УДК 004.056.53
DOI: 10.15587/1729-4061.2022.251570

Розробка алгоритму та програмного забезпечення для захисту маркерів доступу на основі підпису запитів тимчасовим секретом

В. І. Буковецький, В. М. Різак

Запропоновано метод захисту маркерів доступу в клієнт-серверному обміні даними без збереження стану, заснований на формуванні підпису запиту за допомогою тимчасового секрету. Розроблений метод дозволяє не виконувати передачу з кожним запитом маркерів доступу, які дозволили б зловмиснику автентифікуватись дійсним користувачем при компрометації з'єднання, наприклад при використанні атаки «людина посередині».

Запропоновано та обґрунтовано два варіанти методу – спрощений та покращений, сфера використання яких залежить від потреб у захисті та технічних можливостей їх реалізації. Стійкість обох варіантів забезпечується практичною неможливістю підбору початкових вхідних даних геш-функції, яка використовувалась для формування підпису. Покращений варіант дозволяє також захистити маркери доступу на етапі їх отримання, та передбачає захист від атаки повтору запиту. Захист при початковій автентифікації користувача досягається за рахунок використання протоколу Діффі–Геллмана для обміну секретом та маркером доступу. Використання ідентифікаторів запитів та часових міток дозволяє запобігти повторному використанню запиту.

Додатковий захист для маркерів доступу є важливим, адже наявність маркера доступу у зловмисника дає йому повний контроль над обліковим записом користувача. Використання ж SSL/TLS може не дати бажаного рівня захисту для таких важливих даних.

Встановлено, що використання запропонованого методу не додає значних часових витрат. На прикладі використання геш-функції SHA-256 показано, що залежність між розміром повідомлення та додатковим часом на відправку та отримання повідомлення є лінійною. При використанні запропонованого методу в браузері абсолютне значення додаткових витрат часу для повідомлень об'ємом від 100 байт до 2048 Кб становить від 0.4 мс до 142 мс.-Завдяки цьому запропонований метод може бути використаний без значного впливу на досвід користування.

Ключові слова: захист маркерів доступу, підпис клієнт-серверних повідомлень, автентифікація, безпека сесій.

1. Вступ

Стрімкий розвиток мережі Інтернет та технологій відкрив шлях до створення динамічних веб-ресурсів та клієнт-серверних додатків, які можуть надавати користувачеві персоналізовані сервіси. Такий функціонал досягається постійним обміном даними клієнтського додатку із сервером. Найпоширенішим протоколом передачі даних у мережі Інтернет є HTTP, який є протоколом без

збереження стану. Використання протоколу без збереження стану потребує постійної передачі ідентифікуючих користувача даних у кожному запиті. Одним з найрозповсюдженіших методів ідентифікації при такій комунікації є прикріплення певного маркера (токена) доступу до кожного нового запиту. Прикладами таких маркерів є ідентифікатор сесії та JSON Web Token.

Кількість клієнт-серверних додатків які використовуються в повсякденному житті зростає з кожним днем. Відділення банків закриваються, замість них розробляються додатки, які дозволяють отримати всі послуги прямо в смартфоні. Багато держав також взяли курс на цифровізацію, чимало державних послуг стають доступними онлайн. Тепер з комп'ютера чи в смартфоні можна відкрити бізнес, переглянути свою медичну інформацію, переглянути інформацію про наявне майно та навіть виконати передачу права власності на нього. Це дає зловмисникам все більше і більше мотивів для виконання атак. Раніше максимальною наживою зловмисника міг бути профіль на форумі, тепер же їхніми цілями є облікові записи в банківських та державних системах.

Передача даних через мережу відкриває чимало можливостей для атак, головною ціллю яких може стати саме маркер доступу. Отримання цієї інформації дозволить зловмиснику представлятися користувачем навіть не знаючи його основних даних для входу у веб-сервіс (зазвичай це ім'я користувача та пароль). Такими веб-сервісами можуть бути саме онлайн-банкінг, портал державних послуг, керування системою розумного будинку, тощо.

Найрозповсюдженішим методом захисту є використання шифрування протоколів SSL/TLS. SSL/TLS – криптографічні протоколи, які забезпечують встановлення безпечного з'єднання між клієнтом та сервером. Згідно з даними Google, станом на 27 серпня 2021-го року від 79 % до 98 % (в залежності від платформи) сторінок завантажувались через HTTPS (в основі якого лежать криптографічні протоколи SSL/TLS) [1].

Саме на ці криптографічні протоколи покладається функція захисту конфіденційності даних у більшості сучасних веб-додатків.

Однак протоколи HTTPS та SSL/TLS мають свої недоліки та не завжди можуть бути використані через технічні обмеження.

Таким чином, дослідження, присвячені розробці покращених методів захисту даних в системах передачі даних без збереження стану, є актуальними.

2. Аналіз літературних джерел та постановка проблеми

Небезпека атаки MITM з кожним роком стає все більш нагальною. Як вже було зазначено, велика частина сучасних веб-додатків покладає функцію захисту маркерів доступу, які передаються з кожним запитом на протокол HTTPS.

Протокол HTTPS є розширенням протоколу HTTP, де транспорт виконується поверх протоколу SSL(TLS). Протокол забезпечує автентифікацію та шифрування даних, що передаються.

Використання захищеного з'єднання HTTPS активно просувається розробниками сучасних браузерів, так деякі з нових функцій та API доступні лише в захищеному контексті [2].

У літературі детально описані різні методи атак та вразливості цих широковживаних протоколів.

У роботі [3] розглядаються можливі бекдори в багатьох популярних імплементаціях алгоритму Діффі-Хеллмана в TLS, в особливості при його використанні для транспорту HTTP браузерами.

Користувачі інтернету щодня залежать від протоколу HTTPS для безпечної комунікації з сайтами, які вони мають намір відвідати. З плином часу було передбачено, виконано та/або покращено багато атак на HTTPS та модель довіри сертифікатів, яку він використовує. Тим часом кількість центрів сертифікації, яким довіряють веб-переглядачі (відповідно довіряють і їхні фактичні користувачі), зросла, тоді як належний аналіз процедури видачі базових сертифікатів зменшився. В роботі [4] досліджується та класифікуються видимі проблеми безпеки HTTPS, виконується систематичний аналіз попередніх та поточних викликів, задля створення контексту для майбутніх дій. Також пропонується порівняльна оцінка поточних пропозицій щодо вдосконалення інфраструктури сертифікатів, що використовується на практиці.

Посилення впровадження протоколу HTTPS дозволило створити мережу Інтернет, в якій більша частина даних, що передаються – є зашифровані, але ці досягнення безпеки часто стають на шляху уряду, який прагне бачити та контролювати спілкування користувачів.

В 2019 уряд Казахстану здійснив безпрецедентну широкомасштабну атаку перехоплення HTTPS, змусивши користувачів довіряти власному кореневому сертифікату. Авторам роботи [5] вдалося виявити перехоплення та відстежити його масштаб та еволюцію, використовуючи вимірювання з оглядових точок у країні та методи дистанційного вимірювання. Атака була спрямована на підключення, що здійснювались до 37 унікальних доменів, з акцентом на соціальні медіа та комунікаційні послуги, що свідчить про мотив – спостереження. Атака впливала на значну частину зв’язків, що проходять через найбільшого в країні інтернет-провайдера «Казахтелеком». Безперервні вимірювання в режимі реального часу показали, що система перехоплення була вимкнена через 21 день. Згодом, два основних браузери (Mozilla Firefox та Google Chrome) повністю заблокували використання кореневого сертифікату Казахстану. Однак цей інцидент створює небезпечний прецедент не лише для Казахстану, а й для інших країн, які можуть намагатися обійти шифрування в Інтернеті.

В роботі [6] емпірично оцінюється, чи є попередження щодо безпеки підключення в браузерах достатньо ефективними для чіткого інформування користувача про можливу небезпеку, зв’язану з його поточним підключенням. Для дослідження використовувалася телеметрія браузерів Mozilla Firefox та Google Chrome, що дозволило проаналізувати понад 25 мільйонів попереджувальних показів. Під час дослідження користувачі пропускали десяту частину попереджень про шкідливе програмне забезпечення та фішинг Mozilla Firefox, чверть попереджень про шкідливе програмне забезпечення та фішинг Google Chrome та третину попереджень Mozilla Firefox щодо SSL. Це демонструє, що попередження щодо безпеки можуть бути ефективними на практиці; експерти з безпеки та архітектори систем не повинні відкидати мету передачі кінцевим корис-

тувачам інформації про безпеку. Авторами також виявлено, що поведінка користувачів залежить від попереджень. На відміну від інших попереджень, користувачі продовжували пропускати 70,2 % попереджень Google Chrome щодо SSL. Це вказує на те, що досвід користування попередженням може мати значний вплив на поведінку користувача.

Отже, статистика пропуску попереджень про помилки SSL ставить під сумнів можливість повноцінного використання протоколу HTTPS для захисту конфіденційності маркерів доступу в поточній його імплементації.

В роботі [7] проведено дослідження поведінки користувачів, щоб оцінити, чи покращення індикаторів безпеки браузерів та підвищення обізнаності щодо фішингу привели до поліпшення здатності користувачів захищатися від фішингових атак. Учасникам було продемонстровано ряд веб-сайтів та запропоновано ідентифікувати, які з них є фішинговими. Щоб отримати об'єктивні кількісні дані, використовувалось відстеження очей, що дозволяло визначити, на які візуальні сигнали привертають увагу користувачів, коли вони визначають справжність веб-сайтів. Результати показують, що користувачі успішно виявили лише 53 % фішингових веб-сайтів, навіть коли вони були підготовлені їх ідентифікувати, і що вони зазвичай витрачають дуже мало часу, дивлячись на індикатори безпеки порівняно з вмістом веб-сайту під час оцінки. Цікаво, що загальні технічні знання користувачів не корелюють із покращеними показниками виявлення.

В роботі [8] описується метод атаки MITM на протокол HTTPS за допомогою загальнодоступних та безкоштовних інструментів.

З робіт [3–8] можна зробити висновок, що використання протоколу HTTPS не гарантує повної захищеності даних, що передаються. Щоразу з'являються все новіші методи обходу захисту, що дає можливість зловмисникам, в період відомої, але ще не виправленої вразливості здійснити атаку MITM.

Через природу протоколу HTTP, де кожний запит є повністю ізольованим від попередніх, використовується система ідентифікації за допомогою маркера доступу, який може передаватись в Cookies, у заголовках або в тілі повідомлення. Маркер доступу можна вважати ціннішим за інші дані які передаються, адже володіння ним дозволяє отримати тимчасовий або постійний доступ до інформаційної системи, навіть не знаючи початкових даних для автентифікації.

Проблема викрадення даних сесії вже розглядалась в роботі [9], де було описано можливі способи атак, а також методи протидії. Запропонованими методами протидії є використання протоколів типу HTTPS, постійна реавторизація користувача, шифрування маркерів доступу на рівні додатку. В роботі [10] описується захист від викрадення сесії на основі прив'язки маркера доступу до IP-адреси клієнта та даних веб-браузера.

3. Мета та задачі дослідження

Метою роботи є створення алгоритму ідентифікації користувача у додатках, які використовують протокол HTTP або HTTPS, при якому маркери доступу не будуть відправлятись з кожним новим запитом. Це дозволить зменшити можливу шкоду від атаки, при компрометації методів захисту вищого рівня.

При цьому запропонований метод має бути легким у програмній імплементації, зрозумілим, та не створювати додаткове сильне навантаження на сервер та клієнт.

Для досягнення поставленої мети сформовано такі задачі:

– розробити алгоритм обміну запитами без постійної передачі маркерів доступу у відкритому вигляді, а також його імплементацію;

– проаналізувати можливі методи обходу алгоритму та розробити його покращену версію;

– встановити вплив запропонованого алгоритму на швидкість обміну даними.

4. Матеріали та методи дослідження

Для розробки алгоритму вибрано мову програмування JavaScript, яка є дефакто основною мовою для розробки клієнтської частини веб-додатків. Для розробки серверної частини вибрано мову PHP.

Вибір PHP, як мови для розробки серверної частини обумовлюється її широким використанням у розробці веб-додатків, її використовують 78.3 % вебсайтів [11]. Використання JavaScript для розробки клієнтської частини є очевидним – це основна мова для розробки клієнтської частини веб-сайтів, а також може використовуватись для розробки повноцінних програм.

Для виконання серверної частини коду використовувалися сервери: Supermicro X9SCI (виробництво Тайвань) з процесором Intel® Xeon® E3-1230 (випуск – другий квартал 2011-го року, частота 3.20 ГГц, 4 ядра/8 потоків), 12 Гб оперативної пам’яті DDR3 з частотою 1600 МГц та двома жорсткими дисками Seagate ST4000NM033-3ZM (виробництво Китай) об’єднаних у RAID-масив рівня 1. Операційна система Ubuntu 20.04 з PHP версії 7.2; сервер з процесором Intel® Xeon® E-2176G (випуск – третій квартал 2018-го року, частота 4.70 ГГц, 6 ядер/12 потоків), 64 Гб оперативної пам’яті DDR4 з частотою 2666 МГц та двома жорсткими дисками HGST HUH721008AL (виробництво Таїланд) об’єднаних у RAID-масив рівня 1. Операційна система Ubuntu 20.04 з PHP версії 7.2.

Для виконання клієнтської частини використовувався браузер Google Chrome версії 93, запущений на комп’ютері з операційною системою Windows 10 версії 20H2, на процесорі AMD FX-8300 (3.3 ГГц, 8 ядер, 8 потоків), 16 Гб DD3 пам’яті та SSD диском Kingston SHFS37A120G.

5. Результати дослідження методів захисту маркерів доступу на основі підпису запитів тимчасовим секретом

5. 1. Результати розробки алгоритму обміну запитами без постійної передачі маркерів доступу у відкритому вигляді

Розглянуто найрозповсюдженішу методику обміну даними між клієнтом та сервером. Клієнт ініціює нову сесію відправляючи в тілі першого запиту своє ім’я користувача та пароль. На такий запит сервер може відповісти негативно (якщо дані для входу неправильні) або позитивно. У випадку позитивної відповіді сервер надсилає маркери доступу. При кожному наступному запиті клієнт

відправляє отримані маркери доступу в тілі запиту, в окремому заголовку чи в Cookie. По отриманому маркеру доступу, сервер шукатиме в БД відповідного йому користувача, та відносно цього вже буде формувати свій запит.

На рис. 1 зображено знімок екрану програми Charles Web Debugging Proxy, яка дозволяє перехоплювати та аналізувати трафік. На рис. 1 видно маркер доступу, який передається в заголовку Cookie. У випадку компрометації шифрування, зловмисник отримає до нього доступ.

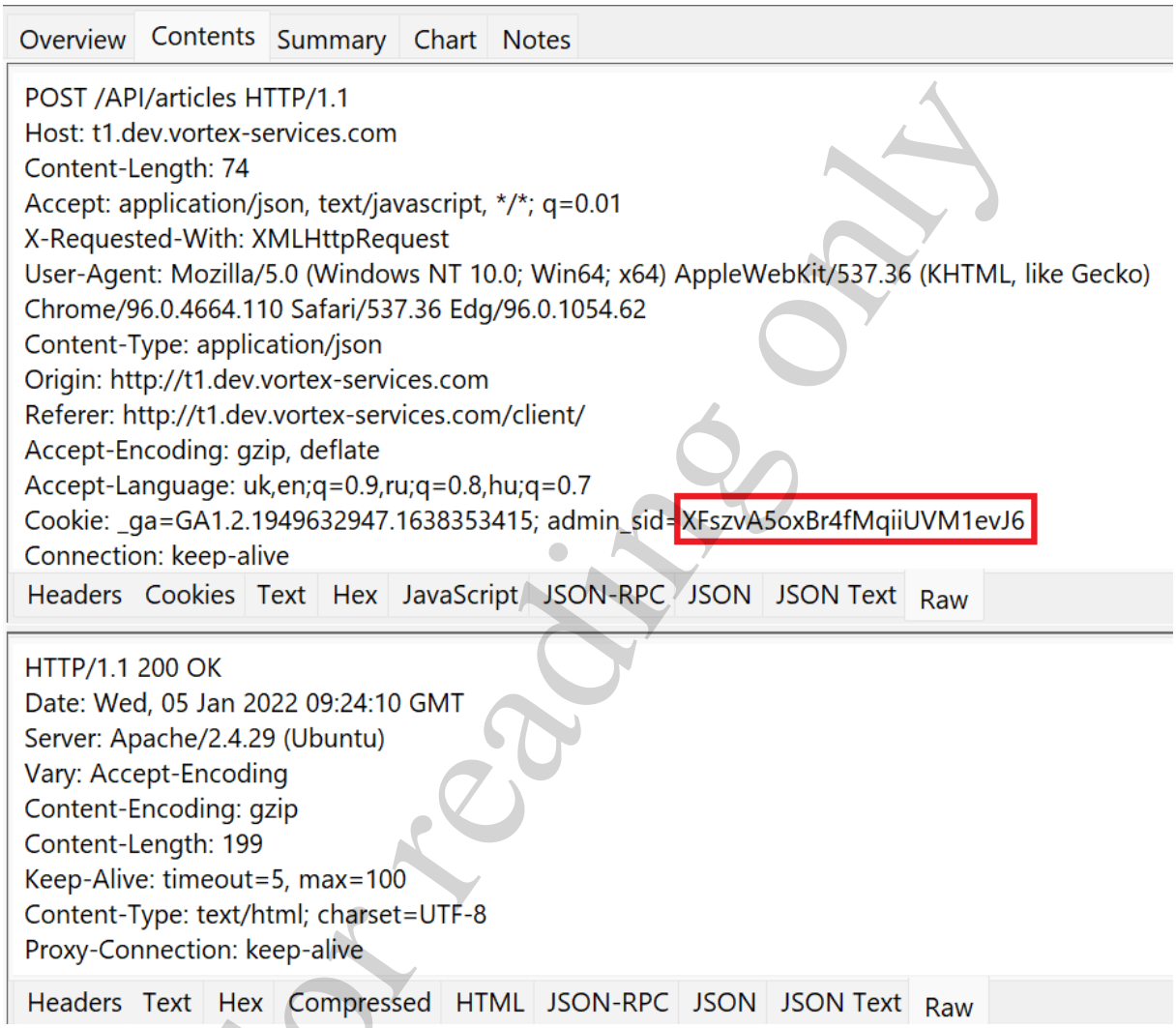

Рис. 1. Незахищений запит на сервер, в якому передається маркер доступу в заголовку Cookie, в полі sid (обведено червоним)

Для вирішення проблеми з постійною передачею маркера, потрібно внести деякі зміни в цю процедуру. При початковій успішній авторизації, замість маркера доступу сервер відправить ідентифікатор маркера та секрет. При цьому, сервер повинен запам’ятати співставлення користувач – ідентифікатор маркера – секрет. Отримані ідентифікатор маркера та секрет клієнт запам’ятовує до кінця сесії.

Для кожного нового запиту на сервер клієнт готує наступні дані: тіло запиту, ідентифікатор маркеру, сіль та підпис запиту. Для формування підпису клієнт об’єднує тіло запиту, секрет та сіль, та застосовує геш-функцію, її результат

і буде підписом поточного запиту. Після цього клієнт може відправити відповідні дані через відповідний транспортний протокол, наприклад HTTP. В такому випадку ідентифікатор маркеру, сіль та підпис запиту будуть відправлені через HTTP-заголовки. На рис. 2 зображено схему роботи такого алгоритму.

Алгоритм не передбачає використання якоїсь конкретної геш-функції, але саме від неї залежить основна частина захисного механізму. Можна стверджувати, що допоки з результату геш-функції не можна отримати початкові дані, доти з підпису запиту не можна отримати відповідні маркери доступу.

Відповідно, в алгоритмі має використовуватись геш-функція, яка є криптографічно стійка, та, відповідно, для якої не було знайдено методів підбору колізій. Через це використання таких популярних геш функцій як MD5 [12–14], SHA1 [15, 16] не рекомендується.

Тим не менш, алгоритм не є прив’язаним до конкретної геш-функції та у випадку знаходження вразливості дозволить швидко здійснити перехід на більш безпечну альтернативу.

При використанні розробленого алгоритму, даних, які відправляються з кожним запитом, не буде достатньо для формування зловмисником нового запиту під видом користувача. Звичайно, такий метод передачі, не захистить дані які відправляються від розкриття, але значно зменшить шанси зловмисника на отримання повного контролю над обліковим записом користувача.

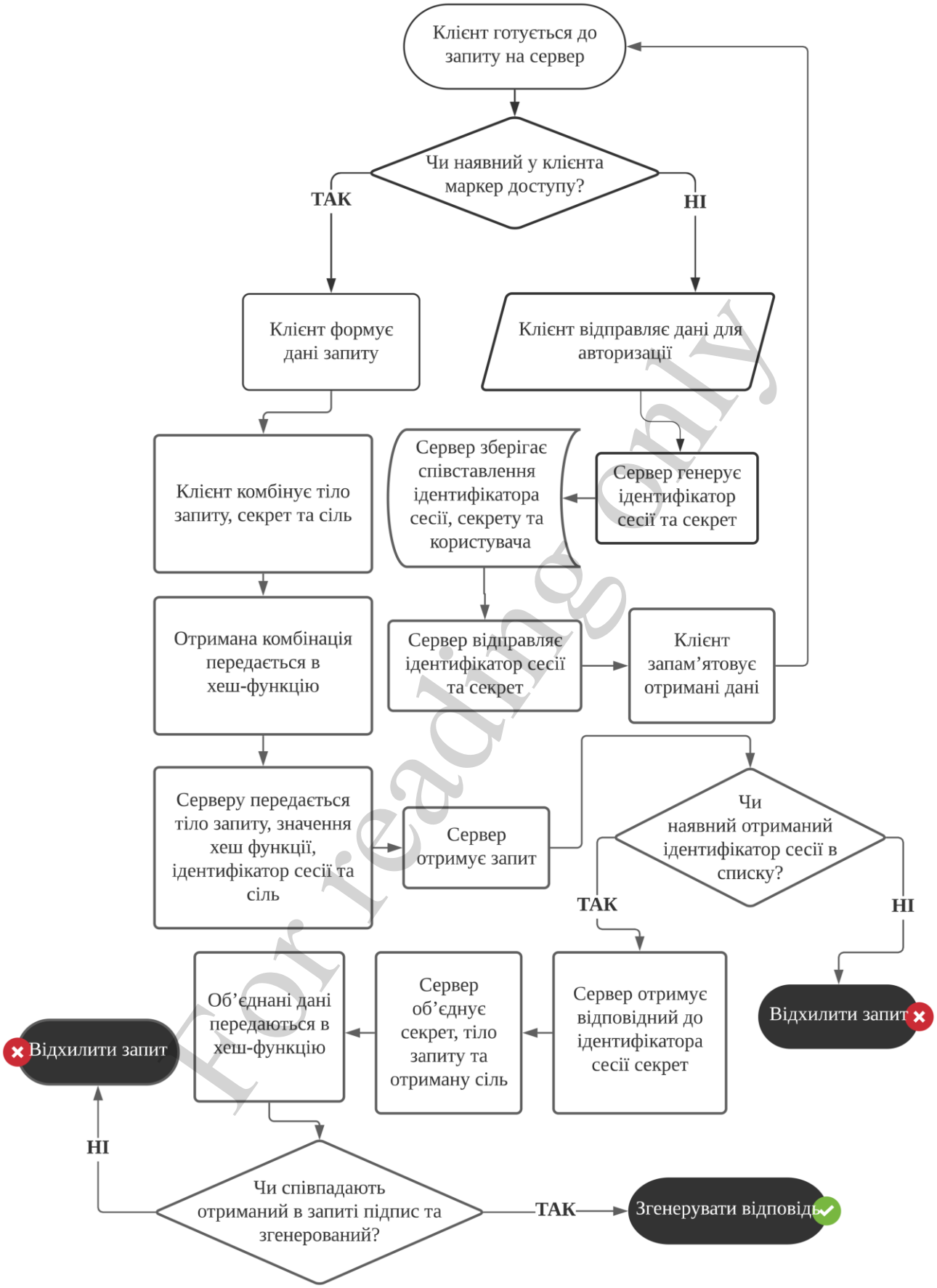

Рис 2. Блок-схема базового алгоритму обміну повідомленнями без постійної передачі маркеру доступу

5. 2. Результати аналізу можливих методів обходу захисту та розробки покращеного алгоритму запитами без постійної передачі маркерів доступу у відкритому вигляді

Слід зауважити, що маркери доступу можуть бути перехоплені при початковому обміну секретом, тому на цьому етапі рекомендується використовувати протокол Діффі-Гелмана (або аналогічний асиметричний протокол) для обміну ключами та подальшій зашифрованій передачі секрету.

Очевидно, що для цієї задачі не підійде симетричний метод, адже клієнт та сервер не можуть наперед знати ключ, з яким би можна було виконати безпечний обмін маркерами доступу.

Також, такий метод не захищає від атаки з повтором запиту. Якщо зловмисник перехопить канал зв'язку та отримає один з відправлених користувачем запитів, він зможе відправити його знову і сервер, перевіривши всі дані, прийме цей запит як автентичний. Це може призвести до значних проблем, адже таким запитом може бути, наприклад, отримання останніх банківських операцій, що дозволить отримувати постійні свіжі дані, поки дійсний поточний маркер доступу.

Для виправлення такого недоліку потрібно додати кілька коректив у процедуру. При обміні маркерами доступу сервер мусить переслати свій поточний час. При цьому, клієнт має запам'ятати різницю свого часу та часу сервера. При створенні нового запиту на сервер, клієнт повинен додавати до кожного запиту свій поточний час, з урахуванням різниці між серверним та своїм.

Крім поточного часу потрібно також відправляти достатньо довге випадкове значення, яке крім відправки у відкритому виді потрібно приєднати до повідомлення, перед створенням значення підпису. Так само, до рядка який буде передано у геш-функцію для генерації підпису, потрібно додати мітку часу клієнта. Найзручніше використовувати для цього мітку часу у форматі UNIX, адже в алгоритмі використовується лише вираховування абсолютної різниці в часі в секундах.

Сервер повинен вести історію випадкових значень і відхиляти всі запити, якщо випадкові значення повторюються. Додавання в запит мітки часу дозволятиме тримати в пам'яті лише короткотривалу історію випадкових значень. Якщо різниця між серверним часом і міткою в повідомленні різниться більше ніж на визначений наперед час зберігання випадкових значень, то такі запити можна автоматично відхиляти. За звичайних умов випадкові значення достатньо зберігати кілька секунд – достатній час для ініціалізації підключення та корекції інших можливих затримок у мережі, через це база таких значень не буде об'ємною. Схема роботи покращеного алгоритму зображена на рис. 3.

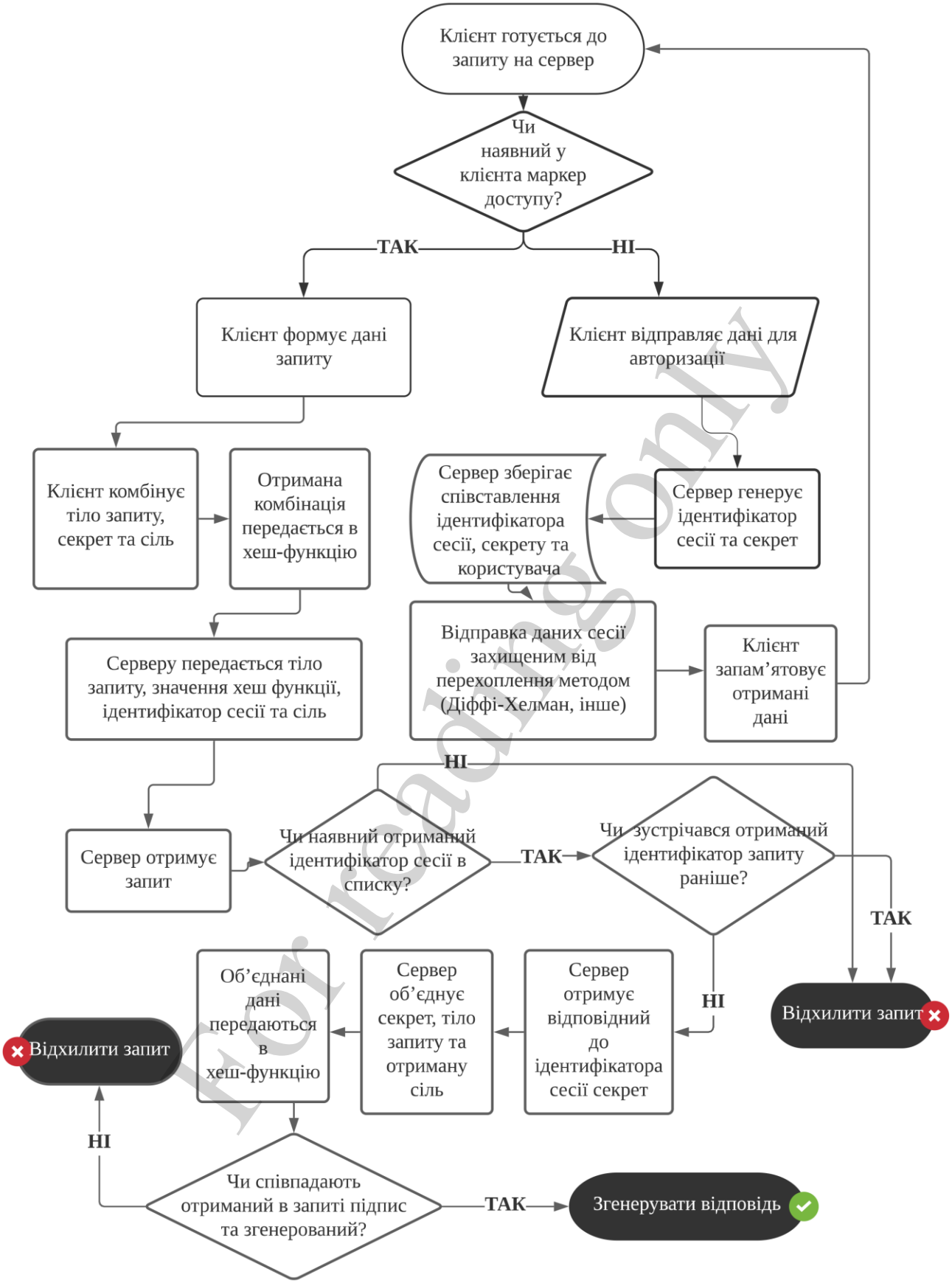

Рис 3. Блок-схема покращеного алгоритму обміну повідомленнями без постій-ної передачі маркеру доступу

Для збереження випадкових значень можна використовувати базу даних типу ключ-значення, такі як Redis, Memcached та інші. Це дозволить вносити лише мінімальні затримки на отримання та збереження цих даних, а також дозволить більш ефективно масштабувати систему для великих навантажень.

Є лише два можливі шляхи отримання маркера доступу за допомогою атаки MITM: знаходженням початкових даних, які були використані в підписі повідомлення та перехоплення маркерів доступу на етапі авторизації сесії. Гарантією безпеки в першому випадку виступає незворотність геш-функції [17], яка використовуватиметься для формування підпису повідомлення, доки не можна з результату геш-функції отримати початкові дані, доти і не можна отримати маркери доступу, які були частиною сформованого підпису.

Опираючись на практичну неможливість зворотного перетворення геш-функції сервер може бути впевненим в автентифікації користувача, адже тільки правильний користувач має одну з частин вхідних даних такої функції: секрет маркера доступу.

5. 3. Вплив запропонованого алгоритму на швидкість обміну даними

Для визначення доцільності розробленого алгоритму потрібно, щоб накладні витрати не були завеликими та значно не сповільнювали обмін повідомленнями між сервером та клієнтом. Для визначення ефективності було реалізовано імплементацію протоколу з використанням мов PHP – серверна частина та JavaScript – клієнтська частина.

Виміри проводились за допомогою вбудованих можливостей JavaScript, від початку формування запиту до отримання повної відповіді сервера. На кожен з розмірів повідомлення проводилось по 100 замірів. В табл. 1 наведено середні значення витрати часу на кожен запит.

Таблиця 1

Витрати часу на захищену та незахищену розробленим алгоритмом передачу для даних різного розміру.

Розмір повідомлення	Час через незахищене підключення (сервер з процесором Intel® Xeon® E3-1230)	Час запиту із використанням простого захисту (сервер з процесором Intel® Xeon® E3-1230)	Час через незахищене підключення (сервер з процесором Intel® Xeon® E-2176G)	Час запиту із використанням простого захисту (сервер з процесором Intel® Xeon® E-2176G)
100 байт	43.2 мс	43.6 мс	39.3 мс	39.4 мс
1 Кб	50.2 мс	51.1 мс	43.4 мс	44.5 мс
8 Кб	52.9 мс	54.2 мс	48.1 мс	50.1 мс
16 Кб	55.8 мс	57.9 мс	50.5 мс	51.3 мс
32 Кб	61.1 мс	63.8 мс	57.6 мс	59.7 мс
64 Кб	71.4 мс	76.3 мс	64.8 мс	68.3 мс
128 Кб	85.1 мс	93.1 мс	75.9 мс	81.8 мс
256 Кб	122 мс	137 мс	110.9 мс	123 мс
512 Кб	194 мс	222 мс	174.2 мс	197.4 мс
1024 Кб	380 мс	452 мс	320.1 мс	381 мс
2048 Кб	890 мс	1032 мс	798.4 мс	903 мс

Як можна побачити з результатів вимірів, приріст затрат є майже лінійним. При чому на малих повідомленнях додаткові затрати є в 10 і більше разів меншими за час на передачу повідомлення звичайним методом.

6. Обговорення результатів розробки алгоритму захисту маркерів доступу

Розроблені алгоритми опираються на стійкість геш-функції до визначення початкових вхідних параметрів. Допоки це твердження є правдивим, з повідомлення, що передається не можна буде відтворити достатньо інформації для представлення користувачем.

На відміну від протоколів SSL/TLS розроблені алгоритми дозволяють захистити маркери доступу не просто шифруючи їх, а взагалі виключає їх з повідомлень, що передаються. При цьому покращений алгоритм виключає також відкриту передачу маркерів доступу на початковому етапі підключення. В комбінації з SSL/TLS, або іншим методом шифрування, розроблений алгоритм дозволить значно покращити захищеність з'єднання.

Як можна побачити з наведених у табл. 1 даних, використання розроблених алгоритмів не вносить значних часових витрат в обмін даними. При цьому, затрати на малих об'ємах достатньо малі, щоб бути повністю непомітними користувачеві. При цьому приріст витрати часу залишається майже лінійним, як і у випадку з незахищеним підключенням. Використання більш потужного серверу не дає значного покращення швидкості роботи алгоритму. Найресурсозатратнішою частиною алгоритму є розрахунок результату геш-функції, при цьому і клієнт і сервер виконують майже ідентичні операції. З великою вірогідністю клієнт буде використовувати менш потужне обладнання ніж сервер, тому в першу чергу всі майбутні оптимізації та покращення в плані швидкості слід проводити в клієнтській частині алгоритму.

Зберігання випадкових значень для контролю повторів запитів не вносить помітних сповільнень у процес обміну даними, так, середня різниця між запитами, що використовували та не використовували порівняння випадкових значень з попередніми становила ~1.2 мс.

Використання алгоритму Діффі-Гелмана не впливає на загальну швидкість роботи, адже використовується лише на початковій стадії отримання маркеру доступу.

Найбільші витрати часу займає розрахунок геш-функції для підпису. Слід зауважити, що для вимірів використовувалася імплементація геш-функції, написана на JavaScript, використання більш оптимальних для цієї задачі інструментів може дозволити зменшити затрати часу в запропонованому алгоритмі.

Запропонований алгоритм має деякі фундаментальні обмеження. Алгоритм захищає від перехоплення лише маркери доступу, а не все повідомлення. Зловмисник все ще зможе прочитати вміст повідомлення, якщо не буде використано додаткових методів захисту.

Запропоноване рішення також має недоліки. Рішенню притаманні додаткові накладні витрати на розрахунок підпису. При чому ці витрати виростають з розміром повідомлення. На дуже великих повідомленнях (наприклад при передачі файлів), такі затрати можуть стати помітними користувачеві. Цей недо-

лік може бути усунутий використанням більш оптимізованим алгоритмом розрахунку геш-функції.

Запропоноване рішення може бути використане на практиці в його поточному виді. Тим не менш, для реального використання потрібний додатковий аналіз та перевірки.

7. Висновки

1. Створено платформонезалежний алгоритм та програмне забезпечення клієнт-серверної передачі повідомлень, який дозволяє не передавати маркери доступу з кожним запитом. На відміну від захисту, що базується виключно на використанні протоколів шифрування SSL/TLS, розроблений алгоритм значно знижує вірогідність отримання маркерів доступу у випадку перехоплення та розшифрування зловмисником повідомлення. Це досягається шляхом зменшення частоти передачі маркерів доступу в процесу обміну даними. Алгоритм є легким в реалізації, адже базується на основі вже доступних в будь-якій сучасній мові програмування функцій. Алгоритм може легко бути інтегрований в уже існуючі системи. Запропонований алгоритм може бути використаний у парі з HTTPS для посилення захисту клієнт-серверного зв'язку та конфіденційності.

2. Розглянуто можливі атаки на запропонований алгоритм, такі як атака повторного запиту та перехоплення маркерів на етапі першого обміну. Запропоновано алгоритм із покращеним захистом. В покращеному варіанті алгоритму змінено процес початкового обміну маркерами доступу для збільшення рівня безпеки. Також в покращений алгоритм додано перевірку на повторну відправку повідомлення. Покращений алгоритм також є легким в реалізації та може бути легко інтегрований в наявні системи. На відміну від рішень запропонованих в роботах [9, 10], розроблені алгоритми значно покращують захист сесії від атак MITM (людина посередині). Це досягається за рахунок відсутності в повідомленнях достатніх даних для формування повідомлення від імені користувача.

3. Встановлено, що розроблені алгоритми захисту не вносять значних додаткових часових затрат у передачу повідомлень. На повідомленнях малого розміру приріст часу не перевищує 10 мс, такі додаткові витрати не будуть помітні користувачеві. На великих повідомленнях (до 2 МБ) додаткові часові витрати складають до 140 мс, що також практично не буде помітно користувачеві в більшості сценаріїв використання алгоритму, при цьому, приріст часових затрат відбувається лінійно, відносно розміру повідомлення.

Література

1. HTTPS Encryption on the Web. Google Transparency Report. URL: https://transparencyreport.google.com/https/overview?hl=en

2. Features restricted to secure contexts. URL: https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts/features_restricted_to_secure_contexts

3. Dorey, K., Chang-Fong, N., Essex, A. (2017). Indiscreet Logs: Diffie-Hellman Backdoors in TLS. Proceedings 2017 Network and Distributed System Security Symposium. doi: https://doi.org/10.14722/ndss.2017.23006

4. Clark, J., van Oorschot, P. C. (2013). SoK: SSL and HTTPS: Revisiting Past Challenges and Evaluating Certificate Trust Model Enhancements. 2013 IEEE Symposium on Security and Privacy. doi: https://doi.org/10.1109/sp.2013.41

5. Raman, R. S., Evdokimov, L., Wurstrow, E., Halderman, J. A., Ensafi, R. (2020). Investigating Large Scale HTTPS Interception in Kazakhstan. Proceedings of the ACM Internet Measurement Conference. doi: https://doi.org/10.1145/3419394.3423665

6. Akhawe, D., Felt, A. P. (2013). Alice in Warningland: A Large-Scale Field Study of Browser Security Warning Effectiveness. 22nd USENIX Security Symposium. Washington, 257–272. URL: https://www.usenix.org/conference/usenixsecurity13/technical-sessions/presentation/akhawe

7. Alsharnouby, M., Alaca, F., Chiasson, S. (2015). Why phishing still works: User strategies for combating phishing attacks. International Journal of Human-Computer Studies, 82, 69–82. doi: https://doi.org/10.1016/j.ijhcs.2015.05.005

8. Chordiya, A. R., Majumder, S., Javaid, A. Y. (2018). Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP Traffic Using Open Source Tools. 2018 IEEE International Conference on Electro/Information Technology (EIT). doi: https://doi.org/10.1109/eit.2018.8500144

9. Kumar Baitha, A., Smitha Vinod, P. (2018). Session Hijacking and Prevention Technique. International Journal of Engineering & Technology, 7 (2.6), 193. doi: https://doi.org/10.14419/ijet.v7i2.6.10566

10. Singh, T., Meenakshi (2020). Prevention of session hijacking using token and session id reset approach. International Journal of Information Technology, 12 (3), 781–788. doi: https://doi.org/10.1007/s41870-020-00486-w

11. Historical trends in the usage statistics of server-side programming languages for websites (2021, November 1). W3Techs. URL: https://w3techs.com/technologies/history_overview/programming_language

12. Dougherty, C. R. (2008). MD5 vulnerable to collision attacks. Vulnerability Note VU#836068. Software Engineering Institute. Carnegie Mellon University. URL: https://www.kb.cert.org/vuls/id/836068/

13. Wang, X., Yu, H. (2005). How to Break MD5 and Other Hash Functions. Lecture Notes in Computer Science, 19–35. doi: https://doi.org/10.1007/11426639_2

14. Libed, J. M., Sison, A. M., Medina, R. P. (2018). Enhancing MD5 Collision Susceptibility. Proceedings of the 4th International Conference on Industrial and Business Engineering. doi: https://doi.org/10.1145/3288155.3288173

15. Wang, X., Yin, Y. L., Yu, H. (2005). Finding Collisions in the Full SHA-1. Lecture Notes in Computer Science, 17–36. doi: https://doi.org/10.1007/11535218_2

16. Stevens, M., Bursztein, E., Karpman, P., Albertini, A., Markov, Y. (2017). The First Collision for Full SHA-1. Lecture Notes in Computer Science, 570–596. doi: https://doi.org/10.1007/978-3-319-63688-7_19

17. Goldreich, O. (2001). Foundations of Cryptography. Volume 1: Basic Tools. Cambridge University Press. doi: https://doi.org/10.1017/cbo9780511546891