Харківський національний університет імені В.Н. Каразіна
Факультет комп'ютерних наук
Кафедра безпеки інформаційних систем і технологій

«Допущено до захисту»

Зав.кафедрою БІСТ

Сватовський І.І._____
« » червня 2023р.

Пояснювальна записка
до кваліфікаційної роботи бакалавра
спеціальність: 125 Кібербезпека

на тему: «Дослідження шляхів та вироблення рекомендацій щодо створення змішаної криптосистеми захисту інформації на основі інфраструктури відкритих ключів»

оцінка « »

Керівник д.т.н. проф. Олійников Р.В.
_____
(прізвище та ініціали/підпис)

Голова ЕК

Рецензент доктор філософії з комп'ютерних наук Родінко М.Ю_____
(прізвище та ініціали/підпис)

Лемешко О.В._____

Виконавець студент групи КБ-41

Нємцев М. О._____
(прізвище та ініціали/підпис)

Харків – 2023

РЕФЕРАТ

Роботу виконано на 45 аркушах, вона містить 1 додаток, 3 таблиці, 15 ілюстрацій, використано 31 джерело.

Дана робота являє собою дослідження та аналіз можливостей створення змішаної криптосистеми на основі інфраструктури відкритих ключів для ефективного захисту інформації. У роботі розглядаються ключові аспекти криптографії, такі як асиметричне та симетричне шифрування, автентифікація, цифрові та електронні підписи, протоколи безпеки та алгоритми обміну ключами.

Представлена дипломна робота висвітлює важливі аспекти створення змішаної криптосистеми на базі інфраструктури відкритих ключів для забезпечення надійного захисту інформації. Дослідження охоплює аналіз наявних алгоритмів шифрування, методів аутентифікації та протоколів безпеки.

Мета роботи являє проектування інформаційної системи, яка забезпечить захист даних для організацій з великою кількістю документів. Програмний комплекс базуватиметься на RSA-шифруванні та використовуватиме принципи електронного цифрового підпису (ЕЦП) для підписання файлів.

Результати роботи можуть бути застосовані в різних сферах, де потрібен надійний захист інформації.

Ключові слова: КРИПТОСИСТЕМА, КРИПТОГРАФІЯ, ЗМІШАНА КРИПТОСИСТЕМА, ІНФРАСТРУКТУРА ВІДКРИТИХ КЛЮЧІВ, АЛГОРИТМИ ШИФРУВАННЯ, ХЕШУВАННЯ, АУТЕНТИФІКАЦІЯ, ЕЛЕКТРОННИЙ ПІДПИС, ЦИФРОВИЙ ПІДПИС, ВІДКРИТИЙ КЛЮЧ, ЗАКРИТИЙ КЛЮЧ, АЛГОРИТМИ ОБМІНУ КЛЮЧАМИ.

ABSTRACT

The report consists of 45 pages and includes 1 appendix. It incorporates 3 tables, 15 illustrations, and references 31 sources.

This work is a study and analysis of the possibilities of creating a mixed cryptosystem based on a public key infrastructure for effective information security. The paper discusses key aspects of cryptography, such as asymmetric and symmetric encryption, authentication, digital and electronic signatures, security protocols, and key exchange algorithms.

The presented thesis highlights important aspects of creating a mixed cryptosystem based on a public key infrastructure to ensure reliable information protection. The research includes an analysis of existing encryption algorithms, authentication methods, and security protocols.

The goal of the project is to develop a software system that provides data protection for local organizations with a large number of documents. The software system will be based on RSA encryption and use the principles of electronic digital signature (EDS) to sign files.

Results of the work can be applied in various areas where reliable information protection is required. The development provides functionality that does not allow copying or transferring a user's signature to another document.

Keywords: CRYPTOSYSTEM, CRYPTOGRAPHY, MIXED CRYPTOSYSTEM, PUBLIC KEY INFRASTRUCTURE, ENCRYPTION ALGORITHMS, HASHING, AUTHENTICATION, ELECTRONIC SIGNATURE, DIGITAL SIGNATURE, PUBLIC KEY, PRIVATE KEY, KEY EXCHANGE ALGORITHMS.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧОК, СКОРОЧЕНЬ ТА СИМВОЛІВ

ЕЦП	Електронно-цифровий підпис
ЦЗО	Центральний засвідчувальний орган
ЦС	Центр сертифікації
CRL	List of revoked certificates
CSR	Certificate signature request
PKI	Public key infrastructure
SHA	Secure hash algorithm

ВСТУП

Протягом останніх років більшої актуальності набуває онлайн-документообіг, і даний тренд продовжує зростати і зміцнюватися з роками по всьому світу. В умовах розвитку сучасного цифрового простору паперові технології зберігання та обробки інформації все частіше замінюються електронними засобами. За таких обставин створення електронного документообігу набуває особливого значення. Розвиток електронних засобів захисту документів має вирішальне значення, оскільки паперові документи мають атрибути, як друк, текстурні поверхні та ін. Електронний цифровий підпис сприяє полегшенню роботи, замінюючи паперові документи на електронні, якими можна вільно обмінюватися використовуючи мережу Інтернет.

Зараз зростає попит на безпечні системи електронного документообігу, які працюють з ЕЦП. Використання таких систем значно прискорює комерційні операції, зменшує кількість паперової документації, економить час і ресурси. Це стосується підписання договорів, звітування перед перевіряючи органами, отримання документів та необхідних довідок у державних підприємств та багато іншого [1].

Основою технології електронного цифрового підпису є криптографічні асиметричні перетворення, при яких підпис створюється за допомогою особистого секретного ключа, а перевіряється за допомогою відкритого ключа. Ця технологія вимагає використання сертифіката ключа - особливого електронного документа, необхідного для перевірки відкритого ключа та підтвердження законного володіння відповідним особистим ключем певною фізичною особою. Це дає змогу поступово перевести в електронний режим багато послуг, що надаються державою, а ще пришвидшити та забезпечити безпеку процедур реєстрації, обробки документів, підписання договорів, здійснення електронних розрахунків, підписання договорів, розвиток інтернет-торгівлі.

З моменту відкриття Центрального засвідчувального органу в Україні у 2002 році електронні цифрові підписи стали широко доступними для користувачів.

Серед основних обов'язків цього органу - реєстрація та акредитація засвідчувальних пунктів та центрів сертифікації ключів, ведення реєстру сертифікатів кореневих ключів.

Засвідчувальні центри, які отримали певні повноваження та акредитацію, що здійснюють генерацію і випуск сертифікатів криптографічних ключів для електронного цифрового підпису, виконують ключову роль у захисті інформаційних систем. Робота спрямована на проектування інформаційної системи для засвідчувального центру, який буде відповідати за генерацію та видачу сертифікатів ключів електронних цифрових підписів для виконання різноманітних вимог.

При створенні електронного цифрового підпису термін «послуги» стосується таких аспектів:

- надання доступу до інструментів електронного цифрового підпису;
- допомога в генерації відкритих і закритих ключів;
- управління сертифікатами відкритих ключів, включаючи їх блокування, скасування та відновлення;
- надавання відомостей про сертифікати;
- надання послуг з позначення часу.

Встановлення довіри є першочерговим завданням в інфраструктурі відкритих ключів. У централізованій системі сертифікації центр сертифікації виконує роль суб'єкта, якому довіряють. Публічні ключі разом з даними про їхніх власників, завірені підписом центру сертифікації, визнаються безпечними, отже центр сертифікації відіграє важливу роль у цьому відношенні.

Актуальністю даної роботи є дослідження та вироблення рекомендацій щодо створення змішаної системи криптографічного захисту інформації на основі інфраструктури відкритих ключів. Ця система спрямована на забезпечення створення, випуску та управління сертифікатами криптографічних ключів ЕЦП для користувачів в рамках будь-якої організації. Це рішення дозволить вирішити проблеми електронного обміну документами для внутрішніх потреб організації та сприятиме подальшому розвитку системи е-комерції.

Об'єктом дослідження є розглядання процесу роботи криптографічних

засобів в асиметричних перетвореннях.

Предметом дослідження є моделі інфраструктури публічних ключів, що використовуються для вирішення завдань створення та підтримки інформаційної системи для центру сертифікації ключів власної розробки.

Метою роботи є дослідження шляхів та вироблення рекомендацій щодо створення змішаної криптосистеми захисту інформації на основі інфраструктури відкритих ключів, що може бути впроваджена в будь-якій організації.

Інформаційна система повинна мати можливість генерувати власний закритий ключ і створювати сертифікат, який містить відкритий ключ центру та підписується самим центром. Система центру сертифікації повинна забезпечувати зберігання закритого ключа свого сертифіката, оскільки він використовується для підписання клієнтських сертифікатів. З іншого боку, відкритий ключ разом із сертифікатом має бути загальнодоступним, щоб користувачі могли вільно верифікувати сертифікати інших учасників.

1 ВИКОНАННЯ АНАЛІЗУ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ

1.1 Аналіз крипто-систем з відкритим ключем

Криптографічна система з відкритим ключем використовується для шифрування та електронного цифрового підпису (ЕЦП). У цій системі відкритий ключ передається відкритим загальнодоступним каналом і використовується для перевірки ЕЦП та шифрування повідомлень. Створення ЕЦП та розшифрування повідомлень вимагає використання секретного ключа. Основна ідея криптографії з відкритим ключем полягає у використанні односторонніх функцій, які дозволяють легко знайти значення функції за відомим вхідним значенням, але ускладнюють зворотну операцію знаходження вхідного значення за значенням функції. Однобічні функції ґрунтуються на відомих обчислювальних завданнях як [4]:

- RSA - факторизація об’ємних цілих чисел;
- EGSA - завдання, виконуюче дискретне логарифмування.

Далі виконаємо аналіз схеми для шифрування використовуючи відкритий ключ. K являє собою множину ключів, а e і d виконують ролі ключа шифрування та розшифрування. Функція шифрування, що використовується для довільного ключа, буде позначатися як E_e, де $e \in K$, тож маємо:

$$E_e(m) = c,$$

де $c \in C, C$ – множина шифртекстів,

$m \in M$, де M є множиною повідомлень.

Функція розшифрування, що дозволяє зробити пошук вихідного повідомлення m на основі шифр-тексту при умові відомого c, будемо називати D_d:

$$D_d(c) = m.$$

На рисунку 1.1 показано схему, що ілюструє передачу інформації між суб'єктом А та суб'єктом Б. У дійсності ці суб'єкти можуть бути як фізичними особами, так і компаніями/організаціями.

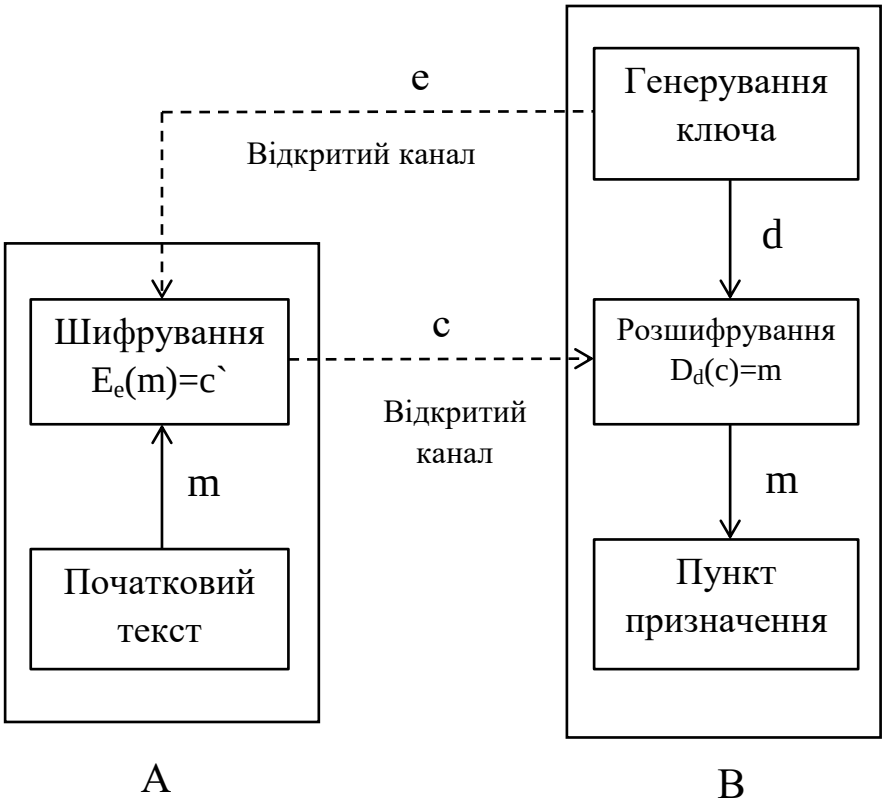

Рисунок 1.1 – Ілюстрація алгоритму обміну інформації між суб'єктами A і B

1.2 Аналіз алгоритмів відкритих ключів

Під час дослідження криптосистеми з відкритим ключем вважається, що вона є ідеальною системою, яка не потребує захищеного каналу для передачі ключа шифрування. За таких умов два законних користувача, A і B, можуть спілкуватися за допомогою відкритого каналу без необхідності фізичної зустрічі для обміну ключами [4]. Однак існує третя сторона, яка виступає в якості перехоплювача, здатного перехопити систему і розшифрувати повідомлення, призначене для особи B, не порушуючи систему шифрування, зображену на рисунку 1.2.

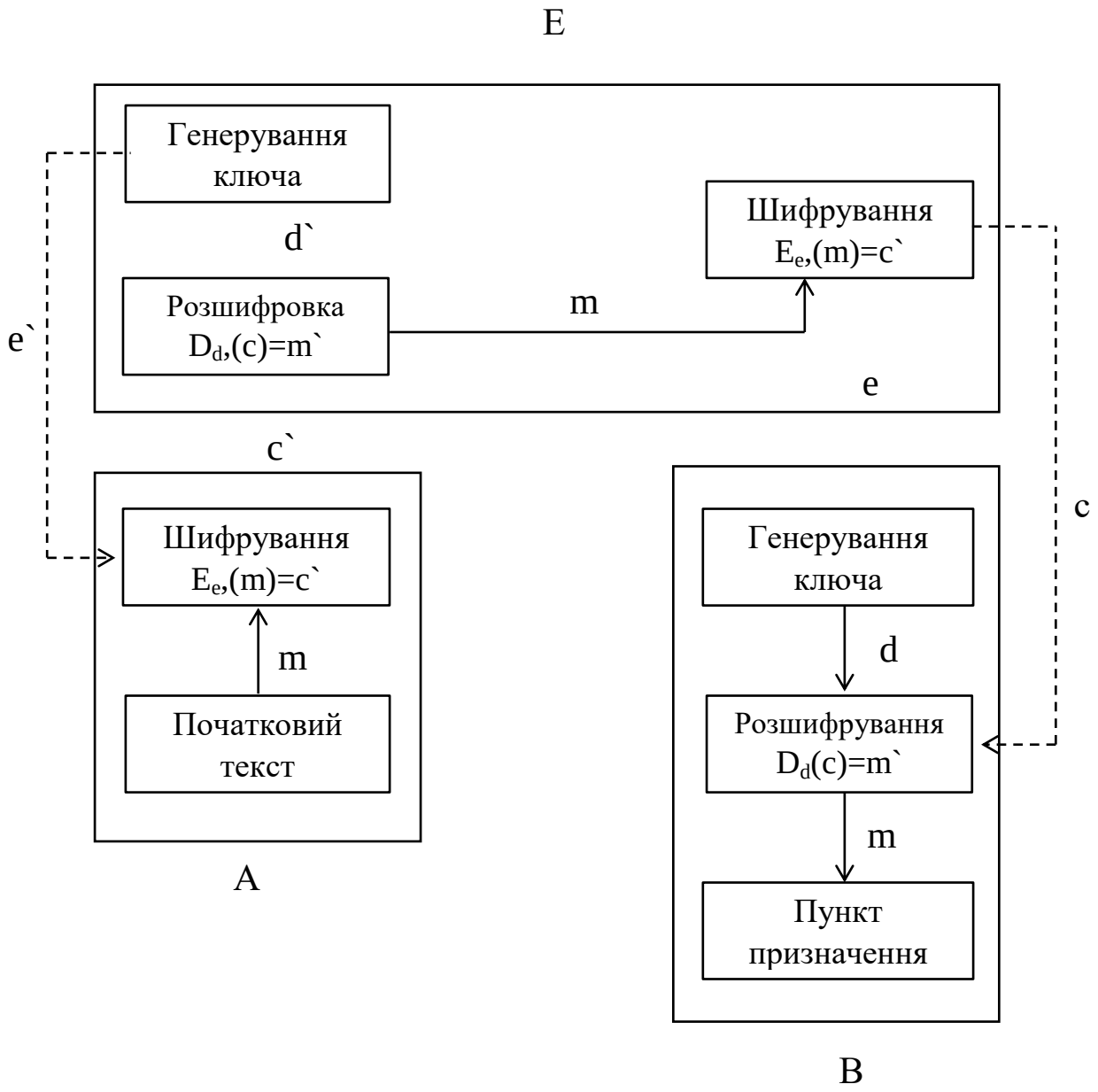

Рисунок 1.2 – Схема несанкціонованого перехоплення системи передачі даних третьою стороною (перехоплювачем)

Таку атаку називають атакою «людини посередині». Згідно з наведеною схемою, жоден з учасників не підозрює про існування третьої сторони, здатної перехопити повідомлення m і навіть підмінити його некоректним повідомленням m'. Це підкреслює необхідність автентифікації відкритих ключів, яка досягається за допомогою використання сертифікатів. У PGP розподілене управління ключами може бути досягнуто за допомогою довірених сторін [4].

Іншою формою атаки є підбір приватного ключа за допомогою відомого відкритого ключа, як показано на рисунку 1.3. Криптоаналітик, знаючи алгоритм

шифрування Ee, намагається знайти відповідний закритий ключ D_d. Процес буде полегшено, за умови, що аналітик зможе здійснювати перехоплення будь-яких зашифрованих текстів, що надсилаються від користувача А до користувача В.

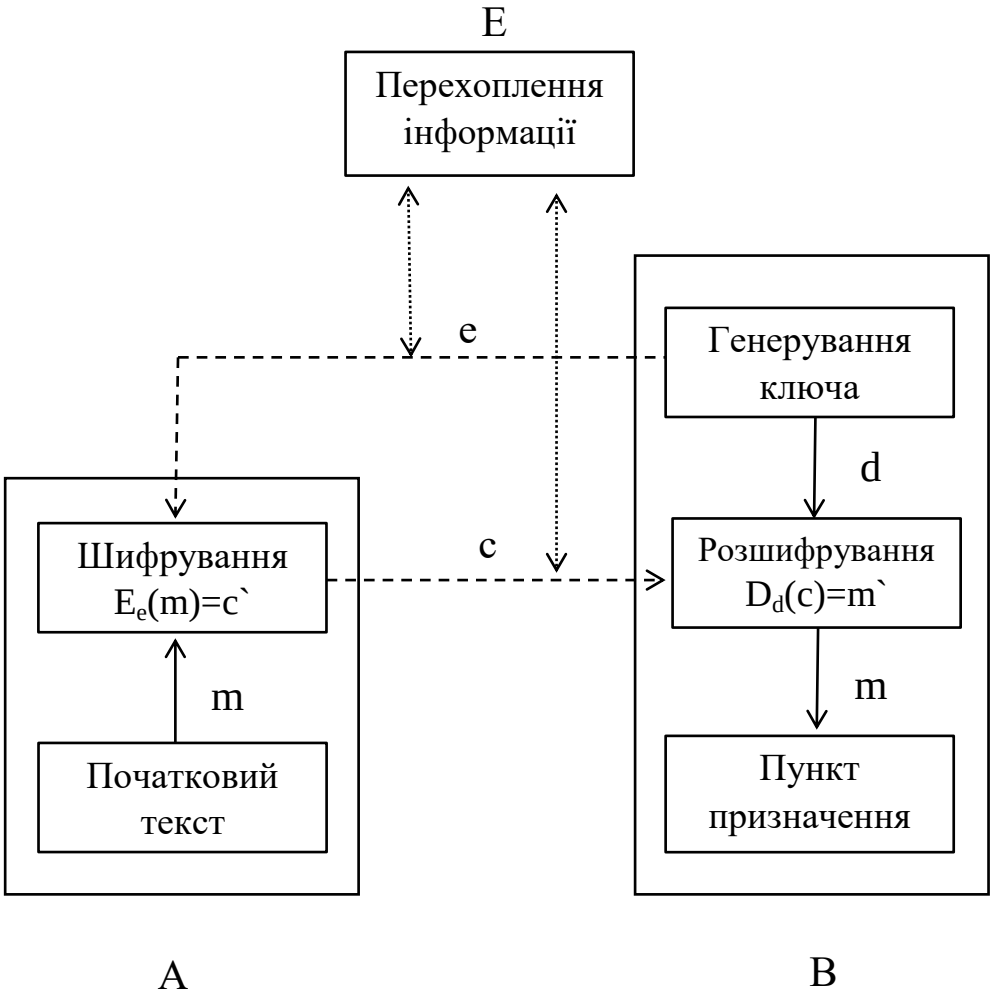

Рисунок 1.3 – Ілюстрація атаки методом підбору закритого ключа за наявності публічного ключа

Багато криптографічних систем з відкритим ключем ґрунтуються на проблемі розкладання великих чисел на множники. Так, наприклад, алгоритм RSA використовує в ролі публічного ключа n твір двох великих простих чисел. Розкладання числа n на прості множники є складним з обчислювальної точки зору завданням, що ускладнює безпеку таких алгоритмів. Однак з роками процес розкладання став швидшим та ефективнішим.

Крім того, з використанням потужних квантових комп'ютерів завдання факторингу потенційно може бути вирішене за алгоритмом Шора. Це становить потенційну загрозу для багатьох асиметричних методів шифрування. Рівень

криптографічної безпеки, заявлений розробниками алгоритмів на основі теоретичних оцінок, може суттєво відрізнятися від реальної безпеки, досягнутої за допомогою криптоаналізу. Це призвело до того, що багато країн законодавчо регулюють використання алгоритмів шифрування [4].

Асиметричне шифрування має свої переваги.

Перевагою асиметричних шифрів над симетричними є те, що немає необхідності передавати секретний ключ захищеним каналом зв'язку до початку комунікації.

Для використання симетричной криптографії обидві сторони повинні мати секретний ключ, тоді як в асиметричній криптосистемі використовується лише один секретний ключ.

У симетричному шифруванні ключі повинні оновлюватися після кожної передачі даних, тоді як в асиметричних криптосистемах ключові пари можуть залишатися незмінними протягом тривалого часу.

Однак асиметричне шифрування має і свої недоліки.

Симетричне шифрування має перевагу над асиметричним в тому, що його простіше змінити.

Незважаючи на те, що повідомлення надійно зашифровані, сам факт надсилання зашифрованого повідомлення може свідчити про те, що між відправником і одержувачем існує зв'язок.

Асиметричні алгоритми вимагають більших за розміром ключів, ніж симетричні.

Процес виконання шифрування і розшифрування за допомогою пари ключів в асиметричних алгоритмах зазвичай більш повільніший, ніж у симетричних алгоритмах.

Асиметричні криптосистеми потребують значних обчислювальних ресурсів, тому на більшості випадків їх використовують у поєднанні з другими алгоритмами.

Для електронного підпису спочатку відбувається процес шифрування повідомлення, а потім застосовується асиметричне шифрування для підписання невеликого результату хеш-функції.

У змішаних криптосистемах симетричні шифри використовуються для шифрування на сеансовому ключі великих обсягів даних, а асиметричне шифрування - тільки для передачі самого сеансового ключа.

1.2.1 Опис та призначення ЕЦП

Електронний цифровий підпис (ЕЦП) є важливим компонентом електронних документів, який забезпечує достовірність та цілісність інформації. Він дозволяє перевірити, що документ не був змінений з моменту створення підпису і що власник сертифіката ключа ЕЦП є уповноваженою особою.

Значення ЕЦП отримується за допомогою криптографічних операцій з використанням особистого ключа. Це дозволяє встановити автентичність підпису та підтвердити його належність конкретному власнику ключа.

Основне призначення електронного цифрового підпису - автентифікація особи, яка підписала електронний документ. Крім того, використання електронного цифрового підпису надає наступні переваги, які будуть описані нижче.

Контроль цілісності переданого документа забезпечується за допомогою перевірки підпису. Якщо документ змінено будь-яким чином, підпис стає непридатним, оскільки він був розрахований на основі початкового стану документа і відповідає лише цьому стану.

Це забезпечує захист від підробки документів, оскільки перевірка цілісності гарантує виявлення будь-яких змін. У більшості випадків підробка стає недоцільною. Крім того, власник не може відмовитися від свого авторства, оскільки створення дійсного підпису вимагає знання приватного ключа, який повинен бути відомий лише власнику.

Підписаний цифровим підписом документ слугує доказом авторства. Власник пари ключів може довести своє авторство підпису на документі, оскільки створення дійсного підпису можливе лише за умови знання закритого ключа. Залежно від призначення документа можуть бути підписані такі поля, як інформація про автора, зміни, які вносив автор, інформація про час генерації та підписання.

Електронний цифровий підпис являє собою шифрування повідомлення за

алгоритмом з використанням відкритого ключа. Зашифрований текст разом з оригінальним повідомленням об'єднуються. Перевірка підпису полягає в дешифруванні за допомогою відкритого ключа, і якщо отриманий текст збігається з оригінальним повідомленням, то підпис є дійсним.

Застосування хеш-функції дає змогу оптимізувати цей алгоритм. Замість того, щоб шифрувати все повідомлення, шифрується значення хеш-функції повідомлення. Цей метод дає наступні переваги:

- Зменшення розрахункової ємності, оскільки розмір документа зазвичай більший за розмір його хешу.
- Сумісність, оскільки переважна частина алгоритмів працює з бітовими рядками даних, а хеш-функція може бути використана для конвертації будь-якого вхідного текстового масиву у відповідний формат.
- Підвищена криптографічна стійкість, оскільки криптоаналітик не може створити підпис для самого повідомлення, використовуючи тільки відкритий ключ, а лише для його хеш-значення.
- Забезпечення цілісності, оскільки без застосування хеш-функції об'ємний електронний файл необхідно розбивати на блоки для накладання ЕЦП, а при перевірці неможливо визначити, чи всі блоки отримані і в правильному порядку.

До схеми асиметричного цифрового підпису входять три процеси: генерація пари ключів, генерація підпису та перевірка підпису. Для успішного використання цифрового підпису необхідно виконати дві умови. Підпис повинен бути перевірений за допомогою відкритого ключа, який відповідає закритому ключу, використаному для підписання. Без наявності приватного ключа створення дійсного цифрового підпису стає обчислювально складним. [6].

На рисунку 1.4 показано процес підписання та верифікації даних.

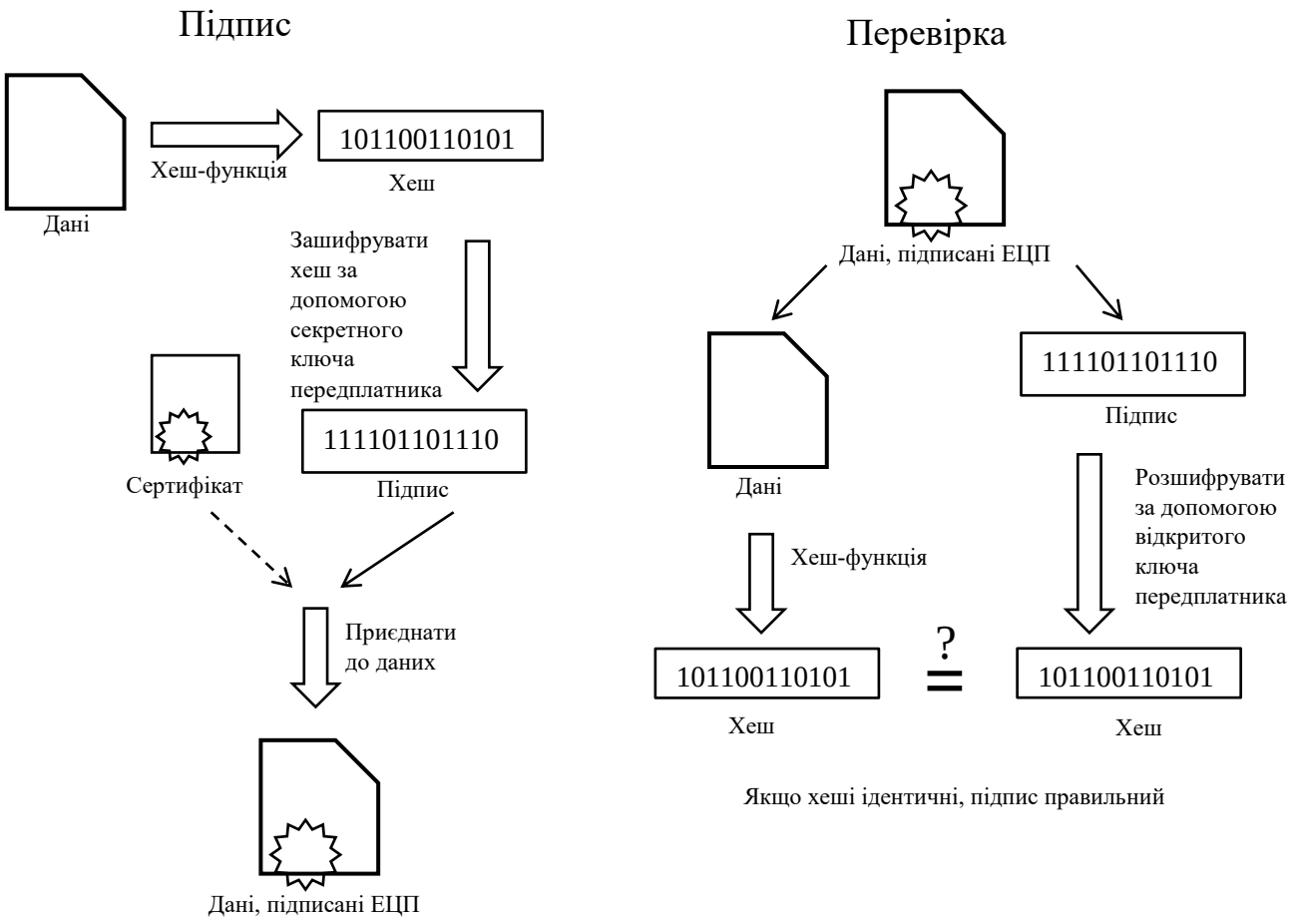

Рисунок 1.4 – Ілюстрація алгоритму підпису та перевірки даних

Опишемо найвідоміші алгоритми цифрового підпису:

1. Американські стандарти ЕЦП:
   - DSA (Digital Signature Algorithm)
   - ECDSA (Elliptic Curve Digital Signature Algorithm)
2. Український стандарт ЕЦП:
   - ДСТУ 4145-2002
3. Стандарт PKCS#1:
   - RSA
4. Схема Шнорра
5. Схема ЕльГамаля

RSA (Rivest-Shamir-Adleman) - широко відомий і широко використовуваний алгоритм цифрового підпису. Його математична схема була розроблена в 1977 році в Массачусетському технологічному інституті (MIT) в США [4].

Першим кроком є генерація пари ключів, що складається з особистого та відкритого ключів. Потім надсилач або власник електронного документа розраховує два великих простих числа, P і Q, а потім обчислює добуток цих чисел

$$N = P * Q$$

після чого виконується розрахунок

значення функції

$$\varphi(N) = (P - 1) * (Q - 1)$$

наступним кроком надсилаючий розраховує число E за наступними умовами:

$$\mathrm{E} \leq \varphi\ (N), \mathrm{НСД}\ (\mathrm{E}, \varphi(N)) = 1$$

і число D з умов:

$$D < N, \mathrm{E} * D \equiv 1\ (\mathrm{mod}\ \varphi\ (N))$$

Автор генерує ключову пару (E, N), яка виступає в якості відкритого ключа для верифікації цифрових підписів. Ці числа передаються партнерам по комунікації для перевірки підписів автора.

Автор також зберігає секретний ключ D для генерації цифрових підписів.

На рисунку 1.5 зображено узагальнену схему генерації та перевірки цифрового підпису RSA.

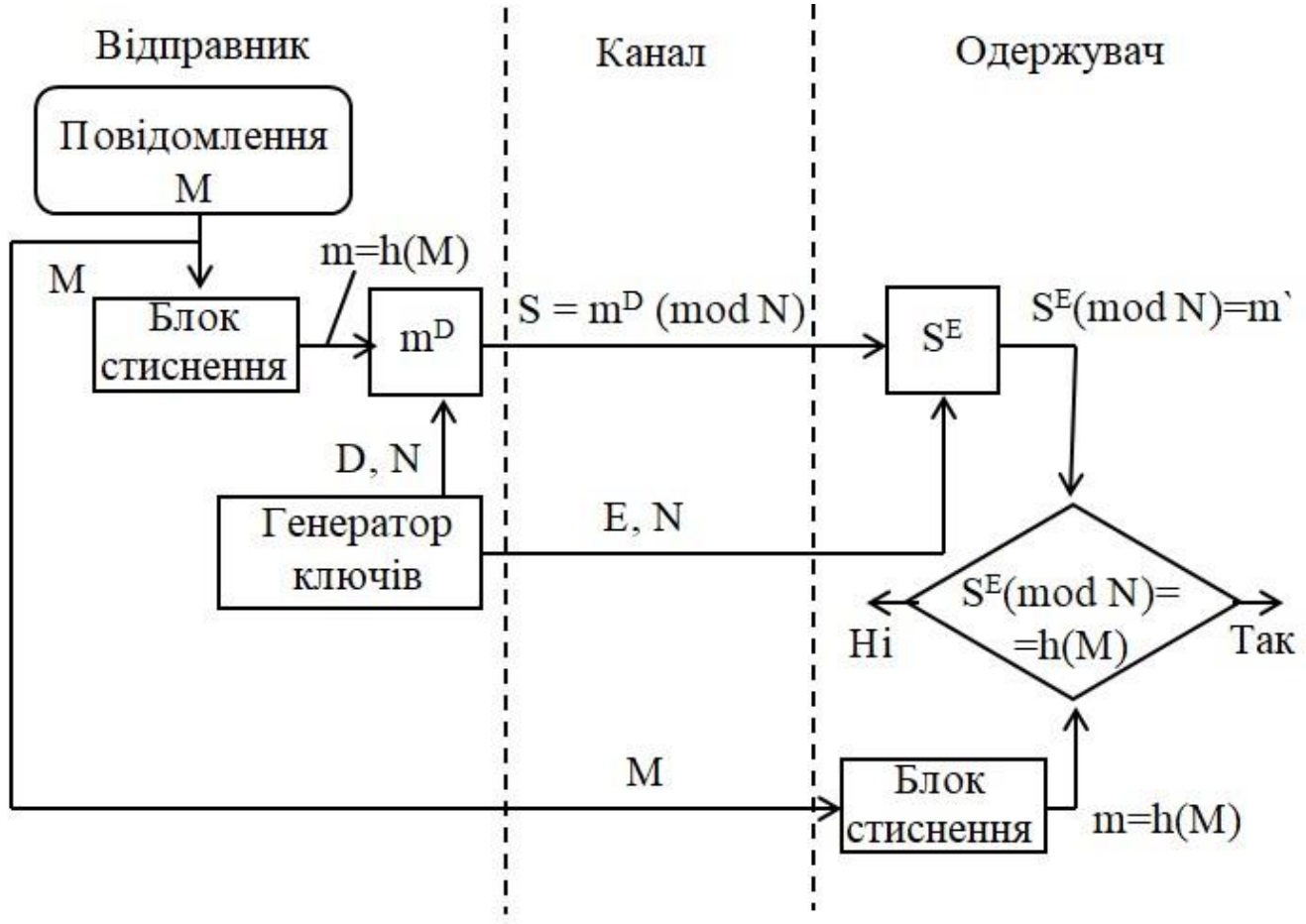

Рисунок 1.5 – Ілюстрація алгоритму генерації та перевірки RSA

Уявімо, наприклад, що відправник хоче підписати повідомлення M перш, ніж відправити його. По-перше, повідомлення M (блок інформації, файл, таблиця) стискається за допомогою хеш-функції h, яка перетворює його в ціле число m [5].

$$m = h(M).$$

Далі виконується розрахунок для цифрового підпису S для електронного документа M з використанням хешу m та приватного ключа D:

$$S = m^d\,(mod\ N).$$

Після передачі пари (M, S) одержувачу у вигляді електронного документа M з цифровим підписом S, де підпис S генерується власником секретного ключа D, одержувач розраховує хеш повідомлення M за допомогою двох способів.

По-перше, одержувач знаходить хеш-значення m', шляхом використання криптографічного алгоритму перетворення підпису S за допомогою відкритого ключа E [5]:

$$m' = S^E(mod\ N).$$

Також, одержувач обчислює хеш-значення отриманого повідомлення M,

використовуючи ту ж саму хеш-функцію h:

$$m = h(M).$$

У випадку виконання рівності при обчислені значень, а саме

$$S^E \,(mod\ N) = h\,(M),$$

одержувач вважає пару (M та S) автентичною. Доведено, що лише власник закритого ключа D має можливість згенерувати підпис S для документа M, а обчислити таємне число D за допомогою відкритого ключа E складніше, ніж поділити модуль N на прості множники.

До того ж, існує математичний доказ того, що верифікація цифрового підпису S дасть позитивний результат тільки в разі, коли під час обчислення S використовувався секретний ключ D, що співпадає з відкритим ключем E. Саме тому відкритий ключ E іноді називають «ідентифікатором підписувача».

Алгоритм цифрового підпису RSA має кілька недоліків.

При розрахунку значення ступеня N та ключів E і D для алгоритму RSA необхідно виконувати перевірку багатьох інших умов, що робить його досить складним на практиці. Порушення будь-якої з цих умов може призвести до підробки ЕЦП тим, хто дізнається про невідповідність. Така можливість під час оформлення особливо значущих документів недопустима ані теоретично, ані практично.

Для досягнення криптографічної стійкості цифрового підпису RSA до спроб підроблення на такому ж рівні, що і, для прикладу, національний стандарт шифрування інформації США зі стійкістю 10^{18}, потрібно застосовувати значення N, D та E у вигляді цілих чисел не менших за 2^{512} (або приблизно 10^{154}), що потребує суттєвих комп'ютерних ресурсів, які на 20-30% вищі за обчислювальні витрати інших алгоритмів цифрового підпису при тому ж самому рівні криптографічної стійкості [5].

Цифровий підпис RSA також є дуже слабким до впливу атак з використанням мультиплікативного підходу. Тобто алгоритм дозволяє зловмиснику створювати підписи без наявності секретного ключа D для документів, в яких результат шифрування обчислюється шляхом добутку значень хешу документів, що були підписані раніше.

1.3 Вироблення рекомендацій щодо створення змішаної криптосистеми захисту інформації

Захист інформації є важливим аспектом у сучасному інформаційному суспільстві. Зі зростанням обсягів переданих даних і збільшенням кількості кібератак, ефективні механізми захисту стають невід'ємною частиною забезпечення безпеки інформаційних систем. Одним із підходів до захисту інформації є використання змішаної криптосистеми на основі інфраструктури відкритих ключів. У цьому пункті наведені рекомендації щодо створення такої криптосистеми.

- Оцінка вимог до системи:

Перед розробкою змішаної криптосистеми необхідно провести аналіз вимог та оцінку загроз безпеці інформації. Це включає визначення конфіденційності, цілісності та доступності даних, а також ідентифікацію потенційних загроз і векторів атак. Ґрунтуючись на цій оцінці, можна визначити необхідні функції та механізми, які мають бути реалізовані в змішаній криптосистемі.

- Вибір алгоритмів шифрування:

Рекомендується вибирати надійні алгоритми шифрування для реалізації змішаної криптосистеми. Для асиметричного шифрування можна використовувати алгоритм RSA або еліптичну криптографію, а для симетричного шифрування - алгоритми AES або 3DES. Важливо обирати алгоритми з доведеною стійкістю і достатньою довжиною ключа для запобігання атак перебором.

- Управління ключами:

Ключовий аспект змішаної криптосистеми - правильне управління ключами. Рекомендується використовувати безпечні методи генерації ключів та їх зберігання. Ключі мають бути захищені від несанкціонованого доступу і регулярно оновлюватися. Також рекомендується використовувати протоколи обміну ключами, такі як протокол Діффі-Хеллмана, для безпечного обміну ключами між учасниками комунікації.

- Аутентифікація та цифрові сертифікати:

Для забезпечення автентифікації та перевірки автентичності учасників комунікації рекомендується використовувати цифрові сертифікати. Цифрові

сертифікати, що видаються довіреними центрами сертифікації, містять відкритий ключ та інформацію про ідентифікацію власника. Перевірка цифрових сертифікатів дає змогу переконатися в тому, що сторони, які комунікують, дійсно є тими, за кого себе видають.

- Реалізація захисту від атак:

Важливим аспектом створення змішаної криптосистеми є реалізація механізмів захисту від різних атак. Рекомендується використовувати методи захисту від підбору ключа, як-от блокування після кількох невдалих спроб аутентифікації. Також рекомендується реалізувати механізми виявлення та запобігання атакам, такі як моніторинг мережевого трафіку та систем аналізу журналів. Вдосконалення існуючих методів протидії новим способам здійснення атак, має ретроспективний характер, оскільки інформація про поточні загрози стає доступною лише після їх реалізації, що обумовлює певну пролонгованість циклу зворотних реакцій і модифікації наявних технологій захисту. [31]

1.4 Постановка задачі

Задача даної роботи – вироблення рекомендацій щодо проектування власного центру сертифікації. Центр сертифікації має вирішувати наступні задачі:

- створення і видачу сертифікатів криптографічних ключів електронного цифрового підпису користувачів;
- гарантія збереження в таємниці закритого ключа електронного цифрового підпису;
- керування сертифікатами (містить у собі можливість припинення і поновлення дії сертифікатів, їх анулювання у випадку компрометації ключа або інших причин);
- управління реєстром ключових сертифікатів, забезпечення його актуальності і зручного доступу для користувачів;
- ведення списку відкликаних сертифікатів – списку скомпрометованих або недійсних, по будь-якій причині сертифікатів, – підтримка його в актуальному стані, випуск відновлень;
- повідомлення власників сертифікатів про закінчення терміну дії сертифіката.

Сертифікаційний центр відповідає за генерацію власного закритого ключа і створення сертифіката, який включає відкритий ключ центру і підписується самим центром. Закритий ключ кореневого сертифіката надійно зберігається в центрі сертифікації і використовується для підписання клієнтських сертифікатів.

Публічний ключ, також як і сертифікат, повинен бути загальнодоступним, оскільки користувачі використовують їх для перевірки сертифікатів кожного окремого учасника.

2 ОЦІНКА СТРУКТУРИ ВІДКРИТИХ КЛЮЧІВ

Технологія аутентифікації на основі відкритих ключів, відома як інфраструктура відкритих ключів (PKI).

Основні складові інфраструктури відкритих ключів включають наступне [5]:

- установлення довіри, що передбачає встановлення взаємного довірчого відношення між суб'єктами в межах певної моделі довіри;
- механізми адміністрування, включаючи керування ключами, контроль доступу та аудит, що забезпечують ефективне управління та безпеку інфраструктурою відкритих ключів;
- механізми керування сертифікатами, включаючи їх видання, відкликання, оновлення та зберігання для забезпечення актуальності та надійності інформації про ключі;
- система іменування суб'єктів, що забезпечує унікальність ідентифікаторів суб'єктів в рамках цієї системи.

Інфраструктура відкритих ключів не забезпечує авторизацію, перевірку достовірності, назву криптографічних об'єктів, інформаційну безпеку або канали зв'язку, але може бути використана як компонент для реалізації цих функцій.

Також інфраструктура відкритих ключів дозволяє використовувати її для забезпечення конфіденційності, цілісності даних, що передаються, та неможливості відмови від них.

2.1 Об'єкти, які входять до складу відкритих ключів

Інфраструктура відкритих ключів реалізується в моделі клієнт-сервер, тобто перевірка будь-якої інформації, надаваною інфраструктурою може відбуватися тільки з ініціативи клієнта.

Основні компоненти інфраструктури відкритих ключів.

Центр, що забезпечує зв'язок ідентичності суб'єкта криптографії (звичайне ім'я і будь-яка додаткова інформація) і його відкритого ключа. У дуже малих інфраструктур відкритих ключів центр, що засвідчує, може бути відсутнім, у

середніх і великих він обов'язково присутній.

Сертифікат відкритого ключа – це підписана засвідчувальним центром структура даних ідентичності суб'єкта криптографії і його відкритого ключа. Для підпису сертифіката засвідчувальний центр випускає сертифікат, який підписує ключем, зазначеним у сертифікаті.

Такий сертифікат називається самопідписаним (самовиданим) [7].

Реєстраційний центр – компонент системи, що перевіряє правильність наданих суб'єктом криптографії даних для формування запису про ідентичність. Засвідчувальний центр цієї інформації, довіряє реєстраційному центру перевірку. Реєстраційний центр, виписує сертифікат тільки тоді, коли перевіривши правильність інформації, підписує її своїм ключем і передає засвідчувальному центру, який, перевіривши ключ реєстраційного центру. Один реєстраційний центр може працювати з декількома засвідчувальними центрами, тобто входити до декількох інфраструктур відкритих ключів. Виходить що один засвідчувальний центр може працювати з декількома реєстраційними центрами.

Репозиторій – сховище випущених ЦС сертифікатів. У Законі України «Про електронний цифровий підпис». Він називається реєстр сертифікатів ключів підписів.

Архів сертифікатів – сховище всіх виданих будь-коли сертифікатів, які включають сертифікати зі строком дії, який є завершеним. Архів використовується для перевірки дійсності електронного підпису, яким засвідчувалися документи.

Центр реєстрації – опціональна компонента інфраструктури, призначена для реєстрації кінцевих користувачів і забезпечення їх взаємодії із центром сертифікації.

Мережний довідник – опціональна компонента інфраструктури, що містить сертифікати і списки відкликаних сертифікатів, що служить для цілей поширення цих об'єктів серед користувачів.

Основні задачі системи інформаційної безпеки, які вирішує інфраструктура керування відкритими ключами, які наведені нижче [8]:

- забезпечення конфіденційності інформації;

- забезпечення цілісності інформації;
- забезпечення аутентифікації користувачів і ресурсів, до яких звертаються користувачі;
- забезпечення можливості підтвердження зроблених користувачами дій з інформацією (незреченність).

2.2 Огляд принципу роботи сертифікату відкритого ключа

Сертифікат відкритого ключа, наприклад, сертифікат електронного підпису або сертифікат ключа підпису, - це електронний чи друкований на папері документ, який засвідчує належність між публічним ключем та даними, що підтверджують особу власника цього ключа. Він включає в себе такі дані: інформацію про його власника, реквізити відкритого ключа, мету та сферу його застосування, орган сертифікації тощо.

За допомогою відкритого ключа можна встановити захищений канал зв'язку з власником ключа кількома шляхами:

- верифікація електронного підпису користувача;
- шифрування переданих даних для забезпечення конфіденційності.

Принцип роботи сертифікату публічного ключа. Для обміну зашифрованими даними у великих мережевих системах використовуються сертифікати. Шифрування з відкритим ключем дозволяє усунути перешкоди для обміну приватними ключами для безпечного обміну даними між користувачами, але не забезпечує довіру до відкритих ключів. Фундаментальна ідея сертифікату полягає в наявності довіреної третьої сторони, яка користується довірою інших сторін для полегшення обміну інформацією. Вважається, що існує лише обмежена кількість таких довірених третіх сторін, а відкриті ключі цих сторін відомі всім за допомогою певних засобів.

Тому будь-яка спроба третьої сторони підробити відкритий ключ може бути досить просто виявлена [8]. Формат сертифіката визначає обов'язкові та необов'язкові елементи, які можуть бути присутніми в ньому (наприклад, X.509). Як правило, сертифікат містить у собі наступні поля:

- ім'я власника сертифіката (ім'я користувача, якому належить сертифікат);

- один або кілька відкритих ключів власника сертифіката;
- ім'я засвідчувального центру;
- серійний номер сертифіката, привласнений засвідчувальним центром;
- термін дії сертифіката (дата початку дії і дата закінчення дії); інформація про використані криптографічні алгоритми;
- електронний цифровий підпис генерується за допомогою секретного ключа центру сертифікації і використовується для підписання результату перевірки всієї отриманої в сертифікаті інформації.

2.3 Основи роботи центру сертифікації

Центр сертифікації - це установа або відділ організації, який займається видачею сертифікатів для електронних цифрових підписів, також відомий як засвідчувальний центр. Як частина глобальної системи каталогів, він відповідає за управління криптографічними ключами клієнтів. Центри сертифікації зберігають відкриті ключі та іншу інформацію про користувачів у форматі сертифікатів та генерують власний секретний ключ. Сертифікат, який містить відкритий ключ даного центру і підписаний їм самим. В майбутньому, після створення і підпису власного сертифіката, центр веде базу даних виданих сертифікатів і списків відкликаних сертифікатів.

Для нормальної роботи із центром сертифікації його власний сертифікат потрібно мати доданий у список довірених на тому комп'ютері, де його потрібно використовувати. Після включення в такий список, будь-який сертифікат, виданий довіреним центром, який є достовірним, а його власник – гідним довіри. Засвідчувальний центр має забезпечувати наступні дії [8]:

- видача сертифікатів на ключі ЕЦП;
- генерація ключів електронного цифрового підпису для учасників інформаційної системи із забезпеченням конфіденційності особистих ключів;
- призупинення, поновлення та скасування чинності сертифікатів ключів підпису;
- Ведення реєстру сертифікатів ключів ЕЦП з актуальною та доступною для учасників інформацією;

- видача сертифікатів ключів підпису у форматі паперових документів чи електронних файлів з відповідною довідкою про їх чинність.;
- ведення обліку сертифікатів ключів підпису клієнтів та перевірка автентичності електронних цифрових підписів щодо виданих сертифікатів;
- Надання користувачам інформаційних систем додаткових послуг, передбачених законодавством, що стосуються застосування ЕЦП.

2.4 Складання структури відкритих ключів

В основному виділяють п’ять видів архітектурних інфраструктур відкритих ключів [7]:

- проста інфраструктура відкритих ключів (одиночний ЦС);
- ієрархічна інфраструктура відкритих ключів;
- мережна інфраструктура відкритих ключів;
- крос-сертифіковані корпоративні інфраструктура відкритих ключів; архітектура мостового ЦС.

В основному інфраструктура відкритих ключів діляться на різні архітектури по наступних ознаках:

- кількість ЦС (а також кількість ЦС, які довіряють один одному); складність перевірки шляху сертифікації;
- наслідок видачі зловмисником себе за ЦС.

Найпростіша з архітектур – архітектура одиночного ЦС. У цьому випадку всі користувачі довіряють одному ЦС і листуються між собою. У даній архітектурі, якщо зловмисник видасть себе за ЦС, необхідно просто перевипустити всі виписані сертифікати і продовжити нормальну роботу.

Ієрархічна інфраструктура відкритих ключів.

Ієрархічна структура – архітектура інфраструктури відкритих ключів, що найчастіше зустрічається. У цьому випадку головним всієї структури стоїть один Головний ЦС, якому все довіряють і йому підкоряються нижчестоящі ЦС. Крім цього головного ЦС у структурі присутній ще один ЦС, який підкоряється вищому, якому у свою чергу приписані будь-які нижчестоящі або користувачі ЦС. Приватний приклад ієрархічної інфраструктури відкритих ключів являє

собою корпоративна інфраструктура відкритих ключів. Якщо в нас є одна велика фірма, у якої в підпорядкуванні безліч філіалів по всій країні. У головному будинку фірми є головний ЦС і в кожному філіалі є ЦС, який підкоряється головному. В ієрархічній інфраструктурі відкритих ключів, навіть якщо зловмисник видав себе за будь-який ЦС, мережа продовжує працювати без нього, а коли він відновлює нормальну працездатність – він просто знову включається в структуру [7].

Верифікація сертифікатів може здійснюватися за допомогою побудови послідовності сертифікатів, починаючи з кореневого центру сертифікації, і підтвердження відповідності суб'єкта сертифіката та відкритого ключа, що його випускає. У процесі перевірки передбачається, що "дійсні" ланцюжки стартують з сертифікатів, випущених одним центром сертифікації.

Існує два методи отримання сертифікатів для перевірки ланцюжка: push-модель і pull-модель. У push-моделі відправник надає всі сертифікати в ланцюжку разом зі своїм власним сертифікатом, що дозволяє одержувачу негайно перевірити їх. У моделі pull відправник додає лише свій власний сертифікат, а одержувач отримує сертифікат центру сертифікації самостійно. Оскільки кожен сертифікат містить ім'я емітента, отримувач може дізнатися, де йому можна верифікувати сертифікат.

Перевага ієрархічної структури полягає в тому, що не потрібно автоматично довіряти всім центрам сертифікації. Насправді, єдиним центром сертифікації, який підлягає довірі, є кореневий центр сертифікації.

Мережна інфраструктура відкритих ключів.

Мережна архітектура інфраструктура відкритих ключів також є і окремим випадком ієрархічної архітектури. Але в цьому випадку немає одного головного ЦС, якому все довіряють. У цій архітектурі всі ЦС довіряють ЦС, які знаходяться поруч, а кожний користувач довіряє тільки тому ЦС, в якому виписав сертифікат. У дану архітектуру інфраструктури відкритих ключів легко додається новий ЦС. У даній архітектурі найбільш складна побудова ланцюжків сертифікації [10].

Незалежні центри сертифікації можуть брати участь у взаємній сертифікації, яка передбачає видачу сертифікатів та обмін ними між собою. Цей

процес уможливлює взаємодію між центрами сертифікації та користувачами з інших доменів системи відкритих ключів. В кінцевому підсумку це призводить до формування довірчої спільноти.

Завдяки мережевій структурі, компрометація одного центру сертифікації в мережі не призводить до повної втрати довіри до всієї інфраструктури публічних ключів.

Цей вид архітектури який можна розглядати як змішаний вид ієрархічної і мережної види архітектур. Є кілька фірм, у кожної з яких організована деяка своя інфраструктура відкритих ключів, але вони прагнуть спілкуватися між собою, у результаті чого виникає загальна міжфірмова інфраструктура відкритихключів.

В архітектурі крос-сертифікованої корпоративної інфраструктури відкритих ключів досить складна система ланцюжка сертифікації [5].

Архітектура мостового ЦС розроблялась щоб убрати недоліки складного процесу сертифікації в крос-сертифікованій корпоративній інфраструктурі відкритих ключів.

У цьому випадку всі компанії довіряють не одній або двом фірмам, а одному певному мостовому ЦС, являє собою практично їх головним ЦС, але він не є основним пунктом довіри, а виступає в ролі посередника між іншими ЦС.

2.5 Базові принципи роботи хеш-функцій

Хеш-функція перетворює довільний масив вхідних даних у фіксований вихідний бітовий рядок. Цей процес, також відомий як хешування або згортка, дозволяє отримати унікальний ідентифікатор вхідних даних, який називається хеш-код чи дайджест повідомлення.

Хеш-функції використовуються для співставлення масивів даних: якщо два масиви мають різні хеш-коди, то вони обов'язково будуть різними. У випадку, якщо хеш-коди однакові, то висока ймовірність того, що масиви також будуть однаковими. Однак варто зазначити, що хеш-функції не забезпечують однозначної відповідності між вхідними даними та хешом через обмежену кількість можливих значень хеш-функції. Це означає, що існує багато варіантів вхідних даних, які дають однаковий хеш-код, і це називається колізією. Ймовірність колізій важлива при оцінюванні ефективності хеш-функцій [5].

У криптографії існують хеш-функції, до яких висуваються особливі вимоги. Щоб хеш-функція H могла бути визнана криптографічно безпечною, вона має відповідати трьом основним критеріям, які є основоположними для переважної частини використання хеш-функцій:

- односторонність: для певного хеш-значення m має бути важко визначити блок даних X, де H(X) = m. Іншими словами, не існує ефективного способу відновити вихідні дані з хеш-коду;
- стійкість до першого попереднього зображення: для певного значення M повинно бути обчислювально неможливо знайти інше значення N, де H(N) = H(M). Це означає, що неможливо знайти два різних повідомлення з однаковим хеш-кодом.
- друга стійкість до попереднього зображення: знайти пару повідомлень, які мають однаковий хеш-код, має бути практично неможливо з розрахунком на обчислення. Це означає, що важко знайти два різних повідомлення, які мають однаковий хеш-код.

Ці вимоги не є самостійними, адже функція, яка не задовольняє вимогу односторонності, також не буде стійкою до колізій першого та другого роду. Важливо також відзначити, що існування односторонніх хеш-функцій, які теоретично унеможливлюють знаходження будь-якого попереднього зображення, що відповідає заданому хешу, не доведено. Звичайне обчислення оберненого значення є лише складним з практичної точки зору завданням.

Атака «день народження» - це техніка атаки, яка дозволяє виявляти колізії для хеш-функцій довжиною n біт в середньому при приблизно $2^{n/2}$ значеннях хеш-функції. Таким чином, n-бітова криптографічна хеш-функція є безпечною, якщо розрахункова трудомісткість пошуку колізій близька до $2^{n/2}$.

Також важливим для функцій криптографічного гешування є те, що навіть найменша зміна вхідного аргументу суттєво змінює хеш-значення (лавиноподібний ефект). В свою чергу, хеш-значення не повинно містити жодної інформації про окремі біти вхідних даних. Ця вимога забезпечує криптографічну безпеку хеш-функцій, які використовуються для хешування паролів користувачів з метою отримання ключа.

Відомі хеш-функції зображені на рисунку 2.1:

Adler-32	PJW-32	SHA-256
CRC	RIPEMD-128	SHA-384
Hashcart	RIPEMD-160	SHA-512
HAVAL	RIPEMD-256	Skein
Keccak	RIPEMD-320	Snefru
LM-хеш	SHA-1	Tiger
MD6	SHA-2	TTH
N-Hash	SHA-224	Whirlpool

Рисунок 2.1 – ілюстрація відомих хеш-функцій

Основним принципом побудови хеш-функцій є ітераційна послідовна схема. Згідно з цим методом, ядром алгоритму є перетворення від k біт до n біт. Тут n - розмір результуючого хешу, а k - будь-яке число, що перевищує n. Основне перетворення повинно володіти всіма властивостями хеш-функції, такими як неінвертованість і відсутність можливості незмінної підстановки вхідних значень [4].

Процес хешування виконується за рахунок використання допоміжної змінної проміжного типу розміром n біт. Початковим значенням цієї змінної може бути будь-яке значення, відоме всім сторонам. Вхідні значення поділяються на блоки довжиною (k-n) біт. Під час кожної наступної ітерації хешування наступна (k-n) бітова частина вхідних даних об'єднується з проміжним значенням, отриманим на попередній ітерації, і до отриманого k-бітного блоку застосовується перетворення піднесення до степеня. Це призводить до "перемішування" всього вхідного тексту з первинним значенням додаткової перемінної.

Через особливості базового перетворення його часто називають функцією стиснення. Величина додаткової змінної після останньої ітерації стає вихідним хешем, як показано на рисунку 2.2. Інколи до отриманого значення застосовуються додаткові перетворення. Однак, якщо функція стиснення розроблена з достатньою стійкістю, ці додаткові перетворення стають непотрібними [7].

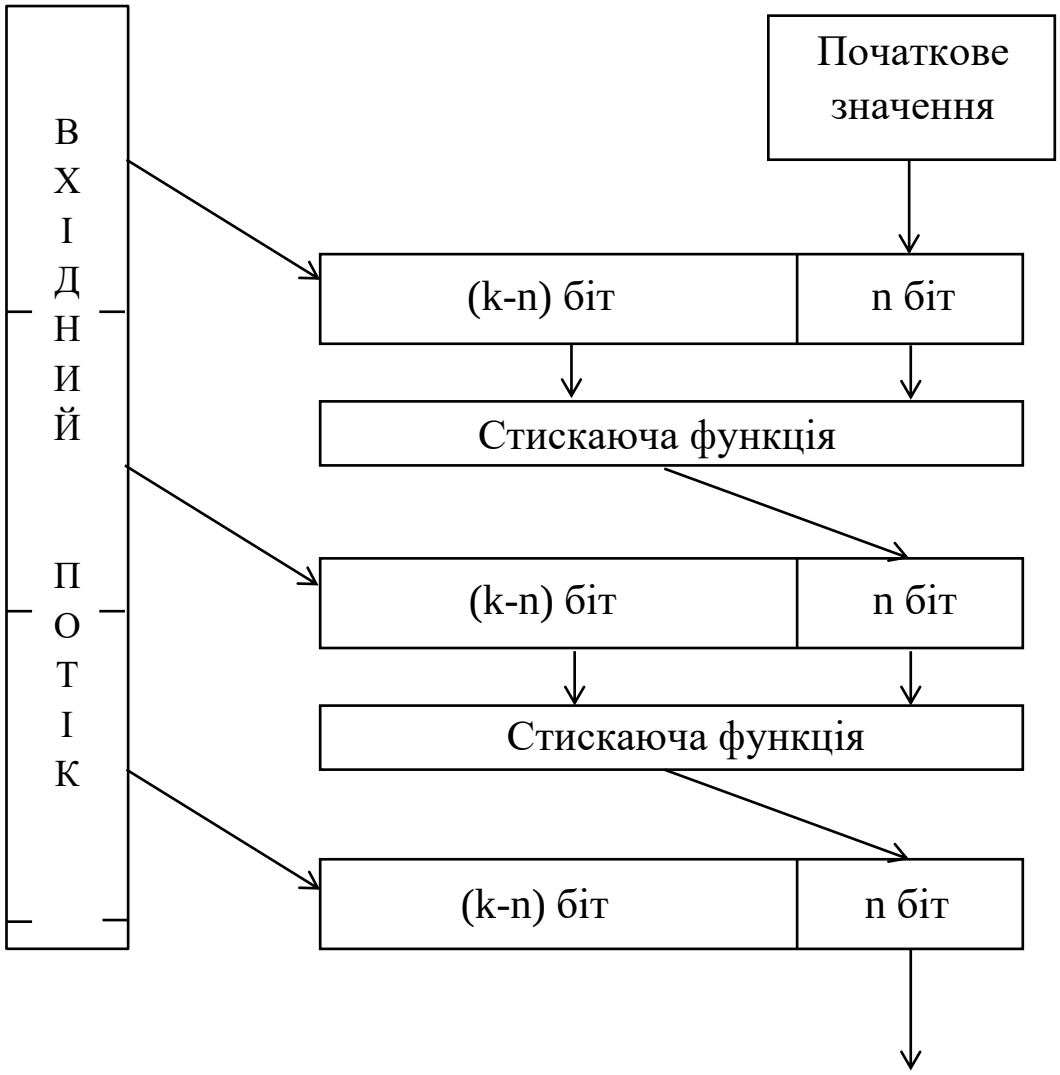

Рисунок 2.2 – Принцип роботи ітеративної функції хешування

При побудові хеш-функцій за ітераційною схемою постають два взаємопов'язані питання: що робити з даними, розмір яких не кратний розміру блоку (k-n), і як врахувати довжину документа в хеш-значенні, якщо це необхідно. Існує два варіанти вирішення цих проблем.

У першій ситуації на початку документа перед хешуванням додається поле фіксованої довжини (32 біти). Це поле містить довжину тексту в двійковому форматі. Після цього комбінований блок даних заповнюється нулями з точністю до найближчого кратного розміру блоку (k-n) біт.

В другій ситуації до документа додається один біт "1" праворуч, а потім біти "0" з точністю до найближчого кратного розміру блоку (k-n) бітів. У цьому випадку поле довжини стає непотрібним, оскільки після вирівнювання не буде двох однакових документів.

На додаток до поширених алгоритмів однопрохідного шифрування, існують

алгоритми багатопрохідного шифрування. У цих випадках блок вхідних даних повторюється декілька разів під час етапу розширення, а потім додається до найближчого кратного розміру блоку.

2.5.1 Використання функцій односпрямованого хешування

Хеш-функція може бути побудована за допомогою симетричного блокового алгоритму в режимі зворотного зв'язку з шифром (CFB) або зворотного зв'язку з виходом (OFB) з фіксованим ключем і вектором ініціалізації (IV). У цьому підході повідомлення M буде зашифровано блоковим алгоритмом, а останній блок зашифрованого тексту вважається хеш-значенням M. Цей метод дозволяє створювати код автентифікації повідомлення (MAC), але не завжди гарантує безпеку хеш-функції.

Захищеніший аналог функції хешування можливо здобути, коли на вхід подається блок повідомлення і попереднє хеш-значення, а на виході - актуальне хеш-значення. Справжні хеш-функції набагато складніші. Розмір значення хеш-функції перевищує розмір блоку, який в свою чергу задається довжиною ключа. Деякі схеми хешування розраховані на довжину хеш-значення, що вдвічі перевищує розмір блоку.

Коректність результуючої функції хешування залежить від безпеки алгоритму, використаного в базовому блоці. На рисунку 2.2 показано схему хешування з довжиною хеш-значення, що дорівнює довжині блоку.

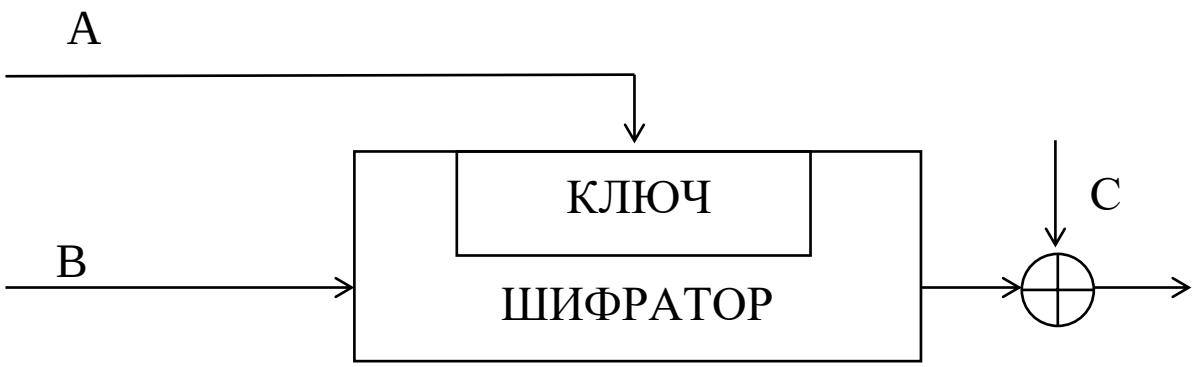

Рисунок 2.3 – Ілюстрація алгоритму утворення функції хешування

Одностороння хеш-функція - це функція, яка приймає довільні вхідні дані будь-якої довжини і обчислює відповідне хеш-значення фіксованої довжини. Одностороння означає, що обчислення хеш-значення є простим завданням, але

знайти вхідні дані за хеш-значенням є майже неможливим завданням.

Принцип роботи односторонньої хеш-функції можна описати за допомогою наступних формул:

- Хешування даних: h = H(m), де m - вхідні дані, H - одностороння хеш-функція, а h - результуюче хеш-значення.
- Одностороння властивість: Нехай f - це функція, яка виконує обернене обчислення хеш-функції. Одностороння властивість вимагає, щоб для будь-якого хеш-значення h було обчислювально важко підбирати вхідні дані m таким чином, щоб f(h) = m.
- Унікальні хеш-значення: Одностороння хеш-функція має забезпечувати унікальні хеш-значення для різних вхідних даних. Для будь-яких двох різних вхідних даних m1 і m2, H(m1) ≠ H(m2).

Для того, щоб забезпечити безпеку і задовольнити дві основні вимоги до хеш-функції, повідомлення M поділяється на блоки Mi фіксованої довжини, що виконуються послідовно. Це дозволяє проводити обчислення з меншою кількістю операцій над вхідними даними. Для гарантування безпеки та цілісності існують різні схеми безпечного хешування, де довжина хеш-значення збігається з довжиною блоку. У таблиці 2.1 наведено ще 12 схем безпечного хешування.

Таблиця 2.1 – Схеми безпечного хешування

Номер схеми	Функція хешування
1	$H_i = E_{Hi-1}(M_i) \oplus M_i$
2	$H_i = E_{Hi-1}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$
3	$H_i = E_{Hi-1}(M_i) \oplus M_i \oplus H_{i-1}$
4	$H_i = E_{Hi-1}(M_i \oplus H_{i-1}) \oplus M_i$
5	$H_i = E_{Mi}(H_{i-1}) \oplus H_{i-1}$
6	$H_i = E_{Mi}(M_i \oplus H_{i-1}) \oplus M_i \oplus H_{i-1}$
7	$H_i = E_{Mi}(H_{i-1}) \oplus M_i \oplus H_{i-1}$
8	$H_i = E_{Mi}(M_i \oplus H_{i-1}) \oplus H_{i-1}$
9	$H_i = E_{Mi} \oplus H_{i-1}(M_i) \oplus M_i$
10	$H_i = E_{Mi} \oplus H_{i-1}(H_{i-1}) \oplus H_{i-1}$
11	$H_i = E_{Mi} \oplus H_{i-1}(M_i) \oplus H_{i-1}$
12	$H_i = E_{Mi} \oplus H_{i-1}(H_{i-1}) \oplus M_i$

Алгоритм повинен бути розроблений спеціально з урахуванням наступних умов: безпека та швидкість. Перші чотири безпечні схеми хешування, стійкі до всіх атак, представлені на рисунку 2.4. [7]

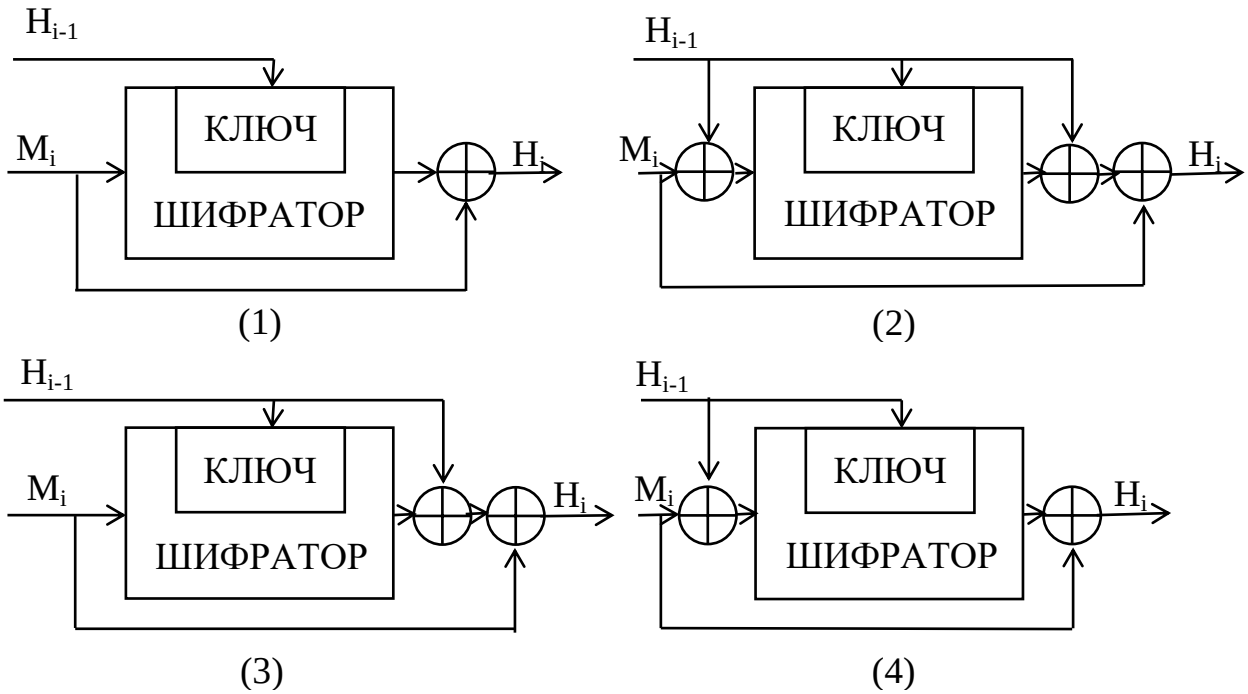

Рисунок 2.4 – Принципи роботи алгоритму безпечного хешування

2.5.2 Аналіз алгоритму SHA

Алгоритм SHA (Secure Hash Algorithm) - один з найбільш поширених і стандартизованих алгоритмів хешування. Він використовується для створення стійких хеш-функцій, які захищають дані від небажаних змін і забезпечують ідентифікацію повідомлень.

Сімейство алгоритмів SHA має кілька варіантів, таких як SHA-1, SHA-256, SHA-384 і SHA-512. Кожен з цих варіантів використовує різну довжину хеш-значення та операційні блоки, що забезпечує різний рівень безпеки та продуктивності.

Алгоритм SHA працює ітеративним способом, де вхідне повідомлення розбивається на блоки, які обробляються послідовно. Кожен блок додається до попереднього хеш-значення, а потім проходить через серію логічних і арифметичних операцій для забезпечення безпечного перетворення даних. Результатом є хеш-значення фіксованої довжини.

Алгоритм хешування SHA є безпечним, бо він спроектований так, щоб було обчислювально неможливо відновити повідомлення, відповідне до даного дайджесту, а також знайти два різні повідомлення, які дадуть однаковий дайджест. Будь-яка зміна повідомлення при передачі з дуже великою ймовірністю викличе зміну дайджесту, і прийнятий цифровий підпис не пройде перевірку [5].

Алгоритм хешування SHA оперує з вхідним повідомленням M, яке доповнюється до числа, кратного 512 бітам. Додаткове доповнення повідомлення виконується наступним чином: спочатку додається біт "1", а потім біти "0", необхідні для отримання довжини повідомлення, яка на 64 біти коротша і ділиться на 512. Потім додається 64-бітне представлення довжини вихідного повідомлення. П'ять 32-бітних змінних ініціалізуються як частина алгоритму, виконуючи роль проміжного сховища під час обчислень, наприклад:

$$A = 0x78981504$$

$$B = 0x56F45E23$$

$$C = 0x9D7A13F4$$

$$D = 0x3DB72343$$

$$E = 0x7E2C1D90$$

Основний цикл алгоритму обробляє 512-розрядні блоки повідомлень один за одним. Цикл складається з чотирьох підциклів, кожен з яких містить 20 операцій. Перед входом у цикл початкові значення п'яти змінних *A, B, C, D, E* копіюються в інші змінні *a, b, c, d, e* [5]:

$$a = A \qquad b = B \qquad c = C \qquad d = D \qquad e = E$$

В результаті кожної операції застосовується нелінійна функція від трьох з п'яти змінних *a, b, c, d, e* з подальшим зміщенням та складанням.

На рисунку 2.5 зображено алгоритм SHA, який використовує набір таких нелінійних функцій:

$$f_t(X,Y,Z) = (X^Y) \vee ((X)^Z) \qquad \text{для } t = 0...19,$$
$$f_t(X,Y,Z) = X \oplus Y \oplus Z \qquad \text{для } t = 20...39,$$
$$f_t(X,Y,Z) = (X^Y) \vee (X^Z) \vee (Y^Z) \qquad \text{для } t = 40...59,$$
$$f_t(X,Y,Z) = X \oplus Y \oplus Z \qquad \text{для } t = 60...79,$$

Рисунок 2.5 – Нелінійні функції алгоритму SHA

де t є номером операції.

Далі зображено чотири константи, за якими працює алгоритм (рисунок 2.6):

$$K_t = 0x5A827999 \qquad \text{для } t = 0...19,$$
$$K_t = 0x6ED9EBA1 \qquad \text{для } t = 20...39,$$
$$K_t = 0x8F1BBCDC \qquad \text{для } t = 40...59,$$
$$K_t = 0xCA62C1D6 \qquad \text{для } t = 60...79.$$

Рисунок 2.6 – Константи алгоритму SHA

Після розбиття блоку повідомлення на шістнадцять слів довжиною 32 біт, а саме – $M_0, M_1, M_2, M_3, M_4, M_5, \ldots, M_{15}$, вони перетворюються на вісімдесят слів довжиною 32-біт – $W_0, W_1, W_2, W_3, W_4, W_5, \ldots, W_{79}$ за допомогою наступного алгоритму, зображеного на рисунку 2.7:

$$W_t = M_t \text{ для } t = 0...15,$$
$$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) <<< 1 \text{ для } t = 16...79,$$

Рисунок 2.7 – Перетворення блока повідомлення з 32 до 80 біт

де $\oplus$ позначає операцію побітового XOR,

<<< позначає циклічний зсув вліво на 1 біт,

а t є номером операції.

Це означає, що перші 16 слів розширеного повідомлення (W) відповідають оригінальним словам повідомлення, а наступні 64 слова обчислюються з використанням попередньо обчислених слів. На рисунку 2.8 показана схема виконання однієї такої операції:

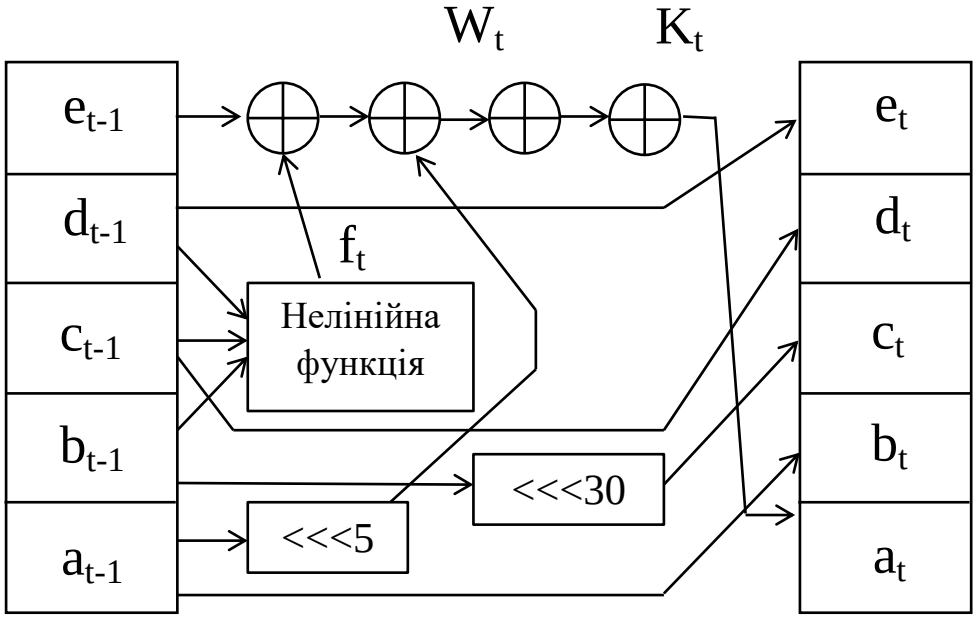

Рисунок 2.8 – Схема алгоритму SHA для виконання однієї операції

Ця операція включає нелінійну функцію від п'яти змінних *a*, *b*, *c*, *d*, *e*, після чого відбувається зсув і додавання. Щоб зрозуміти, як працює ця операція, розглянемо кожен крок окремо:

- Нелінійна функція: Змінні *a*, *b*, *c*, *d*, *e* використовуються як вхідні дані для нелінійної функції, яка виконує операції, що забезпечують змішування та перетворення цих змінних. Ця функція додає додатковий рівень складності і стійкості до алгоритму хешування.
- Зсув: Після виконання нелінійної функції значення змінних *a*, *b*, *c*, *d*, *e* циклічно зсуваються вліво на певну кількість бітів. Це гарантує, що значення бітів перемішуються і переміщуються між змінними.
- Додавання: Після зсуву до змінних *a*, *b*, *c*, *d*, *e* додаються інші значення, отримані в результаті нелінійної функції і зсуву. Це додавання впливає на остаточні значення змінних і допомагає забезпечити дифузію, тобто поширення змін, внесених до всього хеш-значення.

Таким чином, кожна операція включає в себе нелінійну функцію, зсув і додавання, що дозволяє обробляти блоки повідомлень і генерувати хеш-значення крок за кроком.

2.5.3 Виконання управлінням відкритих ключів

Однією з важливих проблем криптографії з відкритим ключем, зокрема систем цифрового підпису, є управління відкритими ключами. Оскільки відкритий ключ надається любому учаснику, важливо мати інструмент для верифікації факту належності ключа його законному власникові. Для вирішення цієї проблеми використовуються сертифікати. Сертифікати дозволяють перевірити дані власника та його відкритий ключ за допомогою цифрового підпису суб'єкта, якому довіряють. Системи сертифікатів поділяються на два типи: централізовані та децентралізовані.

У децентралізованих системах мережа довіри будується шляхом перехресного підписання сертифікатів між відомими та довіреними особами. Це створює мережу довіри для сертифікатів. У централізованих системах використовуються центри сертифікації, що підтримуються організаціями, яким довіряють.

Центр сертифікації генерує власний закритий ключ і сертифікат, видає сертифікати користувачам і перевіряє їхню достовірність за допомогою ЕЦП. Він також зберігає бази даних виданих і відкликаних сертифікатів та здійснює відкликання порушених сертифікатів. Користувачі можуть звернутися до центру сертифікації, щоб замовити власний сертифікат відкритого ключа та отримати інформацію про відкликані ключі.

Зберігання особистого ключа є важливим аспектом криптосистеми цифрового підпису. Викрадення особистого ключа може дозволити зловмиснику створювати дійсні цифрові підписи від імені власника ключа. Особливу увагу слід звернути на умови зберігання особистого ключа. Користувачі можуть зберігати особистий ключ на власному комп'ютері, надійно захищеному паролем. Проте такий метод має вразливі місця, оскільки безпека ключа повністю покладається на рівень безпеки комп'ютера, а власник ключа може використовувати ЕЦП для підписання документів лише на даному конкретному комп'ютері [4].

Сьогодні нам доступні пристрої для зберігання приватних ключів, такі як смарт-карти, USB-пристрої та токени Touch-Memory.

Одним з найбезпечніших методів є зберігання особистого ключа на смарт-картці. Для використання смарт-картки користувач повинен мати її при собі та ввести PIN-код, що забезпечує двофакторну автентифікацію. При підписанні документа його хеш або ключ надсилається на картку, а процесор картки виконує операцію підписання і повертає підпис. Ключ не копіюється під час процесу підписання, що гарантує, що завжди існуватиме лише одна версія ключа.

Дані методи забезпечують захист від підміни ключа, несанкціонованого використання та дозволяють відкликати скомпрометовані ключі.

3 РОЗРОБКА ПРОЕКТУ СИСТЕМИ ДЛЯ ЦЕНТРУ СЕРТИФІКАЦІЇ

3.1 Рекомендації щодо вибору інструментів для створення ІС

Для програмної реалізації ІС для сертифікаційного центру рекомендується використовувати мову програмування PHP, бібліотеку Openssl, що входить до складу PHP у вигляді модуля, і база даних Mysql. При визначенні мови програмування однією з головних причин вибору PHP є її широке використання при розробці багатьох веб-додатків. Вона широко підтримується хостинг-провайдерами і є однією з провідних мов для створення сучасних веб-сторінок. PHP вирізняється своєю легкістю, високою швидкістю роботи та багатими функціональними можливостями завдяки великому набору вбудованих інструментів та додаткових модулів, доступних для веб-розробки. Крім того, PHP - це мова з відкритим вихідним кодом, що дозволяє кастомізувати та адаптувати її до конкретних потреб. Вона також є кросплатформенною, що дозволяє запускати PHP-програми на різних операційних системах з мінімальними модифікаціями. [11].

Мова програмування PHP включає криптографічну бібліотеку OpenSSL як модуль. Ця бібліотека має відкритий вихідний код і розроблена для сумісності з різними платформами. Вона надає функціональність для генерації ключів RSA, DH і DSA, а також для створення і управління сертифікатами X.509. OpenSSL також дозволяє підписувати ключі, генерувати запити на підписання сертифікатів (CSR) та отримувати відповідні сертифікати (CRT). Підтримувані асиметричні алгоритми: RSA, DSA, Diffie-Hellman key exchange. Підтримувані хеш-функції: SHA, MDC-2. Серверна СУБД потрібна для надійного зберігання всіх сертифікатів, даних про них і CRL, а також швидкого доступу і пошуку даних. СУБД Mysql має відмінну швидкодію для читання і пошуку даних, тому відмінно підходить для створення ЦС [12].

3.2 Аналіз функцій бібліотеки Openssl

Для здійснення розробки ІС для сертифікаційного центру була використана бібліотека Openssl – resource openssl_pkey_new ([array51 $configargs]).

Функція генерує ключову пару: відкритий і секретний ключі.

Повертає ресурс ключової пари, або FALSE у випадку невдачі.

Для настроювання генерації ключової пари використовується масив configargs з настроюваннями (табл.3.1), або встановлений у системі файл openssl.cnf у випадку відсутності цього параметра.

Таблиця 3.1 – Масив configargs

Ключ	Тип	Еквівалент openssl.conf	Опис
config	string		Файл конфігурації Openssl (якщо вилучити, буде використовуватися openssl.conf, установлений у системі)
digest_alg	string	default_md	Алгоритм відбитка, наприклад sha1
x509_extension	string	x509_extension	Секція розширень для сертифіката X509
req_extension	string	req_extension	Секція розширень для CSR
private_key_bits	integer		Кількість біт закритого ключа (рекомендується не менш 1280, оптимально - 2048)
private_key_type	integer		Тип закритого ключа: OPENSSL_KEYTYPE_DSA, OPENSSL_KEYTYPE_DH або OPENSSL_KEYTYPE_RSA
encrypt_key	boolean	encrypt_key	Шифрувальний закритий ключ

Функція генерує запит на підпис сертифіката CSR (Certificate Signing Request).

privkey – ресурс ключової пари, попередньо сгенерованої функцією openssl_pkey_new.

```
resource openssl_csr_sign ( mixed $csr, mixed $cacert, mixed $priv_key, int $days [, array $configargs [, int $serial = 0 ]] )
```

Функція підпису сертифікат.

Повертає ресурс сертифіката X509, або FALSE у випадку помилки. Масив dn містить необхідну для створення сертифіката інформацію про унікальне ім’я (Distinguished Name) (табл.3.2).

Таблиця 3.2 – Масив dn

Ключ	Опис
countryname	Назва країни, 2 великі букви
stateorprovincename	Назва області або штату
localityname	Місцезнаходження (наприклад, місто)
organizationname	Назва організації
organizationalunitname	Назва підрозділу організації
commonname	Загальне ім’я (наприклад, ПІБ або host-ім’я сервера)
emailaddress	Адреса e-mail

csr – ресурс попередньо сгенерованого CSR, або шлях до файлу CSR в PEM кодуванні.

cacert – ресурс сертифіката, яким буде підписаний цільовий, або NULL для створення самопідписаного сертифіката.

priv_key – ресурс секретного ключа, за допомогою якого буде підписаний цільовий сертифікат.

days – строк придатності генеруємого сертифіката в днях. serial – серійний номер генеруємого сертифіката.

Одержання ресурсу сертифіката з PEM-закодованих даних сертифіката (x509certdata).

resource openssl_pkey_get_private (mixed $key [, string $passphrase = ""]) Одержання ресурсу секретного ключа. key – PEM-закодовані дані

секретного ключа, або шлях до файлу, що містить ці дані.

passphrase – парольна фраза, необхідно вказати, якщо секретний ключ зашифрований.

array openssl_pkey_get_details (resource $key)

Функція повертає масив даних про ключ: число біт, строкове представлення відкритого ключа і алгоритм.

```
bool openssl_x509_export ( mixed $x509, string &$output [,bool $notext ])
```

Функція для експорту сертифіката X509 у форматі PEM.

```
bool openssl_pkey_export ( mixed $key, string &$out [,
```

```
string $passphrase [, array $configargs ]] )
```

Функція для експорту секретного ключа у форматі PEM.

passphrase – опціональний параметр; пароль для шифрування секретного ключа.

```
bool openssl_pkcs12_export ( mixed $x509, string &$out, mixed $priv_key, string $pass [, array $args ] )
```

Функція для ескпорта сертифіката і секретного ключа у форматі PKCS#12.

Дані в цьому форматі потрібні користувачам для установки їх сертифікатів разом із секретним ключем у систему.

pass – пароль для шифрування файлу PKCS#12; якщо шифрування не потрібно, указати порожній рядок.

```
void openssl_x509_free ( resource $x509cert )
```

Функція для звільнення з пам'яті ресурсу сертифіката.

```
void openssl_pkey_free ( resource $key )
```

Функція для звільнення з пам'яті ресурсу ключової пари.

```
bool openssl_sign ( string $data, string &$signature, mixed $priv_key_id [, int $signature_alg = OPENSSL_ALGO_SHA1 ] )
```

Функція для підпису даних. data – дані для підпису.

3.3 Огляд бібліотеки Openssl

Настроювання функцій бібліотеки Openssl задаються в спеціальному файлі openssl.cnf. Шлях до нього можна змінити, указавши в ключі config5 масиву configargs, параметра відповідних функцій. Він має формат INI і містить безліч тонких настроювань для використання даної бібліотеки, розділених на секції [13].

signature – якщо функція завершилася успішно, підпис вертається в цю змінну.

priv_key_id – ресурс закритого ключа для підпису. signature_alg – хеш-функція:

OPENSSL_ALGO_SHA1 (використовується за замовчуванням);

Далі наведемо найбільш важливі настроювання.

Значення параметра використання ключів keyusage (у настроюваннях

використовуються комбінації, перераховані нижче):

- digitalsignature – цифровий підпис;
- nonrepudiation – незречення від авторства;
- keyencipherment – шифрування криптографічних ключів;
- dataencipherment – шифрування користувацьких даних;
- keyagreement – узгодження ключів (при використанні шифру Діффі-Хеллмана);
- keycertsign – перевірка підпису сертифікатів (тільки для сертифікатів ЦС);
- crlsign – перевірка підпису CRL;
- encipheronly – шифрування даних (має значення тільки при встановленому keyagreement);
- decipheronly – розшифрування даних (має значення тільки при встановленому keyagreement).

Для створення сертифіката ЦС у параметрі basicconstraints потрібно вказати CA:true.

Для декодування й кодування даних у формат ASN1 минулого мають бути написані класи ASN1 (абстрактний) і похідні від нього: ASN1_SEQUENCE, ASN1_SET, ASN1_NULL, ASN1_UTF8STRING, ASN1_TELETEXSTRING, ASN1_ASCIISTRING, ASN1_BITSTRING, ASN1_OCTETSTRING, ASN1_INT, ASN1_GENERALTIME, ASN1_ENUM, ASN1_UTCTIME, ASN1_BOOL, ASN1_OID (для відповідних типів даних).

Для створення CRL потрібні класи X509_CERT і X509_CRL (рис.4.1).

```
ASN1_SEQUENCE X509_CERT::DECODE ( $crt_data_der )
```

Функція декодує дані сертифіката формату DER (crt_data_der), повертає об'єкт класу ASN1_SEQUENCE.

string X509 _CRL:: CREATE ($ci, $ca_pkey, $ca_decoded)

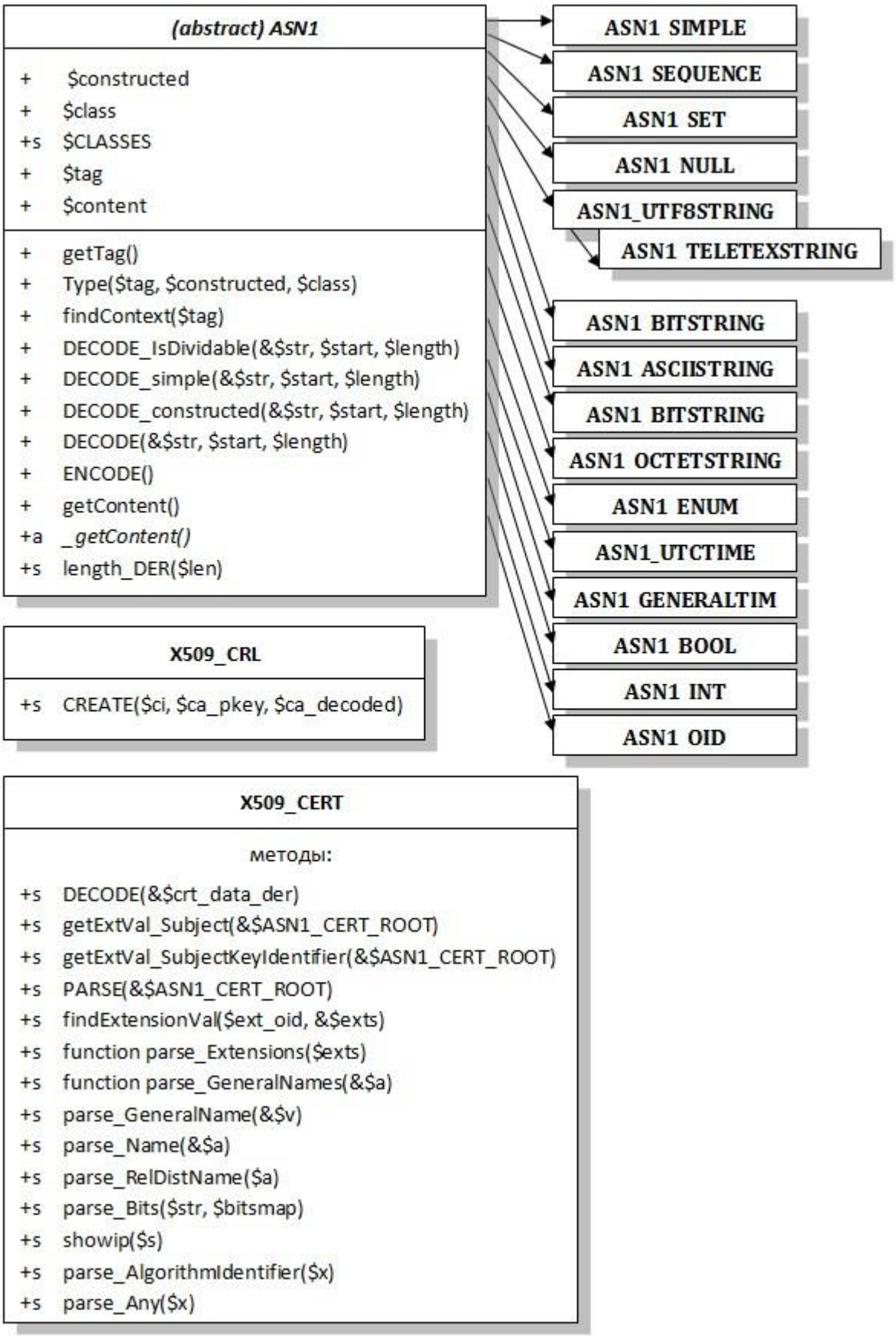

Рисунок 3.1 – Діаграма класів для створення CRL

Функція генерує і підписує CRL, повертає його в Der-Форматі.

Функція створення CRL (Certificate Revokation List). c_i – масив необхідних даних для створення CRL. Ключі:

- (int) no – опціонально; номер CRL;
- (int) version – версія CRL; якщо не зазначена, то v2;
- (int) days – число днів до наступного відновлення CRL;
- (int) alg – хеш-функція (див. константи вище); якщо не зазначена, то OPENSSL_ALGO_SHA1.
- (array) revoked – список відкликаних сертифікатів, кожний елемент списку – масив з наступними ключами:
- (int) serial – серійний номер сертифіката;
- (string) rev_date – дата відкликання (типу Utctime);
- (int) reason – причина відкликання:
- unspecified,
- keycompromise,
- Cacompromise,
- affiliationchanged,
- superseded,
- cessationofoperation,
- certificatehold,
- removefromcrl,
- privilegewithdrawn.

(string) compr_date – (якщо причина відкликання = keycompromise) дата компрометації ключа (типу Generaltime);

(int) hold_instr – (якщо причина відкликання = certificatehold) інструкції затримки: 0 – None, 1 – Callissuer, 2 – Reject.

ca_pkey – ресурс закритого ключа, пов'язаного із сертифікатом ЦС. ca_decoded – декодовані дані сертифіката ЦС, об'єкт класу ASN1_SEQUENCE, повернутий функцією X509::DECODE.

3.4 Створення проекту бази даних для системи центру

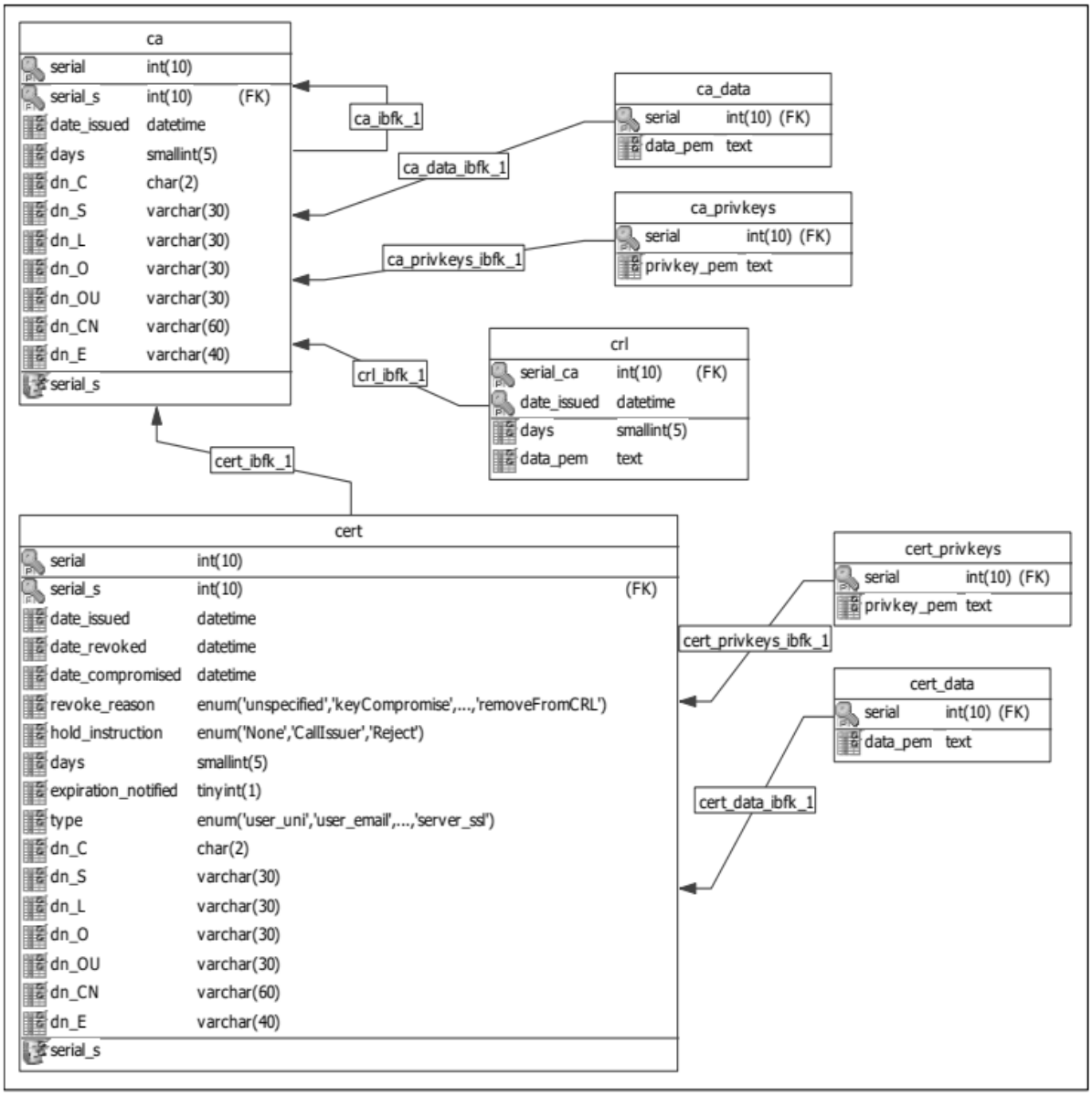

Рисунок 3.2 – Схема бази даних ІС сертифікаційного центру

Проектування бази даних для інформаційної системи (ІС) сертифікаційного центру є одним із важливих етапів розробки системи. Коректне й ефективне проєктування бази даних забезпечує надійне зберігання та управління інформацією про сертифікати криптографічних ключів, їхні статуси, дані користувачів та інші сутності, пов'язані з центром сертифікації.

ВИСНОВКИ

В результаті виконання роботи було проведено проектування інформаційної системи для сертифікаційного центру. Під час дослідження було проведено аналіз інфраструктури відкритих ключів, який дав змогу визначити основні проблеми та рекомендації для створення ІС. Розроблення цієї інформаційної системи дозволяє генерувати, формувати, зберігати та керувати сертифікатами криптографічних ключів ЕЦП для учасників будь-якої організації. Це розв'язує проблему електронного документообігу для забезпечення внутрішніх вимог організації.

Робота ІС заснована на принципі забезпечення безпеки та довіри в електронних комунікаціях. Користувачі зможуть легко створювати нові сертифікати, вказуючи необхідну інформацію про власника та необхідні параметри. Після перевірки та підтвердження запиту, ІС буде генерувати безпечні криптографічні ключі та створювати відповідні сертифікати, які можуть бути використані для електронного цифрового підпису.

Управління сертифікатами в ІС є гнучким і контрольованим процесом, де наявна можливість припинити та відновити дію сертифікатів з різних причин, таких як компрометація ключа або зміна статусу сертифіката. Це буде забезпечувати ефективне управління та контроль над сертифікатами в системі.

Адміністратор буде мати змогу вносити зміни до ЦС, обирати потрібний алгоритм шифрування, хеш-алгоритм тощо.

Ведення реєстру сертифікатів ключів підписів є важливою складовою ІС. Реєстр містить інформацію про сертифікати, включно з даними про власника, термін дії сертифіката. Користувачі можуть вільно отримувати доступ до реєстру та виконувати пошук сертифікатів за різними параметрами, що забезпечує зручність використання та доступність інформації.

Підтримка списку відкликаних сертифікатів (CRL) є важливою функцією ІС. Система забезпечує актуалізацію списку, включно з інформацією про сертифікати, які були скомпрометовані або більше не є дійсними. У разі

необхідності, ІС також випускає відновлення сертифікатів.

В Україні існує високий попит на захист інформації та забезпечення безпечних електронних комунікацій у різних секторах, тож ІС може бути вирішенням цих проблем, забезпечить конфіденційності даних, а також встановить довіру та автентичность в електронних взаємодіях, наприклад, електронний документообіг.

Таким чином, застосування ІС для центру сертифікації в Україні зможе забезпечити безпеку та автентифікацію даних, спростити процеси взаємодії та підвищити довіру користувачів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Портал знань. Система електронного документообігу. Електронний цифровий підпис. URL: http://www.znannya.org/?view=e-government-documents (дата звернення: 30.03.2023).
2. Радник. Закон України «Про електронні довірчі послуги» URL: https://radnuk.com.ua/praktyka_zakupivel/nadporogi/zakon-ukrainy-pro-elektronni-dovirchi-posluhy-novi-pravyla-zastosuvannia-elektronnykh-pidpysiv-ta-pechatok/ (дата звернення: 30.03.2023).
3. Електронний реєстр чинних, блокованих та скасованих сертифікатів відкритих ключів URL: https://czo.gov.ua/ca-registry (дата звернення: 02.04.2023).
4. Остапов С.Е., Валь Л.О. Основи криптографії. Навчальний посібник. Ч: Книги-XXI, 2008. – 188 с.
5. Франчук В.М. Захист інформаційних ресурсів: криптографічні та стеганографічні методи захисту даних – 120 с.
6. В. Ємець, А. Мельник, Р. Попович. Сучасна криптографія. Основні поняття. К.: БаК, 2003. – 144 с.
7. Остапов С.Е., Євсеєв С.П., КорольО.Г. Технології захистуінформації. Навчальний посібник. Х.: ХНЕУ, 2013. – 249 с.
8. Гончарова Л.Л., Возненко А Д., Стасюк О. І., Коваль Ю. О. Основи захисту інформації в телекомунікаційних та комп'ютерних мережах. К.: 2013. – 435 с.
9. Акредитований центр сертифікації ключів – ПриватБанк. URL: https://acsk.privatbank.ua/main (дата звернення: 05.04.2023).
10. О.Г. Додонов, Д.В. Ланде, В.Г. Путятін. Інформаційні потоки в глобальних комп’ютерних мережах. К.: Наук. думка, 2009. – 295 с.
11. Люк Веллинг, Лора Томсон. Разработка веб-приложений с помощью PHP и MySQL. М.: Вильямс, 2010. – 848 с.
12. Ярцев В.П. Організація баз даних та знань: навчальний посібник.

К., ДУТ, 2018, – 214 с.

13. Гутманс Э., Баккен С., Ретанс Д. PHP 5. Профессиональное программирование. СПб.: Символ-Плюс, 2006. – 704 с.
14. Шнайер Б. Прикладная криптография: Протоколы, алгоритмы и исходные тексты на языке С / Б. Шнайер.− М. : Триумф, 2002. – 816 с.
15. Яковлев А. В. Криптографическая защита информации : учебное пособие/ А. В. Яковлев, А. А. Безбогов, В. В. Родин, В. Н. Шамкин. – Тамбов : Изд-во Тамб. гос. техн. ун-та, 2006. – 140 с.
16. Гапак О. М. Визначення довжини періоду генераторів псевдовипадкових послідовностей на основі регістрів зсуву зі зворотнім зв’язком та перенесення / О.М. Гапак // Моделювання та інформаційні технології. – 2014. – Випуск 73. – С. 92 – 97.
17. NIST SP 800-22rev1a. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications//National Institute of Standards and Technology Special Publication 800-22rev1a, 2010, - 131 p.
18. Національні стандарти України, «ДСТУ 4145-2002. Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка», Київ, 2002.
19. D. Bernstein, T. Lange and R. Rezaeian Farashahi, «Binary Edwards Curves», Cryptographic Hardware and Embedded Systems – CHES 2008, pp. 244-265.
20. M. Rivain, «Fast and regular algorithms for scalar multiplication over elliptic curves», IACR Cryptology ePrint Archive, p. 388, 2011.
21. M. Li, A. Miri and D. Zhu, «Fast Algorithm for Converting Ordinary Elliptic Curves into Binary Edward Form», International Journal of Digital Content Technology and its Applications, vol. 6, no. 1, pp. 405-412, 2012.
22. М. Ковтун, «Применение кривых Эдвардса для защищенной реализации механизмов электронной цифровой подписи согласно ДСТУ 4145- 2002», Системи обробки інформації, том. 5, №. 151, с. 130-137, 2017.
23. M. Kovtun, Z.B. Hu, S. Gnatyuk, N. Seilova. «Method of Searching Birationally Equivalent Edwards Curves over Binary Fields», in Proc. of the 1 st International Conference on Computer Science, Engineering and Education Applications

(ICCSEEA2018), Jan 18-20, 2018, Kiev, Ukraine.

24. M.G. Kovtun, V.Y. Kovtun, A.A. Okrimenko and S.A. Gnatyuk. «Search method development of birationally equivalent binary Edwards curves for binary Weierstrass curves from DSTU 4145-2002», in Proc. PIC S&T, Kharkov, Ukraine, Oct. 13-15, 2015. pp. 5-8. DOI: 10.1109/INFOCOMMST.2015.7357253.
25. B. Koziel, R. Azarderakhsh and M. Mozaffari-Kermani, «Low-Resource and Fast Binary Edwards Curves Cryptography», Progress in Cryptology -- INDOCRYPT 2015, pp. 347-369, 2015. [9] «Hyperelliptic org», Hyperelliptic.org, 2018. [Online]. Available: http://www.hyperelliptic.org (дата звернення: 15.04.2023).
26. K. Kim, C. Lee and C. Negre, «Binary Edwards Curves Revisited», Progress in Cryptology -- INDOCRYPT 2014, pp. 393-408, 2014. 119
27. P. Montgomery, «Speeding the Pollard and elliptic curve methods of factorization», Mathematics of Computation, vol. 48, no. 177, pp. 243-243, 1987.
28. D. Bernstein, «Batch Binary Edwards», Advances in Cryptology - CRYPTO 2009, pp. 317-336, 2009.
29. N.Sklavos, A.Fournaris, «Binary Edwards Curve Design Strategy for Efficient and Power Attack Resistant Architectures», Proceedings of the 4th Workshop on Secure Hardware & Security Evaluation, Cryptographic Hardware and Embedded Systems (CHES'15), 2015.
30. Азаров, С., Нємцев, М., & Малахов, С. Огляд аналогій та обгрунтування принципів створення демон юнітів відстеження мережевої активності користувачів. Proceedings of the XX International Scientific and Practical Conference. Graz, Austria. 2023. Pp. 447-453. URL: https://isg-konf.com/technologies-innovative-and-modern-theories-of-scientists/ Available at: DOI: 10.46299/ISG.2023.1.20 (дата звернення: 25.05.2023).
31. Михайленко Д., Нємцев М. Особливості технології мережевих пасток як інструменту активного захисту та аналізу дій атакуючої сторони. Proceedings of the XXI International Scientific and Practical Conference. Melbourne, Australia. 2023. Pp. 483-487 URL: https://isg-konf.com/scientists-and-methods-of-using-modern-technologies/ Available at: DOI: 10.46299/ISG.2023.1.21 (дата звернення: 31.05.2023).

ДОДАТОК А

КОПІЇ НАУКОВИХ ПУБЛІКАЦІЙ

ОГЛЯД АНАЛОГІЙ ТА ОБГРУНТУВАННЯ ПРИНЦИПІВ СТВОРЕННЯ ДЕМОН ЮНІТІВ ВІДСТЕЖЕННЯ МЕРЕЖЕВОЇ АКТИВНОСТІ КОРИСТУВАЧІВ

Азаров Сергій Ігорович
студент факультету комп'ютерних наук, (магістратура)
Харківський національний університет імені В.Н. Каразіна

Нємцев Максим Олександрович
студент факультету комп'ютерних наук, (бакалавріат)
Харківський національний університет імені В.Н. Каразіна

Малахов Сергій Віталійович
канд. техн. наук, ст. науковий співробітник, доцент кафедри
Харківський національний університет імені В.Н. Каразіна, Україна

Вступ. В даний час найважливішим аспектом проблематики протидії кіберзагрозам є забезпечення безперервності збору, обробки та узагальнення відповідної інформації. Відповідно, чим швидше вдається отримати інформацію про існуючі загрози безпеки та/або аномалії мережевої активності, тим вище шанси на ефективне парирування відповідних загроз.

Однак, не усі мережеві користувачі мають час/можливість збирати відповідні дані [1] та відправляти їх у потрібні департаменти (служби або відділи безпеки) задля її завчасного аналізу. Одним із можливих шляхів вирішенням цих труднощів є використання ***демон юнітів***, або ***демон процесів***. Тому має сенс визначитися у понятійному апараті, стосовно трактовок цих термінів в межах питань, що розглядаються [1].

Юніт - це текстовий файл, який описує системний ресурс, яким *systemd* «знає», як керувати, наприклад: - службу, сокет, пристрій, точку монтування та/або таймер.

Блок (*у контексті проблематики синтезу демон юніту*) - це набір конфігураційних параметрів і команд для ***демон юніту*** *(див. нижче)* а саме: – команда для запуску процесу; – параметри залежностей; – порядок запуску тощо. Використання *блоків* дозволяє налаштувати та управляти «демонами» в системі з використанням системи ініціалізації *systemd*.

Демон - це типи блоків, які представляють собою довготривалі фонові процеси (або «демони»), які надають послуги системі або іншим процесам. Вони використовуються для автоматичного запуску під час завантаження і безперервної роботи у фоновому режимі, управляючи системними ресурсами та, за потреби, надаючи послуги іншим процесам.

Демон юніт - це типи блоків, які за допомогою інших процесів збирають, аналізують та *відтворюють процес мережевої активності реального*

користувача з потрібними або характерними для нього властивостями поведінки (або інакше, мережевої присутності).

Модуль - самостійна частина програмного коду, яка має чітко визначену функціональність і може використовуватись для будь-якої кількості програм або проектів.

Етап - позначення окремої фази або частини процесу розробки програмного продукту.

Самописний модуль (у контексті даної статті) - цей термін сигналізує про самостійно розроблений, протестований та інтегрований у проект компонент.

Основна частина.

Коротко розглянемо декілька тезисів, які описують важливість процесу синтезу й використання ***демон юнітів***:

1. Автоматизація фонових завдань: - демони зазвичай використовуються для виконання автоматизованих фонових завдань без втручання користувача (*в нашому випадку адміністратору з безпеки*). Це включає такі завдання, як обслуговування системи, моніторинг мережі, а також процеси резервного копіювання та відновлення.

2. Управління ресурсами: - демони допомагають керувати системними ресурсами, контролюючи та відстежуючи їх використання. Застосовуються для обмеження наявних ресурсів процесора, пам'яті та дискового простору, які використовує процес, запобігаючи споживанню занадто великої кількості ресурсів і впливу на продуктивність системи.

3. Забезпечення стабільності і надійності системи: - сприяють підвищенню експлуатаційних параметрів системи, відстежуючи її продуктивність та вживаючи відповідних дій у разі виникнення проблем. Наприклад, демон може перезапустити аварійну службу, щоб запобігти небажаного простою системи.

4. Покращення параметрів безпеки: - демони можуть допомогти покращити безпеку системи, в автоматичному режимі відстежуючи мережевий трафік, виявляючи й блокуючи підозрілу активність та/або поведінкові колізії. В цьому сенсі вони можуть виконувати такі завдання, як сканування вірусів і виявлення вторгнень, щоб захистити систему від шкідливих програм та інших загроз безпеці.

Розглянемо деякі відомі технології, де використовується ***демон***, як основний компонент програми.

1. Unix-подібні операційні системи. Unix-подібні операційні системи включаючи Linux і macOS, значною мірою покладаються на демони [1]:

Systemd - це менеджер системи та послуг, який використовується в багатьох дистрибутивах *Linux*. Він ініціалізує та керує системними службами, включаючи демони, у спрощений та ефективний спосіб;

cron - планувальник завдань на основі Unix-подібних операційних систем. Це дозволяє користувачам планувати автоматичне виконання завдань або сценаріїв через певні проміжки часу;

OpenSSH - реалізація протоколу Secure Shell (SSH), включає демон під назвою *sshd*. Він забезпечує безпечний віддалений доступ і можливості передачі файлів.

2. Веб-сервери: - веб-сервери часто використовують демони для обробки вхідних запитів і обслуговування веб-вмісту. Наприклад [2]:

HTTP-сервер Apache - це популярний веб-сервер із відкритим кодом. Він використовує процеси-демони для обробки вхідних запитів клієнтів і обслуговування веб-сторінок.

3. Системи баз даних: - багато систем управління базами даних (БД) використовують демони для обробки підключень клієнтів, керування зберіганням даних і виконання фонових завдань [3]:

MySQL і *MariaDB* - є широко використовуваними системами керування реляційними базами даних. Вони використовують процеси-демони для обробки підключень клієнтів і виконання запитів.

PostgreSQL - ще одна популярна система управління реляційними БД з відкритим кодом. Він використовує процес демона під назвою *postmaster* для обробки підключень клієнтів і керування БД.

4. Системи обміну повідомленнями: - системи обміну повідомленнями часто використовують демони для полегшення зв'язку між клієнтами та обробки доставки повідомлень. Наприклад [4]:

RabbitMQ - широко використовуваний брокер повідомлень із відкритим кодом. Він покладається на процеси-демони для отримання, маршрутизації та доставки повідомлень між програмами.

Apache Kafka - це розподілена потокова платформа. Він використовує процеси-демони, які називаються брокерами, для зберігання та розповсюдження повідомлень.

Узагальнюючи все вище зазначене і враховуючи основну мету загального напряму досліджень *(це, компіляція поведінкових профілів мережевих користувачів [1])*, можна стверджувати, що процес створення демон юніту обробки мережевої активності користувачів можна умовно поділити на декілька етапів:

1 етап - створення оболонки демон юніту, який буде обробляти усі необхідні процеси;

2 етап - впровадження збору та збереження трафіку;

3 етап - впровадження обробки/аналізу трафіку;

4 етап - впровадження обмеження користування трафіком на основі отриманих сигнатур мережевої активності користувачів;

5 етап - впровадження механізмів захисту для отриманих поведінкових сигнатур.

До переваг подібної структуризації можна віднести:

- Поступовість. При створенні демон юніту, ці модулі розробляються один за одним. Розробка наступного модулю починається лише після вдалого тестування попереднього.

- Незалежність. Кожен з етапів, може бути легко заміненим/видаленим з ресурсу, що ніяк не вплине на його роботу, ресурс буде функціонувати.
- Варіативність. Кожен з етапів може бути з легкістю заміненим.

Велику увагу при розробці цього демон юніту, слід приділити саме варіативності. Отже наш ресурс повинен виконувати необхідні функції та до того ж мати своє продовження, так як не усі рішення будуть актуальні через певний час.

Зупинимось на основних технологіях, які можуть бути використані або можуть замінити самописні модулі в межах реалізації зазначених вище етапів.

1 етап - «Оболонка демон юніту».

Systemd – популярна система ініціалізації, яка використовується в багатьох дистрибутивах Linux. Забезпечує простий і потужний спосіб керування системними службами, включаючи демони. *Systemd* містить команду *«systemctl»*, яку можна використовувати для створення та керування системними одиницями обслуговування. [5]

Upstart – ще одна система ініціалізації, яка використовується в деяких дистрибутивах Linux, наприклад Ubuntu. Він забезпечує структуру керування послугами, подібну до *systemd*, і може використовуватися для створення та керування демонами.

SysV init – це «стара» система ініціалізації, яка використовується в багатьох ранніх випусках дистрибутивів Linux. Використовує серію сценаріїв оболонки в каталозі */etc/init.d* для керування системними службами. Хоча ініціалізація *SysV* сьогодні менш поширена, його все ще можна використовувати для створення та управління демонами в деяких системах.

Launchd – це система управління демонами, яка використовується в macOS та деяких інших системах на базі Unix. Він забезпечує простий і гнучкий спосіб керування системними службами, включаючи демони. *Launchd* використовує файли списку властивостей *XML* для визначення служб і пов'язаних із ними властивостей.

Supervisor – це система управління процесами, яка використовується для адміністрування процесами в системах на основі Unix. Він забезпечує простий і потужний спосіб керування процесами, що довго виконуються, включно з демонами. *Supervisor* використовує конфігураційні файли в каталозі */etc/supervisor/conf.d* для визначення процесів і пов'язаних з ними властивостей.

2 та 3 етапи - «Накопичення та аналіз трафіку».

tcpdump: tcpdump – це популярний інструмент командного рядка [6], який використовується для захоплення та аналізу мережевого трафіку. Його можна використовувати для перехоплення трафіку на певному мережевому інтерфейсі та запису захоплених пакетів у файл для подальшого аналізу.

Wireshark – аналізатор мережевих протоколів, який може фіксувати та відображати мережевий трафік у реальному часі. Підтримує широкий спектр протоколів і може використовуватися для аналізу трафіку з кількох мережевих інтерфейсів.

Tshark – інструмент командного рядка, який є частиною пакета *Wireshark* [7]. Його можна використовувати для захоплення та аналізу мережевого трафіку, так само, як *tcpdump*, але надає більш розширені параметри фільтрації та аналізу.

Suricata – система виявлення та запобігання вторгненням (IDS\IPS) у мережу з відкритим кодом, яка може фіксувати та аналізувати мережевий трафік у режимі реального часу [8]. Містить потужний механізм правил, який можна використовувати для виявлення та блокування шкідливого трафіку.

Snort – це IDS\IPS із відкритим кодом, яка може фіксувати й аналізувати мережевий трафік у реальному часі [9]. Містить потужний механізм правил, який можна використовувати для виявлення та блокування шкідливого трафіку.

Zeek – є відкритою структурою аналізу вихідної мережі, яка може фіксувати та аналізувати мережевий трафік у режимі реального часу, містить потужну мову сценаріїв, яку можна використовувати для налаштування аналізу та звітності [10].

4 етап - «Поведінкові обмеження» [11].

Брандмауер – це система безпеки мережі, яка відстежує та контролює вхідний і вихідний мережевий трафік на основі заздалегідь визначених правил безпеки. Брандмауери можна налаштувати для блокування небажаного трафіку з певних IP-адрес та/або портів на основі аналізу мережевого трафіку.

IPS – це система безпеки мережі, яка відстежує мережевий трафік на наявність ознак зловмисної діяльності. У разі виявлення зловмисної активності IPS, автоматично блокує небажаний трафік.

VPN – це технологія, яка дозволяє користувачам отримувати безпечний доступ до приватної мережі через Інтернет. *VPN* можна налаштувати для обмеження доступу користувачів на основі аналізу мережевого трафіку, наприклад блокування доступу до певних веб-сайтів або служб.

WAF (Wireless Access Point) – це система безпеки, розроблена для захисту веб-додатків від таких атак, як впровадження *SQL* і міжсайтовий сценарій. *WAF* можна налаштувати для блокування доступу до веб-додатків на основі аналізу мережевого трафіку [12].

5 етап – «Захист поведінкових сигнатур» [13].

Шифрування повного диска (*FDE - Full Disk Encryption*) – це технологія, яка шифрує весь жорсткий диск комп'ютера або мобільного пристрою, роблячи його недоступним для будь-кого без ключа шифрування.

Диски з самошифруванням (*SED - Self-Encrypting Drive*) – це жорсткі диски, які мають вбудовані можливості шифрування. Вони використовують апаратне шифрування для шифрування даних на диску та можуть бути налаштовані на запит паролю або іншого механізму автентифікації для доступу до даних. Приклади SED включають Samsung T7 Touch Portable SSD та Kingston IronKey D300.

Зашифровані файлові системи: - це файлові системи, які шифрують дані на рівні файлів. Вони забезпечують додатковий рівень захисту конфіденційних даних, що зберігаються на жорсткому диску.

Шифрування віртуального диска: - шифрування віртуального диска – це технологія, яка створює віртуальний зашифрований диск на комп'ютері або мобільному пристрої. Віртуальний диск виглядає як звичайний жорсткий диск для користувача, але всі дані, записані на диск, зашифровані.

Апаратні модулі безпеки (*HSM - Hardware Security Module*): - апаратні модулі безпеки, це фізичні пристрої, які забезпечують криптографічний захист конфіденційних даних. Забезпечують зберігання ключів шифрування та виконання криптографічних операцій, таких як шифрування та дешифрування, не відкриваючи ключі головному комп'ютеру.

Висновки.

1. Розглянута загальна концепція створення модульного, автономного демон юніту, який є основою для подальшої розробки проекту компіляції поведінкових демон юнітів [1].

2. Запропоновано приклад побудови демон юніту, котрий дозволяє систематизувати процес створення власного демон юніту для аналізу мережевої поведінки користувачів з можливістю налаштування під необхідні для адміністратора з безпеки, параметри мережі, та підтримкою автономного блокування дій користувачів при розбіжностях їх мережевої активності з параметрами поведінкових сигнатур, що зберігаються.

3. Для кожного з етапів наведено перелік можливих аналогів, що додає можливість оцінки компонентів існуючого демон юніту, та заміну його модуля іншими *open-source* компонентами.

4. Наведені приклади відомих технологій, де демон юніти є одним із основних компонентів, що підтверджує реалізованість обраної концепції синтезу демону на основі раніше сформованих поведінкових сигнатур [1].

Список літератури:

1. Азаров, С., Малахов, С., & Мелкозьорова, О. Аналіз структури та функції основних елементів алгоритму автоматизованої компіляції поведінкового профілю мережевих користувачів. Proceedings of the XVI International Scientific and Practical Conference. Prague, Czech Republic. 2023. Pp. 504-509. URL: https://isg-konf.com/methods-of-solving-complex-problems-in-science/

2. Офіційна документація проекту fedora, яка містить інформацію про роботу з демонами та можливість їх адміністрування. URL: http://surl.li/hcthe

3. Офіційна сторінка проекту altex-soft, де приводиться порівняння існуючих систем управління базами даних. URL: http://surl.li/hcthw

4. Офіційна сторінка проекту oreilly, яка пояснює основну концепцію обміну повідомленнями. URL: https://www.oreilly.com/library/view/java-message-service/0596000685/ch01.html

5. Офіційна документація проекту systemd, яка містить інформацію про концепцію демонів та її впровадження в Linux-системи з використанням systemd. URL: https://www.freedesktop.org/wiki/Software/systemd/

6. Офіційна документація проекту tcpdump, котра містить повну інформацію яка описує актуальність та спосіб роботи даного модулю. URL: https://www.tcpdump.org/manpages/tcpdump.1.html

7. Офіційна документація проекту wireshark, яка містить усю необхідну інформацію про даний проект. URL: https://www.wireshark.org/docs/

8. Офіційний User-guide проекту suricata. URL: http://surl.li/hctjr

9. Офіційна документація проекту snort. URL:https://www.snort.org/documents

10. Офіційна документація проекту zeek. URL: https://docs.zeek.org/en/master/

11. Джон Маллери, & Джейсон Занн (2007). *Безопасная сеть вашей компании.* (Е. Линдеманн, пер. с англ.). - Москва: НТ Пресс

12. Офіційна енциклопедія британського університету. URL:http://surl.li/hctlm

13. Офіційна документація компанії Microsoft про способи захисту даних. URL: http://surl.li/hctla

CERTIFICATE

Google Scholar
Bowker
INTERNATIONAL SCIENCE GROUP
OUCI Open Ukrainian Citation Index
doi Crossref Content Registration

is awarded to

Немцев Максим Олександрович

for active participation
XX International Scientific and Practical Conference
«TECHNOLOGIES, INNOVATIVE AND MODERN THEORIES OF SCIENTISTS»

May 23-26, 2023, Graz, Austria
24 Hours of Participation
(0,8 ECTS credits)

Organizing committee Ekaterina Zvereva

ОСОБЛИВОСТІ ТЕХНОЛОГІЇ МЕРЕЖЕВИХ ПАСТОК ЯК ІНСТРУМЕНТУ АКТИВНОГО ЗАХИСТУ ТА АНАЛІЗУ ДІЙ АТАКУЮЧОЇ СТОРОНИ

Михайленко Дмитро Денисович
студент факультету комп'ютерних наук (бакалаврат)
Харківський національний університет імені В. Н. Каразіна, Україна

Нємцев Максим Олександрович
студент факультету комп'ютерних наук (бакалаврат)
Харківський національний університет імені В. Н. Каразіна, Україна

Науковий керівник:
Малахов Сергій Віталійович
канд. техн. наук, ст. науковий співробітник, доцент кафедри
Харківський національний університет імені В.Н. Каразіна, Україна

Вступ. Темпи зростання числа кіберзагроз та постійне удосконалення використовуваних технік атаки програмно-апаратних ресурсів користувачів, ставлять сторону захисту у вкрай невигідне, відносно атакуючої сторони, становище. Фахівці у галузі інформаційної безпеки (ІБ) змушені постійно аналізувати відомості про нові інциденти безпеки та методи атак, якими зловмисники скористалися при реалізації чергової атаки. В цьому сенсі слід враховувати, що система реагування на інциденти безпеки побудована таким чином, що розслідування та притягнення до відповідальності відбувається вже після завдання збитків. Саме це надає хакерам суттєвий запас часу на підготовку атак, що забезпечує певну перевагу атакуючій стороні. Крім того, вдосконалення існуючих методів протидії новим способам здійснення атак, має ретроспективний характер, оскільки інформація про поточні загрози стає доступною лише після їх реалізації, що обумовлює певну пролонгованість циклу зворотних реакцій і модифікації наявних технологій захисту [1-4]. Саме тому у профільних фахівців з ІБ з'явилася ідея переходу від класичної (*пасивної або реактивної*) моделі захисту до активної моделі.

Основна частина. Активний захист - це нова модель захисту, яка передбачає використання технологій, що намагаються заздалегідь спровокувати зловмисників до їх виявлення та/або введення їх в пастку, замість того, щоб тільки захищати систему від вторгнень. Ця модель базується на активній взаємодії зловмисника *(або ресурсів зловмисника, наприклад бот-системи та/або бот-мережі)* та системи захисту потенційної жертви, де система захисту «намагається» змінити спосіб дій зловмисника або перехопити його дії. Це досягається шляхом використання в тому числі таких технологій, як *Honeypots* та *Honeynets* [5-6], що синтезують та використовують штучні пастки для потенційних зловмисників і таким чином намагаються перехопити їх дії.

Використання можливостей *Honeypots* (HP) передбачає навмисну обфускацію ПЗ та/або заздалегідь підготовленої інформації, що захищається, шляхом введення додаткових компонентів, які представляють інтерес для кіберзлочинців. В контексті створення HP, під обфускацією мається на увазі ускладнення системи захисту задля створення для атакуючої сторони додаткових труднощів при аналізі цих компонентів (тобто при атаці ресурсу).

В залежності від типу загроз, HP умовно поділяють на [7]: - *Email trap or Spam trap; Decoy Database; Malware Honeypot* та *Spider Honeypot*.

Можливі зворотні реакції HP називають сценарним полем. Створюване сценарне поле [6] відрізняється залежно від характеру взаємодії з хакером, місця розгортання, обсягом параметрів конфігурації тощо. Наприклад, можуть використовуватися такі захисні поведінкові реакції, як: - маршрутизація трафіку; - збільшення часу очікування; - імітація процедур автентифікації при здійснені доступу; - імітаційне шифрування контенту та ін.

Система HP здатна аналізувати і ефективно реагувати на загрози безпеки, тільки якщо цій ресурс виглядає досить правдоподібно, щоб привернути увагу хакера, а також підтримує досить широке поведінкове поле для адекватного реагування на можливі дії нападників. Однак слід мати на увазі, що такий ресурс навіть при коректному налаштуванні може бути розкрито та використано (*скомпрометовано*) проти основної системи яка захищається. Тобто система захисту, яка використовує тільки Honeypot, не є завершеним рішенням для захисту комп'ютерних мережі окремих вузлів та потребує її комплексної підходу шляхом інтеграції з іншими рішеннями безпеки [8-12].

Спираючись на зміст основних завдань [5-7,13] та вимоги до системи *Honeynet*, має бути сформовано перелік компонентів, які дозволяють виконувати поставлені завдання. Традиційно *Honeynet* має наступний перелік складових компонентів (*підсистем*):

1. HP, що є безпосередньою пасткою і відтворює для нападника «синтетичне» середовище для здійснення їм «успішної» атаки на «призовий» ресурс та реалізує функції формування та збереження відповідних *log-файлів*;

2. Підсистема виявлення аномалій мережевого трафіку. Під аномаліями слід розуміти будь-яке відхилення параметрів мережевого трафіку від штатного режиму функціонування елементів мережі, що захищається. Нетипова мережева активність, повинна бути проаналізована на предмет спроби реалізації атак, шляхом порівняння властивостей трафіку з наявною базою даних (БД) поведінкових сигнатур, що містить система аналізу. Ця БД зберігає властивості, що притаманні різним атакам;

3. Підсистема маршрутизації трафіку, основним завданням якої є перенаправлення підозрілого або аномального трафіку на ресурси HP Дана система має керуватись рішеннями, що прийняті у рамках *системи виявлення аномалій мережевого трафіку*.

4. Підсистема взаємодії із хакером. Під взаємодією мається на увазі поведінка HP у відповідь на дії хакера, тобто створення сценарного поля *Honeypot*. Однією з поведінкових реакцій повинно бути розірвання сессії зв'язку із нападником.

Така підсистема має отримувати актуальні дані, щодо стану всіх елементів *Honeynet*.

Вочевидь, що кількість наявних підсистем впливає на якість імітації усієї *Honeynet*. Використання в структурі *Honeynet* лише однієї НР можливо та є достатнім, однак практично не має великого сенсу, оскільки така система не відповідає реальній інфраструктурі *Intranet*. Задля створення кращої імітації реальної мережі потрібно відтворення роботи серверів обробки, зберігання, та передавання інформації в цій мережі. Тому, систему *Honeynet* слід декілька розширити до: - сервер зберігання інформації (*Data server*); - сервер корпоративної пошти (*Corporate mail server*) та *FTP* сервер що дублює процес роботи *Data server*. Одночасне існування *Data server* та *FTP серверу* потрібно задля переправлення трафіку зловмисника з одного серверу на інший.

Зрозуміло, що одночасне існування 4-х окремих систем управління є надлишковим та неефективним з точки зору розгортання *Honeynet* та її подальшої підтримки. Оскільки ці системи залежать одна від іншої, повинні взаємодіяти з адміністратором безпеки та мають доступ до компонентів *Honeynet system,* то найкращим рішенням є інтеграція цих систем у єдиний модуль управління, варіант складу і структури якого наведено на Рис.1.

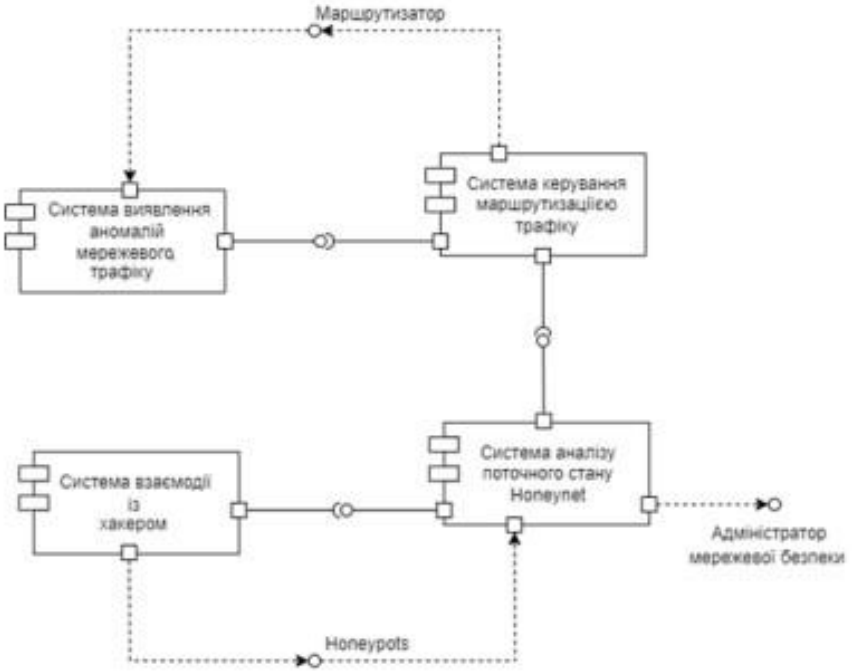

Рисунок 1. Структура логіки взаємодії елементів інтегрованого модулю управління *Honeynet*

Такий модуль дозволяє здійснювати централізоване управління системою *Honeynet* і виконує наступні завдання:

1. Керує «активним» *Access Control List* [8] для вхідного маршрутизатору, завдяки чому здійснюється зворотна реконфігурація поведінки *Honeynet* у

відповідь на дії зловмисника. Такі реакції спрацюють незалежно від режиму функціонування компонентів *Honeypot*s, та впроваджують єдину поведінкову стратегію реакції системи на конкретну спробу [6] реалізації атак, що підвищує шанси на «довіру» із боку хакеру;

2. Прослуховує трафік з вхідного маршрутизатора, перевіряючи наявність поведінкових аномалій та спрямовує трафік до мережі *Intranet* або до *Honeypots* на основі результатів цього аналізу [13];

3. Відслідковує поточний стан всіх складових *Honeynet* (включаючи використовувані HP) та логує відомості про стан подій в межах умовного DMZ середовища [8], що відтворює *Honeynet* [13].

При створенні структури *Honeynet* слід мати на увазі, що хоча головною ціллю нападника і виступають «призові» інформаційні ресурси цієї системи, але потрібно враховувати і спроби атакуючого блокувати саме вхідний маршрутизатор у рамках проведення, наприклад *DoS* атаки [8] на, як він вважає, справжній мережевий ресурс. Якщо основний маршрутизатор буде виведено з ладу, то функціонування усієї системи буде не таким корисним. Тому, потрібен ще один резервний вузол, з функцією маршрутизації, що буде підтримувати безперервну роботу системи у разі тимчасової відмови її основного вхідного компоненту. В цьому разі резервування маршрутизаторів (*в зоні створюваної Honeynet DMZ*) можливо забезпечити за допомогою використання наступних протоколів [14]: - HSRP *(Hot Standby Router Protocol)*, VRRP *(Virtual Router Redundancy Protocol)* та GLBP *(Gateway Load Balancing Protocol)*.

Висновки.

1. При використанні *HP* необхідно провести налаштування середовища для максимальної схожості з реальним джерелом, оскільки в іншому випадку атакуючий швидко зрозуміє, що має справу лише з аватаром бажаного ресурсу.

2. Підсистеми (елементи) виявлення аномалій мережевого трафіку, маршрутизації та взаємодії з хакером є основними вузлами *Honeynet*, що тісно взаємодіють між собою.

3. Усі основні вузли *Honeynet* повинні мати канал зв'язку з адміністратором безпеки задля створення можливості постійного моніторингу та своєчасного корегування реакцій системи в разі виявлення нових (нетипових) дій.

4. Основні елементи *Honeynet* здатні коректно функціонувати лише у випадку, коли підтримують крос читання всіх наявних *log-файлів Honeynet*.

Список літератури:

1. Богданова, Є., Чорна, Т., & Малахов, С. (2022). Огляд поточного стану загроз, що обумовлені впливом експлойтів. *Комп'ютерні науки та кібербезпека*, (2), 35-40. Вилучено з URL: https://periodicals.karazin.ua/cscs/article/view/21039/19745

2. Лахтін, І., Михайленко, Д., & Нарєжній, О. (2022). Порівняння комерційних сканерів вразливостей веб-додатків та сканерів з відкритим кодом. *Комп'ютерні науки та кібербезпека*, (2), 41-49. Вилучено з URL: https://periodicals.karazin.ua/cscs/article/view/21040/19746

3. Погоріла К.В., Богданова Є.С., Колованова Є.П. Огляд можливостей та узагальнення специфіки реалізації XDR-технології, як засобу комплексної протидії актуальним загрозам інформаційної безпеки. Технології, інструменти та стратегії реалізації наукових досліджень: матеріали IV Міжнародної наукової конференції, м. Суми, 07.10.2022 р. / МЦНД. – Вінниця: Європейська наукова платформа, 2022. - 142 с. DOI 10.36074/mcnd-07.10.2022

4. 10 Best XDR Solutions: Extended Detection And Response Services in 2023. URL: https://www.softwaretestinghelp.com/xdr-security-solutions/ (дата звернення: 24.05.2023).

5. Рузудженк, С., Погоріла, К., Кохановська, Т., & Малахов, С. (2020). Особливості захисту корпоративних ресурсів за допомогою технології Honeypot. Комп'ютерні науки та кібербезпека, (4), 22-29. Вилучено з URL: https://periodicals.karazin.ua/cscs/article/view/15751/14600

6. Кохановська, Т., Нарежний, О., & Дьяченко, О. (2020). Дослідження можливостей технології Honeypot. Комп'ютерні науки та кібербезпека, 1(1), 33-42. Вилучено з URL: http://surl.li/hglmt

7. CrowdStrike team. HONEYPOTS IN CYBERSECURITY EXPLAINED [Електронний ресурс] – 2022. Вилучено з: http://surl.li/hgncb

8. Джон Маллери, & Джейсон Занн (2007). *Безопасная сеть вашей компании.* (Е. Линдеманн, пер. с англ.). Москва: НТ Пресс.

9. Яремчук, К., Воскобойников, Д., & Мелкозьорова, О. (2022). Сучасні загрози та способи забезпечення безпеки веб-застосунків. *Комп'ютерні науки та кібербезпека*, (2), 28-34. Вилучено з URL: https https://periodicals.karazin.ua/cscs/article/view/21038/19744

10. Рондалєв, Д., Мелкозьорова, О., & Нарєжній, О. (2019). Особливості функціонування корпоративного міжмережевого екрану та питання взаємодії з системою IDS. *Комп'ютерні науки та кібербезпека*, (3), 11-21. https://periodicals.karazin.ua/cscs/article/view/15614/14707

11. Мелкозьорова, О., Лєсная, Ю., & Малахов, С. (2022). Особливості інтеграції систем захисту від несанкціонованих дій в сучасних інформаційних системах. *Комп'ютерні науки та кібербезпека*, (1), 39-44. https://periodicals.karazin.ua/cscs/article/view/20912/19616

12. Азаров, С., Нємцев, М., & Малахов, С. Огляд аналогій та обгрунтування принципів створення демон юнітів відстеження мережевої активності користувачів. Proceedings of the XX International Scientific and Practical Conference. Graz, Austria. 2023. Pp. 447-453. URL: https://isg-konf.com/technologies-innovative-and-modern-theories-of-scientists/

13. Михайленко, Д., Чорна, Т. & Малахов, С. Використання можливостей AI при реалізації Static та Dynamic Honeypot для покращення параметрів захисту інформаційних ресурсів. *Технології, інструменти та стратегії реалізації наукових досліджень:* матеріали IV *Міжнародної наукової конференції,* (с. 54-57). 7.10.2022 р. Суми, Україна: МЦНД. DOI 10.36074/mcnd-07.10.2022

14. HSRP vs VRRP vs GLBP Protocols – GeeksforGeeks. (2022). Вилучено з: https://www.geeksforgeeks.org/hsrp-vs-vrrp-vs-glbp-protocols/

CERTIFICATE

Google Scholar
Bowker
INTERNATIONAL SCIENCE GROUP
OUCI Open Ukrainian Citation Index
doi Crossref Content Registration

is awarded to

Нємцев Максим Олександрович

for active participation
XXI International Scientific and Practical Conference
«SCIENTISTS AND METHODS OF USING MODERN TECHNOLOGIES»

May 30 – June 2, 2023, Melbourne, Australia
24 Hours of Participation
(0,8 ECTS credits)

Organizing committee — Ekaterina Zvereva