

МЕТОДИ ЗА ЗАЩИТЕНО СЪХРАНЕНИЕ НА ДАННИ

Дако Димов

Икономически университет – Варна/Катедра „Информатика“, Варна, България,
dako.dimov@ue-varna.bg

РЕЗЮМЕ

Докладът има за цел да разгледа заплахите за сигурността на чувствителни данни в съвременната дигитална ера като поставя акцент върху начина на съхранението им. Представят се методи за криптиране, хеширане и хеширане с осоляване на данни. Криптирането преобразува данните в шифрован текст като използва симетрични и асиметрични алгоритми, които се прилагат според нуждите на сигурността. Хеширането генерира уникални стойности за удостоверяване на целостта на данните, докато осоляването добавя допълнителни данни, за да предотврати атаки с предварително изчислени хеш стойности. Тези методи за защитено съхранение на данните, заедно с допълнителни техники като добавяне на "пипер", създават комплексен защитен механизъм, който повишава сигурността и намалява риска от злонамерени атаки.

Ключови думи: криптиране, хеширане, защита на данните

METHODS FOR SECURE STORAGE OF DATA

Dako Dimov

University of Economics Varna/Department of Informatics, Varna, Bulgaria, dako.dimov@ue-varna.bg

ABSTRACT

The report aims to review the security threats to sensitive data in today's digital era, focusing on the way it is stored. Methods for encryption, hashing and salting of data are presented. Encryption transforms the data into ciphertext using symmetric and asymmetric algorithms, chosen depending on the use case. Hashing generates unique values for checking the integrity of the data, whereas salting adds additional data to prevent attacks with precomputed hash values. These methods of secure data storage, alongside additional techniques, such as adding "pepper", create a complex defense mechanism, which increases the security and lowers the risk of malicious attacks.

Keywords: encryption, hashing, data security

ВЪВЕДЕНИЕ

Живеем в дигитална ера и кражбата на лични данни или друга потребителска информация е най-голямата заплаха. От значение е не само защитата на достъпа до информацията, а и начина ѝ на съхранение, за да бъде достатъчно надеждно защитена от злонамерени атаки. Сигурността на данните в интернет е от решаващо значение в днешно време, когато информацията става все по-дигитализирана и все повече хора използват онлайн услуги и съхранение на данни. С подобряването на технологиите хакерите също стават все по-находчиви. С откраднатите данни те могат да превземат банкови сметки, да имат достъп до лични данни и още много други. Защитата на личните данни става все по-трудна поради нарастващия брой потребители в интернет. Това налага нуждата за използване на методи за сигурно съхранение на данните, които да намалят риска от нейното изтичане.

1. СЪЩНОСТ И ВИДОВЕ

За да се осигури защитата на данните е необходимо прилагането на различни методи за защита - криптиране, хеширане, хеширане и осоляване. Тези техники представляват основни принципи на криптографията и са от съществено значение за защитеното съхранение на потребителската информация. Съществуват различни методи за защита на данните – запазване на данните в необработен вид, представяне на данните в криптиран вид, използване на хеширане на данните, и прилагане на хеширане и осоляване върху данните (Cloudian, 2023).

В момента криптирането е един от най-популярните и ефективни използвани методи за защита на данните. Според Националния институт по стандарти и технологии (NIST), обикновеният текст се отнася до „разбираеми данни, които имат значение и могат да бъдат разбрани без прилагане на декриптиране“ (The National Institute of Standards and Technology, 2023).

Криптирането на данни е метод за защита на поверителността на данните чрез преобразуването им в кодирана информация, наречена криптиран текст, която може да бъде декодирана само с ключ за дешифриране. Произлиза от гръцката дума „*kryptos*“, което означава „скрит“ и „графика“ – наставката, означаваща „писане“ (Techtarget, 2021).

Данните или обикновеният текст се криптират с алгоритъм за криптиране и ключ за криптиране. Процесът води до шифрован текст, който може да се види в оригиналната му форма само, ако е дешифриран с правилния ключ. Съществуват два основни вида криптиране на данни - симетрично криптиране и асиметрично криптиране, известно още като криптиране с публичен ключ (De Groot, 2015).

1.1. СИМЕТРИЧНО КРИПТИРАНЕ

Симетричното криптиране на данни е метод за защита на информация, при който се използва една и съща ключова информация както за криптиране, така и за декриптиране на данните. Този ключ може да бъде предварително споделен между комуникиращите страни или генериран на база сложни алгоритми. Основната идея зад симетричното криптиране е, че данните се преобразуват със специален алгоритъм и ключ, които ги правят неразбираеми за неоторизирани лица. Единствено тези, които разполагат със съответния ключ, могат да декриптират и възстановят оригиналната информация. Примери за алгоритми за симетрично криптиране са DES (Data Encryption Standard), AES (Advanced Encryption Standard) и Blowfish. Тези алгоритми се използват за защита на данни при трансфер през незащитени канали, съхранение на данни на устройства (нар. данни в покой) и много други приложения.

Data Encryption Standard (DES) е един от най-ранните и популярни алгоритми за симетрично криптиране. Този алгоритъм е разработен през 1970-те години от IBM. За преобразуване на данните DES използва метод на Feistel network, при който блокът от данни се разделя на две половини. Едната половина служи като „лява“ част, а другата като „дясна“ част на Feistel мрежата. След като са завършени всички етапи на преобразуване, последният етап се изпълнява в обратен ред, за да се получи криптиран резултат. Въпреки своята сложност, дължината на ключа (56 бита) на DES се оказва недостатъчна, за да гарантира сигурност срещу съвременните атаки (Diffie et al., 1977).

Blowfish е симетричен криптиращ алгоритъм, разработен от Брус Шнайер през 1993 година. Този алгоритъм се използва за криптиране на данни и се отличава със своята скорост и относително проста структура. За преобразуване на данните Blowfish използва структура на Файстъл мрежа, като всеки блок от данни минава през определен брой итерации на преобразуване. Всяка итерация включва операции като XOR с подключове, замяна със S-блокове и пермутации. Този алгоритъм е популярен заради своята скорост и относително лесна

реализация. Въпреки това, по-късите ключове могат да бъдат податливи на атаки от бързи изчислителни машини, особено ако бъдат използвани със слабо управление на ключовете. Поради това, Blowfish не се счита за най-сигурния алгоритъм за съвременни криптографски приложения (Schneier, 1994).

Advanced Encryption Standard (AES) е съвременен симетричен криптиращ алгоритъм, който замени Data Encryption Standard (DES) и Blowfish като стандарт за криптиране. Той предоставя по-висока сигурност и ефективност в сравнение с DES. AES е използван широко и се счита за един от най-сигурните методи за криптиране. AES е известен със своята криптографска сигурност и ефективност, тъй като използва по-дълги ключове и по-силни криптиращи методи. Подходящ е за широк спектър от приложения - от защита на данни при трансфер през мрежи до шифроване на данни в покой. Предоставя по-добра сигурност и отговаря на съвременните изисквания за криптографска защита.

Въпреки предимствата си, симетричното криптиране има и някои ограничения. Основното е, че обменът на ключове между комуникиращите страни може да бъде рисков и да доведе до компрометиране на сигурността, ако ключът попадне в ръцете на злонамерени лица. Затова е важно да се използват сигурни методи за обмен на ключове или да се избират алтернативни подходи като публично-частни ключове (асиметрично криптиране) (Padhiar, Sneha, 2021).

Симетричното криптиране на данни е по-бърз метод от асиметричното криптиране и се използва най-добре от индивидуални потребители или в затворени системи. Използването на симетрични методи с множество потребители в отворени системи, като например през мрежа, изисква предаване на ключа и създава възможност за кражба.

1.2. АСИМЕТРИЧНО КРИПТИРАНЕ

С развитието на компютърните технологии и интернет, се прилага криптография с публичен ключ (public key) и частен ключ (private key), които са математически свързани и могат да се използват само заедно, също позната като асиметрична криптография. При асиметричната криптография се използват два ключа, единият - публичен ключ се използва за криптиране, а другият - частен ключ се използва за дешифриране. Този метод позволява сигурна комуникация и обмен на информация между страни, без да се изисква предварително споделяне на общ ключ за криптиране и декриптиране. Освен за криптиране, асиметричната криптография се използва и за създаване на електронни подписи. Това позволява на страни да подпишат съобщения или документи с частния си ключ, което дава увереност в автентичността на информацията и нейния подател. Асиметричното криптиране се използва от множество потребители и в отворени мрежи, като интернет, тъй като публичният ключ може да се споделя свободно, без риск от кражба на данни. Най-често използваните видове асиметрично криптиране са ElGamal, RSA, DSA.

ElGamal е алгоритъм за асиметрично криптиране, който може да бъде използван както за криптиране на данни, така и за създаване на електронни подписи. Той се базира на сложни математически операции с групи и пресмятане на корени. Друга съвременна форма за криптиране с публичен ключ е RSA (Rivest-Shamir-Adleman), по името на своите създатели. Това е алгоритъм, който съществува от 1978 г. и все още се използва широко. Прилага се за криптиране на данни и за създаване на електронни подписи. Той използва факторизацията на големи прости числа и извършва сложни математически операции с модули, за да генерира ключовете и извършва криптирането и декриптирането.

Методът DSA (Digital Signature Algorithm) се използва за създаване на електронни подписи и автентикация. Той е специализиран да се използва за създаване на подписи за проверка на идентичността на изпращача на информация. Подобен на него е алгоритъмът ECDSA (Elliptic Curve Digital Signature Algorithm), който е също алгоритъм за създаване на електронни подписи, който използва елиптични криви в криптографията. Той е предпочитан поради своята ефективност и сравнително малкия размер на ключовете.

IES (Integrated Encryption Scheme) е метод, който комбинира асиметрично и симетрично криптиране, за да осигури бърз и ефективен процес на шифроване и декриптиране. Сходен на този метод са хибридните криптосистеми Hybrid Cryptosystems, съчетаващи предимствата на симетричното и асиметричното криптиране. Публичният ключ се използва за криптиране на симетричен ключ, който по-късно се използва за криптиране на самите данни. Това осигурява баланс между сигурност и изчислителна ефективност (IEEE, 2004).

Върху данните може да бъде приложено симетрично и асиметрично криптиране. Между двете форми на криптиране, асиметричното криптиране има тенденция да бъде по-силно поради еднопосочната си същност. От друга страна, това може да се приеме за недостатък, защото процесът на криптиране и декриптиране с асиметрични ключове може да бъде по-бавен, понеже изисква повече изчислителни ресурси в сравнение със симетричното криптиране. Като неудобство е също управлението на ключовете, особено частния ключ. Неговото компрометиране може да се окаже критично за сигурността на информацията.

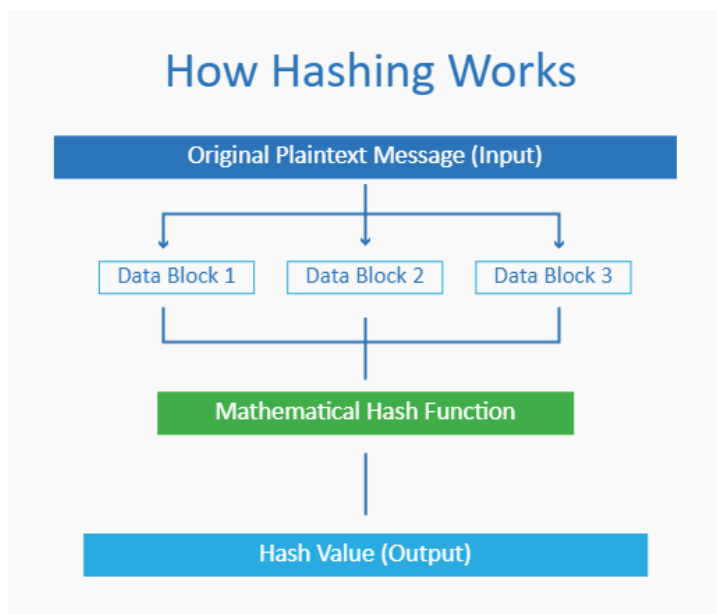
Всеки метод за криптиране на данни има своите предимства и недостатъци. Видът на използваното криптиране зависи от начина на достъп до данните и от кого. Важно е да се избере начинът, който ще осигури максимална защита за конкретния тип данни и за приложението, в което ще бъдат използвани.

2. ХЕШИРАНЕ

Хеширането е практика за трансформиране на даден ключ или низ от знаци в друга стойност с цел сигурност. При него се прилага алгоритъм за съпоставяне на данни от всякакъв размер към фиксирана дължина. Това се нарича хеш стойност, която представлява уникално представяне на входните данни. Хеш функциите са математически алгоритми, които генерират тези хеш стойности. Всеки алгоритъм за хеширане извежда данни с фиксирана дължина. Така всяка хеш стойност е уникална. Дори малка промяна във входните данни трябва да доведе до значително различаваща се хеш стойност. Това позволява лесно откриване на дори най-малките изменения в данните.

Хеш функциите са бързи за изчисление и подходящи за приложения, които изискват бързи проверки. Използват се за проверка дали данните не са били променени след генерирането на хеша. Използва се за създаване на уникални идентификатори за обекти, файлове, съобщения и други (Nohe, 2018).

Следната графика илюстрира как работи хеширането, като приетия вход се разделя на множество по-малки блокове данни (вж. Фигура 1).



Фигура 1. Схема на хеширане, чрез разделяне данните на блокове при входа

Източник: Създадена от (www.thesslstore.com, 2023)

Например, за да се подпише електронно част от софтуер и да се направи той достъпен за изтегляне от уебсайт, трябва да се създаде хеш на скрипта или изпълнимия файл, който се подписва. След добавяне на електронния подпис, всичко отново се хешира.

Когато потребителят изтегли софтуера, неговият браузър проверява двете уникални хеш стойности (вж. Фигура 2). Ако браузърът изведе една и съща хеш стойност, тогава той знае, че и подписът, и файлът са автентични - не са били променени. Ако не е, браузърът издава предупреждение за несъответствие (Code Signing Store, 2023).



Фигура 2. Хеширане на текст с алгоритъм SHA-2

Източник: Създадена от (www.thesslstore.com, 2023)

Основните видове алгоритми за хеширане са класифицирани в техните семейства. Най-разпространените семейства хеширащи алгоритми са Secure Hash Algorithm (SHA), Message Digest (MD), Whirlpool и други.

Алгоритъмът за хеширане MD5 (Message Digest Algorithm 5) работи като приема входни данни и ги обработва чрез няколко итерации на функции за преобразуване. В края на този процес се генерира 128-битова хеш стойност. Днес, той се счита за недостатъчно сигурен, поради възможността за колизии (различни входни данни, които генерират една и съща хеш стойност) и варианти на атаки. Whirlpool е метод за хеширане на данни, проектиран от Виктор Раймен (създател на алгоритъма AES) и Пауло Барето през 2000 г. Този алгоритъм е базиран на конструкцията "Меркъл-Дамгор", който генерира 512-битови хеш стойности, които обикновено се представят като 128-цифрени шестнадесетични числа. Той използва различни

нелинейни преобразувания и операции за миксиране на данните. Друг алгоритъм за хеширане е BLAKE2, който е бърз и сигурен, използващ паралелни процеси и битови операции. Той предлага по-добра производителност в сравнение с други хеш алгоритми. Поддържа разнообразни размери на хеш стойностите и се използва за широка гама от приложения (Code Signing Store, 2024).

Алгоритъмът за хеширане TIGER е сравнително нов алгоритъм, който започва да набира популярност в мрежите за споделяне на файлове и торент сайтовете. Разработен е от Рос Андерсън и Ели Бихам през 1995 г. Tiger е проектиран да работи бързо предимно на 64-битови компютри. Tiger няма патентни ограничения и може да се използва свободно. Понастоящем няма известни ефективни атаки срещу него. Сходен на него е алгоритъмът Scrypt. Той използва 128-битова произволна сол, която се добавя към съобщението и допълнително 192-битово число, което се използва за криптиране с алгоритъма Blowfish, който е труден за изчисление и добавя сложност при генерирането на хеш (Шнайер, 2002).

Тези са само някои от многото алгоритми за хеширане. Те се използват в различни сигурни приложения, включително проверка на целостта на данните, създаване на електронни подписи, хешове на пароли и други. Всеки алгоритъм има ясни предимства в зависимост от целта, за която е предназначен да постигне. Изборът на най-подходящ алгоритъм е свързан с необходимостта от получаване на точно очакваните резултати. При избор на алгоритъм е важно да се обърне внимание на неговата сигурност, скорост на обработка и предназначение на конкретното приложение. Също така, добра практика е да се използват актуални алгоритми и да се следват стандартите за криптографска сигурност.

Ключова разлика между криптиране и хеширане е, че криптирането е обратимо, докато хеширането не е. Не може да се възстановят оригиналните данни от хеш стойността. Това го прави полезен при удостоверяване на данни и съхранение на пароли, защото те не се съхраняват директно, а само техните хеш стойности. Въпреки, че термините „хеширане“ и „криптиране“ могат да се използват взаимозаменяемо, хеширането винаги се използва за целите на еднопосочно криптиране и хешираните стойности са много трудни за декодиране. Шифроването винаги предлага ключ за дешифриране, докато хешираната информация не може да бъде декодирана лесно и е предназначена да се използва като метод за валидиране на целостта на пакет информация или част от данни. Докато криптирането е предназначено да защитава данните при пренос и покой, хеширането служи като контролна сума, която има за цел да провери дали даден файл или част от данните не са били променени, т.е. че са автентични (Aggarwal, 2015).

3. ОСОЛЯВАНЕ

Осоляването включва добавяне на произволни данни, преди да бъдат пуснати през криптографска хеш функция. Прилагането на осоляване на данните е честа практика за повишаване на сигурността на паролите, както и за защита на чувствителни данни по време на съхранение. Това увеличава сигурността при атаки срещу предварително генерирани таблици с хеш стойности (таблица със съответствия). При осоляване на данните се генерира случайна сол (salt), която е уникална за всяка парола или чувствителни данни. Обикновено солта е низ с фиксирана дължина, като например 16 или 32 байта. Преди да се изпълни хеширането, солта се комбинира с данните (например паролата). Този процес може да бъде извършен по различни начини, като например конкатенация (слепване) на солта и данните. Комбинираният низ (солта

+ данните) се подлага на хеширане с подходящ алгоритъм за хеширане (като например SHA-256 или bcrypt.) В базата данни се съхранява и солта, и резултатът от хеширането (хеш стойността). Солта не е тайна информация и може да бъде съхранена заедно с хеш стойността. При проверка на парола (или други данни) се търси съответният запис в базата данни със съответната сол и хеш стойност. Комбинира се въведената парола със съхранената сол и се изчислява хеш стойността за комбинирания низ. След това се сравнява тази хеш стойност със съхранената в базата данни и, ако те съвпадат, паролата е валидна (Grassi et al.,2017).

Осоляването на данните е важен аспект на сигурността и криптографията. Като всеки метод за защита на чувствителни данни, то има своите предимства и недостатъци. Този подход прави речниковите атаки (brute-force) и атаките срещу предварително генерирани таблици с хеш стойности (rainbow tables) много по-трудни за реализиране, тъй като дори идентични пароли имат различни хеш стойности поради уникалността на добавената сол. Осоляването повишава сигурността дори при използването на слаби пароли. Солта допринася за увеличаване на сложността на атаките, тъй като всяка сол е уникална за всеки запис. Това прави атаките по анализ на честотата (frequency analysis) или други статистически методи много по-трудни. Недостатък на осоляването на данните е допълнителният код и обработка. Осоляването изисква допълнителен код за генериране и управление на солите. Този процес може да забави разработката и имплементацията. Понякога са възможни грешки при създаване на алгоритъма за генериране на солта. В случай, че генерирането на сол не се извърши правилно, това може да доведе до слаби места в сигурността и по-голяма уязвимост. Също неудобство е, че прилагането на осоляване може да доведе до по-голяма сложност при управлението на данните, тъй като всеки запис изисква уникална сол. Въпреки, че осоляването затруднява речниковите атаки, то не предпазва от атаки, насочени срещу използваните хеш функции (като криптоанализа). Използването на осоляване на данните е важен механизъм за подобряване на сигурността, особено при съхранението на пароли и чувствителни данни. Но е важно да бъде правилно имплементирано и управлявано, за да се избегнат възможни уязвимости (Майер, 2022).

Важен аспект на сигурността при съхранението на пароли и други чувствителни данни е запазването им във вид на хеширане и осоляване. Този процес помага да се предпазят данните от атаки, като например речникови атаки (brute-force). Запазването на данни във вид на хеширане и осоляване е широко прилагана практика и има своите предимства и недостатъци. Предимство на запазването на данни във вид на хеширане и осоляване е повишаване на тяхната сигурност при изтичане на данни. Запазването на данни във вид на хеширане и осоляване е силно средство за повишаване на сигурността на данните, особено при съхранение на пароли.

4. ДОБАВЯНЕ НА „ПИПЕР“

Друг метод за повишаване на сигурността при съхранение на данни е така нареченият „пипер“, който може да се добави към криптираните или хеширани данни и е подобен на сол. Това е низ, който се добавя към данните, но за разлика от солта, която обикновено се запазва заедно с паролата, пиперът трябва да се съхранява отделно, за да се предотврати получаването му от нападателя в случай на пробив в базата данни. Неговата идея е да затрудни операциите по проверка (в това число и опитите за разбиване) като добави етап на отгатване на пипера. Ако например сме добавили пипер към хеширащата функция и той е число от 0 до 9, то при проверка за съвпадение на дадена информация с тази, запазена на сървър, трябва да се извършат до 10 хеширащи операции, като се пробва с всяка възможна стойност от 0 до 9. Това

усложнява и забавя операцията за хеширане и я прави по-трудна за разбиване с груба сила, тъй като в зависимост от възможните стойности за пипера, трябва да се извършат толкова на брой различни хеширания докато се открие правилната стойност на пипера. Тъй като този пипер трябва да е еднакъв за всички потребители, а всички криптографски операции се извършват при клиента, няма как да се извършат тези операции без да се разкрие тайната на пипера (Lin, 2016).

ЗАКЛЮЧЕНИЕ

Данните са ценни, независимо дали се прехвърлят между потребители или се намират на сървър. Те трябва да бъдат защитени по всяко време. Защитата на данните е процес на обезпечаване на сигурността на цифровите данни и предотвратяване на загубата им чрез неототоризиран достъп. Включва сигурност на данните срещу атаки, които могат да доведат до промяна на съдържанието им, което от своя страна да ги направи ненадеждни. Защитата на данните гарантира, че те са достъпни единствено за този, който има право на достъп до тях.

REFERENCES / ИЗПОЛЗВАНА ЛИТЕРАТУРА

1. МАЙЕР, А. (2022). Управление на данните. Какво е това? [Online] Available from <https://bg.six.ms/blog/yprablenie-na-dannite> [Accessed 10/11/2024]
2. ШНАЙЕР Б. (2002). Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си, ISBN 5-89392-055-4
3. AGGARWAL K., VERMA H. K. (2015). Hash_RC6 — Variable length Hash algorithm using RC6 2015 *International Conference on Advances in Computer Engineering and Applications*, Ghaziabad, pp. 450-456,
4. DIFFI W., HELLMAN M. E. (1977) Special Feature Exhaustive Cryptanalysis of the NBS Data Encryption Standard, *Computer*, 10 (6), pp. 74-84
5. GRASSI P. A., GARCIA M. E., FENTON J. L. (2017), Digital Identity Guidelines, *NIST Special Publication 800-63-3*, pp. 54
6. IEEE. (2004). IEEE Standard Specifications for Public-Key Cryptography - Amendment 1: Additional Techniques, pp.1-167
7. PADHIAR, SNEHA. (2021). A Comparative Study on Symmetric and Asymmetric Key Encryption Techniques. pp. 133.
8. SCHNEIER B. (1994). The Blowfish Encryption Algorithm, *Dr. Dobbs's Journal*, 19 (4), pp. 38-40.
9. CLOUDIAN. (2023). [Online] Available from <https://cloudian.com/guides/data-protection/data-encryption-the-ultimate-guide/> [Accessed 10/11/2024]
10. CODE SIGNING STORE. (2024). *What Is a Hashing Algorithm? A Look at Hash Functions*. [Online] Available from <https://codesigningstore.com/what-is-hashing-algorithm-how-it-works> [Accessed 10/11/2024]
11. DE GROOT, J. (2015). [Online] Available from <https://www.digitalguardian.com/blog/what-data-encryption> [Accessed 10/11/2024]
12. LIN, R. (2016). *Salt & Pepper: Spice up your hash!* [Online] Available from <https://medium.com/@berto168/salt-pepper-spice-up-your-hash-b48328caa2af> [Accessed 10/11/2024]

13. NOHE, P. (2018). [Online] Available from <https://www.thesslstore.com/blog/difference-encryption-hashing-salting/> [Accessed 10/11/2024]
14. TECHTARGET. (2021). [Online] Available from <https://www.techtarget.com/searchsecurity/definition/plaintext> [Accessed 10/11/2024]
15. THE NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. (2023). *Glossary*. [Online] Available from https://csrc.nist.gov/glossary/term/clear_text/ [Accessed 10/11/2024]