент їх створення. Але збільшення потужностей КС і розповсюдження GPU, ASIC та FPGA, робить КС, які використовують ці алгоритми вразливими. Більш сучасні альтернативні алгоритми були створені для захисту від цих загроз.

В результаті виконаного аналізу найкращим алгоритмом для використання на ПП .NET є Argon2. Даний алгоритм забезпечує найбільш надійний захист при оптимальній швидкодії, можливість конфігурації пам’яті забезпечує захист від ASIC та GPU. Для підтримки застарілих застосунків можна використовувати Bcrypt, основою загрозою якого є ASIC. PBKDF2 при досить великій кількості ітерацій здатний забезпечити досить надійний захист, його слід використовувати для застосунків, які.

Через відсутність популярних реалізацій алгоритм Scrypt не є рекомендованим.

Подальша робота в даному напрямку може включати створення КС та ПЗ, для повного локального аналізу всіх алгоритмів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Методи та алгоритми хешування паролів на платформі .NET. [Текст] : матеріали десятої міжнародної науково-технічної конференції., 24 – 25 листопада 2022 року Черкаси – Баку – Бельсько-Бяла – Харків. – 87 с.

2. Методи та засоби забезпечення безпеки у застосунках на основі платформи .NET. [Текст] : матеріали дев'ятої міжнародної науково-технічної конференції., 18 – 19 листопада 2021 року Черкаси – Баку – Бельсько-Бяла – Харків. – 62 с.

3. Towards reliable storage of 56-bit secrets in human memory. [Text] : In Proceedings of the 23rd USENIX Conference on Security Symposium (SEC'14), 20 – 22 August 2014 San Diego, CA, USA; – pp. 607–623.

4. Bart Preneel. Cryptographic hash functions. [Text] / Bart Preneel // European Transactions on Telecommunications July/August 1994. – Volume 5, Issue 4. p. 431–448. – DOI: https://doi.org/10.1002/ett.4460050406

5. Coles M. Expert SQL Server 2008 Encryption [Text] / M. Coles, R. Landrum. – Berkeley: Apress Berkeley, CA, 2009. – 320 p. – DOI: https://doi.org/10.1007/978-1-4302-3365-7_7

6. Key derivation function: The SCKDF Scheme [Text] : Security and Privacy Protection in Information Processing Systems: 28th IFIP TC 11 International Conference Proceedings 2013 Germany – pp. 125–138. – DOI: https://doi.org/10.1007/978-3-642-39218-4_10

7. Password Storage Cheat Sheet Sheep [Electronic resourse] – Regime of access : www/ URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html. – 23.11.2022 y. – Title from the screen.

8. Tilborg H. Encyclopedia of Cryptography and Security [Text] / H. van Tilborg, S. Jajodia. – New York: Springer New York, NY, 2011. – 1416 p. – DOI: https://doi.org/10.1007/978-1-4419-5906-5_1197

9. Poisonous MD5 – Wolves Among the Sheep [Electronic resourse] – Regime of access : www/ URL: https://blog.silentsignal.eu/2015/06/10/poisonous-md5-wolves-among-the-sheep/. – 23.11.2022 y. – Title from the screen.

10. Cimpanu Catalin. A quarter of major CMSs use outdated MD5 as the default password hashing scheme. [Text] / Cimpanu Catalin // ZDNet. 17 June 2019.

11. FIPS Publication 180-1: Secure Hash Standard. National Institute of Standards and Technology (NIST). [Text] 1995. – p. 28.

12. FIPS Publication 180-2: Secure Hash Standard. National Institute of Standards and Technology (NIST). [Text] 2002. – p. 76.

13. Haunts S. Applied Cryptography in .NET and Azure Key Vault [Text] / Stephen Haunts. – Berkeley: Apress Berkeley, CA, 2019. – 228 p. – DOI: https://doi.org/10.1007/978-1-4842-4375-6_5

14. Bcrypt algorithm [Text] : In Proceedings of the FREENIX Track: 1999 USENIX Annual Technical ConferenceProvos, June 6–11, 1999, Monterey, California, USA. – p. 34.

15. Scrypt Is Maximally Memory-Hard. [Text] : In Advances in Cryptology – EUROCRYPT 2017., vol 10212. Springer, Cham. DOI: https://doi.org/10.1007/978-3-319-56617-7_2

16. Argon2: new generation of memory-hard functions for password hashing and other applications. [Text] : In 2016 IEEE European Symposium on Security and Privacy (EuroS&P) March 21-24, 2016 at the Congress Center Saar, Saarbrücken, Germay. IEEE, 2016.

17. Stronger Key Derivation via Sequential Memory-Hard Functions, [Text] : In BSDCan'09, May 09, 2009, Ottawa, Canada.

18. How expensive is it to crack a password derived with Argon2? Very [Electronic resourse] – Regime of access : www/ URL: https://research.redhat.com/blog/article/how-expensive-is-it-to-crack-a-password-derived-with-argon2-very/. – 16.11.2022 y. – Title from the screen.

19. Are Your Passwords in the Green? [Електронний ресурс] – Режим

доступу : www/ URL: https://www.hivesystems.io/blog/are-your-passwords-in-the-green. – 22.11.2022 р. – Назва з титул. екрана.

20. Bcrypt password cracking extremely slow? Not if you are using hundreds of FPGAs! [Electronic resourse] – Regime of access : www/ URL: https://scatteredsecrets.medium.com/bcrypt-password-cracking-extremely-slow-not-if-you-are-using-hundreds-of-fpgas-7ae42e3272f6. – 23.11.2022 y. – Title from the screen.

21. What Is a Hash Function in Cryptography? A Beginner’s Guide [Electronic resourse] – Regime of access : www/ URL: https://www.thesslstore.com/blog/what-is-a-hash-function-in-cryptography-a-beginners-guide/. – 23.11.2022 y. – Title from the screen.

22. Key derivation functions. [Electronic resourse] – Regime of access : www/ URL: https://cryptography.io/en/latest/hazmat/primitives/key-derivation-functions/. – 23.11.2022 y. – Title from the screen.

23. Hash Algorithm in .NET Core. [Electronic resourse] – Regime of access : www/ URL: https://jintechflow.wordpress.com/2021/06/28/hash-algorithm-in-net-core/. – 16.11.2022 y. – Title from the screen.

24. HMAC (Hash-Based Message Authentication Codes) Definition. . [Electronic resourse] – Regime of access : www/ URL: https://www.okta.com/identity-101/hmac/. – 17.11.2022 y. – Title from the screen.

25. What is a key derivation function (KDF) and how do they work? . [Electronic resourse] – Regime of access : www/ URL: https://www.comparitech.com/blog/information-security/key-derivation-function-kdf/. – 20.11.2022 y. – Title from the screen.