УДК 004.056/.57.4:681.5
DOI: https://doi.org/10.17721/ISTS.2024.8.66-73

Лариса МИРУТЕНКО, канд. техн. наук, доц.
ORCID ID: 0000-0003-1686-261X
e-mail: myrutenko.lara@gmail.com
Київський національний університет імені Тараса Шевченка, Київ, Україна

Олексій ШАЙНА, асп.
ORCID: 0009-0003-5707-0544
e-mail: p.papyge@gmail.com
Київський національний університет імені Тараса Шевченка, Київ, Україна

Дмитро ПАЛКО, асп.
ORCID ID: 0000-0002-2886-1975
e-mail: palko.dmytro@gmail.com
Київський національний університет імені Тараса Шевченка, Київ, Україна

ПРОТОКОЛИ БЕЗПЕКИ В КІБЕРФІЗИЧНИХ СИСТЕМАХ

Вступ. ***Кіберфізичні системи займають важливе місце у сучасних технологіях, оскільки вони поєднують фізичні об'єкти та криптографічні механізми захисту для забезпечення надійної роботи мережних пристроїв, зокрема й у фінансовій галузі, Інтернеті речей і промисловому Інтернеті речей. Основна проблема таких систем полягає у забезпеченні надійного й ефективного захисту даних у разі обмежених ресурсів обчислювальної потужності й енергоспоживання. Криптографічні протоколи, що використовуються у кіберфізичних системах, повинні бути високоефективними, адже від їхньої роботи залежить як безпека, так і загальна продуктивність систем. У цій статті досліджено шляхи підвищення ефективності криптографічних протоколів у кіберфізичних системах.***

Результати. ***Під час дослідження встановлено, що використання протоколів MTLS підвищує рівень захисту даних, але водночас потребує значно більшої кількості ресурсів кіберфізичної системи порівняно з TLS та SSL. До того ж TLS усе ще використовує більшу спроможність кіберфізичних систем аніж SSL, чим підвищує вартість пристроїв кіберфізичних систем. Оптимізація алгоритмів шифрування та дешифрування в протоколі TLS може допомогти зменшити вартість пристроїв і підвищити швидкість передачі даних.***

Висновки. ***Отримані результати показують, що підвищення ефективності криптографічних протоколів у кіберфізичних системах можливе завдяки використанню ефективніших алгоритмів шифрування. Оптимізація протоколів безпеки може значно покращити швидкість передачі даних і продуктивність кіберфізичних систем, особливо у середовищах з обмеженими ресурсами. Варто звернути увагу на концепцію використання існуючих протоколів безпеки, які об'єднують у собі використання симетричних та асиметричних алгоритмів шифрування. В подальшому саме швидкість шифрування та розшифрування відіграватиме визначну роль у підвищенні ефективності. Оскільки саме цей чинник зменшить витрачення ресурсів у кіберфізичних системах, а також отримає перевагу в часі, за рахунок передачі більшої кількості інформації за одиницю часу, не втрачаючи криптостійкості. Подальші дослідження можуть бути зосереджені на розробленні власного протоколу криптографічного протоколу.***

Ключові слова: ***кіберфізична система, асиметричний алгоритм, симетричний алгоритм, протокол безпеки.***

Вступ

Кіберфізичні системи (КФС) займають важливе місце у сучасних технологіях, оскільки вони поєднують фізичні об'єкти та криптографічні механізми захисту для забезпечення надійної роботи мережних пристроїв, зокрема й у фінансовій галузі, Інтернеті речей (IoT) і промисловому Інтернеті речей (IIoT). Основна проблема таких систем полягає у забезпеченні надійного й ефективного захисту даних за обмежених ресурсів обчислювальної потужності та енергоспоживання. Криптографічні протоколи, що використовуються у кіберфізичних системах, мають бути високоефективними, адже від їхньої роботи залежить як безпека, так і загальна продуктивність систем. Ця стаття спрямована на дослідження шляхів підвищення ефективності криптографічних протоколів у КФС.

Метою статті є аналіз існуючих криптографічних протоколів, що використовуються в кіберфізичних системах, та розроблення підходів до підвищення їхньої ефективності. Основну увагу приділено зменшенню енергоспоживання, підвищенню швидкодії алгоритмів шифрування та розробленню методів оптимізації для середовищ з обмеженими ресурсами. Дослідження також охоплює аспекти безпеки, стійкості до фізичних атак і захисту персональних даних.

Результати

Під час дослідження встановлено, що використання протоколів MTLS підвищує рівень захисту даних, але водночас потребує набагато більшої кількості ресурсів КФС порівняно з TLS (Transport Layer Security) та SSL (Secure Sockets Layer). Зазначимо, що TLS усе ще використовує більшу спроможність кіберфізичних систем аніж SSL, чим підвищує вартість пристроїв КФС. Оптимізація алгоритмів шифрування та дешифрування в протоколі TLS може допомогти зменшити вартість на пристрої та підвищити швидкість передачі даних.

Безпека кіберфізичних пристроїв є критично важливою через те, що вони часто працюють у відкритих мережах, й існує ризик перехоплення даних. Для забезпечення безпечного з'єднання між складовими КФС широко використовують криптографічні протоколи, наприклад, TLS, SSL, MTLS.

КФС можуть представляти собою різні IoT-пристрої, з окремою архітектурою безпеки. Для захисту зв'язку між складовими цих девайсів застосовують протоколи безпеки, що шифрують мережний трафік.

Як приклад розглянимо такі КФС.

Смарт-колонки Google Home, Amazon Echo. Смарт-колонки Google Home і Amazon Echo використовують TLS для захисту даних під час їхньої передачі на хмарні сервери, де обробляються голосові команди користувачів.

Фітнес-трекери (Fitbit, Apple Watch). Багато фітнес-трекерів використовують TLS для захисту з'єднань із мобільними додатками або хмарними сервісами для зберігання даних про здоров'я та активність.

Розумні лічильники (Smart Meters). Більшість енергетичних компаній використовують розумні лічильники для моніторингу споживання електроенергії або газу. Ці лічильники використовують TLS для захищеної передачі даних про споживання на центральні сервери компаній.

ISSN 2707-1758

Системи "розумного дому" (Nest, Philips Hue). Системи керування освітленням та опаленням також використовують TLS для забезпечення безпеки у процесі взаємодії із хмарними сервісами або додатками. Усі ці КФС застосовують TLS-протокол для захисту зв'язку. Щоб зрозуміти недоліки та переваги цього протоколу, потрібно розглянути ще два протоколи безпеки – SSL та MTLS. SSL, схема якого зображена на рис. 1 (Tencent Cloud HTTPS) – це попередник TLS, але нині використовується зрідка через вразливості старих версій протоколу (SSL 2.0, SSL 3.0). Більшість сучасних КФС застосовують надійніший протокол TLS замість SSL. Проте раніше SSL використовували в таких системах, як ранні моделі розумних камер відеоспостереження. Деякі моделі камер мали SSL для захисту відеопотоку під час передачі на сервер або мобільний додаток. SSL досі зустрічається на деяких старих або менш оновлених КФС, але більшість сучасних систем перейшли на TLS, оскільки SSL вважається небезпечним.

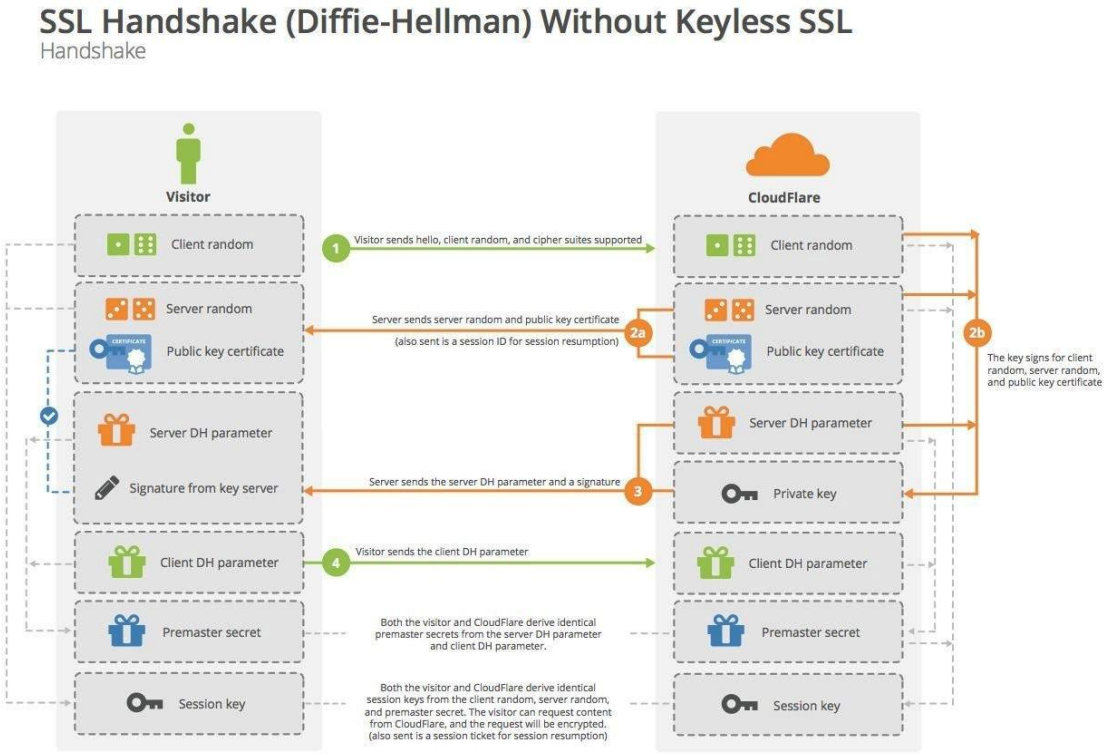

Рис. 1. Схема роботи SSL

Тепер розглянемо приклади використання протоколу MTLS у сучасних КФС. Нині цей протокол захисту з'єднання зустрічається в таких системах: *медичні пристрої* (Smart Health Devices). Деякі медичні IoT-пристрої, такі як пристрої для моніторингу серцевого ритму, інсулінові помпи або інші життєво важливі пристрої, використовують MTLS для забезпечення захищеної передачі медичних даних між пристроєм і медичним сервером. Це важливо для запобігання несанкціонованому доступу до конфіденційних даних пацієнтів. *Індустріальні КФС (IIoT).* У промислових застосуваннях, таких як автоматизовані системи управління заводами, сенсори та контролери часто використовують MTLS для автентифікації серверів і забезпечення безпеки обміну даними. Це допомагає запобігти атакам, які можуть скомпрометувати критичні системи виробництва. *Банкомати та платіжні термінали.* Банківські термінали використовують MTLS для безпечної передачі транзакційних даних між терміналом і сервером банку. Це запобігає атакам типу "людина посередині" (MitM), гарантуючи, що і термінал, і сервер банку є автентичними. *Розумні транспортні системи.* Розумні автомобілі та транспортні системи використовують MTLS для безпечного з'єднання між автомобілем та інфраструктурою (напр., зарядні станції, системи навігації). Це дозволяє зменшити ризики зловмисних втручань у системи автомобіля (Stallings, 2017).

На основі прикладів використання указаних протоколів можна зробити висновок: застосування MTLS у КФС, доречне тільки тоді, коли прилад має великий обсяг електронного ресурсу, що по суті робить такий прилад більш вартісним і габаритним. Варто зазначити, що TLS-протокол використовують безпосередньо в будь-яких кіберфізичних системах, тому що це потребує менший ресурс для його імплементації. Відповідно протокол SSL теж досі зустрічається і потребує ще менших характеристик до приладу на відміну від MTLS і TLS. Розглянемо детальніше наведені протоколи, відмінності між ними та проаналізуємо доцільність їхнього використання в тій чи інший КФС. Основні переваги цих протоколів насамперед у тому, що вони шифрують усі дані, які передаються між клієнтом і сервером, а це запобігає перехопленню або зловмисним змінам даних. Також TLS і особливо MTLS забезпечують і серверу, і клієнту впевненість у справжності кожного з них, що запобігає атакам типу "людина посередині" та несанкціонованому доступу. На додачу протоколи гарантують, що дані не були змінені під час їх передачі (Koblitz, 1987).

Проблеми використання даних систем безпеки полягають у тому, що КФС часто мають обмежену

обчислювальну потужність, пам'ять та енергетичні ресурси. Шифрування й обчислення, пов'язані з TLS і MTLS, можуть бути важкими для таких пристроїв, тому важливо використовувати оптимізовані криптографічні алгоритми (напр., ECC замість RSA). Скажімо, взаємна автентифікація в MTLS вимагає керування цифровими сертифікатами і для серверів, і для клієнтів, що може бути складним і ресурсомістким процесом для великих мереж КФС. Для КФС, які часто мають обмежені мережні ресурси, час установлення захищеного з'єднання може бути критичним. Тому використання легких криптографічних алгоритмів і протоколів, таких як DTLS (легка версія TLS для UDP), може бути кращим варіантом для деяких КФС (Stallings, 2017). Щоб зрозуміти, чому вказані протоколи потребують збільшених ресурсів для пристроїв, пропоную розглянути кожен протокол окремо (Menezes, Van Oorschot, & Vanstone, 1996).

TLS працює на транспортному рівні моделі OSI, як зображено на рис. 2 (OpsFlow Blogs, Mastering HTTPS), і використовується для захисту даних, що передаються між клієнтом (напр.,браузером) і сервером (напр., вебсайтом або хмарним сервісом). Основними функціями TLS є:

- шифрування – захист переданих даних від перехоплення;
- цілісність – гарантування, що дані не були змінені під час передачі;
- автентифікація – підтвердження, що клієнт і сервер є довіреними сторонами.

Тепер пропонуємо розглянути основні етапи роботи цього протоколу: рукостискання (Handshake) – це перший етап, під час якого клієнт і сервер домовляються про параметри з'єднання: привітання від клієнта Client Hello: клієнт ініціює з'єднання, надсилаючи список підтримуваних алгоритмів шифрування, протоколів і версій (Rivest, Shamir, & Adleman, 1978). Привітання від сервера (Server Hello): сервер вибирає підтримуваний клієнтом алгоритм шифрування і надсилає свій сертифікат (цифровий сертифікат SSL/TLS), який містить його публічний ключ (Boneh, & Franklin, 2003). Перевірка сертифіката: клієнт перевіряє сертифікат сервера, щоб переконатися в його справжності. Якщо сертифікат довірений (зазвичай перевіряється за допомогою інфраструктури публічних ключів – PKI (Public Key Infrastructure)), клієнт приймає його. Генерація сесійного ключа: клієнт генерує сесійний ключ для шифрування подальшої комунікації і надсилає його серверу. Цей ключ зашифрований публічним ключем сервера. Захищений обмін: сервер використовує свій приватний ключ для розшифрування сесійного ключа, після чого клієнт і сервер починають використовувати цей ключ для шифрування та розшифрування всіх подальших повідомлень. Шифрування даних (Data Encryption). Після завершення рукостискання весь трафік між клієнтом і сервером шифрується сесійним ключем. TLS підтримує кілька алгоритмів шифрування, таких як AES, ChaCha20, RC4, та інші, які можна використовувати залежно від конфігурації (Boneh, & Franklin, 2003). Перевірка цілісності (Data Integrity): TLS використовує хеш-функції, такі як HMAC Hash-based Message Authentication Code, для перевірки цілісності даних. Це дозволяє виявляти, чи були повідомлення змінені під час передачі (Diffie, & Hellman, 1976). Шифрування даних (Data Encryption): після завершення рукостискання весь трафік між клієнтом і сервером шифрується сесійним ключем. TLS підтримує кілька алгоритмів шифрування, наприклад AES, ChaCha20, RC4, та інші, які можуть використовуватися залежно від конфігурації (Giovanni et all, 2018).

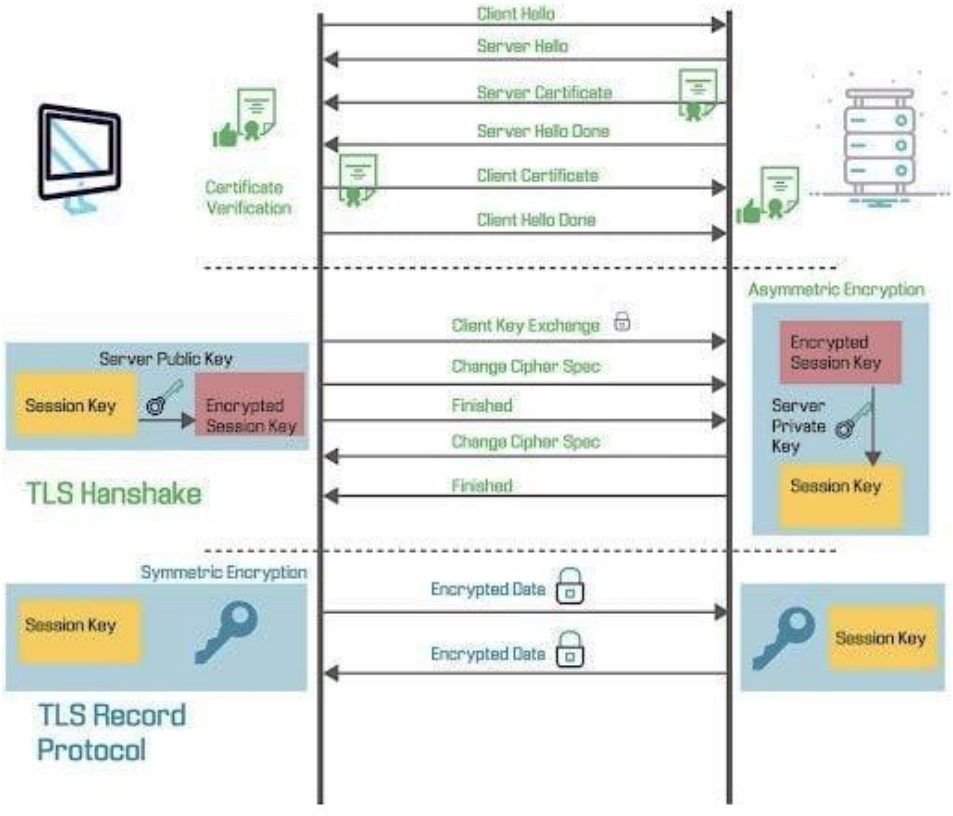

Рис. 2. Схема роботи протоколу TLS

ISSN 2707-1758

Перевірка цілісності (Data Integrity): TLS використовує хеш-функції, такі як HMAC (Hash-based Message Authentication Code), для перевірки цілісності даних. Це дозволяє виявляти, чи були повідомлення змінені під час передачі. Закриття з'єднання (Connection Closure): після завершення передачі даних з'єднання закривається. Це робиться за допомогою спеціальних повідомлень, які підтверджують, що передачу завершено і з'єднання можна безпечно закрити. Чому TLS вимагає більше ресурсів, ніж SSL: TLS підтримує новіші та складніші криптографічні алгоритми, що підвищують безпеку, але також збільшують обчислювальне навантаження. Наприклад, у TLS використовують алгоритми AES і SHA-256, які є надійнішими, але вимагають більше ресурсів, ніж старіші алгоритми, які застосовували в SSL. Покращене рукостискання: у TLS реалізовано покращені механізми рукостискання, які гарантують надійність автентифікації та обміну ключами (Rives, 1978). В TLS відбувається обмін сесійним ключем за допомогою алгоритмів, таких як Diffie-Hellman або його версія на основі еліптичних кривих (ECDH), що забезпечує захист від атак, на зразок "людина посередині". Однак цей процес є обчислювально складнішим порівняно з простішими алгоритмами в SSL. Покращена цілісність даних: SSL використовував слабкіші хеш-функції (напр., MD5 або SHA-1), які були пізніше визнані небезпечними. TLS перейшов на новіші, більш стійкі до колізій функції, такі як SHA-256 або SHA-384. Хоча ці функції значно підвищують безпеку, їхня обчислювальна складність також є вищою. Захист від атаки повторного використання сесій (Replay Attack): TLS забезпечує захист від Replay Attack, яка була слабкіше реалізована в SSL. Це додатковий механізм безпеки, який вимагає більше ресурсів для перевірки послідовності й автентичності кожного пакета. Використання прямих секретів (Perfect Forward Secrecy, PFS): у TLS впроваджено підтримку PFS, це означає, що компрометація одного сесійного ключа не призведе до компрометації всіх попередніх сесій і досягається застосуванням алгоритмів обміну ключами, таких як Diffie-Hellman. Однак PFS вимагає більше обчислювальних ресурсів для генерації нових ключів під час кожного з'єднання. Щоб краще зрозуміти, в чому саме перевага вдосконаленої криптографії, необхідно розглянути різницю між криптографічними алгоритмами DES, AES та SHA-1, SHA-256 відповідно. Спочатку розглянемо основні параметри та структуру SHA-1 та SHA-256, оскільки саме ці функції хешування використовують у протоколах TLS та SSL (Menezes, Van Oorschot, & Vanstone, 1996): алгоритм SHA-1. Ініціалізують п'ять 32-бітових значень:

$$H0 = 0x67452301, H1 = 0xEFCDAB89, H2 = 0x98BADCFE, H3 = 0x10325476, H4 = 0xC3D2E1F0.$$

Кожен 512-бітовий блок ділиться на 16 слів по 32 біти: $W0, W1, \ldots, W15$, де для розширення слів, генерується 80 слів за допомогою розширення вхідних слів:

$$Wt = (Wt-3 \oplus Wt-8 \oplus Wt-14 \oplus Wt-16) \ll 1.$$

Формула для кожного раунду така:

$$T = (A \ll 5) + ft(B,C,D) + E + Wt + Kt,$$
$$E = D, D = C, C = B \ll 30\ C\ =\ B \backslash ll\ 30C = B \ll 30, B = A, A = T.$$

Оновлення значень *H*, після оброблення кожного блока значення $H0, H1, H2, H3, H4$ змінюються:

$$H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D, H4 = H4 + E.$$

Результат хешування: після оброблення всіх блоків результатом є конкатенація значень $H0, H1, H2, H3, H4,$ що утворюють 160-бітовий хеш. Алгоритм SHA-256 встановлює вісім 32-бітових початкових значень:

$$H0 = 0x6a09e667, H1 = 0xbb67ae85, H2 = 0x3c6ef372, H3 = 0xa54ff53a\ ,$$
$$H4 = 0x510e527f, H5 = 0x9b05688c, H6 = 0x1f83d9ab, H7 = 0x5be0cd19.$$

Розбиття повідомлення на блоки по 512 бітів, де кожен блок розбивається на 16 слів по 32 біти, розширюються для створення 64 слів за такою формулою:

$$Wt = \sigma 1(Wt-2) + Wt-7 + \sigma 0(Wt-15) + Wt-16,$$

де

$$\sigma 0(x) = (x \gg 7) \oplus (x \gg 18) \oplus (x \gg 3),$$
$$\sigma 1(x) = (x \gg 17) \oplus (x \gg 19) \oplus (x \gg 10).$$

Оброблення блока даних (64 раунди):

$$T1 = H + \Sigma 1(E) + Ch(E,F,G) + Kt + Wt,$$
$$T2 = \Sigma 0(A) + Maj(A,B,C).$$

Оновлення значень:

$$H = G, G = F, F = E, E = D + T1,$$
$$D = C, C = B, B = A, A = T1 + T2.$$

Оновлення значень хеша:

$$H0 = H0 + A, H1 = H1 + B, H2 = H2 + C, H3 = H3 + D,$$
$$H4 = H4 + E, H5 = H5 + F, H6 = H6 + G, H7 = H7 + H.$$

ISSN 2707-1758

Результат хешування: після оброблення всіх блоків вихідним значенням є конкатенація $H0, H1, \ldots, H7$, що утворює 256-бітовий хеш. Довжина хеша: SHA-1: повертає 160-бітове хеш-значення (20 байтів). SHA-256: повертає 256-бітове хеш-значення (32 байти). Більша довжина хеша в SHA-256 робить його стійкішим до атак за принципом "дня народження" (birthday attack), оскільки кількість можливих комбінацій збільшується з 21602160 (SHA-1) до 22562256 (SHA-256). Для атаки колізій на SHA-256 необхідно обчислити 21282128 хешів, у той час як для SHA-1 потрібно обчислити 280280 хешів. Стійкість до колізій: SHA-1: підтримує теоретичну стійкість до колізій на рівні 280280 (що є вразливим у сучасних умовах). SHA-256: має теоретичну стійкість до колізій на рівні 21282128, що набагато безпечніше для сучасних обчислювальних ресурсів. Рівень криптостійкості: SHA-1 використовує 80 раундів оброблення даних, тоді як SHA-256 застосовує 64 раунди, але на довших 32-бітових блоках. SHA-256 також використовує складнішу структуру та різноманітні константи, що ускладнює відновлення вхідних даних. Теоретична криптостійкість у просторі можливих хешів: SHA-1: 21602160 можливих хешів. SHA-256: 22562256 можливих хешів. Для SHA-1 імовірність знайти колізію (тобто два різних блоки вхідних даних, що дають однаковий хеш) значно більша через менший розмір хеша. Теоретично атака на колізію для SHA-1 потребує 280280 операцій, що вже під силу сучасним суперкомп'ютерам, тоді як атака на колізію SHA-256 потребує 21282128 операцій, що робить її надзвичайно складною і практично неможливою. Атака на обчислення обернення (preimage attack): SHA-1: потребує в середньому 21602160 обчислень для знаходження вхідних даних, що відповідають певному хеша. SHA-256: потребує 22562256 обчислень для тієї самої задачі. Така різниця в обчисленнях робить SHA-256 значно стійкішим до brute-force атак. Рівень ентропії: довший хеш у SHA-256 надає більший рівень ентропії, що значно ускладнює криптоаналітичні атаки. Оскільки кожен біт у хеші SHA-256 має більше можливих варіантів, передбачити вихідне значення або підібрати колізію стає значно складніше.

Переваги SHA-256 над SHA-1. Складніша структура: SHA-256 використовує більшу кількість операцій із довшими блоками, що створює надійнішу хеш-функцію, стійку до сучасних методів криптоаналізу. Стійкість до сучасних обчислень: навіть із розвитком квантових обчислень SHA-256 матиме суттєву перевагу в безпеці порівняно із SHA-1, як зазначено у порівняльній таблиці на рис. 3.

Параметр	SHA-1	SHA -256
Довжина хеша	160 біт	256 біт
Блок повідомлення	512біт	512 біт
Кількість раундів	80	64
Структура	Меркле-Демгард	Меркле-Демгард
Колізійна стійкість	Знижена (ефективність атаки)	Висока
Безпека	Скомпрометований	Сучасний стандарт

Рис. 3. Порівняльна таблиця алгоритмів SHA-1 та SHA-256

Як було зазначено, блокові шифри також використовують у протоколах SSL і TLS, тому слід розглянути відмінність між DES і AES, аби зрозуміти що в сучасних алгоритмах більше не використовують шифр DES. Алгоритм DES:

$$Li + 1 = Ri,$$
$$Ri + 1 = Li \oplus F(Ri, Ki),$$

$F(Ri, Ki)$ – функція раунду, яка залежить від правої частини Ri і підключа Ki для цього раунду. $\oplus$ – операція XOR.

$$Ri \to E(Ri),$$

де E – функція розширення, яка перетворює 32 біти Ri на 48 бітів за допомогою додавання бітів за певною схемою XOR із підключем

$$B = E(Ri) \oplus Ki.$$

B ділиться на 8 частин по 6 бітів, кожна частина обробляється *S*-блоком:

$$S(Bj) \to 4 \text{ біт}.$$

Результати конкатенуються у 32-бітове значення.

Відбувається заміна P:

$$P(S(B)) \to F(Ri, Ki)$$

Фінальний результат: після 16 раундів результати *L* та *R* об'єднуються, і до них застосовується кінцева перестановка FP. Алгоритм AES: 128-бітовий вхідний блок перетворюється на матрицю state розміром 4×4 байти. Введений ключ обробляється для отримання підключів для кожного раунду. SubBytes**:** заміна байтів із використанням *S*-блока.

$$state[i, j] = S(state[i, j]).$$

MixColumns**:** перемішування колонок множенням на фіксовану матрицю в полі Галуа GF(2^8):

$$C(x) = state[i, j] \cdot M,$$

де *M* – фіксована матриця: $M = \begin{matrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{matrix}$

ISSN 2707-1758

AddRoundKey: XOR із підключем раунду:

$$state = state \oplus RoundKey.$$

Формула ключового розкладу: AES генерує підключі для кожного раунду за допомогою функції розширення ключа:

$$W[i] = W[i-4] \oplus T(W[i-1]),$$

де *T* включає *S*-блок і операцію Rcon, яка додає змінну залежність для кожного раунду. Довжина ключа: DES: використовує ключ завдовжки 56 бітів (хоча початково є 64 біти, 8 із них застосовують для перевірки парності).

AES: підтримує три довжини ключа – 128, 192, і 256 біт. Короткий ключ DES значно зменшує рівень безпеки, оскільки потребує лише 256256 операцій для повного перебору, тоді як навіть для найкоротшого 128-бітового ключа AES потрібно 21282128 спроб. Розмір блока: DES: працює з блоками по 64 біти. AES: використовує 128-бітові блоки. Більший розмір блока AES робить його стійкішим до атак, які базуються на шаблонах (patterns), а також підвищує ефективність оброблення великих обсягів даних. Структура алгоритму: DES: використовує 16 раундів обробки з Feistel-структурою, де кожен раунд здійснює операції з підключами. AES: залежно від довжини ключа має 10 (128-бітовий ключ), 12 (192-бітовий ключ), або 14 (256-бітовий ключ) раундів. AES базується на замінно-перестановочній мережі (SP-network), яка містить такі основні етапи: заміну байтів (SubBytes), зсув рядків (ShiftRows), змішування стовпців (MixColumns) і додавання підключа (AddRoundKey). Структура SP-мережі, на відміну від Feistel-структури, забезпечує кращу дифузію та заплутування даних, що підвищує стійкість AES до атак. Повний перебір ключів (Brute-force): DES: із ключем у 56 бітів є вразливим до повного перебору, оскільки сучасні обчислювальні потужності можуть перебрати всі можливі ключі за лічені години чи навіть хвилини. AES: найменший ключ у 128 бітів потребує 21282128 можливих комбінацій, що значно перевищує можливості навіть найпотужніших сучасних обчислювальних систем. Криптоаналіз: DES: вразливий до атаки, відомої як "диференційний криптоаналіз" і "лінійний криптоаналіз". Крім того, є атакована версія – 3DES, яка використовує три послідовні операції DES (шифрування-дешифрування-шифрування) для підвищення безпеки, але навіть вона починає поступатися AES через обмежену стійкість і вимоги до потужності (Koblitz, 1987). AES: стійкий до всіх відомих атак, включаючи лінійний і диференційний криптоаналіз. Його структура SP-мережі і більший розмір ключів роблять AES надійним проти методів криптоаналізу, спрямованих на пошук шаблонів і регулярностей у процесі шифрування. Ефективність та швидкість: DES: вимагає менше ресурсів і є швидшим на застарілому обладнанні, але вже не відповідає сучасним вимогам безпеки. AES: швидший та ефективніший на сучасному обладнанні, особливо на процесорах, які мають апаратну підтримку AES (напр., AES-NI).

Переваги AES над DES. Криптостійкість: завдяки довшим ключам і складнішій структурі AES є більш захищеним від сучасних атак. Стійкість до квантових атак: жоден симетричний алгоритм не є повністю захищеним від квантових обчислень, але AES із більшими ключами має більшу криптостійкість порівняно з DES. Універсальність і швидкість: AES є ефективним як на програмному, так і на апаратному рівні, що робить його універсальним стандартом для захисту даних. Порівняння двох блокових шифрів представлено на рис. 4.

Параметр	DES	AES
Довжина блока	64 біти	256 біт
Блок повідомлення	56 біт	512 біт
Кількість раундів	16	64
Структура	Feistel network	Substitution-Permutation Network
Колізійна стійкість	Вразливий до атак на перебір	Висока, захищений від багатьох атак
Безпека	Скомпрометований	Сучасний стандарт

Рис. 4. Порівняльна таблиця алгоритмів DES та AES

Отже, можна зробити висновок, що вдосконалюючи криптографічні алгоритми, можна покращити рівень безпеки протоколу TLS над SSL, але водночас виникає потреба в більших ресурсах, тим самим підвищується вартість девайсу через його характеристики. Тепер розглянемо протокол MTLS детальніше. MTLS – це розширення стандартного протоколу TLS, яке забезпечує взаємну автентифікацію обох сторін з'єднання: і клієнта, і сервера. У той час як стандартний TLS забезпечує автентифікацію лише сервера, MTLS дозволяє також переконатися, що клієнт є довіреною стороною (Luo, Xu, & Zheng, 2020). Це робить MTLS безпечнішим, але також потребує більше ресурсів для встановлення та підтримання з'єднання. У стандартному протоколі TLS клієнт перевіряє автентичність сервера за допомогою сертифіката X.509, який видає сертифікаційний центр (CA). Відтак клієнт шифрує дані для передачі серверу. Однак сервер не перевіряє, хто є клієнтом, і не знає, чи є він довіреним. MTLS розв'язує цю проблему, вимагаючи автентифікації з обох сторін (Luo, Xu, & Zheng, 2020). Основні етапи роботи такі.

Клієнт і сервер обмінюються сертифікатами: у випадку MTLS і клієнт, і сервер повинні надати цифрові сертифікати під час процесу рукостискання. Це дозволяє серверу перевірити автентичність клієнта і переконатися, що він є довіреною стороною (Schneier,

2000). Автентифікація обох сторін: після обміну сертифікатами клієнт перевіряє сертифікат сервера, щоб переконатися, що він справжній і виданий надійним сертифікаційним центром. Сервер, зі свого боку, перевіряє сертифікат клієнта, якщо обидва сертифікати є дійсними, сторони продовжують встановлення сесії. Шифрування з'єднання: як і у звичайному TLS, MTLS використовує сесійний ключ для шифрування всіх подальших даних після встановлення автентифікації клієнта і сервера. Це забезпечує конфіденційність, цілісність і захист даних від перехоплення. Перевірка сертифікатів через PKI: взаємна автентифікація потребує інфраструктури публічних ключів для керування сертифікатами як клієнта, так і сервера. Кожна сторона повинна мати чинний сертифікат, підписаний сертифікаційним центром, якому довіряють обидві сторони.

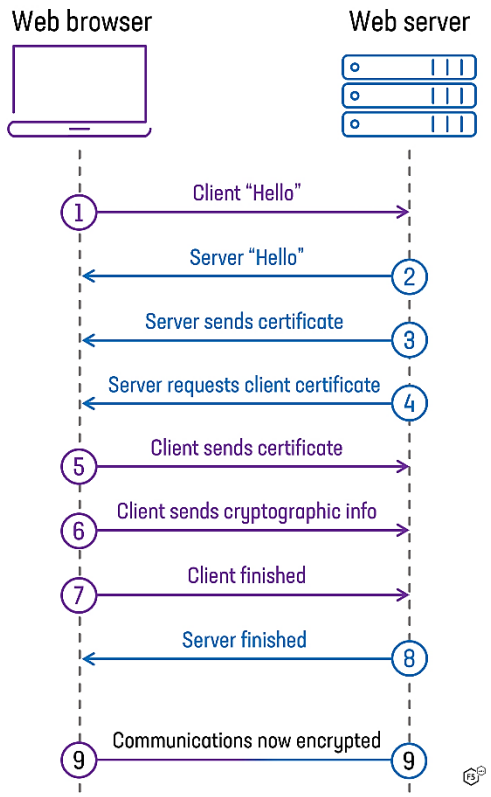

Рис. 5. Схема роботи MTLS

Тепер розглянемо, чому MTLS, схема якого зазначена на рис. 5, вимагає більшої ресурсоспроможності від КФС: додаткова автентифікація клієнта: у MTLS сервер має перевіряти автентичність клієнта за допомогою його сертифіката, що вимагає додаткових обчислювальних ресурсів. Процес перевірки сертифіката включає валідацію сертифіката, перевірку його терміну дії, пошук відмічених сертифікатів у списку відкликаних сертифікатів (CRL) або через протокол OCSP (Online Certificate Status Protocol). Вимога сертифікатів з обох сторін: на відміну від звичайного TLS, де сертифікат надає лише сервер, у MTLS клієнт також повинен мати чинний сертифікат. Це додає оброблення на стороні клієнта, що може бути критичним для пристроїв з обмеженими ресурсами, таких як IoT або КФС. Більший обсяг рукостискання: процес рукостискання в MTLS складніший і включає більше етапів, ніж у стандартному TLS. Кожна сторона повинна обмінятися сертифікатами і виконати обчислювальні операції для перевірки автентичності, що збільшує час установлення з'єднання і вимагає більше обчислювальних ресурсів для оброблення цього процесу. Керування сертифікатами: MTLS покладається на інфраструктуру PKI для керування сертифікатами клієнта і сервера. Це вимагає додаткових зусиль для керування сертифікатами, їхнього видання, поновлення та відкликання. Керування великими обсягами сертифікатів може бути складним і ресурсомістким процесом, особливо в середовищах із великою кількістю клієнтів (напр., у корпоративних мережах або середовищах КФС) (Perez, & Garcia, 2019). З огляду на це можна сказати, що для того, щоб використовувати той чи інший протокол безпеки з'єднання в системах із малою пропускною здатністю, можна спробувати покращити взаємодію симетричних та асиметричних алгоритмів шифрування. Тому необхідно також розглянути можливість зміни складних математичних алгоритмів на легші, що дозволить покращити швидкість шифрування та дешифрування, знизить вартість КФС. Причому потрібно звернути увагу на доцільність використання складних алгоритмів шифрування в кіберфізичних системах, в яких швидкість відіграє визначальну роль. Наприклад, використання складних алгоритмів шифрування в таких КФС, як FPV-дрони, може негативно впливати на роботу самого пристрою та обмін даними між клієнтом і пристроєм. Потрібно враховувати специфіку вказаного девайсу у процесі проєктування аріхтектури безпеки.

Дискусія і висновки

Щоб використовувати той чи інший протокол безпеки з'єднання в системах із малою пропускною здатністю, необхідно покращити взаємодію симетричних та асиметричних алгоритмів шифрування. Запропоновано розглянути можливість зміни складних математичних алгоритмів на легші, що покращить швидкість шифрування та дешифрування, знизить вартість КФС. Варто звернути увагу на концепцію використання існуючих протоколів безпеки, які об'єднують в собі використання симетричних та асиметричних алгоритмів шифрування. В подальшому саме швидкість шифрування та розшифрування відіграватиме визначальну роль у підвищенні ефективності, оскільки саме цей чинник зменшить витрачання ресурсів у КФС, а також отримає перевагу в часі, за рахунок передачі більшої кількості інформації за одиницю часу, майже не втрачаючи у криптостійкості. Подальші дослідження можуть бути зосереджені на розробці власного криптографічного протоколу.

Внесок авторів: Олексій Шайна – збір та аналіз даних про наявні системи захисту в кіберфізичних системах; Лариса Мирутенко – огляд існуючих проблем у протоколах безпеки та пропозиція щодо поліпшення наявних криптографічних протоколів; Дмитро Палко – концептуалізація, аналіз теоретичних засад дослідження.

Список використанних джерел

Boneh, D., & Franklin, M. (2003). Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3), 586–615.

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6).

Rivest, R., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), 120–126.

Schneier, B. (2000). *Secrets and lies. Digital security in a networked world*. Wiley.

Menezes, A., Van Oorschot, P., & Vanstone, S. (1996). *Handbook of applied cryptography*. CRC Press.

ISSN 2707-1758

Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.). Pearson.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177), 203–209.

Luo, J., Xu, L., & Zheng, P. (2020). Security analysis of network protocols in industrial cyber-physical systems. *Journal of Industrial Information Integration*, 17, 100–110.

Perez, R., & Garcia, S. (2019). Application of cryptography in IoT and cyber-physical systems. Current state. *Sensors*, 19(4).

References

Boneh, D., & Franklin, M. (2003). Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3), 586–615.

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6).

Rivest, R., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2), 120–126.

Schneier, B. (2000). *Secrets and lies. Digital security in a networked world.* Wiley.

Menezes, A., Van Oorschot, P., & Vanstone, S. (1996). *Handbook of applied cryptography*. CRC Press.

Stallings, W. (2017). *Cryptography and network security: Principles and practice* (7th ed.).Pearson.

Koblitz, N. (1987). Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177), 203–209.

Luo, J., Xu, L., & Zheng, P. (2020). Security analysis of network protocols in industrial cyber-physical systems. *Journal of Industrial Information Integration*, 17, 100–110.

Perez, R., & Garcia, S. (2019). Application of cryptography in IoT and cyber-physical systems. Current state. *Sensors*, 19(4).

Отримано редакцією журналу / Received: 20.11.24
Прорецензовано / Revised: 27.11.24
Схвалено до друку / Accepted: 13.12.24

Larisa MYRUTENKO, PhD (Engin.), Assoc. Prof.
ORCID ID: 0000-0003-1686-261X
e-mail: myrutenko.lara@gmail.com
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Oleksii SHAINA, PhD Student
ORCID ID: 0009-0003-5707-0544
e-mail: p.papyge@gmail.com
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

Dmytro PALKO, PhD Student
ORCID ID: 0000-0002-2886-1975
e-mail: palko.dmytro@gmail.com
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine

EXISTING SECURITY PROTOCOLS IN CFS

Background. ***Cyber-Physical Systems (CPS) play an important role in today's technology, as they combine physical objects and cryptographic security mechanisms to ensure the secure operation of networked devices, particularly in the financial industry, the Internet of Things (IoT), and the Industrial Internet of Things (IIoT). The main problem of such systems is to ensure reliable and effective data protection with limited resources of computing power and energy consumption. Cryptographic protocols used in cyber-physical systems must be highly efficient, because both the security and the overall performance of the systems depend on their operation. This article is aimed at researching ways to improve the efficiency of cryptographic protocols in KFS.***

Results. ***During the research, it was established that: the use of MTLS protocols increases the level of data protection, but at the same time requires a much larger amount of CFS resources compared to TLS and SSL. At the same time, TLS still uses more capacity of cyber-physical systems than SSL, which increases the cost of CFS devices. Optimizing encryption and decryption algorithms in the TLS protocol can help reduce device costs and increase data transfer speeds.***

Conclusions ***The obtained results show that increasing the efficiency of cryptographic protocols in cyber-physical systems is possible by using more effective encryption algorithms. Optimizing security protocols can significantly improve the data transfer rate and performance of cyber-physical systems, especially in resource-constrained environments. It is worth paying attention to the concept of using existing security protocols that combine the use of symmetric and asymmetric encryption algorithms. In the future, it is the speed of encryption and decryption that will play a significant role in increasing efficiency. Since it is this factor that will reduce the use of resources in the CFS and will also gain an advantage in time, due to the transfer of more information per unit of time, with almost no loss in crypto-resistance. Further research may focus on the development.***

Keywords: ***cyberphysical system, asymmetric algorithm, symmetric algorithm, security protocol.***

Автори заявляють про відсутність конфлікту інтересів. Спонсори не брали участі в розробленні дослідження; у зборі, аналізі чи інтерпретації даних; у написанні рукопису; в рішенні про публікацію результатів.

The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.