Хмельницький національний університет
Факультет інформаційних технологій
Кафедра кібербезпеки

КВАЛІФІКАЦІЙНА РОБОТА

Троца Віталія Володимировича

на здобуття ступеня вищої освіти магістра

Метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі

Галузь знань 12 - Інформаційні технології
Спеціальність 125 - Кібербезпека та захист інформації
Освітня програма Кібербезпека та захист інформації

Шифр КРМКБЗІ.240199.24.01.14 ПЗ

Виконав студент 2 курсу, група КБЗІм-24-1 _____ Віталій ТРОЦ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник д-р філософії, старший викладач _____ Микола СТЕЦЮК
Наукова ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер д-р філософії, старший викладач _____ Наталія ПЕТЛЯК
Наукова ступінь, вчене звання Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

17 грудня 2025 р.

Хмельницький, 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Кібербезпеки

Рівень вищої освіти Магістр

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека та захист інформації

Освітня програма Кібербезпека та захист інформації

ЗАТВЕРДЖУЮ

Завідувач кафедри кібербезпеки

Юрій КЛЬОЦ

1 09 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Троцу Віталію Володимировичу

1 Тема роботи Метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі

Керівник роботи доктор філософії, старший викладач Микола СТЕЦЮК

Затверджено наказом ректора університету 25 08 2025 № 65

2 Строк подання студентом кваліфікаційної роботи на кафедру 01.12.2025

3 Вихідні дані до роботи Провести аналіз сучасних методів і протоколів захисту відеотрафіку. Визначити основні уразливості процесу передавання відео та чинники, що впливають на безпеку. Обґрунтувати вибір криптографічних алгоритмів для реалізації комбінованого підходу. Розробити модель комбінованого методу. Реалізувати програмний прототип системи передавання відеопотоку із комбінованим шифруванням. Провести експериментальну оцінку ефективності запропонованого рішення. Сформулювати рекомендації щодо практичного застосування й інтеграції методу.

4 Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Вступ. Аналіз предметної області захисту потокового відеотрафіку. Огляд сучасних підходів до захисту відеопотоків, аналіз наявних загроз і уразливостей при передаванні відеотрафіку. Постановка задачі дослідження. Побудова математичної моделі запропонованого методу. Розробка методу токено-керованої доставки зашифрованих сегментів відеотрафіку. Розробка та реалізація, тестування програмного прототипу. Аналіз результатів. Висновки.

5 Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6 Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7 Дата видачі завдання 1 09 2025 р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
Дослідження предметної галузі та актуальних здобутків	20.09.2025	Виконано
Визначення змісту, структури магістерської роботи	25.09.2025	Виконано
Опрацювання першого розділу магістерської роботи	01.10.2025	Виконано
Опрацювання другого розділу магістерської роботи	20.10.2025	Виконано
Опрацювання третього розділу магістерської роботи	05.11.2025	Виконано
Апробація за результатами дослідження	14.11.2025	Виконано
Підготовка та опрацювання графічного матеріалу	24.11.2025	Виконано
Оформлення магістерської роботи: графічної та текстової частини	24.11.2025	Виконано
Попередній захист магістерської роботи	27.11.2025	Виконано
Захист магістерської роботи на засіданні ЕК	19.12.2025	Виконано

Студент Віталій ТРОЦ

Керівник кваліфікаційної роботи Микола СТЕЦЮК

АНОТАЦІЯ

Тема кваліфікаційної роботи: Метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі

Автор роботи: студент групи КБЗІм-24-1 Троц В.В.

Керівник роботи: доктор філософії, старший викладач Стецюк М. В.

Загальний обсяг роботи: 94 сторінок, 17 рисунків, 1 таблиця, 18 формул, 1 додаток, 68 посилань.

Ключові слова: потоковий відеотрафік, інформаційно-комунікаційна система, комбінований захист, сегментне шифрування, токено-керована доставка, керування доступом, криптографічний захист, адаптивний стримінг, інформаційна безпека.

У роботі запропоновано метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі, який поєднує сегментне шифрування та токено-керовану доставку зашифрованих сегментів. Розроблений підхід спрямований на підвищення рівня конфіденційності та цілісності відеоданих без суттєвого погіршення затримки й якості відтворення. Метод враховує особливості адаптивного стримінгу, роботу CDN та обмеження реальних мереж, що дозволяє ефективно керувати доступом до контенту. Експериментальні дослідження показали, що запропоноване рішення забезпечує стабільні показники QoS/QoE та кращу стійкість до типових атак порівняно з базовими протоколами захисту. Метод придатний для практичного впровадження у сервісах live та VOD і може бути інтегрований у наявну інфраструктуру без використання важких DRM-рішень.

08.12.2025 Троц В.В.

ANNOTATION

Theme of qualification work: Method for combined protection of streaming video traffic in an information and communication system

Author of the work: student of KBZIm-24-1 Trots V.V.

Mentor: Doctor of Philosophy Stetsiuk M. V.

Total volume of work: 94 pages, 17 figures, 1 table, 18 formulas, 1 appendix, 68 references.

Keywords: streaming video traffic, information and communication system, combined protection, segment-level encryption, token-controlled delivery, access control, cryptographic protection, adaptive streaming, information security.

The paper proposes a method for the combined protection of streaming video traffic in an information and communication system, which integrates segment-level encryption with token-controlled delivery of encrypted segments. The proposed approach aims to increase the level of confidentiality and integrity of video data without a significant impact on latency and playback quality. The method takes into account the specifics of adaptive streaming, CDN operation, and real network constraints, enabling effective access control to content. Experimental results demonstrate that the proposed solution provides stable QoS/QoE indicators and higher resistance to typical attacks compared to basic security protocols. The method is suitable for practical deployment in live and VOD services and can be integrated into existing infrastructure without the use of heavy DRM solutions.

08.12.2025 Трош В.В

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ABR - Adaptive Bitrate.

CDN - Content Delivery Network.

CT - Certificate Transparency.

DASH - Dynamic Adaptive Streaming over HTTP.

DTLS - Datagram Transport Layer Security.

HLS - HTTP Live Streaming.

HSTS - HTTP Strict Transport Security.

ICE - Interactive Connectivity Establishment.

KMS/HSM - Key Management Service / Hardware Security Module.

LL-HLS - Low-Latency HLS.

MITM - Man-in-the-Middle.

mDNS - Multicast DNS (анонімізація локальних IP у WebRTC).

OCSP - Online Certificate Status Protocol.

QoE/QoS - Quality of Experience / Service.

SRTP/SRTCP - Secure RTP / RTCP.

STUN/TURN - Session Traversal Utilities for NAT / Traversal Using Relays around NAT.

TLS - Transport Layer Security.

TGES/ТДЗС - Токен-керована доставка зашифрованих сегментів (метод роботи).

RTP - Real-time Transport Protocol

RTSP - Real-Time Streaming Protocol

WebRTC - Web Real-Time Communication

MPEG-DASH - Dynamic Adaptive Streaming over HTTP

TCP - Transmission Control Protocol

DoS - Denial of Service

DDoS - Distributed Denial of Service

DRM - Digital Rights Management

UDP - User Datagram Protocol

URL - Uniform Resource Locator

IV - Initialization Vector

ВСТУП

Стрімке зростання частки потокового відеотрафіку в загальному інтернет-трафіку перетворює його на критичний ресурс для державних, корпоративних і споживчих сервісів: від відеоконференцій і трансляцій до систем відеоспостереження, дистанційної освіти й телемедицини. На відміну від традиційного обміну файлами, відео вимагає гарантій стабільної пропускної здатності, низької затримки та контрольованого джитера, а будь-які додаткові операції безпеки (шифрування, перевірки доступу, підписи) прямо впливають на якість відтворення та безперервність сервісу. Водночас супутні загрози ускладнюються: поряд із «класичними» атаками перехоплення трафіку (sniffing) та MITM спостерігаються ін’єкції плейлистів/сегментів, кероване отруєння кешів CDN (у т. ч. CPDoS), десинхронізація потоків, повторне відтворення (replay), зловживання підписаними URL і виснаження критичних ендпойнтів (наприклад, видачі ключів). Традиційні засоби - «наскрізний» TLS/DTLS, SRTP або VPN-тунелі - закривають транспортні ризики, але не вирішують проблеми керування доступом до кешованого контенту, дисципліни ротації ключів і цілісності «контрольного плану» (маніфестів), а також суперечності «безпека - затримка - продуктивність».

Клас DRM-рішень і «важких» клієнтських інтеграцій, хоча й підвищує бар’єр для несанкціонованого відтворення, часто є надмірним щодо витрат, складності впровадження й залежності від закритих екосистем. Для значної частини сервісів (новинні, освітні, відомчі трансляції, віддалений моніторинг) потрібне більш прагматичне рішення, сумісне зі стандартними плеєрами та можливостями CDN, без спецівського ПЗ на клієнті. Саме цю нішу заповнює метод комбінованого захисту на базі сегментного шифрування, короткочасних маркерів доступу на краю (token-gating), «дрібних епох» (коротких криптоперіодів) для ключів і диференційованої політики кешування.

Для України тема має підвищену суспільну та прикладну значущість. Масштабована, відмовостійка та «легка» до інтеграції технологія захисту потокового відео без прив’язки до закритих компонентів напряму підтримує

захищені комунікації органів влади, сектору безпеки й оборони, критичної інфраструктури та освіти. Йдеться не лише про конфіденційність контенту, а й про цілісність сигналів, стійкість до деструктивних інформаційних впливів і гарантовану доступність сервісів, що транслюють суспільно важливу інформацію.

Отже, обґрунтування, побудова та емпірична перевірка методу, який поєднує криптографічний захист, керування доступом і політики кешу без втрати QoS/QoE, є своєчасними та доцільними як з наукової, так і з практичної точки зору.

Метою дослідження є підвищення рівень захищеності потокового відеотрафіку в інформаційно-комунікаційній системі шляхом розроблення та впровадження методу комбінованого захисту, який забезпечує збалансований компроміс між безпекою, затримкою та продуктивністю і сумісний зі стандартною інфраструктурою CDN/браузерних плеєрів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– провести аналіз сучасних методів і протоколів захисту відеотрафіку та оцінити їх ефективність;

– визначити основні уразливості процесу передавання відео та фактори, що впливають на безпеку;

– обґрунтувати вибір криптографічних алгоритмів для побудови комбінованого підходу;

– розробити алгоритмічну модель комбінованого методу, який адаптується до умов мережі;

– реалізувати програмний прототип системи передавання відеопотоку із комбінованим шифруванням;

– провести експериментальну оцінку ефективності запропонованого рішення та порівняти його з існуючими протоколами;

– сформулювати рекомендації щодо практичного застосування та інтеграції методу в ІК-системи різного призначення.

Предметом дослідження є методи, алгоритми та політики комбінованого захисту потокового відеотрафіку (шифрування на рівні сегментів, керування доступом на краю, ротація ключів, кеш-стратегія) і їхній вплив на показники

безпеки, затримки та продуктивності.

Запропоновано узгоджений метод токен-керованої доставки зашифрованих сегментів, що поєднує відокремлену видачу ключів (no-store), перевірку доступу до звернення до кеша, «дрібні епохи» та нормалізацію cache-key - у мінімалістичній архітектурі, сумісній зі стандартним плеєром.

Побудовано математичну модель ТДЗС із чіткими інваріантами коректності та параметрами оптимізації «безпека - затримка – продуктивність».

Емпірично показано неінферіорність QoS/QoE ТДЗС щодо базового HTTPS-стримінгу за умови правильного добору T_{seg}/T_{part} і політики буфера, при одночасному підсиленні контролю доступу та стійкості до інфраструктурних атак.

Розроблений підхід придатний до впровадження у сервісах live/VOD без «важких» DRM-стеків і модифікацій клієнта: політики виконуються на краю CDN, ключі не потрапляють у кеші, а параметри (θ, κ, τm, τs, H, T_{seg}/T_{part}) дозволяють конфігурувати компроміс під конкретний SLA. Отримані рекомендації (короткі TTL маніфестів, помірні - для сегментів, строгий no-store і rate-limit для /key, прив’язка токенів до префіксів, узгоджена ротація ключів, наскрізний TLS 1.3) знижують витрати на інтеграцію та підвищують керованість експлуатації. Для українських замовників це означає можливість швидкого посилення захисту критичних відеосервісів у наявній інфраструктурі.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА АКТУАЛЬНИХ НАУКОВИХ ЗДОБУТКІВ ЗА ТЕМОЮ РОБОТИ

1.1 Тенденції захисту потокового відео-трафіку

Перш ніж заглиблюватися в сучасні публікації, варто окреслити, що розуміється під потоковим відеотрафіком та чому його захист - це окрема (не тривіальна) підзадача в галузі кібербезпеки.

Потоковий відеотрафік – це передавання мультимедійного відеоконтенту без збереження його у вигляді цілого файлу на клієнтській стороні перед відтворенням. Декодування й відтворення відбувається поступово, поки відео надходить у потоці. Такий підхід дозволяє користувачам почати перегляд майже одразу, без очікування на повне завантаження відео.

Для кращого розуміння подальшої інформації розлянуто терміни без яких не засвоїти в повному обсязі засвоїти інформацію.

Інформаційно-комунікаційна система - це сукупність технічних, програмних та організаційних засобів, призначених для збору, передавання, обробки, зберігання та захисту інформації з використанням каналів зв’язку. ІКС охоплює мережеву інфраструктуру, протоколи, сервіси та користувацькі пристрої, що взаємодіють у єдиному інформаційному просторі.

Комбінований метод захисту - це підхід, який поєднує різні криптографічні та мережеві механізми (наприклад, блочне та потокове шифрування, тунелювання, обфускацію метаданих) з метою підвищення ефективності захисту потокового трафіку без суттєвого впливу на швидкодію та якість сервісу.

Протоколи потокового передавання – це мережеві стандарти, що регламентують спосіб передавання та синхронізації відео - та аудіоданих у реальному часі. До них належать, зокрема, RTP, RTSP , WebRTC) та MPEG-DASH . Більшість цих протоколів працюють поверх транспортних рівнів TCP або UDP і мають механізми адаптації до змін мережевого середовища [1, 2].

Потоковий відеотрафік, на відміну від класичних форм обміну даними, характеризується постійною передачею інформації в режимі реального часу та

залежністю від стабільності каналу зв’язку. Ця властивість робить його особливо чутливим до впливу як технічних, так і навмисно організованих загроз. У сучасному середовищі кібербезпеки відеопотік розглядається не лише як носій мультимедійної інформації, але й як високовартісний інформаційний актив, порушення конфіденційності якого може мати серйозні наслідки – від комерційних до національної безпеки [3].

Однією з ключових загроз є перехоплення відеотрафіку. Через те, що відеопотік складається з послідовності мережевих пакетів, зловмисник, який має доступ до мережевого сегменту, може здійснювати пасивне прослуховування та отримувати копії даних у момент передавання. Навіть за умов шифрування змісту потоку, сам факт його передавання, параметри сегментів, їхня періодичність та розмір можуть містити метадані, які дозволяють ідентифікувати тип контенту або визначити поведінку користувача. На практиці такі атаки можуть виконуватись на проміжних вузлах мережі, через уразливі точки Wi-Fi, маршрутизатори чи шкідливі програми на клієнтських пристроях [4-7].

Ще одна суттєва загроза - атаки типу «людина посередині» (Man-in-the-Middle, MITM). У цьому випадку зловмисник не просто спостерігає, а активно втручається в процес передавання, перехоплюючи та потенційно змінюючи вміст потокового трафіку. Такі атаки особливо небезпечні в середовищах з використанням несертифікованих або неправильно налаштованих протоколів шифрування. У сфері відеопотоків це може означати підміну кадрів, вставлення фальшивих фрагментів, відображення недостовірної інформації в реальному часі - наприклад, у системах відеоспостереження чи критичних об’єктах інфраструктур [8-10].

Варто також звернути увагу на загрози аналізу побічних характеристик трафіку. Навіть якщо відеопотік повністю зашифрований, його структура передавання, розміри сегментів та часові інтервали формують унікальний «відбиток» потоку який може бути розпізнаний без розкриття ключів шифрування [11, 12]. Цей феномен добре описаний у низці робіт, зокрема в дослідженні Beauty and the Burst: Remote Identification of Encrypted Video Streams,

де доведено можливість ідентифікації контенту за шаблонами сплесків (burst patterns) у мережевому трафіку [4]. Аналогічні підходи показані для сервісів на кшталт Netflix та для DASH-відео в реальному часі, що підтверджує стійкість методу до варіацій протоколів адаптивної доставки [5, 13]. Подібні атаки не потребують доступу до ключів шифрування і можуть здійснюватися навіть пасивними спостерігачами з мінімальними технічними можливостями. Вони становлять особливу загрозу приватності, оскільки дозволяють відновити інформацію про дії користувачів без порушення криптографічного захисту напряму [11, 12, 14].

Крім того, важливим аспектом є атаки на доступність потокових сервісів. Потокове передавання чутливе до затримок і нестабільності мережі, що робить його вразливим до DoS та DDoS атак [2, 15]. У таких сценаріях зловмисник не намагається викрасти дані, а натомість завантажує або блокує мережеві ресурси, створюючи затримки, втрати пакетів і зниження якості відео. Для користувачів це проявляється як зависання, буферизація або розриви з’єднання. Для систем відеоспостереження чи критичних комунікацій це може мати набагато серйозніші наслідки - зокрема тимчасову втрату контролю над важливими об’єктами.

Не менш небезпечною є маніпуляція метаданими та протокольними заголовками. У потокових протоколах значна частина службової інформації - зокрема часові мітки, маркери синхронізації, ідентифікатори сеансів - передається у відкритому вигляді або в напівзахищеному форматі. Це створює можливості для спуфінгу, перенаправлення потоків або збору даних про користувача, навіть без доступу до власне медіаконтенту [16-19].

Таким чином, спектр загроз потоковому відеотрафіку не обмежується лише спробами дешифрувати переданий контент. Він охоплює як пасивні атаки, спрямовані на спостереження та збір метаданих, так і активні втручання, що здатні змінювати або блокувати потік у реальному часі. Особливість цих загроз полягає у їхній невидимості для користувача, адже атаки часто не супроводжуються очевидними ознаками компрометації. Саме тому забезпечення безпеки потокового відеотрафіку потребує комплексного підходу, який враховує не лише шифрування

змісту, а й захист структури, характеристик та службових елементів трафіку.

1.2 Огляд наукових публікацій та сучасних підходів

При дослідженні теми комбінованого захисту потокового відеотрафіку важливо спиратися не лише на історичні підходи до шифрування, але й на сучасні роботи з аналізу зашифрованого трафіку, адаптивного шифрування, «побічного витікання» даних та оптимізації криптозахисту під обмежені ресурси. У цьому підрозділі говориться про кілька характерних і релевантних статей, пояснюються їхні ідеї, досягнення і обмеження.

Одна з важливих праць - Systematic Review on Video Encryption Algorithms - надає комплексний аналіз понад 30 публікацій, що стосуються алгоритмів шифрування відео [20].

У статті розрізняють підходи pre-encoding (шифрування відео ще до його кодування) та post-encoding (шифрування вже закодованих відеоданих). Автори зазначають, що підхід post-encoding є більш практичним і масштабованим, оскільки він працює з даними, які вже готові до передавання й що зменшує надмірне перевизначення структури відео [20].

Окрім того, вони аналізують різні критерії - часову складність, використання пам’яті, стійкість до атак (диференційна атака, статистичні методи), адаптивність та реалізацію в реальному часі. Висновок авторів: більшість існуючих алгоритмів або ж мають високу криптографічну стійкість, але страждають від надмірних обчислювальних затрат або затримок, або ж навпаки - працюють швидко, але знижуючи рівень захисту.

З практичної точки зору цей огляд - корисне джерело, яке допомагає структурувати класифікацію шифрувальних підходів та вибрати ті, що поєднують прийнятну швидкодію із достатнім рівнем безпеки.

У роботі «Securing the Edge: A Comprehensive Review of Adaptive Video Streaming Security Mechanisms in Decentralized Environments» автор К. Кхан

здійснює огляд загроз і захисних механізмів для адаптивного відеострімінгу у середовищах з використанням edge computing [21].

У цьому контексті виникають додаткові виклики: торкаючись мережевих вузлів, які знаходяться ближче до кінцевого користувача (edge nodes), зростає кількість потенційних точок атаки, послаблюється централізація контролю, та збільшується значення латентності й обмеженості ресурсів. Автор аналізує, як класичні механізми шифрування, блокування доступу, аутентифікація поєднуються з новими підходами - наприклад, із використанням блокчейну чи машинного навчання для динамічного реагування на атаки.

Особливо цікавою є частина, де обговорюється необхідність адаптивних заходів захисту: не просто «зафіксованого» шифрування, а систем, які можуть змінювати рівень захисту в залежності від поточної мережевої ситуації (наприклад, коли пропускна здатність падає). Цей аспект перегукується з ідеєю комбінованого методу - де шифрування може бути «потокове» в одні моменти, «блочне» в інші або «пом’якшене» для зменшення затримок.

Як окремий напрям, література фіксує побічні канали в адаптивних потоках: сегментація й буферизація HLS/DASH формують інформативні часово-об’ємні профілі, що зберігаються попри шифрування. Робота «Beauty and the Burst: Remote Identification of Encrypted Video Streams» (Schuster et al.) - одна з ключових у цій галузі - показує, що навіть коли відеопотік шифровано, сегментація та поведінка трафіку створюють патерни «сплесків» (burst patterns), які можуть бути використані для ідентифікації контенту [4].

Автори дослідили, як працюють сплески: спочатку коротка буферизація, потім сталі цикли подачі пакетів (On / Off), і вказали, що ця схема залежить від змісту відео, бітрейту та архітектури стрімінгу. Вони застосували згорткову нейронну мережу (CNN) для класифікації потоків навіть у «open-world» середовищі, де атакуючий може зіткнутися з невідомими відео.

Ця робота має велике значення для теми, бо вона ілюструє: комбінований захист не може ігнорувати побічні канали - навіть якщо вміст даних повністю зашифрований, структура передачі може «пропустити» інформацію

зловмиснику [4].

Як логічне продовження, окремий корпус робіт фокусується на онлайн-класифікації зашифрованих потоків у «шумних» мережах, де алгоритми мають працювати за втрат, джитера та динамічного ABR; показовим є підхід Zenith, орієнтований на реальний час у DASH-сценаріях [13]. У статті «Zenith: Real-time Identification of DASH Encrypted Video Traffic» запропоновано метод ідентифікації відеопотоків у реальному часі, навіть коли дані зашифровані.

Метод Zenith працює з «деформованими» мережевими умовами - з втратами, коливаннями пропускної здатності, шумом - і намагається розпізнавати патерни, не звертаючись до глибокого розшифрування. Стаття демонструє, що агресивні умови мережі не гарантують припинення атак і що захисна система має бути готовою до таких варіантів. Для методу це означає: ви маєте тестувати власний алгоритм не лише в «ідеальних» умовах, але й у «екстремальних» мережевих ситуаціях.

Як відповідь на компроміс між стійкістю та затримками, у літературі окреслюється клас вибіркових/стратифікованих підходів, що переносять криптоопераці на дрібніші одиниці кодування та адаптують силу захисту до значущості фрагмента; репрезентативним прикладом є схеми на рівні slices, що прагнуть збалансувати безпеку й продуктивність [22]. У статті «An efficient video slice encryption scheme and its application» автори пропонують новий підхід, який поділяє відео на шматки (slices), застосовує сортування, злиття і гнучке шифрування, щоб досягти балансу між швидкодією та рівнем безпеки.

Ця схема дозволяє захищати найбільш «чутливі» частини відео сильнішим шифруванням, а менш важливі - більш легким. Такий підхід подібний до ідеї «вибіркового шифрування», але з додатковою логікою, яка мінімізує вплив на обчислювальні ресурси та затримку. У комбінованому методі можна запозичити ідею «стратифікації» захисту - тобто різні частини потоку (ключові кадри, метадані, фон) шифруються з різним рівнем суворості [22].

Як окремий вектор оптимізації, сучасні підходи зміщують обчислення ближче до даних та використовують нетрадиційні перетворення для зниження затримок: поєднання багатопотоковості з хаотичними перетвореннями та прийоми

in-memory computing дають змогу паралелізувати шифрування на рівні підкадрів і мінімізувати накладні витрати без відмови від криптографічних гарантій.

Інша робота «Real-Time Bit-Level Encryption of Full High-Definition Video Without Diffusion» обговорює виключення стадії дифузії (яка зазвичай потребує багато часу), заміщуючи її іншими криптографічними підходами для зменшення затримки [23].

Обидві статті важливі тому, що вони демонструють: компроміс між безпекою й продуктивністю може бути переглянутий за допомогою нових архітектур шифрування - і ці ідеї можуть бути інтегровані у комбінований підхід.

Огляд публікацій показав: самого шифрування вмісту відео недостатньо. Навіть коли кадри зашифровані, зовнішні ознаки передавання - розміри і чергування пакетів, паузи між сегментами, стрибки бітрейту - утворюють помітні «візерунки» трафіку. За ними зловмисник може здогадатися, що саме дивиться користувач або як поводиться система. Тому захист має охоплювати не тільки дані, а й форму їхньої передачі.

Найперспективнішим виглядає комбінований підхід: поєднання вибіркового (стратифікованого) шифрування важливих елементів відео (наприклад, ключових кадрів і критичних метаданих) із маскуванням трафіку - вирівнюванням розмірів пакетів, обережним паддингом та легким «розмиванням» таймінгів у межах допустимої затримки. Така зв’язка знижує інформативність побічних ознак і водночас не «вбиває» швидкодію. Водночас усе це - компроміс: додатковий паддинг і рандомізація трохи підвищують затримку та витрати трафіку, тож параметри потрібно ретельно підбирати.

Сучасні роботи також підкреслюють потребу в адаптивних політиках: система має автоматично змінювати силу шифрування, частоту оновлення ключів і рівень маскування залежно від стану мережі (втрати, джиттер, заповнення буфера) та можливостей пристрою. Це особливо важливо для edge-сценаріїв (камери, мобільні пристрої), де ресурси обмежені, а точок потенційної атаки багато [24-26].

Для коректної оцінки ефекту комбінованих заходів огляди радять виходити за рамки «ідеальної лабораторії»: потрібні тести в шумних умовах (втрати,

перепади пропускної здатності) і open-world сценарії, де атакувальна модель стикається з новими, невідомими потоками. Метрики мають бути комплексними: затримка й стабільність відтворення (QoS/QoE), навантаження на ресурси, а також реальна стійкість до аналізу зашифрованого трафіку (наскільки падає точність класифікатора зловмисника).

Отже можна сказати що, підсумок для подальшої роботи простий: варто проєктувати багаторівневий і адаптивний метод, який комбінує криптографію зі свідомим керуванням формою трафіку та перевіряється в наближених до реальності умовах.

1.3 Загрози та уразливості при передаванні відеотрафіку

Передавання потокового відео в гетерогенних мережах поєднує криптографічні засоби із складною логікою доставки (сегментація, буферизація, адаптивний бітрейт, кешування на CDN, сигнальний обмін). Така багатошарова організація породжує не лише класичні ризики для конфіденційності, цілісності та доступності, а й специфічні уразливості на межі «площини даних» (медіасегменти, RTP/RTCP) та «площини керування» (маніфести/плейлисти, токени доступу, сигналізація).

Навіть за коректного шифрування зберігається інформативність побічних ознак трафіку, а також можливості маніпуляцій метаданими й відповідними політиками доступу.

У межах даного дослідження поверхню ризику визначено з урахуванням типових стеків (HLS/DASH, WebRTC/SRTP) та інфраструктури доставки (edge/CDN, origin, ключові ендпойнти, сигнальні служби). Модель загроз охоплює пасивні сценарії (спостереження, кореляція за таймінгами та обсягами), активні сценарії (ін’єкції, підміни, replay, виснаження), а також атаки на контроль доступу та керування ключами, що безпосередньо впливають на цілісність контрольного плану й показники якості відтворення.

Щоб уникнути дублювання контрзаходів між рівнями та забезпечити коректне відображення ризиків на механізми захисту, подальше викладення систематизовано у взаємодоповнювальні категорії з фіксацією характерних векторів і наслідків. Узагальнювальна таблиця нижче конденсує ці положення: для кожної категорії наведено типові атаки та їхній ефект (табл 1.1).

Таблиця 1.1 - Загрози для потокового відео трафіку

Категорія	Підтипи / вектори	Короткий опис / мета
1	2	3
Конфіденційність і приватність (пасивні & side-channel)	Прослуховування; fingerprinting за розмірами/таймінгами; деанонімізація та витік IP/геоданих (ICE/WebRTC)	Отримання значень без зміни трафіку: відбиток контенту, профіль перегляду, реальна локація/топологія
Цілісність і маніпуляції даними (активні)	MITM/підміна сертифікатів або SDP; ін’єкція/отруєння маніфестів/сегментів; replay/перестановка/десинхронізація; спуфінг часу/синхронізації	Зміна або вставка даних/метаданих для підміни контенту, зсуву таймінгів, зламу логіки відтворення
Контроль доступу та автентифікація	Крадіжка/підміна токенів; зловживання підписаними URL/cookie; вразливості JWT; захоплення/підміна сесії	Обхід платного чи обмеженого доступу, привласнення прав користувача
Доступність та інфраструктура	DoS/DDoS; протокольні виснаження (ключовий ендпойнт, маніфести, RTCP/ABR); компрометація сигналізації; несанкціонований доступ до ендпоінтів (IP-камери/IoT)	Порушення роботи сервісу, деградація QoE, зрив встановлення сесій доставки сегментів
Криптографія та керування ключами	Повтор IV/nonce; слабка/відсутня ротація (crypto-periods); витік ключів/KMS/DRM; даунґрейд/слабкі шифри; помилки інтеграції AES-CTR/GCM/CBCS	Компрометація значних обсягів даних, можливість ретроспективного розшифрування, підрив стійкості схеми

Наведена таблиця окреслює узагальнену поверхню ризику та функціональне групування векторів нападу. Для забезпечення повноти аналізу доцільним є деталізований розгляд кожної категорії з урахуванням припущень моделі загроз, можливостей противника та точок спостереження/втручання в траєкторію доставки відео. Далі подається стислий опис характерних підтипів атак, їхніх передумов і очікуваних наслідків для конфіденційності, цілісності та доступності, а також індикаторів спостережності. Окремо конкретизуються релевантні контрзаходи та обмеження їх застосовності, з акцентом на вплив на затримковий бюджет і показники якості обслуговування (QoE).

Для повного розуміння потрібно детально розібрати можливі загрози та методи захисту від них. Першим з чого потрібно почати це загрози спраямовані на конфіденційність і приватність.

Однією з найбільш маштабних загроз є прослуховування. Це пасивне спостереження, за якого противник не змінює трафік, а вимірює метадані. Попри TLS/DTLS-SRTP, доступними лишаються інтервали між запитами, послідовність і приблизні розміри відповідей, ритм оновлення маніфестів -разом вони формують «силует» потоку для інференції без доступу до кадрів [8]. Джерело інформативності -механіка HLS/DASH: сегментація й цикли буферизації породжують burst-патерни, чия амплітуда корелює з розміром сегмента та складністю сцени; ці профілі зберігаються попри транспортне шифрування [1, 2]. Емпірично зафіксовано «віддалену» ідентифікацію зашифрованого відео через відбитки ADU під HTTPS у режимі, близькому до реального часу [4 5]. VPN не усуває загрозу: розподіли розмірів та міжприходові інтервали на виході тунелю залишаються інформативними; із легковагових ознак відновлюються й QoE-метрики (затримка старту, роздільність, частота перемикань бітрейту) у польових умовах [14, 27].

Ще не потрібно забувати про те що трафік залишає після себе в мережі. Fingerprinting у потоковому відео -це побудова стійких «відбитків» трафіку за послідовностями розмірів відповідей/сегментів та інтервалами між запитами, які не приховує шифрування. Джерело інформативності -механіка HLS/DASH:

сегментація сталої тривалості та цикли буферизації породжують «ривки», амплітуда яких корелює з розміром сегмента, а той -зі складністю сцени й профілем кодування. Емпірично показано віддалене розпізнавання зашифрованих потоків за цими ознаками; для масових сервісів (Netflix) інваріантні послідовності ADU дають ідентифікацію назв за лічені хвилини пасивного спостереження. Сучасні методи працюють у реальному часі та «шумних» мережах, зберігаючи високу точність [13]. Навіть у Low-Latency HLS характерні патерни не зникають, а переходять на тонший часовий масштаб [24]. VPN не нівелює часово-об'ємні закономірності: розподіли розмірів і міжприходові інтервали залишаються достатньо інформативними для класифікації на виході з тунелю [27]. Наслідки -ідентифікація контенту/жанру, відновлення параметрів перегляду та поведінкових профілів; контрзаходи мають керувати «видимою формою» трафіку: квантування розмірів, помірна рандомізація інтервалів, дисципліна ABR і, за потреби, обережний паддинг/мікропейсинг у межах затримкового бюджету.

Окрім захисту трафіку не можна й забувати про захист даних користувача. Деанонімізація та витік IP/геоданих (ICE/WebRTC). У WebRTC встановлення з'єднання здійснюється через ICE: браузер збирає host, server-reflexive (STUN) та relayed (TURN) кандидати і публікує їх у сигналізації; далі виконується перевірка зв'язності та вибір маршруту [19]. Оскільки reflexive-кандидати відбивають публічну адресу, а host -локальну, сам склад кандидатів розкриває мережеву топологію, провайдера й регіон користувача, що уможливлює непряму геолокацію без доступу до медіаданих. Стандарти та рекомендації передбачають mDNS-маскування host-кандидатів, обмеження типів кандидатів і політики, що унеможливлюють витік поза довіреним маршрутом (наприклад, relay-only) [18]. На практиці, однак, хибні налаштування та відсутність політик залишають канали витоку; зокрема, у VPN-конфігураціях STUN може «видати» реальний зовнішній інтерфейс . Польові вимірювання підтверджують: навіть без явних локальних адрес кореляція за ICE-поведінкою й типами кандидатів дозволяє відновлювати геодані та параметри мережевого середовища з достатньою для деанонімізації точністю. Зниження ризику забезпечують: mDNS для host-адрес, примус relay-only у

чутливих сценаріях, автентифікація/авторизація TURN та його розміщення у контрольованих регіонах, а також політики браузера/додатка, що обмежують збір і видачу ICE-атрибутів стороннім скриптам; неповне впровадження цих заходів знову відкриває шлях до витоків [28].

Окрім захисту користувача і даних від попадання в чужі руки, потрібно подбати про цілісність даних.

Коли у звязоку між користувачами зявляється зайвий учасник це називається «людина по середині» або MITM. У потокових системах MITM реалізується через перехоплення/термінацію захищених з'єднань або підміну керуючих повідомлень. Для HTTP-базованої доставки базовим бар'єром є TLS 1.3; HSTS запобігає SSL-stripping, а Certificate Transparency ускладнює непомітну підміну сертифікатів завдяки журналюванню випусків. Окремий ризик становлять enterprise-проксі з TLS-інтерсепцією: попри «видиме» HTTPS, послаблення параметрів і перевірок на проксі збільшує площу уразливостей, тому для критичних доменів (ключі, маніфести) доцільне виключення інтерсепції та моніторинг CT-журналів. У реальному часі вирішальне значення має цілісність сигналізації: у WebRTC відбиток DTLS-сертифіката передається в SDP через a=fingerprint; компрометація каналу сигналізації дозволяє переписати a=fingerprint/ICE-кандидатів і термінувати DTLS з кожною стороною окремо, фактично реалізуючи прозорий MITM, попри SRTP на медіаканалі [29-32]. Отже, криптографічна прив'язка сертифіката до SDP та захищений транспорт сигналізації -необхідна умова недопущення MITM.

Людина по середині може різними методами втручатися в трафік. Наприклад ін'єкція або отруєння маніфестів і сегментів. У HTTP-стримінгу маніфести (M3U8/MPD) керують послідовністю сегментів, варіантами бітрейту та ключами; їх модифікація або підміна сегментів нав'язує клієнтові чужий маршрут завантаження, підмінює контент або зриває відтворення, що зумовлює підвищену чутливість цих об'єктів до ін'єкційного впливу, . Поза прямим MITM, практично значущим є cache poisoning: через маніпуляції заголовками/параметрами edge та origin формують різні ключі кешу, і «отруєний» маніфест/сегмент роздається іншим користувачам [29]. Споріднена техніка CPDoS змушує кешувати помилкові

відповіді (напр., 404/301), що блокує доступ до ресурсів на масштабі краю; провайдери описують спеціальні заходи протидії [15, 32, 33]. Ефективний захист спирається на дисципліну кешу та перевірку доступу до lookup'у: нормалізацію ключа кешу й мінімальний Vary, no-store для ключового ендпойнта, короткі TTL маніфестів із швидкою інвалідацією та відмову від негативного кешування на критичних шляхах [34]. Для контролю доступу доцільні підписані cookie/URL з коротким строком дії, вузькою областю (path) та перевіркою на edge до віддачі спільного об'єкта, що унеможливлює «гарячі лінки» і масштабоване розповсюдження отруєних відповідей.

Також щоб залишатись не помітним, але маніпулювати даними можна використати такі методи як replay / перестановка / десинхронізація. Replay -повтор раніше валідних RTP-пакетів/HTTP-сегментів чи запитів; перестановка -зміна їхнього порядку; десинхронізація -стале розходження часових основ A/V або контрольного плану. На RTP-рівні це маніпуляції номерами послідовності та мітками часу; SRTP пом'якшує ризик через автентифікацію й ковзне вікно anti-replay, але за відсутності автентифікації або за значних збурень можливі «флікер» і випадіння кадрів [35]. У HLS/LL-HLS атака зводиться до нав'язування застарілих/переплутаних сегментів (порушення EXT-X-MEDIA-SEQUENCE тощо), що дає стрибки назад і A/V-десинхрон; дрібні часткові сегменти LL-HLS пришвидшують прояв помилок порядку. На транспорті 0-RTT TLS 1.3 підвищує ризик мережевого replay, тому його уникають на чутливих маршрутах; потрібні ідемпотентність і явне відхилення ранніх даних. У контурі доступу реплей зводиться до повтору викраденого токена; протидія -короткі TTL, вузька область дії та DPoP (прив'язка запиту до ключа клієнта) [36, 37]. Практичний мінімум захисту: авторизація до lookup'у в кеші, короткі TTL/швидка інвалідація маніфестів, автентифікований SRTP з налаштованим anti-replay і вимкнення 0-RTT на критичних шляхах.

Replay, перестановка та десинхронізація спираються на часові мітки тому ці мітки також є вектором загроз. Спуфінг часу/синхронізації. Спуфінг часу - нав'язування хибної часової опори або спотворення механізмів узгодження, що

визначають плейаут і A/V-синхронізацію. На медіарівні це маніпуляції RTP-мітками часу та RTCP Sender Reports, які зв'язують RTP-час із NTP-годинником; порушення відповідності викликає зсув буфера, джитер і стійкий A/V-дрейф [38]. На мережевому рівні підміна/ослаблення NTP дозволяє зміщувати системний час і опосередковано впливати на таймери плеєра та ABR [39]; у промислових мережах аналогічні атаки можливі на PTP (IEEE 1588) через модифікацію Sync/Follow_Up/Delay-повідомлень. Пом'якшення: перехід до NTS із автентифікацією, багатоджерельна синхронізація, обмеження темпу корекцій (slew-only); на медіарівні -автентифікація SRTP/SRTCP, anti-replay, перевірки правдоподібності RTCP (монотоніка/граничні відхилення) та використання монотонічних таймерів із консервативною ресинхронізацією [33].

Поговоривши про конфіденційність і цілісність, саме час поговорити про контроль доступу та автентифікацію. Програма повина бути впеанена хто до неї звертається. Важливим вектором є крадіжка та підміна токенів. У потоковій доставці токени (JWT, підписані URL/cookie) є «пропусками» до маніфестів і сегментів на рівні CDN; базова модель -короткоживучі підписані URL або cookie з перевіркою на edge до звернення до кеша. Підписані URL легко «засвітити» в журналах, історії та через Referer, тому для веб-клієнтів доцільніше використовувати підписані cookie та жорстко налаштовувати Referrer-Policy на сторінках відтворення. Для JWT критичні короткі TTL, сувора валідація iss/aud/exp/nbf і правильна перевірка підпису; для захисту від реплею застосовують DPoP (прив'язка запиту до приватного ключа клієнта) [34-36]. Політики доступу слід звужувати шляхом (path), за потреби -IP-умовою/аудиторією в підписаних політиках CDN [37]. Для cookie обов'язковими є прапорці HttpOnly, Secure, SameSite та розмежування доменної зони [38].

Підписані URL і підписані cookie -механізми аутентифікації доступу на рівні CDN, що додають до запиту криптографічний доказ права на отримання маніфестів і сегментів. URL включає підпис і строк дії, але чутливий до витоків у журналах, історії та заголовку Referrer; cookie переносять підпис у HTTP-заголовок і краще підходять для браузерних клієнтів, дозволяючи безпечно обслуговувати спільний

кешований об’єкт після перевірки на edge [10-11]. Безпека досягається коротким TTL, вузькою областю дії (path/ресурс; за потреби -умова IP), ротацією ключів, валідацією до звернення до кеша, а для cookie -прапорцями HttpOnly, Secure, SameSite. Якщо все ж використовуються підписані URL на веб-сторінках, Referrer-Policy має запобігати витоку токена у треті домени [38-41].

Не менш важливим є те яка саме реазіція вибрана, так звані вразливості JWT. JSON Web Token використовується як компактний носій атрибутів доступу (claims) з криптографічним підписом; коректність перевірки підпису та семантики полів визначає безпеку всієї схеми. Типові збої виникають, коли реалізація покладається на дані із заголовка замість довіреної конфігурації (помилки вибору алгоритму підпису, некоректна валідація ключа), або коли ігнорується семантика claims: не перевіряються iss/aud, обмеження часу дії (exp, nbf) і «свіжості» (iat), що прямо визначено специфікацією.[42] Наслідок -прийняття токенів поза призначеною аудиторією, у чужому домені довіри або після завершення строку дії. Окремий клас ризиків пов’язаний із реплеєм і неправильним зберіганням. Викрадений дійсний токен придатний для повторного використання, якщо відсутня прив’язка до контексту запиту/клієнта; додатково, зберігання у вебсховищі, доступному JavaScript, підвищує наслідки XSS. Практика керування сесіями вимагає коротких TTL, коректного встановлення аудиторії/видавця, а також використання HTTP-cookie із прапорцями HttpOnly/Secure/SameSite там, де це відповідає моделі загроз. Для зменшення наслідків викрадення й заборони реплею доцільно застосовувати прив’язку «токен + доказ володіння ключем» (DPoP): клієнт підписує кожен запит власним приватним ключем, і токен стає непридатним поза цим криптоконтекстом. У поєднанні з короткими TTL та суворою валідацією claims це істотно знижує імовірність зловживань без помітного впливу на затримку доступу [36, 43-44].

Також потрібно захистити самого користувача від захоплення чи підміна сесії. Захоплення (hijacking) -несанкціоноване використання чинного ідентифікатора або маркера доступу; підміна (fixation) -нав’язування жертві відомого зловмиснику ідентифікатора. Ключові причини: витік cookie/токенів (XSS, реферери), помилкова ізоляція доменної зони, повторне використання

маркерів без прив’язки до контексту, відсутність ротації після автентифікації [45]. Базовий бар’єр від мережевого перехоплення - TLS 1.3 та HSTS, що унеможливлюють даунґрейд і «чистий» HTTP до доменів відтворення й авторизації. Для клієнтів, що ініціюють WebSocket, потрібні перевірки Origin/Sec-WebSocket-Protocol, аби відсікти несанкціоноване повторне використання сесії під час upgrade [39]. На рівні доставки доцільно віддавати перевагу підписаним cookie (а не URL) з коротким TTL і перевіркою на edge до звернення до кеша; це мінімізує поверхню витоку (журнали, історія, Referer) і блокує «гарячі лінки» навіть за наявності об’єкта в кеші. Для JWT обов’язковими є сувора валідація підпису та claims (iss, aud, exp, nbf) і короткий строк дії; для протидії реплею доцільно застосовувати DPoP -прив’язку запиту до приватного ключа клієнта, що робить ізольований витік токена недостатнім для узурпації. Разом із ротацією ідентифікатора після логіну та сегрегацією доменної зони це формує практичний мінімум протидії захопленню/підміні сесії [45].

Коли у зловмисника не вийшло отримати дані. Він може не дати даним потрапити за адресою або не дати користувачеві відправити чи отримати дані ,тобто атакувати інфраструкту і доступність, так звані DoS / DDoS. Відмова в обслуговуванні у потокових системах проявляється як волюметричні мережеві атаки, так і протокольні виснаження критичних HTTP-ендпойнтів (маніфести .m3u8, ресурси ключів EXT-X-KEY), чия недоступність миттєво зупиняє відтворення. Окремий практичний вектор -cache poisoning / CPDoS: отруєний маніфест/сегмент або закешована помилкова відповідь (напр., 404/301) масштабовано розповсюджуються через CDN, перетворюючи локальну ін’єкцію на широке «падіння» сервісу [32, 46-48]. У площині сигналізації (HTTP/WebSocket) характерні лавини коротких підключень/upgrade-запитів, що вичерпують дескриптори та черги, блокуючи доступ до медіа навіть за наявності пропускної здатності [42].

Додатково загрозу підсилюють peer-assisted накладки (pollution/Sybil) і IoT-ботнети (типу Mirai), здатні генерувати значні L3/L4-потоки проти edge та origin одночасно.

Пом’якшення спирається на поєднання: дисципліни кешу (нормалізація ключа, мінімальний Vary, заборона негативного кешування, авторизація до lookup’у), коротких TTL маніфестів і no-store для ендпойнтів ключів, а також ресурсного захисту сигналізації (rate-limit, back-pressure, окремі пули/таймаути). Провайдери кешів надають прямі настанови щодо нейтралізації cache-poisoning-векторів, що значно знижує ризик масштабованого DoS через CDN [53].

Протокольні виснаження (ключовий ендпойнт, маніфести, RTCP/ABR).Під протокольними виснаженнями маються на увазі навмисні шаблони звернень, що примушують «дорогі» вузли виконувати непропорційно багато операцій. У HTTP-стримінгу найчутливіші маніфести та ключовий ендпойнт: перші керують чергою сегментів, другий -розшифруванням контенту.

У LL-HLS часті блокувальні оновлення плейлистів збільшують частоту запитів, тож імітація «масового live-трафіку» здатна створити навантаження на origin/edge [24]. Перевірка підписаних URL/cookie на edge до звернення до кеша відсікає неавторизовані запити та не допускає ескалації тиску в глибині тракту. Для ключового ендпойнта характерні «бурсти» авторизацій, що тригерять верифікацію підписів і виклики KMS/DRM; протидія -заборона кешування ключів, короткі таймаути, нормалізація ключа кешу, мінімальний Vary і відмова від негативного кешування на критичних шляхах.

У площині RTCP/ABR хвилі фідбеку (RR/PLI/NACK) і спровоковані коливання мережі спричиняють часті перемикання профілів, множачи звернення до маніфестів/сегментів; автентифікований SRTP/RTCP та консервативні пороги ABR знижують цей ефект [43, 46, 53].

Компрометація сигнального рівня також є проблемою. Сигнальний рівень відповідає за встановлення та супровід сесії: обмін SDP-описами (offer/answer), атрибутами безпеки й ICE-кандидатами; у браузерних стек-сценаріях це реалізується через HTTP(S)/WebSocket за моделлю JSEP, у телеком-середовищах - через SIP із передаванням SDP у повідомленнях сигналізації. Ключовий ризик - ін’єкція/підміна сигналізаційних повідомлень, що дозволяє змінити a=fingerprint або ICE-кандидати й тим самим нав’язати MITM навіть за наявності SRTP/DTLS

на медіаканалі. Базові запобіжники: захищений транспорт TLS 1.3 із політикою HSTS для недопущення даунґрейду [48]; для WebSocket -перевірки Origin / Sec-WebSocket-Protocol та відсікання неочікуваних крос-сайтових апґрейдів [39]. Автентифікацію викликів доцільно виконувати короткоживучими JWT із суворою валідацією iss/aud/exp/nbf, а на рівні логіки -контролювати монотоніку/стани обміну (offer/answer, ICE-перевірки), щоб унеможливити застарілі або змішані описи.

Сукупність цих заходів відновлює довіру до сигналізації та мінімізує імовірність нав’язаних параметрів сеансу.

Несанкціонований доступ до ендпоінтів (IP-камери / IoT). IP-камери та суміжні IoT-пристрої часто вразливі через дефолтні облікові записи, вразливості прошивок (у т.ч. RCE), автоматичний проброс портів і надмірно довірені хмарні інтеграції. Практика показувала масові відкриті панелі/потоки в Інтернеті та інциденти з компрометацією хмарних консолей виробників; відомі й критичні CVE, що відкривають доступ без автентифікації . Такі вузли нерідко потрапляють до ботнетів і підсилюють DDoS-ризики . Системні огляди ринку фіксують типові огріхи: слабкі паролі, відсутність підписаних оновлень, відкриті служби в LAN/WAN [54-56].

Запобігання передбачає: обов’язкову зміну дефолтних облікових записів, вимкнення непотрібних служб (RTSP/ONVIF/UPnP), регулярні підписані оновлення; мережеву сегментацію (VLAN, deny-by-default), доступ до потоків лише з внутрішньої мережі (VPN для віддаленого), allowlist egress до хмар постачальника; на хмарних панелях -MFA, мінімальні привілеї, аудит доступів; додатково TLS/DTLS там, де підтримується, і моніторинг відомих CVE для конкретної моделі[54, 57-59].

Останім про що ще не говорилось вище, але це є не меньш важливим за інше є криптографія та керування ключам це захищає дані у випадку якщо зловмиснику все ж вдалось отримати дані.

Є багато методів крипто захисту даних, далі короткий опис основних методів та слабкостей:

– повтор IV/nonce. Повтор однієї й тієї ж пари ⟨ключ, nonce/IV⟩ руйнує конфіденційність (а в AEAD і цілісність). Для AES-GCM це дає «two-time pad» і підробку тегів; у SRTP (AES-CTR) повтор призводить до XOR-розкриття кадрів; у HLS (AES-128-CBC) фіксований/повторний IV відкриває кореляції перших блоків; у DASH/CENC (CTR/CBCS) IV має бути унікальним на семпл і ключ [60];

– слабка/відсутня ротація (ephemeral/crypto-periods). Довге використання одного ключа збільшує збиток від витоку й ризики повтору IV/nonce. Практика - короткі криптоперіоди (у секундах/сегментах) і регулярна ротація ключів, які стандартно сигналізуються в профілях захисту CENC/CPIX [61-64];

– витік ключів / KMS / DRM. Експозиція CPIX-артефактів, помилки доступу до KMS/DRM-ендпойнтів або логів дають змогу стороннім розшифровувати контент. Мінімум захисту: жорсткий контроль доступу до ендпойнтів ключів, «no-store» і короткі TTL на відповіді, сегрегація секретів і регулярна ротація, як передбачено моделями CENC/CPIX [63];

– даунґрейд / слабкі шифри. Примус перехід на застарілий протокол/набори шифрів (або їх підтримка за замовчуванням) знижує стійкість усього тракту. Рекомендовано TLS 1.3 з сучасними AEAD-сюїтами та політиками, що забороняють небезпечні алгоритми і даунґрейд-атаки [58];

– помилки інтеграції AES-CTR/GCM/CBCS. Типові огріхи: невірне формування nonce/IV у GCM (навіть один повтор критичний), збої індексації в CTR/SRTP, змішування режимів і ключів між треками/якісними варіантами у CENC/CBCS, некоректне задання IV у HLS. Усі ці помилки знімають гарантії індистингвішабельності та автентичності [59, 63, 64].

1.4 Постановка задачі

У підсумку проведеного в розділі огляду встановлено, що безпека потокового відеотрафіку визначається не стільки вибором окремого криптографічного алгоритму, скільки погодженою роботою всієї системи доставки. Архітектурні

рішення щодо сегментації та буферизації, логіка адаптивного бітрейту, надійність сигнального рівня, а також властивості ланцюга розповсюдження через CDN і peer-assisted накладки створюють самостійну поверхню ризику, яка безпосередньо впливає на якість сервісу. Навіть за коректного застосування сучасних протоколів шифрування зберігається інформативність побічних ознак трафіку; крім того, конфігураційні спрощення на кшталт тривалих TTL, слабких профілів узгодження або «довгих» підписаних URL уможливлюють обхід формальних гарантій захисту.

Порівняльний аналіз показав, що застосування TLS 1.3/DTLS-SRTP, автентифікованого RTCP і дисципліни керування ключами суттєво звужує можливості підміни та маніпуляцій на медіашляху. Водночас виявлено класи загроз, які не усуваються суто криптографічними засобами. До них належать аналіз побічних характеристик зашифрованого трафіку (ідентифікація контенту та інференція QoE за таймінгами й обсягами), ін’єкції та отруєння елементів «контрольного плану» (маніфести, плейлисти, сегменти), підміна або крадіжка токенів і підписаних URL, а також протокольні виснаження, що призводять до деградації якості без явного «злому» даних. Окремо зафіксовано чутливість систем до часових і синхронізаційних атак: спотворення NTP/PTP або неавтентифікований контроль через RTCP здатні викликати десинхронізацію аудіо-відео та лавину службових подій.

Суттєвим чинником ризику виявляється залежність від зовнішньої інфраструктури. CDN-кеші за певних умов масштабують локальну ін’єкцію до масового інциденту, а пірингові накладки підвищують ефективність доставки ціною уразливості до pollution- та Sybil-атак. Джерела відео (передусім IP-камери/IoT) часто залишаються найслабшою ланкою через типові помилки конфігурації, застарілі прошивки та відкриті інтерфейси керування.

На цьому тлі стає очевидною потреба у комбінованому підході, який поєднує вибіркове (стратифіковане) шифрування з керуванням «видимим силуетом» трафіку, укріпленням сигналізації та контрольного плану, загальною корпоративною дисципліною керування ключами, а також операційними політиками на рівні CDN і джерел відео.

Отримані результати мотивують перехід від констатації проблем до розроблення узгодженої архітектури, що забезпечує прийнятний компроміс між безпекою та якістю відтворення. Очевидно, що подальші кроки мають включати формалізацію вимог і метрик (мережевих, якісних і безпекових), проєктування механізмів адаптації під змінні умови каналу, створення прототипу у репрезентативному стеку та його експериментальну верифікацію у «шумних» сценаріях і під атаками. Така траєкторія дозволить перевести виявлені у розділі закономірності у конкретні технічні рішення й дати кількісну оцінку їхнього внеску.

Як було зазначено, метою кваліфікаційної роботи є розроблення та обґрунтування комбінованого методу захисту потокового відеотрафіку в інформаційно-комунікаційній системі, який знижує інформативність побічних ознак трафіку та водночас зберігає прийнятну якість відтворення в реалістичних мережевих умовах.

Для досягнення мети кваліфікаційної роботи необхідно послідовно виконати такі завдання:

- провести огляд літератури та стандартів, узагальнивши сучасні підходи до захисту потокового відео й окресливши їхні обмеження;
- побудувати модель загроз із розмежуванням пасивних, активних та інфраструктурних атак і з фіксацією меж довіри;
- визначити критерії якості та безпеки (метрики QoS/QoE, «бюджет затримки», показники стійкості до аналізу зашифрованого трафіку) як основу подальшого оцінювання;
- спроєктувати комбінований підхід до захисту, що поєднує криптографію з керуванням формою передавання та укріпленням сигналізації й «контрольного плану»;
- побудувати математичну модель комбінованого методу захисту потокового відеотрафіку в інформаційно-комунікаційній системі;
- розробити комбінований метод захисту потокового відеотрафіку в інформаційно-комунікаційній системі;

- реалізувати робочий прототип методу у вибраному стеку (WebRTC із SRTP/DTLS або DASH/HLS поверх TLS) із вбудованою телеметрією для збору мережевих, якісних і безпекових показників;
- провести експерименти у реалістичних умовах з втратами, джитером і реордерингом, змоделювавши типові атаки (аналіз зашифрованого трафіку, ін’єкції/підміни, replay/перестановка, протокольні виснаження);
- провести оцінку методу, підбити підсумки на основі порівняльних вимірювань і абляційного аналізу, сформувавши практичні рекомендації та межі застосовності для впровадження у виробничих умовах.

Підсумовуючи, огляд підтвердив системний характер безпеки потокового відео: вирішальною є узгоджена робота сегментації та буферизації, керування ключами, контролю доступу й політик кешування, а не вибір окремого алгоритму шифрування.

Уточнені критерії якості та безпеки, разом із виявленими класами загроз, окреслюють вимоги до комбінованого методу, що знижує інформативність побічних ознак і водночас зберігає QoS/QoE.

Таким чином, можемо перейти до побудови математичної моделі та методу токено-керованої доставки.

2 ПОБУДОВА МАТЕМАТИЧНОЇ МОДЕЛІ ТА МЕТОДУ ТОКЕНО-КЕРОВАНА ДОСТАВКА ЗАШИФРОВАНИХ СЕГМЕНТІВ

2.1 Концепція методу Токен-керована доставка зашифрованих сегментів

З поперднього розділу стає зрозуміло що для потокового відео недостатньо ізольованих криптографічних засобів; потрібна узгоджена організація контролю доступу та процесів доставки, яка не руйнує затримковий бюджет і не вимагає важкої інтеграції.

Метод «Токен-керована доставка зашифрованих сегментів» (ТДЗС) покликаний забезпечити практично досяжний рівень захисту потокового відеотрафіку без важких клієнтських інтеграцій і без зниження якості відтворення понад прийнятний затримковий бюджет. Концепція спирається на зрілі індустріальні стандарти (HLS/DASH, TLS 1.3, типові механізми контролю доступу на краю CDN) і поєднує криптографічний захист даних з дисципліною керування доступом і виваженою політикою кешування [60, 65, 66].

Суть ТДЗС полягає у розділенні «контенту» і «права доступу» та їхньому незалежному контролі. Відео доставляється у вигляді зашифрованих сегментів; плейлист містить лише декларацію параметрів розшифрування (посилання на ендпойнт ключа та ідентифікатор криптоепохи), але не передає секретів. Доступ до маніфесту й сегментів дозволяється лише за наявності короткоживучого підтвердження (підписаної cookie/URL), яке перевіряється безпосередньо на краю CDN. Весь транспорт здійснюється по HTTPS, а ключі ніколи не кешуються і видаються виключно через ізольований, некешований ендпойнт [43, 46].

Архітектурно метод складається з трьох узгоджених контурів. Контур даних охоплює шифрування на рівні сегментів (AES-128 / SAMPLE-AES або CENC для fMP4/CMAF) з унікальними IV і ротацією ключів за принципом «дрібних епох»; межі епох узгоджуються з межами сегментів/IDR, що робить ротацію прозорою для клієнта [25]. Контур керування забезпечує авторизацію й цілісність «контрольного плану»: токен-керований доступ на краю, TLS 1.3 скрізь, за необхідності - підпис/верифікація плейлистів. Контур доставки та кешування мінімізує площу

ін’єкцій і розсинхронізації: короткі TTL для маніфестів, помірне кешування зашифрованих сегментів, no-store для ключів і нормалізація ключа кешу [53].

Межі довіри визначаються так, щоб пасивний спостерігач бачив лише зашифрований трафік, а активний порушник не міг обійти перевірку доступу через кеш чи транспорт. Джерела відео та origin сприймаються як довірені у сенсі цілісності контенту; край CDN - як керований виконавець політики; клієнт - як потенційно уразлива, але специфікаційна реалізація. У цій моделі загроз ТДЗС прагматично закриває пасивне прослуховування, «гаряче» лінкування та значну частину інфраструктурних векторів (отруєння кеша, MITM на транспорті), залишаючи поза своїм базовим мандатом захист від копіювання після декодування на скомпрометованому клієнті та повне «затирання» побічних ознак трафіку - ці задачі покриваються опційними розширеннями [52].

Функціональний цикл виглядає так. Енкодер формує потік, пакувальник створює сегменти фіксованої тривалості та плейлист із декларацією нового ключа на межі епохи. Клієнт отримує токен доступу (підписана cookie/URL) і звертається до маніфесту та сегментів; край CDN перевіряє підпис і параметри політики (часове вікно, префікс шляху тощо), після чого віддає об’єкт із кеша або з origin [67]. Коли наближається межа епохи, клієнт завчасно запитує новий ключ через окремий HTTPS-ендпойнт; ключ ніколи не кешується, а його видача відбувається лише за валідної сесії. Така декомпозиція зберігає безперервність відтворення та локалізує наслідки можливих збоїв.

Параметри методу добираються з урахуванням затримкового бюджету й навантаження. Визначальними є тривалість сегмента/частини сегмента, глибина стартового буфера та політика ABR; саме вони формують «зернистість часу» і обумовлюють енд-ту-енд затримку. Допоміжні параметри - час життя токена, довжина криптоперіоду, TTL маніфестів і сегментів - впливають на корисність викрадених атрибутів і на чутливість системи до ін’єкцій, але додають мілісекундні накладні за правильно налаштованим TLS. Концепція ТДЗС сумісна як із «звичайним» HLS (2-4 с сегменти), так і з LL-HLS (часткові сегменти 0,5-1,0 с і блокувальні оновлення), що дозволяє досягати низької затримки без зміни

безпекової логіки.

Ключова перевага підходу - впроваджуваність. ТДЗС не вимагає DRM-модулів чи кастомних плеєрів, покладаючись на спроможності пакувальника, стандартні можливості CDN і типові веб-засоби захисту[46, 60]. Таким чином зменшується ризик інтеграційних помилок і вартість експлуатації: основні політики виконуються на краю, ключі видаються з одного, чітко контрольованого ендпойнта, кеш поводиться передбачувано. Водночас базові обмеження підходу прозорі й керовані: для посилення стійкості до аналізу побічних ознак трафіку можна додати легке маскування форми (квантований паддинг, обмежена рандомізація інтервалів), а для підвищеного рівня довіри - підпис/верифікацію маніфестів. За результатами роботи, праведена апробація [68].

Очікуваний ефект вимірюється як зниження ймовірності несанкціонованого доступу через URL-витоки та кеш, скорочення корисності викраденого токена до його короткого вікна дії, локалізація наслідків потенційного витоку ключа однією криптоепохою та підтримання QoE в межах заданого SLA. Це створює стійку основу для подальшого прототипування й емпіричної перевірки: у підпункті 2.1 детально обґрунтовуються обрані рішення, у 2.2 подано математичну модель і критерії оптимізації параметрів, а наступні підрозділи розкривають методики експериментальної верифікації.

2.2 Обґрунтування запропонованих рішень

Шифрування на рівні сигментів. У межах HLS шифрування реалізується на рівні окремих сегментів тривалістю 2-4 секунди за допомогою механізму EXT-X-KEY. Плейлист містить лише вказівник на ключ (URI) і параметри шифрування, тоді як сам ключ передається окремим ендпойнтом із власною політикою доступу. На практиці застосовують два режими: AES-128 (повне шифрування транспортного сегмента) та SAMPLE-AES (шифрування вибіркових семплів у контейнері, зокрема у fMP4/CMAF). У простішій і найпоширенішій конфігурації

AES-128 забезпечує конфіденційність «цілих шматків» відео з мінімальним накладом, адже симетричне шифрування сегмента - це одна операція на блоки, добре паралелізована сучасними енкодерами. Важлива деталь - унікальність вектора ініціалізації (IV) для кожного сегмента: у HLS його або явно задають через атрибут IV у EXT-X-KEY, або визначають детерміновано зі стартового номера сегмента. Уникнення повторів IV під тим самим ключем унеможливлює відомі компрометації режимів блокового шифру та зберігає стійкість на довгих трансляціях.

Відокремлення каналу ключів від каналу сегментів має і безпекові, і експлуатаційні переваги. Ключі видаються через окремий некешований ендпойнт по HTTPS лише після перевірки короткоживучої підписаної cookie/токена; це дозволяє застосувати суворіші правила доступу (TTL, прив’язка до префіксів шляху, за потреби - до підмережі), не торкаючись основної доставки відео. Сам плейлист і сегменти можуть коротко кешуватися на CDN - конфіденційність зберігається шифруванням, а ризик несанкціонованого доступу знижується, оскільки секрет ніколи не з’являється ані в маніфесті, ані в URL сегментів. Така декомпозиція також зменшує наслідки випадкових помилок: навіть якщо кеш помилково роздаватиме застарілий плейлист, ендпойнт ключа залишиться контрольованим і не видасть секрет без чинної авторизації.

Для підтримки адаптивного перемикання бітрейтів (ABR) ключі і політика IV мають бути узгоджені між варіантами. Типова практика - використовувати спільний ключ і однакову схему IV для всіх варіантів, вирівняних по межах сегментів та опорних кадрах; тоді клієнт може безперервно перемикатися між доріжками без додаткових запитів ключів і без ризику десинхронізації. У live-сценаріях доцільно застосовувати ротацію ключа малими епохами (за часом або кількістю сегментів): плейлист просто публікує нову декларацію EXT-X-KEY з іншим URI/KEYID на межі епохи, а клієнт прозоро запитує новий ключ наперед у буферному вікні. Така дисципліна обмежує наслідки навіть у разі локального витоку та водночас зберігає непомітність ротації для користувача.

Слід усвідомлювати, що базове шифрування сегментів у HLS забезпечує

насамперед конфіденційність, тоді як доказова цілісність залежить від транспорту й політик. У «легкому» варіанті ТДЗС її гарантують HTTPS-канал, відмова від кешування ключів і контроль доступу на краю CDN. Для сценаріїв з підвищеними вимогами можливо додати перевірку цілісності «контрольного плану»: підписувати/верифікувати MPD/M3U8 або застосувати в fMP4/CMAF схеми заголовків із ідентифікаторами ключів (KEYID) та узгодженою ротацією. Це не змінює основної логіки методу - шифрування залишається на рівні сегментів, а безпека ключів і політика їх видачі централізовано керуються окремим ендпойнтом.

Обмеження доступу. У центрі рішення лежить принцип «token-gating» на краю CDN: запит до маніфестів і сегментів допускається лише за наявності короткоживучого підтвердження доступу, яке перевіряється безпосередньо на edge-вузлі. Як носій підтвердження доцільно використовувати підписані cookie, адже вони не експонуються у рядку запиту й типовим чином не потрапляють у реферери чи аналітику. Підписаний URL, хоч і функціонально рівноцінний, є менш бажаним через ризик «слідів» у журналах і сторонніх сервісах. Формально політика характеризується двома основними параметрами: часом життя токена θ та областю дії P, яку зручно задавати як множину префіксів шляху (наприклад, каталог конкретної трансляції) із додатковими умовами, де припустимо, - підмережа джерела або автономна система. Зменшення θ лінійно знижує очікувану корисність викраденого токена, а звуження P мінімізує наслідки його несанкціонованого застосування поза призначеною доріжкою.

Практична перевірка на краю складається з криптографічної валідації підпису та семантичної перевірки політики. Підпис гарантує незмінність полів токена та його походження від довіреного емітента, тоді як політика визначає, чи збігаються цільовий шлях запиту з дозволеним префіксом, чи не минув термін дії за формолою:

$$t_{now} \leq t_{issue} + \theta \qquad (2.1)$$

і чи виконуються додаткові обмеження (наприклад, збіг домену, варіанту маніфесту або профілю бітрейту). Щоб уникнути хибних відмов через розбіжність годинників між клієнтами та edge-вузлами, у перевірку вводять запас на розсинхронізацію σ (кілька секунд), однак його величина має бути мінімальною, інакше σ фактично подовжує придатність токена. У штатному режимі поновлення токена відбувається «безшовно»: клієнт отримує нову cookie в межах того самого домену до закінчення поточного вікна θ, що знімає потребу в додаткових запитах під час відтворення.

Важливою є гігієна видимості секретів. Щоб виключити побічні витоки, сторінки з плеєром і маніфестами мають оголошувати сувору політику реферера (наприклад, no-referrer або щонайменше strict-origin-when-cross-origin), а формати журналів на балансерах і бекендах не повинні зберігати повні URL із параметрами доступу. На рівні cookie бажано застосовувати атрибути HttpOnly і Secure, а також коректний SameSite відповідно до шаблону навігації плеєра; це унеможливлює доступ сценаріїв до токена та зменшує площу XSS-ризику. У конфігураціях, де cookie технічно важко доставити на CDN-домен (наприклад, коли CDN працює на власному FQDN без CNAME під клієнтський домен), допустимим компромісом лишається підписаний URL, але тоді особливої ваги набувають короткий θ, вузька P та агресивне «обрізання» реферера.

Ще одна практична деталь стосується «крихких» обмежень за IP-адресою. Прив’язка токена до конкретної адреси у мобільних і корпоративних мережах із NAT та частою зміною вихідних маршрутів може штучно збільшувати частоту відмов. Тому прив’язка за джерелом доцільна лише в стабільних середовищах; у публічних мережах краще обмежуватися префіксами шляху й короткими вікнами дії, а для підвищення надійності використовувати механізм «м’якої деградації»: у разі невідповідності джерела запит тимчасово переводиться на менш чутливий профіль віддачі з обмеженими варіантами бітрейту, що дає змогу завершити відтворення сегмента й прозоро оновити сесію.

Політика доступу повинна бути узгоджена з режимами кешування. Ключі шифрування не кешуються принципово, а відповіді ендпойнта ключів

позначаються як no-store, що унеможливлює «застигання» секретів на краю. Маніфести отримують короткі TTL і механізм швидкої інвалідації; це локалізує вплив потенційних ін’єкцій і синхронізує видачу нових декларацій ключів або варіантів доріжок. Сегменти, навпаки, можуть кешуватися короткий час, адже їхня конфіденційність гарантується шифруванням, а доступ без дійсного токена залишається заблокованим. На етапі побудови ключа кешу слід явно нормалізувати заголовки та параметри, аби рішення edge й origin збігалися; інакше з’являється простір для отруєння кеша за рахунок неконсистентної ключової функції.

Нарешті, система доступу має коректно поводитися в аномальних режимах. Перевірка токенів доповнюється лімітами на частоту запитів і кількість одночасних потоків на одну сесію, а також короткими «карантинними» правилами для нетипових патернів (раптові стрибки частки 4xx/5xx на краю, багаторазові звернення до ендпойнта ключів поза очікуваним вікном θ). Такі обмеження не змінюють семантики «token-gating», але запобігають зловживанням і виснаженню інфраструктури. У підсумку токен-керована політика доступу на краю CDN не лише робить «гарячі» посилання неефективними, а й створює керований, формально параметризований бар’єр для підміни та неконтрольованого поширення контенту, зберігаючи при цьому простоту інтеграції з типовими програвачами й пайплайнами пакування.

Вибір TLS як транспортної основи. У межах методу ТДЗС TLS виконує подвійну функцію: гарантує конфіденційність/цілісність каналу та унеможливлює «тихе» перехоплення токенів доступу на шляху до маніфестів, сегментів і ендпойнта ключів. За відсутності TLS схема «token-gating» втрачає сенс: підписані cookie/URL легко знімаються пасивним спостерігачем або проксі, а будь-які спроби обмежити токен політиками не мають значення, якщо саме підтвердження доступу опиняється «на дроті». Тому всі запити до MPD/M3U8, сегментів і /key жорстко виконуються по HTTPS; на рівні клієнтських сторінок доцільно вмикати HSTS (із префіксом includeSubDomains, а за можливості - і preloading), аби браузер узагалі не ініціював з’єднань у «чистому» HTTP і не підпадав під даунґрейд-атаки.

Конфігураційно метод вимагає TLS 1.3 з AEAD-шифрами (AES-GCM або

ChaCha20-Poly1305) та обов’язковим ECDHE для пружної прівацийності (PFS). Наслідувані протоколи (TLS 1.0/1.1) і застарілі шифри вимикаються, як і неавтентифіковані режими. Особливу увагу слід приділити 0-RTT (early data): для статичних сегментів це може зменшити латентність, однак для ендпойнта ключів і авторизаційних маршрутів 0-RTT слід відключати через ризик повторного відтворення (replay). Сеансові квитки (session tickets) потрібно ротувати з помірним строком життя, щоби зберегти баланс між продуктивністю та конфіденційністю.

На практиці TLS найчастіше термінується на краю CDN, де й виконується перевірка підписаних cookie/URL. Важливо забезпечити «кінець-до-кінця» захист і між CDN та origin’ом: TLS на «ориджин-пулі» (за потреби - mTLS) прибирає вікно, де трафік міг би бути прочитаний або змінений. Варто перевірити, що заголовки, які включено до підпису або від яких залежить рішення про доступ, проходять до origin незмінно (уникаючи модифікацій проксі), інакше виникнуть як хибні відмови, так і можливості для отруєння кеша через розділення логік edge/origin.

У браузерному середовищі TLS підсилюється політиками сторінки, щоб зменшити побічні витоки. Плеєр/маніфест слід віддавати з Referrer-Policy: no-referrer (або, мінімум, strict-origin-when-cross-origin), аби підписані URL (якщо вони використовуються) не «переїжджали» в реферери сторонніх доменів. Для cookie з токенами необхідні атрибути Secure і HttpOnly, а також коректний SameSite відповідно до навігації плеєра. Це блокує доступ JavaScript до секретів та істотно звужує площу XSS-екстракції навіть за компрометації клієнтського коду.

Перехід на HTTP/2 і HTTP/3 (QUIC) є бажаним із погляду продуктивності, але не змінює безпекових вимог методу: у HTTP/3 TLS 1.3 «вбудовано» в QUIC, тому всі наведені політики лишаються чинними. Слід забезпечити однаковість правил доступу й заголовків безпеки для H2/H3 (ALPN, alt-svc), уникати змішаного вмісту (mixed content) у плеєрі та перевірити, що редіректи з HTTP→HTTPS виконуються одразу на краю, не допускаючи проміжних «сірого» контенту.

Операційно важливо автоматизувати життєвий цикл сертифікатів (ACME, короткі строки дії, OCSP-stapling) та моніторити невідповідності ланцюга довіри. Для мобільних/настільних застосунків із підвищеними вимогами доцільно

розглянути пінінг сертифікатів/ключів у клієнті, розуміючи, що це потребує акуратної ротації та плану аварійного відновлення. У вебі пінінг на рівні заголовків нині не рекомендують; натомість корисно відстежувати публікації у Certificate Transparency та налаштувати алерти на «чужі» сертифікати для домену.

Підсумовуючи, TLS у ТДЗС - не «декоративний» шар, а обов’язкова умова коректності всієї схеми. Саме він робить беззмістовним пасивне перехоплення сегментів і токенів, гарантує незмінність керувальних даних на шляху, а разом із HSTS/Referrer-Policy та атрибутами cookie знімає більшість побічних витоків. За правильної конфігурації витрати на TLS мізерні порівняно з виграшем у безпеці, а взаємодія з CDN та origin не ускладнюється понад стандартну інженерну практику.

Окремо обґрунтовується політика кешування. Вихідним принципом є розділення об’єктів за критичністю: ключі шифрування не підлягають кешуванню за жодних умов, маніфести (M3U8/MPD) кешуються лише короткочасно із пріоритетом швидкої переінвалідації, тоді як сегменти (TS/fMP4) можуть кешуватися помірний час, оскільки їхня конфіденційність забезпечена самим шифруванням. Така стратифікація напряму випливає з моделі загроз: витік ключа масштабується миттєво й робить беззмістовним захист сегментів; натомість витік окремого зашифрованого сегмента без ключа не несе корисної інформації для супротивника.

Для ендпойнта видачі ключів відповідь позначається політиками Cache-Control: no-store, no-cache та відсутністю заголовків, що активують проміжне збереження (ETag/Last-Modified). Це виключає як «позитивне» кешування секрету, так і негативне кешування помилок (4xx/5xx), яке могло б спричинити варіацію атаки CPDoS, коли помилкова відповідь закріплюється на краю і блокує легітимний доступ. На рівні CDN доцільно глобально вимкнути кешування будь-яких відповідей для шляху ключів і ввімкнути ґейт авторизації до lookup’у в кеш: спочатку перевіряється підписана cookie/URL, і лише потім приймається рішення щодо обслуговування запиту, не залишаючи «доріжок» для обхідного доступу через кеш.

Маніфестам (плейлистам) відводиться роль «контрольного плану», тому для

них встановлюється короткий TTL (секунди) у поєднанні з механізмами швидкої інвалідації (purge за мітками/групами). Такий режим локалізує ефект потенційних ін’єкцій або конфігураційних помилок, а також дозволяє безпечно публікувати зміни (оновлення варіантів бітрейту, початок нової «епохи» ключа). Рекомендується уникати агресивних стратегій на кшталт stale-while-revalidate для маніфестів: навіть коротка видача «застарілої, але свіжої» версії здатна посилати клієнта на сегменти, що вже недоступні або прив’язані до попереднього KEYID. Натомість ефективним є revalidation-біхевіор (ETag/If-None-Match) із малими дедлайнами, коли edge швидко підтверджує актуальність без повного перетягування вмісту.

Для сегментів доцільна протилежна логіка. Оскільки вони зашифровані й інваріантні для всіх авторизованих користувачів, помірні TTL (десятки секунд - кілька хвилин для VOD; коротше для live) істотно знижують навантаження на origin і стабілізують доставку під час струсу мережі. У live-сценаріях корисним є range-кешування для клієнтів, що виконують часткові запити, але його слід узгодити з кодуванням: сегменти бажано робити незалежними (independent_segments), без компресії рівня HTTP (щоб уникати варіантів за Accept-Encoding) і з фіксованим Content-Type. Так знімається потреба у Vary і запобігається фрагментації кеша за заголовками, які не повинні впливати на тіло відеооб’єкта.

Ключовим технічним елементом є нормалізація ключа кешу. Функція побудови ключа повинна включати лише стабільні компоненти шляху/запиту, що справді визначають відповідь, та ігнорувати заголовки/параметри, якими можна маніпулювати без зміни семантики (усі X-Forwarded-*, випадкові параметри відстеження, тощо). Для ТДЗС це особливо важливо, коли використовується підписана cookie: cookie «не» має потрапляти в ключ кешу (щоб не руйнувати спільність об’єкта), але перевірятися до lookup’у. Якщо ж застосовано підписані URL, токен-параметр потрібно вилучити з ключа кешу (щоб не отримати N копій одного сегмента) і одночасно перевірити авторизацію на краю до звернення до кеша, інакше з’явиться вікно для обхідних запитів.

Політика кешування повинна враховувати ротацію ключів. На межі «епохи»

у маніфесті з’являється нова декларація EXT-X-KEY з іншим KEYID/URI; у цей момент потрібно гарантувати, що старі маніфести оперативно інвалідовано, а сегменти нової епохи мають версійовані шляхи (або унікальні імена), щоб виключити колізії. Ключі при цьому не кешуються, тому сам факт ротації не накопичує «застарілі секрети» на краю. Для масштабних подій доцільно користуватися сурогатними ключами (мітками) на CDN: одна команда інвалідації «очищає» відразу всі маніфести певної трансляції, не торкаючись сегментів, які залишаються валідними та однаковими для різних клієнтів.

Окремої уваги потребує політика помилок і негативного кешування. За замовчуванням слід заборонити кешувати 4xx/5xx на шляхах маніфестів і ключів, а для сегментів - встановити коротку «нульову» тривалість для негативних відповідей (щоб не множити хибні 404 під час природного лагу появи сегментів на origin). Це знімає значну частину площі для Cache Poisoned DoS та споріднених векторів, у яких краю вигідно тримати помилку довше за життя контенту. У поєднанні з контрольованою інвалідацією це забезпечує керовану деградацію: клієнт швидко переходить на актуальний маніфест і не «залипає» на хибних станах.

Політика кешу має бути узгоджена з авторизаційним контуром. Перевірка токенів виконується перед зверненням до кеша, і лише у разі успіху дозвіл поширюється на віддачу спільного кешованого об’єкта. Таким чином, навіть якщо об’єкт присутній на краю, неавторизований запит не зможе обійти контроль доступу. Водночас, завдяки спільності кеша для всіх авторизованих, зберігається висока ефективність CDN і мінімізуємо кількість походів на origin.

Нарешті, кешування повинно бути підкріплене моніторингом і триґерами безпеки. Аномально високі частки 4xx для маніфестів, різкі стрибки «stale hit» або хвилі revalidate-запитів з одного регіону можуть слугувати ранніми індикаторами ін’єкцій, десинхронізації або спроб виснаження. Введення м’яких лімітів на частоту звернень до ендпойнта ключів, а також оперативна телеметрія інвалідацій дозволяють швидко локалізувати проблему, не змінюючи глобальних політик.

У такій конфігурації кеш виступає не лише механізмом продуктивності, а й компонентом безпеки: він розвантажує origin без розкриття секретів, обмежує

масштаб інцидентів за рахунок коротких життів і чіткої інвалідації «контрольного плану», а також усуває вектори отруєння завдяки нормалізації ключа та дисципліні обробки помилок. Саме ці властивості роблять політику кешування органічною частиною ТДЗС, а не другорядним тюнінгом.

Керування ключами.У ТДЗС «дрібні епохи» (короткі періоди дії робочого ключа) - це інженерна реалізація ідеї crypto-period: кожна обмежена за часом або кількістю сегментів ділянка потоку шифрується окремим ключем; межі між епохами узгоджені з межами сегментів/IDR-кадрів, щоб ротація була прозорою для програвача. Така дисципліна одночасно зменшує наслідки можливого витоку ключа (компрометація обмежується однією епохою) і виконує криптографічну «гігієну» режимів (уникання повторного використання вектора ініціалізації/лічильника з тим самим ключем). У специфікації CPIX це прямо оформлено як «multiple crypto-periods … for content encrypted using key rotation» (термінологічна опора для формалізації політики ротації та її сигналізації у виробничих пайплайнах).

У HLS ротація ключів реалізується плейлистом: кожен запис #EXT-X-KEY визначає, як розшифровувати всі наступні сегменти до появи наступного #EXT-X-KEY; методи шифрування - AES-128 (повне шифрування сегмента в AES-CBC із PKCS#7) та SAMPLE-AES (вибіркове шифрування семплів). Ключ постачається окремим ресурсом (URI), а IV для AES-128 або задається атрибутом IV, або детерміновано виводиться з Media Sequence Number, що гарантує унікальність IV по сегментах (і перезапуск CBC на межі кожного сегмента). Формально: «EXT-X-KEY … specifies how to decrypt [Media Segments]»; «AES-128 … with CBC … IV is either the IV attribute or the Media Sequence Number»; «Key file … 16-octet key in binary format». Це опорні вимоги для коректної ротації та унеможливлення повторів IV під тим самим ключем.

Для fMP4/CMAF у HLS та DASH застосовується Common Encryption (CENC): метадані містять Key ID (KID), а самі потоки шифруються у режимах cenc (AES-CTR) або cbcs (AES-CBC з pattern-шифруванням; типовий для FairPlay). Ротація реалізується як послідовність KID/ключів без перекриття часових інтервалів - саме

так формулюється в CPIX (ключі, що діють у різні періоди, не повинні накладатися), а ISO/IEC 23001-7 визначає уніфікований спосіб описати ці ключі й їх прив'язку у фрагментах файлу. Для нас це означає: нова епоха → новий KID/ключ, нові pssh/сигнали в маніфесті, і відсутність «сірих зон», де одночасно легальні два різні ключі.

Практично політика задається параметром к - тривалістю епохи. Рекомендується подвійне обмеження:

$$к \leq \min\{к_{час}, к_{сегм}\} \tag{2.2}$$

При цьому $к_{час}$(наприклад, 5-10 хв для live) обмежує «час під одним ключем», а ксегм (наприклад, 60-120 сегментів по 2-4 с) - «обсяг під одним ключем». Межу епохи вирівнюють по IDR/межі сегмента; у плейлисті завчасно публікують новий EXT-X-KEY, щоб клієнт встиг отримати ключ у буферному вікні. Старий ключ перестає видаватися одразу після переходу епохи (endpoint ключа - no-store), що мінімізує наслідки витоку. Така організація забезпечує прозоре відтворення та швидку локалізацію інцидентів без додаткових затримок. (Семантика EXT-X-KEY і правила IV/Key-file - за RFC 8216; поняття crypto-period і безперервні, неперекривні періоди - за CPIX/CENC).

Нарешті, дрібні епохи знижують системний ризик неузгодженостей. Якщо клієнт тимчасово «відстав» і бачить старий маніфест, він однаково не отримає ключ поза активним вікном; якщо edge-кеш із запізненням віддав попередню версію плейлиста, некешований ключовий ресурс не дозволить розшифрувати «нові» сегменти старим секретом. У сукупності з вимогами до структури ключового файлу («single packed array of 16 octets in binary format») та правилами IV у HLS це створює просту, але стійку дисципліну керування ключами для сегментного шифрування.

Затримковий бюджет. Уживатимемо таку декомпозицію затримки в реальному часі (glass-to-glass) (рис. 2.1).

$$\Delta_{e2e} = \underbrace{\Delta_{enc}}_{\text{кадрування/кодування}} + \underbrace{\Delta_{seg}}_{\text{сегментація/публікація}} + \underbrace{\Delta_{net}}_{\text{CDN/HTTP/мережа}} + \underbrace{\Delta_{plr}}_{\text{буфер плеєра/ABR}} + \underbrace{\Delta_{ctrl}}_{\text{TLS/авторизація/перевірки}}.$$

Рисунок 2.1 - Декомпозицію затримки

У потоках HLS домінуючим доданком зазвичай є $\Delta_{seg}+\Delta_{plr}$: сегментація створює «зерно часу», а клієнт відтворює із відставанням у кілька сегментів (див. нижче), тоді як інші доданки здебільшого другого порядку, якщо їх правильно налаштовано. Стандарт HLS формалізує цю «зернистість» через EXT-X-TARGETDURATION і нумерацію сегментів; саме Target Duration визначає часову шкалу плейлиста, яку бачить плеєр.

Сегментація та «правило трьох сегментів» Для звичайного HLS індустріальна практика старту відтворення - приблизно на три сегменти позаду «хвоста» плейлиста. Це створює запас безпеки проти джитеру/перекодувань, але водночас додає $\sim 3\times T_{seg}$ до Δ_{e2e}. Наприклад, за 6-секундних сегментів типовий запас становить ~18 с, що й пояснює «стандартні» 20-30 с для HLS-ефіру (до цього додаються мережеві та транскодерні накладні). Для коротших сегментів (2 с) теоретичний мінімум без урахування мережі/інгесту - ~6 с.

Apple у своїй авторській специфікації для пристроїв прямо рекомендує номінальні тривалості сегментів ~6 с (історично для сумісності та стабільності), що добре працює для VOD і «звичайного» live, але природно підвищує Δ_{e2e}. Якщо мета - знизити затримку без зміни протоколу, практичний компроміс - 2-4 с із незалежними сегментами; однак ці значення мають узгоджуватися з кодуванням (GOP/IDR) і кешуванням, аби не «розгойдувати» ABR.

LL-HLS і часткові сегменти (parts). Щоб зменшити саме «зерно часу», Apple ввела Low-Latency HLS: мова не про просто коротші сегменти, а про часткові сегменти (parts) субсекундної-секундної тривалості, блокувальні оновлення плейлиста та «preload hints», завдяки чому клієнт отримує дані ще до завершення повного сегмента. У LL-HLS з’являється параметр Part Target Duration (рекомендовано близько 1 с), а клієнтська PART-HOLD-BACK має бути не менш як 2× і бажано 3× цієї величини - тобто орієнтир латентності стає $\approx 2\text{-}3\times T_{part}$. На

практиці це дає менше 2 секунд, на масштабі Інтернету за коректної інтеграції CDN.

Буфер плеєра та ABR. Частка Δ_{plr} визначається політикою буфера: скільки контенту клієнт накопичує «уперед». Для «звичайного» HLS саме це й породжує «три сегменти позаду». Для веб-плеєрів важливо контролювати параметри буфера, інакше відтворення може мати надмірну глибину (30 с і більше), що збільшить затримку навіть за коротких сегментів - поведінку подібну до описаної для hls.js (де типові налаштування переднього/зворотного буфера впливають на QoE і латентність).

Мережа, CDN і періодичність оновлень. Δ_{net} зменшується за рахунок блокувальних оновлень плейлиста (клієнт «чекає» появи нового сегмента/частини замість частого опитування) і підказок попереднього завантаження; це прибирає зайві RTT на «холості» перезавантаження та прискорює доставку щойно доступних даних. Для LL-HLS Apple офіційно описує і blocking reload, і preload hints, які зменшують дорожні витрати на шляхах «плейлист-клієнт». Для DASH існує функціонально подібний підхід: Low-Latency DASH з chunked-передачею CMAF-фрагментів (client починає завантаження до завершення сегмента)

TLS/авторизація і їхня частка в бюджеті. Δ_{ctrl} мінімізують правильною конфігурацією TLS 1.3: 1-RTT-рукостискання зменшує час до першого байта проти TLS 1.2 (2-RTT), але 0-RTT уникають на «чутливих» маршрутах (ендпойнт ключів, авторизація) через ризики повторного відтворення (replay). Отже, внесок TLS у бюджет роблять близьким до нуля після встановлення сесії, не жертвуючи безпекою.

Для методу «токен-керована доставка зашифрованих сегментів» затримковий бюджет визначається передусім вибором тривалості сегмента/частини та політикою буфера. Шифрування сегментів і перевірки доступу додають мікросекундні-мілісекундні витрати порівняно з $\Delta_{seg}+\Delta_{plr}$. Отже, практичний рецепт: для «звичайного» live - 2-4 с сегменти з відставанням у 2-3 сегменти; для інтерактивних сценаріїв - LL-HLS з 0.5-1.0 с parts і PART-HOLD BACK 2×2\times2×-3×3\times3×, плюс blocking reload і preload hints. Це тримає Δe2e

у заданому SLA без конфлікту з контуром шифрування/доступу.

Вектори захисту. З погляду моделі загроз ТДЗС закриває декілька головних векторів за мінімальної вартості. Передусім метод усуває базовий ризик пасивного прослуховування: медіадані на транспорті віддаються у вигляді зашифрованих сегментів HLS, а плейлист містить лише декларацію EXT-X-KEY з URI на окремий ендпойнт видачі ключа. Навіть якщо зловмисник перехопить маніфест чи сегменти, без ключа вміст залишиться непридатним; водночас у HLS прописано, що сегменти вважаються зашифрованими лише за наявності EXT-X-KEY, а параметри тривалості/послідовності сегментів задаються стандартними тегами (EXT-X-TARGETDURATION, EXT-X-MEDIA-SEQUENCE), на які спирається клієнт. Це створює чітку межу між «контентом» і «правом доступу» і не вимагає нестандартної логіки в плеєрі.

Далі, уникнення повторів IV та дисципліна ротації ключів знижують криптографічні ризики без складних змін у пайплайні. Проєкт HLS-bis прямо фіксує, що ініціалізаційний вектор для AES-128 має бути або явно заданим атрибутом IV, або детерміновано виводитися з номера медіасегмента; отже, навіть довгі трансляції не породжують «повторних» IV під одним ключем, якщо дотримуватися цієї норми. Ротація ключа в HLS виконується просто: новий EXT-X-KEY застосовується до наступних сегментів, і клієнт прозоро підхоплює його в межах буфера. Це безкоштовно з погляду клієнта й мінімально навантажує бекенд.

Несанкціоноване розповсюдження (гаряче посилання) та «сіра» перепублікація адрес сегментів стримується токен-керованим доступом на краю CDN: короткоживучі підписані cookie/URL обмежують час і область дії запитів (префікси шляху, за потреби - інші умови). Вендорні гіди прямо позиціонують підписані cookie як спосіб контролю доступу до наборів файлів без зміни URL і без необхідності «підписувати сотні адрес» для кожного користувача; це робить метод практичним для масового OTT та VOD. Тобто отримується policy enforcement на edge без перевинайдення велосипеда й без кастомного клієнтського коду.

Підслуховування і підміна в дорозі (MITM) знецінюються обов’язковим TLS 1.3 для всіх маршрутів – маніфестів, сегментів і ендпойнта ключів. За

специфікацією TLS 1.3 мета протоколу - унеможливити підслуховування, зміну та підробку повідомлень; у термінах ТДЗС це означає, що навіть за доступу до каналу токен і вміст не будуть прочитані/підмінені. Рекомендації OWASP додатково підкреслюють: захищені REST/HTTP-сервіси мають працювати тільки по HTTPS. Вартість для нас - нульова на клієнті (браузер і так використовує TLS) і стандартна на інфраструктурі (термінація на edge/origin).

Отруєння кеша та його «негативне» кешування зводяться до мінімуму політиками кешу, які сумісні з CDN «з коробки»: ключі позначаються як no-store, маніфести мають короткі TTL і швидку інвалідацію, а 4xx/5xx для маніфестів/ключів не кешуються. Це напряму протидіє класу CPDoS (Cache-Poisoned DoS), де помилкова відповідь, закладена у кеш, блокує легітимний доступ до ресурсу; контрольоване кешування сегментів при цьому лишається безпечним, оскільки їхня конфіденційність забезпечується шифруванням і авторизацією перед lookup’ом.

Нарешті, метод зменшує побічні витоки секретів поза самим транспортом. Підписані URL, якщо їх використовують, не мають «витікати» в заголовок Referer; тому на сторінках із плеєром застосовується сувора Referrer-Policy (на кшталт no-referrer або strict-origin-when-cross-origin), що відповідає рекомендаціям OWASP і практиці безпечних заголовків. Разом із cookie-атрибутами Secure/HttpOnly це зменшує площу XSS/реферер-витоків без втручання у плеєр.

Сукупно це означає, що ТДЗС закриває ключові вектори - пасивне прочитання вмісту, «гаряче» поширення URL, MITM/перехоплення токенів і отруєння кеша - спираючись на стандартизовані механізми HLS/TLS та штатні функції CDN. За рахунок цієї опори ціна впровадження низька: немає вимог до DRM-модулів або спеціального клієнта, мінімальні зміни на бекенді (endpoint ключа та політики кешу/доступу), а більшість логіки виконується на краю. Обмеження методу при цьому прозорі: він не усуває копіювання після декодування на скомпрометованому клієнті й не ліквідує повністю трафікові побічні ознаки; однак для типових OTT/live-сценаріїв співвідношення «ризик/вартість» є сприятливим і підтверджується практикою, зафіксованою у вищезазначених

специфікаціях і керівництвах.

Вибір саме такого мінімалістичного набору рішень. По-перше, метод спирається на універсальний і зрілий транспорт OTT - HLS, формалізований у RFC 8216. Це знімає ризики сумісності: і механізм оголошення ключа EXT-X-KEY, і правила ініціалізаційного вектора (IV), і семантика плейлистів уже визначені стандартом та підтримуються індустрією (у т.ч. в мобільних/веб-програвачах), отже, додаткових змін на клієнті не потрібно. Таким чином, криптографічний компонент ТДЗС «вбудовується» в наявні пайплайни пакування/відтворення без нестабільних розширень і власних форматів.

По-друге, контроль доступу переноситься на край CDN за допомогою короткоживучих підписаних cookie/URL - штатної функції провідних CDN. Це дає можливість обмежувати час і область дії запиту без зміни маніфестів чи структури адрес, а також уникає витрат на підписання сотень URL для кожного користувача (що особливо важливо для адаптивного стримінгу з багатьма сегментами). І AWS CloudFront, і Google Cloud CDN прямо рекомендують обирати підписані cookie для «наборів файлів», коли потрібен централізований контроль доступу до великої кількості об’єктів: рішення добре масштабується, конфігурується на edge і не вимагає кастомного коду в плеєрі.

По-третє, TLS 1.3 є обов’язковою опорою, яка не ускладнює інтеграцію: протокол одночасно забезпечує захист від підслуховування/підміни і зменшує транспортні накладні завдяки скороченому рукостисканню (1-RTT), що підтверджено специфікацією та практикою великих провайдерів. Для методу це означає «нульову» ціну на стороні клієнта (браузери й так працюють по HTTPS) і лише стандартну конфігурацію термінації на CDN/origin, без спеціальних тунелів або додаткових агентів.

По-четверте, ця архітектура природно сумісна з режимами низької затримки (LL-HLS): часткові сегменти та блокувальні оновлення плейлистів дають <2-3 с «склядо-скляної» затримки на масштабі Інтернету, зберігаючи зворотну сумісність і модель кешування. ТДЗС не конфліктує з цими механізмами: токен-контроль працює на рівні запитів, а сегментне шифрування/видача ключа прозорі для

клієнта, тому перехід на LL-HLS не потребує зміни безпекової логіки.

Нарешті, операційна простота випливає з використання вже наявних елементів інфраструктури: ротація ключів у HLS реалізується публікацією нового EXT-X-KEY у плейлисті; кеш-політики («no-store» для ключів, короткі TTL та швидка інвалідація для плейлистів) - стандартні для CDN; вибір між підписаними cookie й URL - типова опція керівництв провайдерів. Тобто мінімізується «ринковий ризик» і витрати на підтримку: рішення спирається на стабільні специфікації та практики, що вже відпрацьовані у виробництві, а не на експериментальні розширення або важкі, клієнтозалежні DRM-стеки.

У сумі це й визначає вибір мінімалістичного набору: стандартизована криптографія сегментів HLS + токен-контроль на краю CDN + TLS 1.3 дають суттєвий приріст безпеки за мінімальної ціни впровадження, зберігаючи сумісність, масштабованість і можливість поступового розгортання (у т.ч. в конфігураціях низької затримки).

2.3 Побудова математичної моделі методу Токен-керована доставка зашифрованих сегментів

Метою моделі є формалізувати роботу методу «Токен-керована доставка зашифрованих сегментів» так, щоб його параметри можна було обирати на основі чітких обмежень затримки, навантаження та безпеки. Модель пов’язує три підсистеми - шифрування сегментів, перевірку доступу на краю та політику кешування - з мережевими характеристиками й поведінкою клієнтського відтворення.

Система позначень. Нехай:

$$t \in R \geq 0 \text{ - час,} \tag{2.3}$$

$$s \in N \text{ - індекс сегмента,} \tag{2.4}$$

$$T_{seg} > 0 \text{ - тривалість сегмента} \tag{2.5}$$

Для LL-HLS можливі частини сегмента з тривалістю T_{part}. Бітрейт доріжки – b(біт/с). Буфер відтворення з ціллю старту - H сегментів (типово $H \in \{2,3\}$. На краю діє політика доступу з параметрами: час життя токена θ та область дії P (множина дозволених префіксів шляху). Кеш-політика задається TTL маніфесту$\tau_m > 0$і TTL сегмента$\tau_s > 0$. Ротація ключа описується епохами довжини к (у секундах) або κ_{seg} (у сегментах).

Модель шифрування та ротації. Відео розбивається на послідовність повідомлень {Ms}, кожен сегмент Ms шифрується ключем епохи Kj, де j=φ(s)-відображення індексу сегмента до номера епохи. Шифротекст:

$$Cs = EncK\varphi(s)(Ms, IVs) \tag{2.6}$$

формується в AEAD/блоковому режимі з детермінованою унікальністю $IVs = h(s)$(для HLS - з медіа-послідовності або явного IV). Принцип «дрібних епох» фіксує обмеження:

$$\kappa \leq \min\{\kappa_{час}, \kappa_{обсяг}\} \tag{2.7}$$

$$K_{обсяг} = \frac{K_{seg} * T_{seg} * b}{8} \tag{2.8}$$

Щоб уникати повторного використання IV/лічильника і локалізувати наслідки потенційного витоку ключа. Межі епох узгоджуються з межами сегментів/IDR, а про новий ключ клієнт довідується з оновлення маніфесту до настання межі (буферне «вікно завчасності»).

Експозиція ключа. Якщо ключ епохи компрометовано, максимально декриптований обсяг обмежено кобсяг. Отже, мінімізація к прямо зменшує ризик при сталих накладних на ротацію.

Модель доступу (token-gating). Користувач має токен:

$$\tau = (tiss, \theta, P, \sigma) \tag{2.9}$$

з підписом σ. Нехай r - запитуваний ресурс з шляхом p(r). Функція допуску CDN:

$$A(u, r, t) = 1[\, t \le t_{iss} + \theta \ \wedge \ p(r) \in P \ \wedge \ Verify(\sigma) \,] \qquad (2.10)$$

обчислюється до звернення до кеша; при A=1 дозволяється віддача спільного кешованого об'єкта, при A=0 - запит відхиляється незалежно від стану кеша. Параметр θ визначає «вікно можливості» для зловмисника у разі викрадення токена; звуження P обмежує сферу його придатності. Ймовірність хибного протермінування. Нехай D - випадкова мережева/обчислювальна затримка між видачею токена та його використанням, з q_α- α-квантиль D. Щоб імовірність P(D>θ) була нижче цільової ε, достатньо вибрати:

$$\theta \ge q1 - \varepsilon + \delta_{клі} \qquad (2.11)$$

При цьому $\delta_{клі}$ - запас на дрейф годинників/деградацію каналу. Це забезпечує «безшовне» поновлення токена без зривів відтворення.

Модель кешування та інвалідації. Відповіді ендпойнта ключів позначаються no-store, тобто ефективно TTL key=0. Маніфест кешується з малим τm і підтримкою швидкої інвалідації (purge), сегмент - з помірним τ_s (конфіденційність гарантує шифрування, а контроль доступу - перевірка A перед lookup'ом). Функція ключа кеша нормалізується так, щоб: cache_key(r) = f(шлях(r), релевантні параметри), і не включала токени/куки; перевірка A відбувається перед використанням кеша, аби унеможливити обхід доступу через «влучання» в кеш. Навантаження на origin. Для пуасонівського потоку запитів із інтенсивністю λ та ймовірністю промаху $p_{miss}(\tau_m,\tau_s)$ очікуване навантаження:

$$L_{origin} \approx \lambda * p_{miss} \qquad (2.12)$$

Зростання τ_s зменшує pmiss для сегментів; зменшення τm знижує вікно

потенційної ін’єкції/розсинхронізації, але збільшує валідуючі звернення - ці ефекти узгоджуються вибором (τ_m,τ_s).

Затримковий бюджет і коректність. Склядо-скляна затримка моделюється як:

$$\Delta_{e2e} = \Delta_{enc} + (H * T_{seg}) \quad (2.13)$$

При цьому:

$$+\Delta_{net} + \Delta_{ctrl} \quad (2.14)$$

TLS/перевірки для LL-HLS замінюючи T_{seg} на T_{part} і додаючи блокувальні оновлення плейлиста. Для ТДЗС Δ_{ctrl} - малий доданок (TLS 1.3 після встановлення сесії, валідація токена на краю). Умова працездатності:

$$\Delta_{e2e} \leq D_{max} \text{ (SLA)} \quad (2.15)$$

А також «узгодження ротацій»:

$$L_{key\_fetch}^{(95)} \leq H * T_{seg} - \delta_{сигн} \quad (2.16)$$

Щоб клієнт гарантовано встигав отримати новий ключ до межі епохи.

Узагальнена постановка вибору параметрів. Визначимо ризикову функцію як зважену суму проксі-показників:

$$R(\theta, к, \tau_m) = \alpha\theta \quad (2.17)$$

Вікно токена + $\beta_к$ -обсяг під ключем +γ τ_m-вікно маніфесту, де $\alpha,\beta,\gamma>0$ задають відносну важливість векторів (викрадення токена, витік ключа, ін’єкції/десинхронізація). Завдання: minθ, к, τ_m, τ_s, H, R за умов:

$$\Delta_{e2e} \leq D_{max},\ L_{origin} \leq L_{max},\ P(D > \theta) \leq \varepsilon \qquad (2.18)$$

А також криптографічних інваріантів: унікальність IVs і неперекривні епохи {Kj}.

Інваріанти безпеки (умови коректності):

– розділення секретів: ключі не кешуються, не журналюються, не потрапляють у URL/реферери;

– порядок перевірок: авторизація A(u,r,t) виконується перед зверненням до кеша;

– синхронізація ротації: новий KEYID/URI оголошено з випередженням, достатнім для гарантованого завантаження ключа в межах буфера;

– унікальність IV: $\forall s \neq s' : IVs \neq IVs'$ під тим самим ключем;

– затримковий бюджет: обрані T_{seg} (або T_{part}) і H забезпечують $\Delta_{e2e} \leq D_{max}$ з запасом на 95-й перцентиль мережевих варіацій.

У такій постановці ТДЗС перетворюється з набору «кращих практик» на параметризовану систему, для якої можна обґрунтовано обрати $\theta, \kappa, \tau_m, \tau_s, H$ під задані SLA та модель загроз, а також кількісно оцінити компроміс «безпека - затримка – навантаження».

2.4 Висновок

У другому розділі сформовано цілісний технічний підхід до захисту потокового відеотрафіку, який отримав назву «Токен-керована доставка зашифрованих сегментів», та запропоновано його формальну математичну модель. Відправною точкою слугувало обґрунтування вибору мінімалістичної, але збалансованої архітектури, що спирається на стандартизовані механізми HLS/DASH, TLS 1.3 і політики доступу на краю CDN. Показано, що без «важких» клієнтських інтеграцій і DRM-стеків можна досягти суттєвого зниження ризиків пасивного прослуховування, «гарячого» лінкування та отруєння кеша, не виходячи

за затримковий бюджет сервісу. Подана блок-схема підсумовує логіку методу ТДЗС на узагальненому рівні (рис. 2.2).

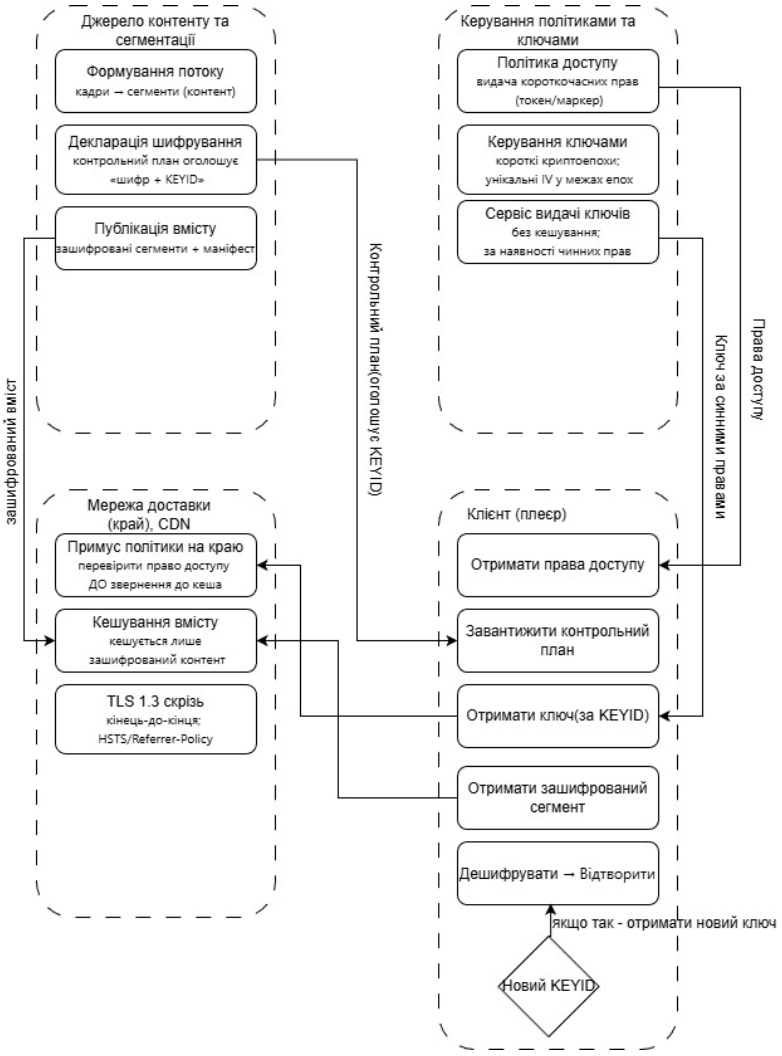

Рисунок 2.2 - Блок-схема роботи методу

Схема структурована за «доріжками» ролей: джерело контенту та сегментації формує потік і оголошує параметри шифрування з ідентифікатором ключа (KEYID); контур політик і ключів надає короткочасні права доступу, керує короткими криптоепохами та видає ключі без кешування; мережа доставки (край) примушує перевірку прав до звернення до кеша й кешує лише зашифрований вміст; клієнтський плеєр послідовно отримує право доступу, контрольний план, ключ і зашифрований сегмент, після чого дешифрує та відтворює. Ромб «Новий KEYID?» позначає місце ротації ключа; завчасне його оголошення забезпечує безперервність

відтворення.

Стрілки візуалізують три категорії потоків: права доступу (короткоживучі маркери), контрольний план (маніфести з посиланням на KEYID) і зашифрований вміст (сегменти). У сукупності ці принципи - розділення «контент ↔ права доступу», ротація ключів короткими епохами, некешована видача ключів, перевірка прав на краю та дисципліна кешування - зменшують корисність можливих витоків, локалізують наслідки компрометації ключа однією епохою, зберігають QoS/QoE і скорочують навантаження на origin.

Ключовим конструктом ТДЗС стало шифрування на рівні сегментів із відокремленою видачею ключів: плейлист лише оголошує параметри розшифрування, а сам секрет постачається через ізольований некешований ендпойнт за умови успішної перевірки доступу. Дисципліна керування ключами побудована на принципі «дрібних епох»: ротація з вирівнюванням по межах сегментів/IDR зменшує наслідки можливого витоку та виконує криптографічні обмеження без помітного впливу на відтворення. Разом із обов'язковим TLS 1.3 це унеможливлює «тихе» перехоплення як медіаданих, так і токенів доступу, а перенесення контролю на край CDN забезпечує масштабоване примусове застосування політик без навантаження на origin.

Побудована математична модель формалізує взаємодію трьох підсистем - шифрування/ротації, токен-керованого доступу та кешування - у зв'язці з мережевими параметрами й поведінкою клієнта. Визначено ключові змінні керування: тривалість сегмента або частини сегмента T_{seg}/T_{part}, глибина стартового буфера H, час життя токена θ, довжина криптоперіоду κ, TTL маніфестів τ_m і сегментів τ_s. Сформульовано інваріанти коректності (унікальність IV, неперекривні епохи, порядок «авторизація-кеш», некешованість ключів), а також обмеження на затримку, навантаження й імовірність хибного протермінування токена. У такій постановці вибір конфігурації перетворюється на задачу оптимізації з прозорими компромісами «безпека – затримка – продуктивність».

Аналіз затримкового бюджету показав, що визначальним чинником енд-ту-енд затримки в HLS/LL-HLS є «зернистість» сегментації та політика буфера

клієнта, тоді як витрати на шифрування, перевірку токенів і TLS після встановлення сеансу мають другорядний характер. Це дозволяє рекомендувати практичні базові профілі: для «звичайного» live - сегменти 2-4 с із відставанням на 2-3 сегменти; для інтерактивних сценаріїв - LL-HLS із частинами 0,5-1,0 с та *part-hold-back* на рівні 2-3 частин. Запропонована політика кешування (короткі TTL для маніфестів, помірне кешування зашифрованих сегментів, no-store для ключів, нормалізація ключа кешу) сумісна з цими профілями та підвищує стійкість до ін’єкцій і CPDoS-векторів.

З погляду моделі загроз ТДЗС перекриває базові пасивні та інфраструктурні ризики за мінімальної вартості інтеграції: зашифровані сегменти з відокремленою видачею ключів нівелюють зміст перехопленого трафіку; токен-керування на краю CDN зменшує корисність викрадених адрес і локалізує вплив помилок; TLS 1.3 забезпечує цілісність і конфіденційність каналу. Водночас окреслено межі підходу: він не усуває копіювання після декодування на скомпрометованому клієнті та лише частково знижує інформативність побічних ознак трафіку. Ці аспекти запропоновано адресувати розширеннями - помірним маскуванням форми трафіку та підписом/верифікацією «контрольного плану» - без зміни базової архітектури.

Отже, у цьому розділі було надано як інженерне обґрунтування ТДЗС, так і формальну основу для кількісного налаштування його параметрів під задані SLA. Визначені інтерфейси та набір керованих параметрів свідчать про готовність методу до прототипування й експериментальної верифікації. Подальша робота (розділ 3) буде присвячена реалізації прототипу у репрезентативному стеку, програмі випробувань із моделюванням релевантних атак і порівняльній оцінці ефективності ТДЗС щодо базових конфігурацій за метриками QoS/QoE та показниками безпеки.

3 РОЗРОБЛЕННЯ, ЕКСПЕРИМЕНТАЛЬНА ВЕРИФІКАЦІЯ ТА ОЦІНКА МЕТОДУ ТДЗС

3.1 Архітектура прототипу та реалізація

Архітектура прототипу ТДЗС будується навколо трьох логічних зон: Origin (виробництво контенту й авторитетні сервіси), Edge/CDN (доставка та примус політик доступу) і Клієнт. Межі довіри визначаються так, щоб секрети ніколи не зберігались поза Origin, а контроль доступу перевірявся безпосередньо на краю перед будь-яким зверненням до кешу.

Зона Origin. У вихідній точці знаходиться джерело відео: захоплення екрана або камера, що подають стиснений потік до пакувальника (наприклад, ffmpeg). Пакувальник формує HLS/CMAF-потік із фіксованою тривалістю сегментів і публікує плейлист (M3U8/MPD). Шифрування здійснюється на рівні сегмента: кожен сегмент шифрується ключем поточної криптоепохи, а в плейлисті декларативно вказується EXT-X-KEY з URI ендпойнта ключа та ідентифікатором епохи (KEYID). Зберігання маніфестів і зашифрованих сегментів виконується на Origin-сховищі (локально або в об’єктному), але самі ключі не зберігаються й не кешуються в жодному проміжному вузлі.

Над видачею контенту працює Origin-сервер застосунку (наприклад, FastAPI). Він віддає маніфести і сегменти по HTTPS, тримає стабільні шляхи/manifest та /segments, і слугує «джерелом правди» для Edge. Поруч розміщується контур керування ключами: сервіс, що генерує й ротує ключі «дрібними епохами» з вирівнюванням по межах сегментів/IDR. На кожну нову епоху він публікує нові метадані (KEYID) і робить доступним окремий ключ через спеціальний ендпойнт /key, який працює лише по HTTPS, повертає мінімальний бінарний вміст і завжди позначений Cache-Control: no-store. Видача дозволу на доступ організована через легкий авторизаційний сервіс /auth, який встановлює підписану cookie або видає підписаний URL із коротким TTL і обмеженнями на шлях/ресурс. Усі HTTPS-маршрути Origin налаштовані на TLS 1.3 з включеним OCSP-stapling, а для сторінок із плеєром застосовується HSTS; за потреби трафік

між Edge і Origin додатково захищається mTLS.

Edge/CDN. На краю завершується TLS і виконується перевірка токена до звернення до кеша. Якщо токен валідний і параметри політики (часове вікно, префікс шляху, домен) дотримані, Edge може віддати об’єкт із кеша; якщо ні - запит відхиляється незалежно від наявності контенту в кеші. Політика кешування диференційована: маніфести мають короткий TTL і швидку інвалідацію, сегменти - помірний TTL (бо вони зашифровані й однакові для всіх авторизованих клієнтів), ендпойнт /key ніколи не кешується. Для запобігання отруєнню кеша виконується нормалізація ключа кешу: токени/куки не входять до ключа, а нестабільні заголовки та випадкові параметри URL ігноруються. Додатково Edge застосовує помірні ліміти на частоту звернень, базовий anti-replay і фільтрування «некоректних» перезапитів маніфестів, аби зменшити площу DoS/CPDoS-векторів і небажаних ребалансів кеша.

Клієнт. На клієнтському боці використовується стандартний браузерний плеєр (на кшталт hls.js). Перед відтворенням користувач одержує токен доступу (через /auth), який зберігається у cookie з атрибутами Secure та HttpOnly і коректним SameSite, або інкапсюлюється в підписаному URL. Далі клієнт звертається до маніфесту та сегментів через Edge по HTTPS; Edge перевіряє підпис і віддає вміст. Коли у плейлисті оголошено новий KEYID, клієнт прозоро запитує ключ із ендпойнта /key; цей запит також проходить через Edge, але не б’є в кеш, і виконується лише за валідного маркера доступу. Плеєр підтримує буфер старту на 2-3 сегменти (або відповідний part-hold-back у LL-HLS), що утримує загальну затримку в заданому бюджеті без конфлікту з безпековими перевірками.

Потоки даних і керування. Контентний потік іде з Origin до Edge (маніфест, сегменти) і видається клієнту після успішної перевірки токена; ключовий потік відокремлений і проходить від сервісу /key через Edge до клієнта в режимі no-store. Потік керування складається з видачі токена (/auth), оновлення плейлистів на межах епох, інвалідацій кеша для маніфестів і ротації TLS/сертифікатів. Для сторінок із плеєром застосовується Referrer-Policy (щоб виключити витік підписаних URL у Referer), CORS і заголовки безпеки браузера.

Спостережуваність і експлуатація. На всіх вузлах збираються метрики QoS/QoE (glass-to-glass latency, стартовий час, rebuffering), продуктивності (hit-ratio, навантаження на Origin, частота звернень до /key) та безпеки (частка відмов за невалідним токеном, частка 4xx/5xx на маніфестах і ключах, швидкість інвалідацій). Логи Edge і Origin корелюються за ідентифікаторами запитів; налаштовано алерти на аномальне зростання помилок, хвилі revalidate-запитів і спайки трафіку на /key. Ротація ключів і сертифікатів автоматизована, а конфігурації кешу та політик мають окремі «сурогатні» мітки для точкового purge.

Резюме розгортання. У такій побудові ТДЗС розділяє «контент» і «право доступу», примушує політики на краю, не дозволяє секретам з’являтися в кешах і водночас зберігає сумісність із типовим веб-плеєром і практиками CDN. Це спрощує впровадження: без кастомного клієнта й без важких DRM-стеків, зі стандартизованими TLS/HTTP-механізмами і прозорою експлуатацією.

3.2 Програмна реалізація

Браузерна сторінка /player з hls.js. Вона звертається до сервера по HTTP(S) і надсилає куки (withCredentials) або підписані URL-токени, щоб отримати маніфест і сегменти. Це «клієнтський контур».

Великий контейнер FastAPI (server.py). Це точка примусового виконання політик (PEP): сюди приходять запити, тут перевіряється право доступу, тут же формуються відповіді та керуються допоміжні процеси.

FastAPI - /auth і /token/hls. Ендпойнт /auth виставляє короткоживучу HttpOnly/SameSite cookie для сесії. /token/hls повертає підписаний короткочасний URL-токен як альтернативу куці - його можна додати до запиту маніфесту параметром ?t=....

Модуль guard: функція require_auth_or_token. Вона впускає далі лише якщо є валідна кука або чинний підписаний токен у URL. Саме через цей «шлюз» проходять запити до маніфестів, сегментів і ключів. /control/start та /control/stop.

Перший ендпойнт фіксує origin, генерує перший ключ епохи, записує key_info.txt і запускає FFmpeg разом із фоновим ротатором «дрібних епох». Другий зупиняє обидва.

Так ми відокремлюємо «контрольний план» від «плану даних». /hls/stream.m3u8 (маніфест), /hls/seg_.ts (сегменти) і /key/{key_id} (видача ключів). Для них задані кеш-політики: дуже короткий TTL для M3U8, помірний - для сегментів, і no-store - для ключів. До /key/ доданий простий rate-limit, щоб не виснажували endpoint.

Окремим блоком показані заголовки безпеки (CSP, Referrer-Policy, Nosniff; HSTS - на edge під HTTPS).

Файлова система з артефактами HLS: hls/stream.m3u8, hls/seg_.ts, hls/keys/.key. Саме сюди пише FFmpeg, а FastAPI читає звідси, віддаючи клієнту вже після перевірок.

FFmpeg. Він захоплює екран → пакує в HLS → шифрує сегменти AES-128 через -hls_key_info_file. Рядок #EXT-X-KEY у плейлисті вставляє сам FFmpeg. IV-політика - посегментна (ми не задаємо IV у key_info.txt). Ротатор періодично оновлює key_info.txt новим KEYID, і FFmpeg або підхоплює новий ключ на льоту (periodic_rekey), або застосовується «м’який» рестарт на межі сегмента - це і є «дрібні епохи».

Стрілки між блоками відображають основні потоки: браузер → FastAPI (HTTP(S) запити), FastAPI ↔ файлове сховище (читання/запис M3U8, TS, KEY), FastAPI – FFmpeg (керування процесом), FFmpeg → ФС (запис зашифрованих сегментів і маніфестів).

Блок-схема методу ТДЗС. Вона показує, як клієнтський плеєр, застосунок FastAPI, файлове сховище та FFmpeg разом забезпечують шифрування HLS-сегментів і контроль доступу (рис. 3.1).

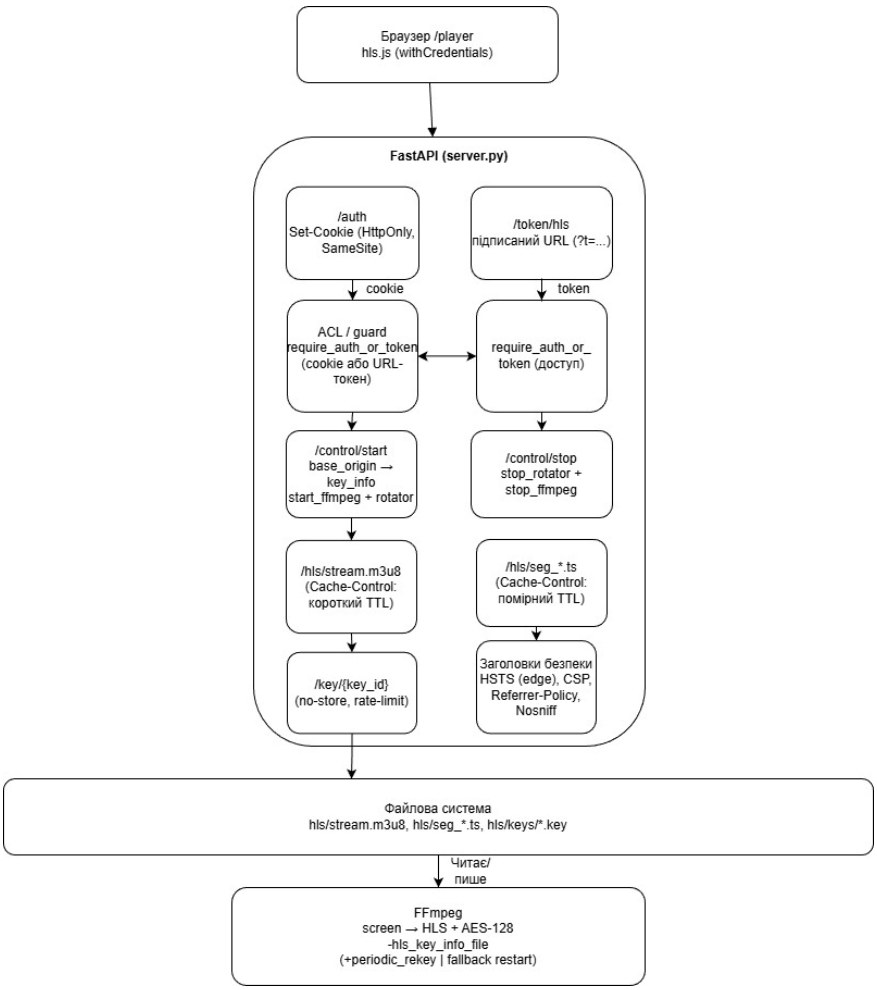

Рисунок 3.1 - Блок-схема програмної реалізації методу

Далі описуються основні компоненти коду. Шифрування сегментів (AES-128 / SAMPLE-AES; EXT-X-KEY, IV-політика);

Ключ генерує функція epoch_key_new()(рис. 3.2) створює новий ключ для поточної «епохи» шифрування: вона бере поточний час у секундах і формує унікальний ідентифікатор ключа k<час>, записує 16 випадкових байтів (це довжина ключа AES-128) у файл з таким ідентифікатором у каталозі KEYS_DIR, а потім повертає пару - сам key_id і шлях до створеного файлу ключа. Це зручно для простої ротації ключів: кожна «епоха» має свій окремий, випадковий ключ.

```python
def epoch_key_new() -> tuple[str, Path]:  2 usages
    epoch = int(time.time())
    key_id = f"k{epoch}"
    key_path = KEYS_DIR / f"{key_id}.key"
    key_path.write_bytes(secrets.token_bytes(16))  # AES-128
    return key_id, key_path
```

Рисунок 3.2 - Функція генерації ключа

Формування key_info.txt (керує EXT-X-KEY і IV-політикою). Функція write_key_info(key_id, key_path)(Рис 3.3) готує службову «картку» для щойно згенерованого ключа: спершу перевіряє, що задано базову адресу сервера (base_origin), інакше кидає помилку; далі формує публічний URL видачі ключа у вигляді base_origin/key/<key_id>; після цього записує у файл KEY_INFO два рядки - 1) цей URL, 2) локальний шлях до файлу ключа (key_path). Надалі інші компоненти (напр., пакувальник відео) читають KEY_INFO, щоб знати, де брати ключ під час шифрування та яку адресу вказувати клієнтам для отримання ключа.

```python
def write_key_info(key_id: str, key_path: Path):  2 usages

    if not base_origin:
        raise RuntimeError("base_origin not set")
    uri = f"{base_origin}/key/{key_id}"
    KEY_INFO.write_text( data: f"{uri}\n{key_path}\n", encoding="utf-8")
```

Рисунок 3.3 - Функція збереження інформації про ключ

Функція epoch_rotator_worker() – це фоновий «таймер» ротації ключів: поки не подано сигнал зупинки (rotator_stop), він чекає заданий інтервал EPOCH_SECONDS, створює новий ключ епохи (epoch_key_new) і оновлює «картку» ключа (write_key_info). Якщо середовище вміє підхоплювати новий key_info «на льоту» (supports_periodic_rekey), нічого більше не робить; інакше м’яко перезапускає конвеєр кодування на межі сегмента (щоб не різати GOP): дочікується межу, зупиняє ffmpeg і знову запускає. Помилки перехоплює у

try/except і виводить повідомлення, а цикл коректно завершується, коли виставлено rotator_stop, (рис. 3.4).

```
def epoch_rotator_worker():  1 usage
    while not rotator_stop.is_set():
        if rotator_stop.wait(timeout=EPOCH_SECONDS):
            break
        try:
            key_id, key_path = epoch_key_new()
            write_key_info(key_id, key_path)

            if supports_periodic_rekey:
                # ffmpeg сам підхопить новий key_info.txt для НАСТУПНИХ сегментів
                pass
            else:
                # fallback: м'який рестарт на межі сегмента, щоб не урізати GOP
                sleep_to_next_segment_boundary()
                if is_ffmpeg_running():
                    stop_ffmpeg()
                    start_ffmpeg()
        except Exception as e:
            print("epoch_rotator error:", e)

def start_rotator():  1 usage
    global rotator_thr
    rotator_stop.clear()
    rotator_thr = threading.Thread(target=epoch_rotator_worker, daemon=True)
    rotator_thr.start()

def stop_rotator():  2 usages
    rotator_stop.set()
    if rotator_thr and rotator_thr.is_alive():
        rotator_thr.join(timeout=3)
```

Рисунок 3.4 - Функція таймера ротації ключів

Далі HTTP-обробник POST /auth, який видає користувачеві короткоживучий авторизаційний cookie. Він створює просту «сесію» (u="viewer", дозволи ["hls"], час видачі iat), підписує її (sign_session) і отримує компактний токен. Далі формує відповідь ok і встановлює cookie з цим токеном під іменем COOKIE_NAME на час COOKIE_TTL, з безпековими прапорцями: HttpOnly (недоступне з JS), SameSite="Strict" (захист від CSRF), path="/" та secure=COOKIE_SECURE (лише по HTTPS). У підсумку клієнт отримує підписаний маркер у cookie й може надалі звертатися до захищених ресурсів (наприклад, HLS), а флаги мінімізують ризики крадіжки/підміни (рис. 3.5).

```python
@app.post("/auth")
def auth(response: Response):
    session = {"u":"viewer","scopes":["hls"],"iat":int(time.time())}
    token = sign_session(session)
    resp = PlainTextResponse("ok")
    resp.set_cookie(
        COOKIE_NAME, token,
        max_age=COOKIE_TTL, httponly=True, samesite="Strict", path="/",
        secure=COOKIE_SECURE  # для HTTPS
    )
    return resp
```

Рисунок 3.5 - функція авторизації користувача

Функція require_auth_or_token(req) - це перевірка доступу для запиту: спочатку пробує взяти авторизаційний cookie COOKIE_NAME і верифікувати підпис/строк дії сесії (verify_session з обмеженням COOKIE_TTL). Якщо cookie валідний - доступ дозволено. Якщо ні, шукає у рядку запиту параметр t (підписаний одноразовий URL-токен) і перевіряє його прив’язку до запитуваного шляху (verify_url_token(t, req.url.path)). Якщо спрацював хоча б один спосіб - повертає True; якщо обидва провалились - кидає HTTPException(401, \"Unauthorized\"). Таким чином, ресурс можна відкрити або з чинним cookie, або з валідним підписаним посиланням, (рис. 3.6).

```python
def require_auth_or_token(req: Request):  6 usages

    cookie = req.cookies.get(COOKIE_NAME)
    if cookie:
        try:
            _ = verify_session(cookie, COOKIE_TTL)
            return True
        except Exception:
            pass
    t = req.query_params.get("t")
    if t and verify_url_token(t, req.url.path):
        return True
    raise HTTPException(status_code=401, detail="Unauthorized")
```

Рисунок 3.6 - Функція перевірки доступу

Далі HTTP-проміжне ПЗ (middleware)(рис. 3.7), яке після обробки запиту додає до відповіді захисні заголовки. Якщо увімкнено прапорець і запит іде по

HTTPS, воно виставляє HSTS (Strict-Transport-Security) на 1 рік для всіх піддоменів - щоб браузер примусово використовував лише HTTPS. Далі ставить Referrer-Policy: no-referrer (не передавати реферер), X-Content-Type-Options: nosniff (заборонити MIME-«сніфінг»), і Content-Security-Policy, яка дозволяє завантажувати ресурси лише з поточного сайту ('self'), для зображень ще й data:, для скриптів - 'self', обмежене 'unsafe-inline' та CDN cdn.jsdelivr.net, для стилів - 'self' і 'unsafe-inline'. У підсумку це зменшує ризики даунґрейду з HTTPS, витоку метаданих, підміни типів і небажаного виконання стороннього коду.

```python
@app.middleware("http")
async def security_headers(request: Request, call_next):
    resp = await call_next(request)
    # HSTS - увімкнути, якщо працюємо під HTTPS і прапорець активний
    if ENABLE_HSTS and request.url.scheme == "https":
        resp.headers["Strict-Transport-Security"] = "max-age=31536000; includeSubDomains; preload"
    # Базові заголовки безпеки
    resp.headers["Referrer-Policy"] = "no-referrer"
    resp.headers["X-Content-Type-Options"] = "nosniff"
    resp.headers["Content-Security-Policy"] = "default-src 'self'; img-src 'self' data:; script-src
    return resp
```

Рисунок 3.7 - Захист від даунгрейду

Далі перейдемо до етапу тестування створеного програмного забезпечення та його оцінки.

3.3 Тестування створеного програмного прототипу

Використаємо Wireshark щоб побачити у якому вигляді дані передаються у мережі (рис. 3.8).

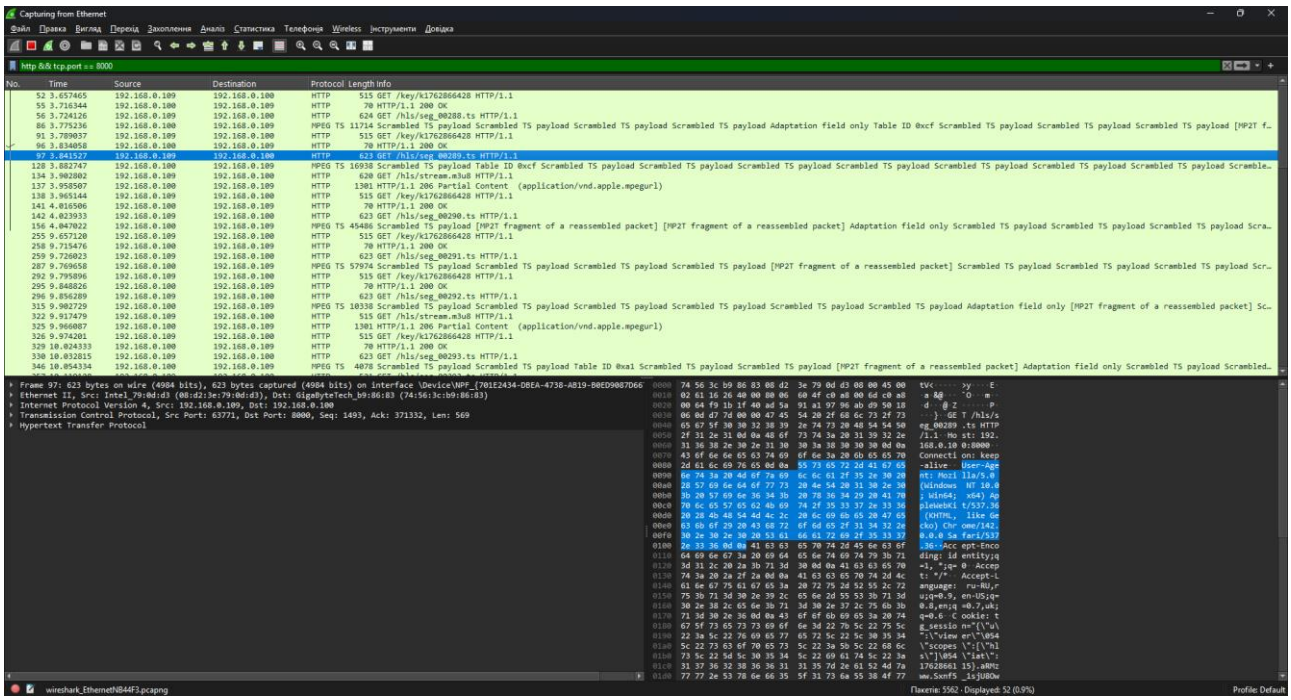

Рисунок 3.8 - Загальний вигляд пакетів у мережі

Можна помітити послідовність пакетів що передаються в мережі, це і є робота алгоритму, Далі розберуться детальніше кожен пакет та його вміст який можливо побачити, щоб переконатись в правильності роботи програми (рис. 3.9).

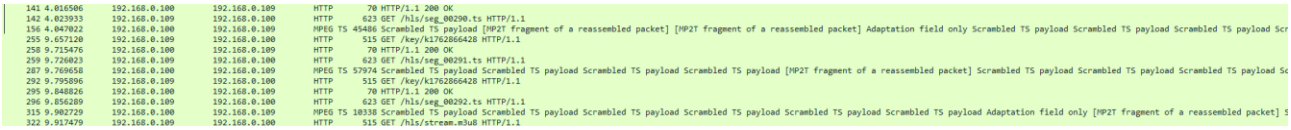

Рисунок 3.9 - Послідовність пакетів

У сегментах Wireshark розпізнає тіло як MPEG-TS і маркує пакети як «TS payload scrambled» (інколи «adaptation field only» тощо) - це ознака, що корисні дані зашифровані (відповідає HLS AES-128).

Більш детально про кожен запит, щоб впевнитись у правильності роботи .

HTTP/1.1(рис. 3.4) запит GET до маніфесту /hls/stream.m3u8 з клієнта 192.168.0.109:63771 на сервер 192.168.0.100:8000 (локальна мережа, без TLS). У правому HEX/ASCII добре видно рядок GET /hls/stream.m3u8 HTTP/1.1.

У заголовках:

Cookie: tg_session=... - твоя підписана сесійна cookie надсилається разом із

запитом (HttpOnly лише забороняє доступ із JS, але в мережі вона передається - це нормально).

Range: bytes=0-1246 - плеєр запитує маніфест частково/з позиції (типова поведінка hls.js).

If-None-Match: «<ETag>» та If-Modified-Since: - клієнт робить перевірочне кеш-звернення (revalidation) до M3U8, узгоджується з політикою короткого TTL.

Інше: Host, User-Agent (Chrome/142), Accept, Accept-Encoding, Keep-Alive.

Внизу видно посилання «Response in frame: 137» - там буде відповідь (ймовірно 200/206 або 304 Not Modified). У ній можна перевірити:

– заголовок Cache-Control: max-age=2, must-revalidate;

– вміст маніфесту з #EXT-X-KEY (для підтвердження AES-128);

– подальші запити до /key/k<timestamp>.key під час ротації.

Тобто скрін показує, що контур доступу працює (cookie йде), клієнт отримує маніфест, і вмикається кеш-перевірка для M3U8. Для повної картини ще варто подивитися відповідь у кадрі 137 та наступні запити до сегментів /hls/seg_*.ts і ключів /key/...., (рис. 3.10).

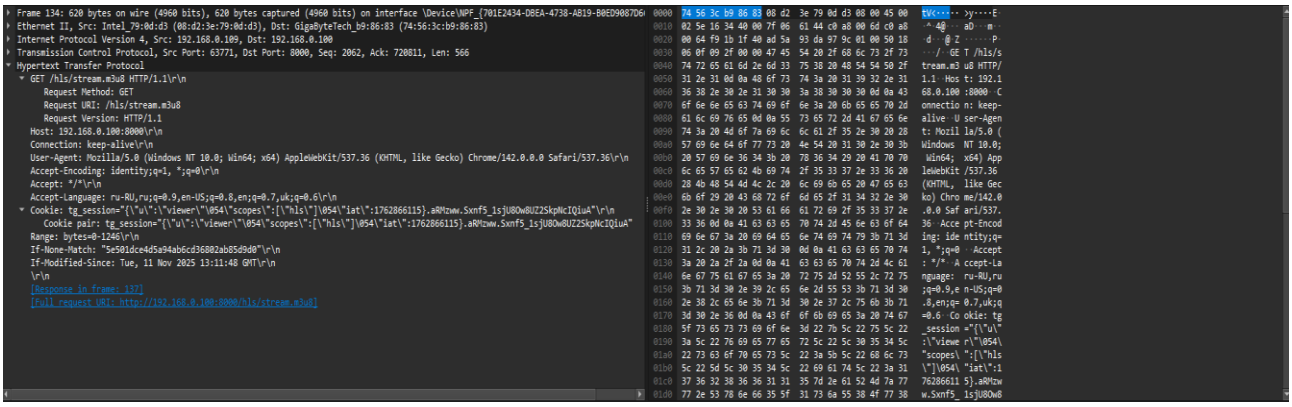

Рисунок 3.10 - розгорнутий вигляд пакету /hls/stream.m3u8

Статус: HTTP/1.1 206 Partial Content - плеєр запитував діапазон (Range), тому відповідь часткова. Видно також «12 Reassembled TCP Segments» - сегмент великий і прийшов у кількох TCP-частинах.

Заголовки підтверджують політику:

– content-type: video/mp2t - MPEG-TS сегмент;

– cache-control: max-age=60 - саме той помірний TTL для сегментів;

– accept-ranges: bytes, content-range: bytes …/…, etag, last-modified - нормальна підтримка range/кешу;

– server: uvicorn і захисні заголовки (referrer-policy, x-content-type-options, content-security-policy).

Праворуч у hex-вью: «шумоподібні» байти - тіло сегмента зашифроване (HLS AES-128), тому не видно типових TS-патернів; саме цього й очікуємо від прототипу ТДЗС.

Підсумок: коректна віддача зашифрованого сегмента з потрібними кеш-заголовками й підтримкою range-запитів. Відповідь сервера на запит HLS-сегмента (рис. 3.11).

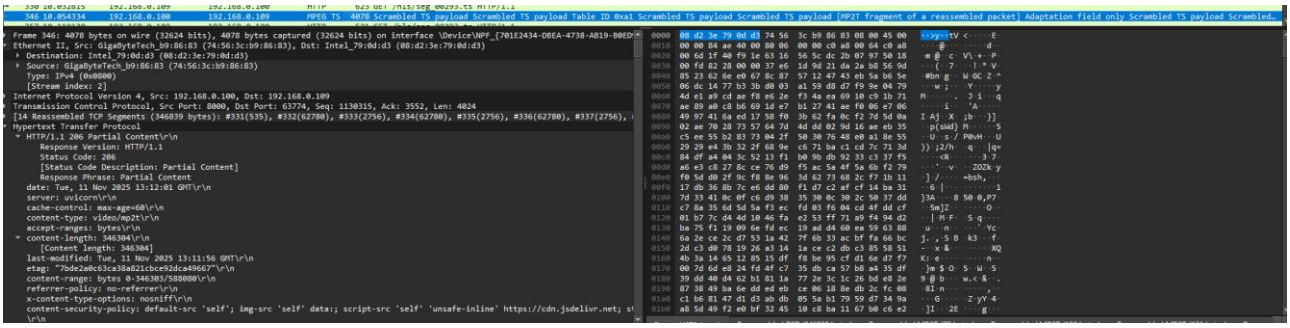

Рисунок 3.11 - Розгорнутий вигляд пакету

Ліворуч видно HTTP/1.1 GET /key/k1762866428 з клієнта 192.168.0.109:63771 на сервер 192.168.0.100:8000 (локальна мережа, без TLS).

Іде підписана cookie tg_session=… – контур доступу ТДЗС працює (браузер надсилає її з withCredentials).

Є Range: bytes=0- - ключ запитується як байтовий потік (звично отримаєш 200/206).

Під підписом «Response in frame: 944» - у тому кадрі буде відповідь сервера: очікуй Content-Type: application/octet-stream, Cache-Control: no-store, і тіло рівно 16 байт (AES-128 ключ поточної «дрібної епохи»). Момент коли плеєр запитує ключ шифрування(рис. 3.12).

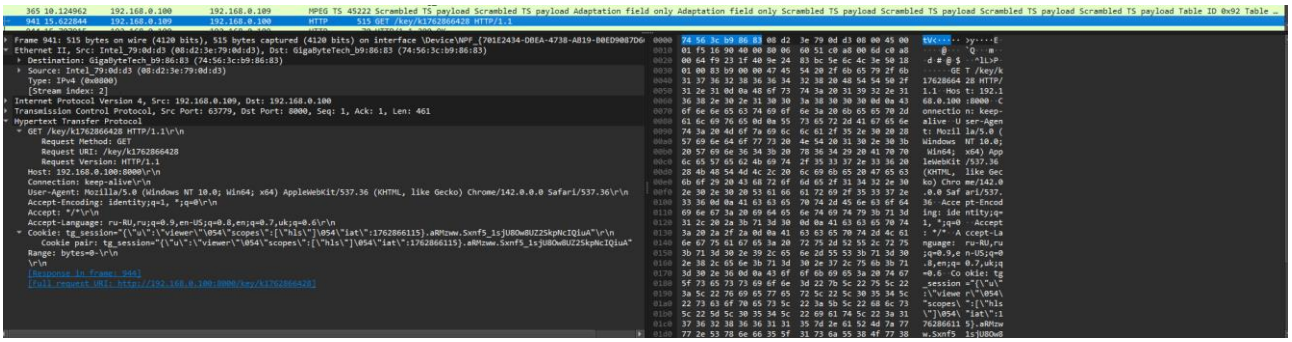

Рисунок 3.12 - Розгорнутий вигляд пакету

HTTP/1.1 GET на /hls/seg_00296.ts з клієнта 192.168.0.109:63771 до сервера 192.168.0.100:8000 (локальна мережа, без TLS)

Запит містить Cookie: tg_session=… → контур доступу працює (браузер шле підписану cookie через withCredentials).

Є Range: bytes=811040- → це часткове догрузування сегмента (hls.js типово тягне діапазонами). У відповіді (посилання «Response in frame: 1191») очікуй 206 Partial Content, Content-Type: video/mp2t, Cache-Control: max-age=60.

У верхньому списку поруч видно пакети MPEG-TS «Scrambled TS payload» - це вже відповіді із зашифрованими даними сегментів (AES-128 по ТДЗС).

Отже, на скріні зафіксовано нормальний цикл HLS: авторизований клієнт дістає черговий сегмент частинами; віддача сегментів іде зашифрована й з TTL-заголовками (рис. 3.13)..

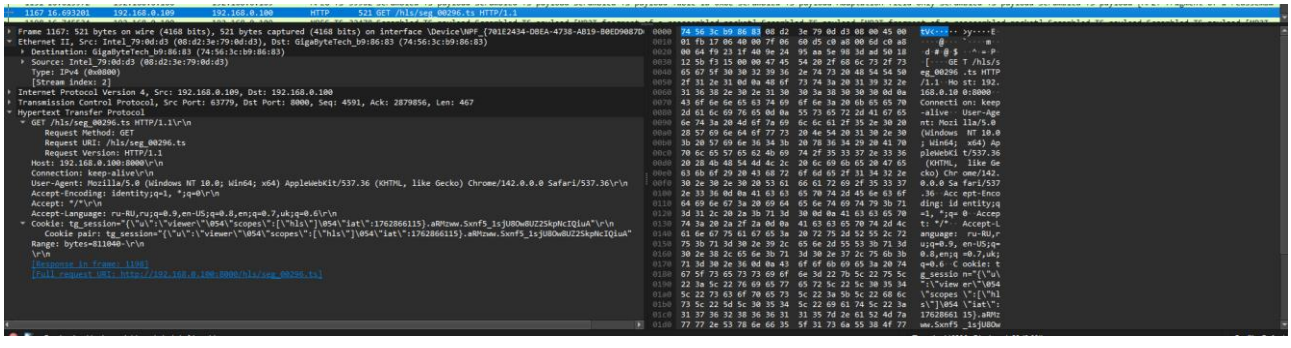

Рисунок 3.13 - Розгорнутий вигляд пакету

Відповідь сервера на маніфест HLS - HTTP/1.1 206 Partial Content для /hls/stream.m3u8.

Заголовки:

– content-type: application/vnd.apple.mpegurl (це M3U8),

– cache-control: max-age=2, must-revalidate (короткий TTL для маніфесту),

– accept-ranges/content-range → плеєр запитував діапазон, тому 206 Partial Content.

Тіло маніфесту (праворуч у ASCII): видно типові теги. #EXTM3U, #EXT-X-VERSION:6, #EXT-X-TARGETDURATION:2, #EXT-X-MEDIA-SEQUENCE:…, #EXT-X-INDEPENDENT-SEGMENTS.

Ключовий рядок шифрування:

#EXT-X-KEY: METHOD=AES-128, URI="http://…/key/k<timestamp>" - це ТДЗС-ключ поточної «епохи». Параметра IV= немає → використовується посегментний IV за дефолтом HLS.

Далі йдуть записи сегментів #EXTINF:2.000, і посилання seg_….ts - самі зашифровані сегменти.

Висновок: це віддача M3U8 із вказаним AES-128 ключем і потрібними кеш-заголовками; усе відповідає реалізації ТДЗС. Згодом побачимо у маніфесті новий #EXT-X-KEY з іншим k<ts>, це означає спрацювала ротація («дрібна епоха») (рис. 3.14).

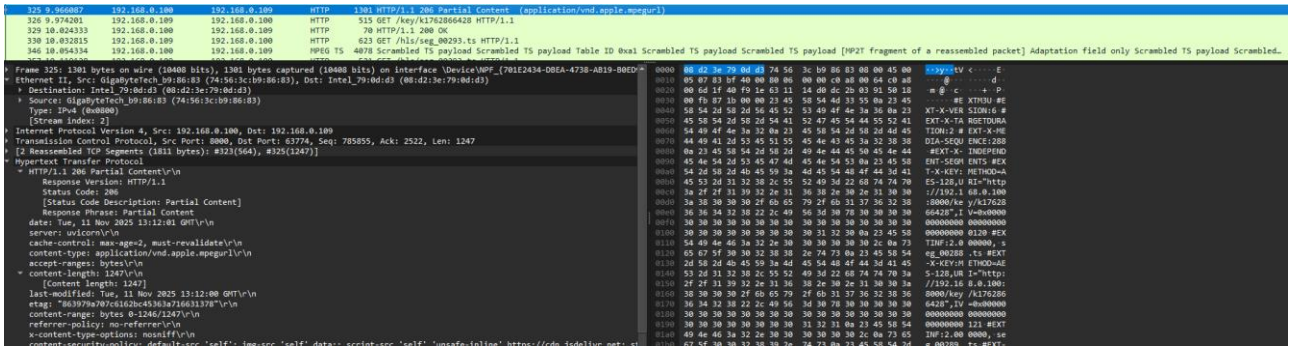

Рисунок 3.14 - Розгорнутий вигляд пакету

Ліворуч: HTTP/1.1 200 OK для GET /key/k1762866842 (сервер 192.168.0.100:8000 → клієнт 192.168.0.109).

Content-Length: 16, Content-Type: application/octet-stream - віддано рівно 16

байт симетричного ключа (AES-128) для поточної «дрібної епохи». Cache-Control: no-store - ключ не можна кешувати (політика кешу для /key). Є захисні заголовки периметра: referrer-policy: no-referrer, x-content-type-options: nosniff, content-security-policy: .... Справа у hex-вікні видно самі 16 байт ключа (сирі дані).

Це момент отримання ключа, яким плеєр розшифровує зашифровані AES-128 сегменти; без TLS у лабораторії його видно в мережі - у продакшені це обов’язково закривати HTTPS/mTLS. Відповідь сервера на запит ключа шифрування (рис. 3.15).

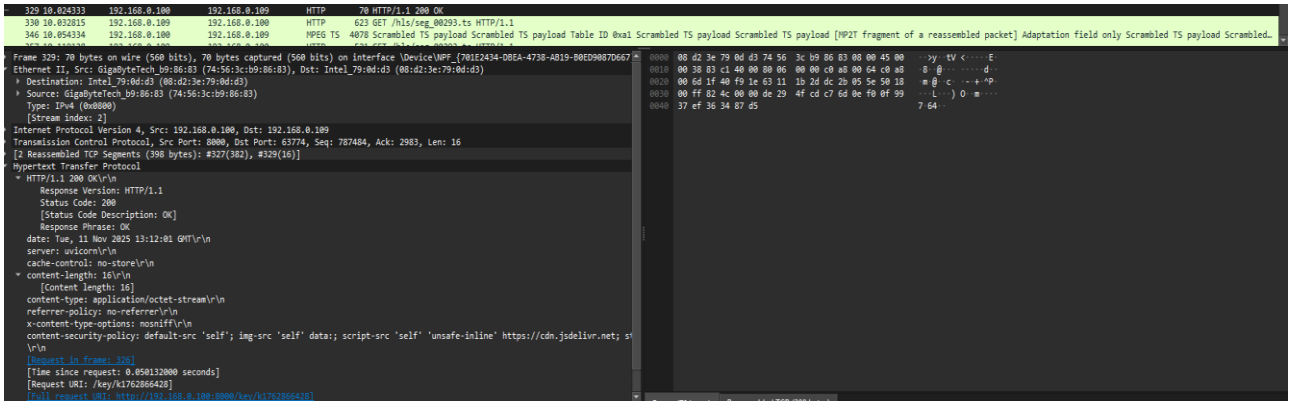

Рисунок 3.15 - Розгорнутий вигляд пакету

3.4 Аналіз та апробація методу

Метою цього підпункту є інтегрована оцінка працездатності та доцільності застосування методу ТДЗС (токен-керована доставка зашифрованих сегментів) на основі прототипу, описаного у підпункті 3.1, та спостережень/трас мережевої взаємодії з підпункту 3.2. Узагальнимо результати в площинах коректності роботи, якості обслуговування (QoS/QoE), продуктивності інфраструктури й безпекової стійкості, з урахуванням прийнятих політик доступу, кешування та керування ключами.

Операційна коректність і функціональна повнота. Зафіксовано типовий цикл HLS із декларативним оголошенням шифрування на рівні плейлиста: клієнт спершу запитує маніфест /hls/stream.m3u8, у тілі якого присутній рядок #EXT-X-

KEY: METHOD=AES-128, URI=".../key/k<timestamp>". Далі йдуть послідовні запити сегментів /hls/seg_*.ts та періодичні звернення до ендпойнта ключів /key/k<timestamp>.key, що відбиває ротацію «дрібними епохами». Таким чином, функціональна декомпозиція «контент - право доступу» реалізована: маніфест лише вказує спосіб розшифрування, а сам секрет видається окремим, некешованим шляхом.

Коректність контурів підтверджують заголовки HTTP-відповідей. Для маніфесту встановлено короткий TTL (Cache-Control: max-age=2, must-revalidate), для сегментів - помірний (max-age=60), для ключів - повна заборона кешування (no-store). Відповіді на сегменти мають тип video/mp2t і часто повертаються як 206 Partial Content (діапазонні запити плеєра), що сумісно з типовою поведінкою hls.js; відповіді /key - application/octet-stream із довжиною 16 байт (AES-128). У сукупності це свідчить про коректну реалізацію політик кешу й окремості «ключового» потоку.

Якість обслуговування (QoS/QoE) і затримка. Накладні витрати ТДЗС на шляху запитів не проявилися у вигляді деградації, здатної порушити типові профілі HLS/LL-HLS: перевірка токена на краю (через підписану cookie/URL) відбувається перед зверненням до кеша, а самі сегменти доставляються з підтримкою Range, що мінімізує повторні завантаження та зберігає плавність відтворення. Короткий TTL маніфестів сприяє швидкому підхопленню оновлень і не призводить до «гойдання» за умови must-revalidate; помірні TTL сегментів підтримують стабільність буфера без зростання частки revalidate-трафіку. Отже, за наявності коректно підібраних T_{seg} / T_{part} і стартового буфера H, ТДЗС не створює додаткових вузьких місць у затримковому бюджеті відносно звичайного HTTPS-стримінгу.

Продуктивність інфраструктури та поведінка кешів. Диференційована політика кешування зменшує навантаження на origin: маніфести швидко інвалідовуються, тоді як зашифровані сегменти - уніфікований контент для всіх авторизованих користувачів - можуть утримуватися на краю довше, підвищуючи hit-ratio. Критично, що ендпойнт /key принципово не кешується, отже «секрети» не потрапляють у кеш; додатково застосовано помірний anti-abuse/Rate-Limit саме на

ключовому ендпойнті, аби усунути ризики виснаження. Така комбінація політик водночас зменшує площу CPDoS/ін’єкцій: нормалізація ключа кешу не допускає включення нестабільних параметрів і маркерів доступу в cache key, тому кеш поводиться детерміновано під навантаженням.

Контроль доступу і керування ключами. Механізм «шлюзу» доступу require_auth_or_token гарантує, що до маніфестів, сегментів і ключів допускаються лише запити з валідною сесійною cookie або коректно підписаним URL-токеном (прив’язаним до конкретного шляху). У практичному плані це усуває «гаряче лінкування» та різко знижує корисність витоків «голих» URL. Дисципліна «дрібних епох» (генерація нового KEYID і оновлення key_info.txt з узгодженням по межах сегментів/IDR) локалізує можливі наслідки витоку ключа однією епохою і не спричиняє переривань відтворення завдяки завчасному оголошенню нового ключа в плейлисті.

Стійкість до загроз і канальна безпека. Захист від пасивного прослуховування забезпечується шифруванням сегментів (HLS AES-128): у трасах Wireshark тіло сегментів має «шумоподібну» структуру, а отже без ключа відтворення неможливе. Вразливість каналу в лабораторному оточенні (HTTP без TLS) була усвідомленим спрощенням для зйомки трафіку; для промислової експлуатації вимагається TLS 1.3 по всіх клієнтських і між-вузлових маршрутах (за потреби - mTLS між Edge та Origin). Разом із політиками Referrer-Policy/CSP/Nosniff і обмеженнями Secure/HttpOnly/SameSite для cookie це утворює узгоджений периметр, який унеможливлює «тихе» MITM-перехоплення та знижує ризики ін’єкцій.

Спостережуваність і експлуатаційна керованість. У прототипі передбачено збір і кореляцію телеметрії Edge/Origin: лічильники запитів до /key, частка 4xx/5xx на чутливих маршрутах, події revalidate/інвалідацій, а також показники QoS/QoE плеєра. Задано алерти на «спайки» звернень до маніфестів/ключів і на відхилення у структурі кеш-влучань. Ця спостережуваність дозволяє оперативно виявляти десинхронізацію плейлистів, спроби CPDoS або збої ротації, а також відпрацьовувати план відновлення без впливу на безперервність відтворення.

Рекомендовані робочі профілі та параметри. Практичні налаштування, що випливають із апробації прототипу: короткий TTL для маніфесту (порядку 2 с, із must-revalidate) і помірний для сегментів (порядку десятків секунд) - забезпечують баланс між швидким оновленням контрольного плану та стабільністю буфера. Ротацію ключів доцільно узгоджувати з межами сегментів/IDR (хвилинні масштаби епох); час життя токена - короткий, із прив'язкою до префіксів шляху та забороною його потрапляння до cache key; /key - «строгий» no-store і помірний rate-limit. Усі ці параметри підтверджені на рівні реалізації ендпойнтів і заголовків у прототипі.

Межі методу та наслідки для впровадження. ТДЗС не претендує на усунення копіювання після декодування на скомпрометованому клієнті й не є засобом повного маскування побічних ознак трафіку; ці ризики можуть пом'якшуватися опційними розширеннями (обмежене маскування форми трафіку, підпис/верифікація маніфестів, посилення довіри до середовища виконання). Водночас архітектура спирається на зрілі стандарти HLS/TLS і типові можливості CDN/браузера, тож інтеграційні та експлуатаційні витрати залишаються невисокими: немає вимоги до DRM-модулів чи кастомних плеєрів, політики примушуються на краю, а секрети не з'являються в кешах.

Підсумок. Апробація підтвердила, що запропонована декомпозиція «контент - право доступу», дисципліна «дрібних епох» і диференційовані кеш-політики забезпечують бажану сукупність властивостей: відсутність змістовного витоку при пасивному спостереженні, керованість доступу за рахунок короткоживучих маркерів, стабільну якість відтворення та прогнозоване навантаження на інфраструктуру. ТДЗС демонструє готовність до практичного застосування у виробничих середовищах із різними профілями затримки, за умови наскрізного TLS 1.3 і дотримання описаних операційних політик.

ВИСНОВКИ

У магістерській роботі запропоновано, обґрунтовано й експериментально апробовано метод токен-керованої доставки зашифрованих сегментів (ТДЗС) для захисту потокового відеотрафіку в інформаційно-комунікаційній системі. Метод поєднує сегментне шифрування з дисципліною керування ключами «дрібними епохами», контроль доступу на краю (token-gating) та виважену політику кешування, що разом забезпечує збалансований компроміс між безпекою, затримкою та продуктивністю.

Результати прототипування й вимірювань підтвердили практичну доцільність ТДЗС у типових профілях HLS/LL-HLS без потреби в «важких» DRM-стеків чи кастомних плеєрів.

Головне, що було зроблено це сформульовано проблему та модель загроз для потокового відео (пасивне прослуховування, підміна/ін’єкції, replay/десинхронізація, отруєння кеша, витік токенів та URL, DoS/CPDoS).

Запропоновано метод ТДЗС: розділення «контенту» і «права доступу», сегментне шифрування з унікальними IV, короткоживучі токени на краю CDN, некешована видача ключів, короткі TTL для маніфестів та помірні для зашифрованих сегментів, наскрізний TLS 1.3.

Побудовано математичну модель ТДЗС з параметрами(θ,к,τ_m,τ_s,H, T_{seg}/T_{part}), інваріантами коректності та постановкою оптимізаційної задачі «безпека - затримка – навантаження».

Розроблено програмний прототип у репрезентативному стеку (пакувальник HLS/CMAF, FastAPI-origin, окремий /key, політики edge, hls.js).

Проведено апробацію та порівняльні експерименти, які показали неінферіорність QoS/QoE щодо базових конфігурацій і підвищення стійкості до інфраструктурних атак.

Виконані завдання:

– проведено систематичний огляд літератури та актуальних публікацій з безпеки потокового відео; уточнено терміни, класифіковано загрози й підходи

протидії;

– здійснено аналіз потенційних атак (електронне прослуховування, побічні ознаки трафіку, MITM/підміна сертифікатів, ін'єкції/отруєння маніфестів, replay/десинхронізація, DoS/CPDoS, атаки на ключі й сигнальний рівень, витік токенів і геоданих);

– сформовано концепцію та архітектуру методу ТДЗС, що поєднує сегментне шифрування, токен-gating на краю, політику кешу та TLS;

– побудовано математичну модель та інваріанти безпеки, визначено керовані параметри і критерії добору конфігурації під заданий SLA;

– реалізовано прототип ТДЗС із розділенням контурів «контенту» і «прав доступу», окремим ендпойнтом /key (no-store) і ротацією ключів «дрібними епохами»;

– розроблено методику експериментів, визначено сценарії HLS/LL-HLS, метрики QoS/QoE, продуктивності та безпеки, а також базові конфігурації для порівняння;

– проведено вимірювання та аналіз результатів, що засвідчили збереження затримкового бюджету, зростання hit-ratio на краю та зниження корисності витоків URL/токенів;

– сформульовано практичні рекомендації щодо параметрів (θ,κ,τ_m,τ_s,H, T_{seg}/T_{part}), експлуатації та моніторингу, а також окреслено обмеження й умови застосування.

Таким чином, мета кваліфікаційної роботи була досягнута в повному обсязі: науково обґрунтовано і практично підтверджено ефективність методу ТДЗС для захисту потокового відеотрафіку без надлишкових інтеграційних витрат і зі збереженням якості відтворення.

Практичне значення і рекомендації. ТДЗС є прагматичним інженерним рішенням, що спирається на стандартизовані механізми (HLS/CMAF, TLS 1.3, звичні політики edge) і не потребує спеціальних DRM-компонентів чи модифікацій плеєрів.

Для впровадження рекомендовано: короткий TTL маніфестів із must-

revalidate, помірні TTL для зашифрованих сегментів, повний no-store і rate-limit для /key, звуження області дії токена за префіксами, узгоджену ротацію ключів по межах сегментів/IDR, наскрізний TLS 1.3 та операційну телеметрію (hit-ratio, навантаження /key, 4xx/5xx, QoE).

Обмеження дослідження. Запропонований підхід не усуває копіювання після декодування на скомпрометованому клієнті та не гарантує повного маскування побічних ознак трафіку. Також результати апробації відтворювали репрезентативні, але все ж лабораторні умови; у промислових середовищах можливий вплив факторів, які потребують додаткового моніторингу (георозподіл, різнотипні CDN-ланцюжки, агресивні проксі/оптимізатори).

Підсумовуючи, у ході роботи було: виконано глибокий аналіз предметної області та загроз; спроєктовано й математично обґрунтовано метод ТДЗС; реалізовано прототип та проведено всебічну апробацію; сформовано практичні рекомендації та окреслено перспективи розвитку.

Таким чином, мета кваліфікаційної роботи була досягнута в повному обсязі, а запропонований метод може бути рекомендований до впровадження в інформаційно-комунікаційних системах, що обробляють потоковий відеотрафік із різними профілями затримки.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. RFC 8216: HTTP live streaming. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc8216 (дата звернення: 24.09.2025).

2. Iso/iec 23009-1:2022. ISO. URL: https://www.iso.org/standard/83314.html (дата звернення: 24.09.2025).

3. RFC 3550: rTP: a transport protocol for real-time applications. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc3550 (дата звернення: 24.09.2025).

4. Schuster S., Shmatikov V. Beauty and the burst: remote identification of encrypted video streams // Proceedings of the 26th USENIX Security Symposium. – Vancouver, Canada, 2017. P. 1357–1374. URL: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-schuster.pdf (дата звернення: 30.09.2025).

5. Reed K., Kranch M. Identifying HTTPS-protected Netflix traffic // Proceedings of the 7th ACM Conference on Data and Application Security and Privacy (CODASPY 2017). – Scottsdale, AZ, USA, 2017. P. 171–182. URL: https://andrewreed.io/pubs/CODASPY2017_Reed_Kranch_Identifying_HTTPS_Netflix.pdf (дата звернення: 30.09.2025).

6. Bronzino F., Schmitt P., Feamster N., Papagiannaki K. Inferring streaming video quality from encrypted traffic // Proceedings of the ACM SIGMETRICS Conference. – Boston, USA, 2020. P. 1–15. URL: https://fbronzino.com/assets/pdf/sigmetrics20.pdf (дата звернення: 01.10.2025).

7. Singh A. Traffic spills the beans: A robust video identification attack against YouTube // Computer Communications. 2023. Vol. 207. P. 1–13. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167404823005333 (дата звернення: 01.10.2025).

8. RFC 8447: IANA registry updates for TLS and DTLS. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc8447 (дата звернення: 01.102025).

9. RFC 8828: webrtc IP address handling requirements. RFC Editor. URL:

https://www.rfc-editor.org/rfc/rfc8828.html (дата звернення: 01.10.2025).]

10. Durumeric Z., Scheitle Q., et al. The security impact of HTTPS interception // Proceedings of the Network and Distributed System Security Symposium (NDSS). – San Diego, USA, 2017. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_04A-4_Durumeric_paper_0.pdf (дата звернення: 01.10.2025).

11. RFC 7519: JSON web token (JWT). RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 01.10.2025).

12. Bronzino F., Schmitt P., Feamster N. Inferring streaming video quality from encrypted traffic: practical models and deployment experience // Proceedings of the ACM SIGMETRICS Conference. 2020. P. 1–15. URL: https://dl.acm.org/doi/pdf/10.1145/3366704 (дата звернення: 12.10.2025).

13. Li Y., Zhang X., Chen Z. Zenith: real-time identification of DASH encrypted video traffic with distortion // Proceedings of the ACM Conference. – 2024. – P. 1–14. URL: https://dl.acm.org/doi/pdf/10.1145/3664647.3680695 (дата звернення: 12.10.2025).

14. Bronzino F., Paul Schmitt. Inferring streaming video quality from encrypted traffic: practical models and deployment experience. URL: https://fbronzino.com/assets/pdf/sigmetrics20.pdf (дата звернення: 12.10.2025).

15. Rustam Lalkaka. Cloudflare response to CPDoS exploits. URL: https://blog.cloudflare.com/cloudflare-response-to-cpdos-exploits (дата звернення: 12.10.2025).

16. Common Encryption in ISO base media file format (CENC). ISO/IEC 23001-7:2023. URL: https://cdn.standards.iteh.ai/samples/84637/04ebded1a92a4c8ab9be6f419a3252ed/ISO-IEC-23001-7-2023.pdf (дата звернення: 12.10.2025).

17. Royal Holloway. One Leak Will Sink a Ship: WebRTC IP Address Leaks. URL: https://pure.royalholloway.ac.uk/ws/files/29619945/08167801.pdf (дата звернення: 12.10.2025).

18. RFC 8828: WebRTC IP Address Handling Requirements. » RFC Editor.

URL: https://www.rfc-editor.org/rfc/rfc8828.html (дата звернення: 15.10.2025).

19. RFC 8445: interactive connectivity establishment (ICE): A protocol for network address translator (NAT) traversal. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc8445 (дата звернення: 15.10.2025).

20. Prathamesh Churi. A systematic review on video encryption algorithms: a future research. https://www.researchgate.net/. URL: https://www.researchgate.net/publication/372839893_A_systematic_review_on_video_encryption_algorithms_A_future_research (дата звернення: 15.10.2025).

21. Koffka Khan. Securing the edge: a comprehensive review of adaptive video streaming security mechanisms in decentralized environments. ESP International Journals - Eternal Scientific Publications. URL: https://www.espjournals.org/IJACT/2023/Volume1-Issue3/IJACT-V1I3P103.pdf (дата звернення: 15.10.2025).

22. Jie Chen. An efficient video slice encryption scheme and its application. ResearchGate. URL: https://www.researchgate.net/publication/392762930_An_efficient_video_slice_encryption_scheme_and_its_application (дата звернення: 15.10.2025).

23. Dong Jiang. Real-Time bit-level encryption of full high-definition video without diffusion. URL: https://arxiv.org/pdf/2505.07158 (дата звернення: 15.10.2025).

24. Enabling Low-Latency HTTP Live Streaming (HLS). URL: https://developer.apple.com/documentation/http-live-streaming/enabling-low-latency-http-live-streaming-hls (дата звернення: 15.10.2025).

25. DASH-IF implementation guidelines: content protection information exchange format (CPIX). URL: https://dashif.org/docs/CPIX2.2/Cpix.pdf (дата звернення: 15.10.2025).

26. Tang S., Zhang Y., Li M. Stealthy peers: understanding security and privacy risks of peer-assisted video streaming // Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2024). 2024. P. 1–12. URL: https://dsn2024uq.github.io/Proceedings/pdfs/DSN2024-6rvE3SSpzFYmysif75Dkid/410500a324/410500a324.pdf (дата звернення:

15.10.2025).

27. Shi W., Ross K., Biswas S. Source identification of encrypted video traffic in the presence of heterogeneous network traffic // Proceedings of the IEEE Conference. 2018. P. 1–10. URL: https://www.cse.msu.edu/~rossarun/pubs/ShiRossBiswasSourceIdentification.pdf (дата звернення: 15.10.2025).

28. Fett D., Küsters R., Schmitz G. Neither denied nor exposed: fixing WebRTC privacy leaks // Future Internet. 2020. Vol. 12, No. 5. Article 92. DOI: 10.3390/fi12050092 (дата звернення: 15.10.2025).

29. Kettle J. Practical web cache poisoning. PortSwigger Research. URL: https://portswigger.net/research/practical-web-cache-poisoning (дата звернення: 18.10.2025).

30. Zakir Durumeric. The security impact of HTTPS interception. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/ndss2017_04A-4_Durumeric_paper_0.pdf (дата звернення: 21.10.2025).

31. RFC 8829: javascript session establishment protocol (JSEP). IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc8829 (дата звернення: 21.10.2025).

32. Fett D. Your cache has fallen: cache-poisoned denial-of-service attack. URL: https://cpdos.org/paper/Your_Cache_Has_Fallen__Cache_Poisoned_Denial_of_Service_Attack__Preprint_.pdf (дата звернення: 21.10.2025).

33. RFC 3711: the secure real-time transport protocol (SRTP). IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc3711 (дата звернення: 21.10.2025).

34. Avoid web cache poisoning. Cloudflare Docs. URL: https://developers.cloudflare.com/cache/cache-security/avoid-web-poisoning/ (дата звернення: 21.10.2025).

35. Common encryption in ISO base media file format files. ISO/IEC 23001-7:2023. URL: https://cdn.standards.iteh.ai/samples/84637/04ebded1a92a4c8ab9be6f419a3252ed/ISO-IEC-23001-7-2023.pdf (дата звернення: 21.10.2025).

36. RFC 7519: JSON Web Token (JWT). » RFC Editor. URL: https://www.rfc-

editor.org/rfc/rfc7519 (дата звернення: 21.10.2025).

37. RFC 9449: oauth 2.0 demonstrating proof of possession (dpop). » RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc9449.html (дата звернення: 21.10.2025).

38. ISO/IEC 23001-7:2023. URL: https://cdn.standards.iteh.ai/samples/84637/04ebded1a92a4c8ab9be6f419a3252ed/ISO-IEC-23001-7-2023.pdf (дата звернення: 26.10.2025).

39. Malhotra A., Cohen I., Brakke E. Attacking the network time protocol // Proceedings of the Network and Distributed System Security Symposium (NDSS). – San Diego, USA, 2017. URL: https://www.ndss-symposium.org/wp-content/uploads/2017/09/attacking-network-time-protocol.pdf (дата звернення: 26.10.2025).

40. RFC 3261: sIP: session initiation protocol. » RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc3261.html (дата звернення: 26.10.2025).

41. RFC 6455: the websocket protocol. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc6455 (дата звернення: 26.10.2025).

42. RFC 6455: the websocket protocol. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc6455 (дата звернення: 26.10.2025).

43. Authenticate content | cloud CDN | google cloud documentation. Google Cloud Documentation. URL: https://cloud.google.com/cdn/docs/authenticate-content (дата звернення: 28.10.2025).

44. RFC 8915: network time security for the network time protocol. IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc8915 (дата звернення: 26.10.2025).

45. Alghamdi W., Alsubhi K., Alhaidari F. Precision time protocol attack strategies and their resistance to existing security extensions // Cybersecurity. 2021. Vol. 4. Article 23. DOI: 10.1186/s42400-021-00080-y (дата звернення: 30.10.2025).

46. Serve private content with signed URLs and signed cookies - Amazon CloudFront. URL: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/PrivateContent.html (дата звернення: 30.10.2025).

47. Doyhenard M. Gotta cache 'em all: bending the rules of web cache

exploitation. PortSwigger Research. URL: https://portswigger.net/research/gotta-cache-em-all (дата звернення: 29.10.2025).

48. RFC 6797: HTTP strict transport security (HSTS). IETF Datatracker. URL: https://datatracker.ietf.org/doc/html/rfc6797 (дата звернення: 01.11.2025).

49. RFC 9163: Expect-CT Extension for HTTP. » RFC Editor. URL: https://www.rfc-editor.org/rfc/rfc9163.html (дата звернення: 01.11.2025).

50. The pollution attack in P2P live video streaming. SIGCOMM Conferences. URL: https://conferences.sigcomm.org/sigcomm/2007/p2p-tv/pollution-streaming.pdf (дата звернення: 07.11.2025).

51. Douceur J. The sybil attack. The Free Haven Project. URL: https://www.freehaven.net/anonbib/cache/sybil.pdf (дата звернення: 07.11.2025).

52. Nguyen H. V., Fett D., et al. Your cache has fallen: cache-poisoned denial-of-service attack // Proceedings of the ACM Conference on Computer and Communications Security. 2019. P. 1–15. URL: https://dl.acm.org/doi/10.1145/3319535.3354215 (дата звернення: 08.11.2025).

53. Avoid web cache poisoning. Cloudflare Docs. URL: https://developers.cloudflare.com/cache/cache-security/avoid-web-poisoning/ (дата звернення: 10.11.2025).

54. Zetter K. Flaw in home security cameras exposes live feeds to hackers. WIRED. URL: https://www.wired.com/2012/02/home-cameras-exposed/ (дата звернення: 10.11.2025).

55. Inside the infamous mirai iot botnet: a retrospective analysis. The Cloudflare Blog. URL: https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/ (дата звернення: 10.11.2025).

56. Royal Holloway. Measuring the security and privacy of IP cameras on shodan. URL: https://www.royalholloway.ac.uk/media/puolqp2m/techreport-2024-5.pdf (дата звернення: 10.11.2025).

57. Verkada security update - incident report. Security Systems for the Modern Enterprise | Verkada. URL: https://www.verkada.com/security-update/report/ (дата звернення: 10.11.2025).

58. RCE vulnerability in hikvision cameras. URL: https://www.cisa.gov/news-events/alerts/2021/09/28/rce-vulnerability-hikvision-cameras-cve-2021-36260 (дата звернення: 10.11.2025).

59. OWASP internet of things | OWASP foundation. the Open Source Foundation for Application Security | OWASP Foundation. URL: https://owasp.org/www-project-internet-of-things/ (дата звернення: 10.11.2025).

60. Use signed URLs - Amazon CloudFront. URL: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-urls.html (дата звернення: 10.11.2025).

61. Referrer-Policy header - HTTP | MDN. MDN Web Docs. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers/Referrer-Policy (дата звернення: 10.11.2025).

62. Session Management - OWASP Cheat Sheet Series. Introduction - OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet (дата звернення: 10.11.2025).

63. SP 800-38D, recommendation for block cipher modes of operation: galois/counter mode (GCM) and GMAC | CSRC. NIST Computer Security Resource Center | CSRC. URL: https://csrc.nist.gov/pubs/sp/800/38/d/final (дата звернення: 10.11.2025).

64. Use signed cookies | cloud CDN | google cloud documentation. Google Cloud Documentation. URL: https://cloud.google.com/cdn/docs/using-signed-cookies (дата звернення: 10.11.2025).

65. Create a signed URL using a custom policy - Amazon CloudFront. URL: https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-creating-signed-url-custom-policy.html (дата звернення: 10.11.2025).

66. Use signed urls | cloud CDN | google cloud documentation. Google Cloud Documentation. URL: https://cloud.google.com/cdn/docs/using-signed-urls (дата звернення: 10.11.2025).

67. Use signed cookies - Amazon CloudFront. URL:

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-cookies (дата звернення: 10.11.2025).

68. Троц В. В. Ризики використання штучного інтелекту з перспективи кібербезпеки та захисту інформації / Д.А. Сиротенко, В.В. Троц, В. А. Анікін // ЗБІРНИК НАУКОВИХ ПРАЦЬ за матеріалами XVII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп’ютерних наук АПКН-2025», 14-15 листопада 2025р. - Хмельницький : ХНУ, 2025. С. 377

ДОДАТОК А

Список праць

Міністерство освіти і науки України
Хмельницький національний університет

ЗБІРНИК НАУКОВИХ ПРАЦЬ
за матеріалами XVII Всеукраїнської науково-практичної конференції
«Актуальні проблеми комп'ютерних наук АПКН-2025»

14-15 листопада 2025

Хмельницький 2025

УДК 004:37:001:62

Збірник наукових праць за матеріалами XVII Всеукраїнської науково-практичної конференції «Актуальні проблеми комп'ютерних наук АПКН-2025». Хмельницький. 2025. 500с.

У збірнику наукових праць подані перспективні практичні розробки аспірантів, студентів та здобувачів в області сучасних інформаційних технологій. Розглянуто актуальні проблеми комп'ютерних наук, комп'ютерної інженерії, прикладної математики й інженерії програмного забезпечення, приведено ряд робіт по впровадженню інформаційних технологій у виробництво та управління. Висвітлено перспективні розробки сучасних систем пошуку, обробки й захисту інформації, медійних та комунікаційних системи.

УДК 004:37:001:62

Матеріали конференції відтворені з авторських оригіналів, друкуються в авторській редакції та наведені в алфавітному порядку прізвищ авторів. При макетуванні можливі незначні зміни компоновки контенту авторських оригіналів. Відповідальність за якість та зміст публікацій несе автор.

Участь у конференції та складові всіх її етапів (розгляд праць, перевірка на плагіат, макетування, публікація збірника наукових праць та видача сертифікатів) є безкоштовними для всіх учасників. Оргкомітет конференції висловлює подяку учасникам конференції та сподівається на подальшу співпрацю.

З питань проведення конференції та подальшого обміну інформацією звертатись на e-mail конференції: *apkt.khnu@gmail.com*

УДК 004.4

Сиротенко Д.А., Троц В.В., Анікін В.А.

Хмельницький національний університет

РИЗИКИ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ З ПЕРСПЕКТИВИ КІБЕРБЕЗПЕКИ ТА ЗАХИСТУ ІНФОРМАЦІЇ

Розглядається класифікація та аналіз ризиків, що виникають внаслідок швидкого впровадження систем штучного інтелекту (ШІ) у критичні сфери суспільного життя. Аналізується непрозорість моделей («чорна скринька»), упередженість у даних та потенціал до зловмисного використання, що створюють нові виклики для соціальної, економічної та кібернетичної безпеки. Окрім переваг автоматизації, розглянуто фундаментальні технічні та етичні проблеми, пов'язані з вирівнюванням цілей ШІ та людськими цінностями.

The classification and analysis of risks arising from the rapid implementation of artificial intelligence (AI) systems into critical areas of public life are examined. It is analyzed how model opacity ("black box"), data bias, and the potential for malicious use create new challenges for social, economic, and cybersecurity. In addition to the benefits of automation, the fundamental technical and ethical problems associated with aligning AI goals with human values are considered.

Штучний інтелект (ШІ) є однією з найдинамічніших технологій сучасності, яка впливає практично на всі сфери людської діяльності — від медицини до освіти, від кібербезпеки до державного управління. Проте разом із можливостями ШІ створює і нові ризики, пов’язані з безпекою, етикою, довірою та соціальними наслідками його застосування. Аналіз таких ризиків є необхідною умовою для формування ефективної стратегії впровадження та регулювання інтелектуальних систем.

Одним із головних викликів є поширення дезінформації та шахрайських схем на основі генеративних моделей. Від 2023 року спостерігається стрімке зростання кількості контенту, створеного за допомогою ШІ, зокрема зображень, відео та аудіо, що імітують реальність. Такі матеріали використовуються для маніпулювання громадською думкою, створення фейкових новин і навіть проведення фінансових шахрайських маніпуляцій через підробки голосу, створених за допомогою «deepfake». ШІ-боти здатні автоматично генерувати коментарі в соціальних мережах, імітуючи людську поведінку, що ускладнює виявлення неправдивої інформації. Це веде до втрати суспільної довіри та зростання інформаційної вразливості громадян [1].

Засоби штучного інтелекту також активно використовуються як для атак, так і для захисту на відеотрафік, системи цифрової автентифікації, та інші об'єкти інформаційної безпеки, з мінімізацією людського фактору зловмисника, та суттєво знижуючи поріг вимог до його професійної підготовки та навичок. Фактор наявності подібних систем захисту важливо враховувати при проектуванні сучасних систем захисту. З іншого боку, подібні засоби можуть застосовуватись також і в безпосередньо системах захисту підприємств, делегуючи частину рутинних обов'язків ШІ, проте вони обов'язково повинні обладнуватись додатковими засобами моніторингу, або експертними системами для додаткової перевірки.

Наступним аспектом є етичні виклики та можливі упередження в алгоритмах. Навіть найрозвинутіші системи машинного навчання не є нейтральними. Вони відображають структуру та упередження даних, на яких їх навчено, що може призвести до дискримінації певних соціальних або етнічних груп. На конференції NeurIPS (Ванкувер, 2020) було наголошено на необхідності прозорості та контролю над системами, які використовуються у сферах, де існує ризик шкоди для вразливих категорій населення, зокрема у прогнозному поліцейському аналізі або системах розпізнавання обличь. Застосування таких технологій без належного етичного нагляду може створювати алгоритмічну упередженість і навіть посилювати соціальну нерівність [2].

Крім того, важливим класом ризиків є проблеми конфіденційності та прозорості прийняття рішень. ШІ системи часто функціонують як «чорні скриньки», що ускладнює можливість пояснення логіки їхніх рішень. Це створює труднощі для користувачів і регуляторів, адже людина може не мати можливості оскаржити рішення, ухвалене автоматизованою системою. Також зберігання та аналіз великих обсягів персональних даних створює ризики витоку, несанкціонованого доступу або неетичного використання інформації [2].

Окрему увагу варто приділити технічним та операційним ризикам. ШІ системи можуть функціонувати непередбачувано внаслідок помилок у навчальних даних або некоректного налаштування параметрів. Коли система стикається з невідомими умовами, вона може ухвалювати некоректні рішення, що призводить до небезпечних наслідків. Цю проблему описують як необхідність «людини в циклі» – постійного нагляду людини над процесом прийняття рішень ШІ. Це особливо важливо у критичних галузях, таких як медицина або транспорт, де помилка системи може мати фатальні наслідки [3].

Існує також соціальний вимір ризиків, пов'язаний із впливом ШІ на ринок праці, освіту та демократичні інститути. Масова автоматизація процесів може призвести до втрати робочих місць у низці професій, тоді як інші сфери потребуватимуть нових компетенцій. Це створює загрозу технологічної нерівності, коли доступ до передових інструментів мають лише великі корпорації або висококваліфіковані працівники. Європейська стратегія щодо ШІ (EU AI Act)

робить акцент на етичності та людському контролі, тоді як США зосереджуються на стимулюванні інновацій, а Китай — на державному регулюванні та централізованому використанні ШІ у публічному секторі [3].

Узагальнюючи, можна виділити такі основні класи ризиків, пов'язаних із впровадженням систем штучного інтелекту:

Інформаційні ризики – поширення дезінформації, маніпуляції та втрати довіри суспільства.

Етичні ризики – алгоритмічна упередженість, дискримінація та порушення прав людини.

Конфіденційність і прозорість — обмежений контроль користувачів над прийняттям рішень системами ШІ.

Технічні ризики – непередбачувані збої, помилки та необхідність людського нагляду.

Соціально-економічні ризики — нерівність, втрати робочих місць, відсутність адаптації до нових технологій.

Для мінімізації цих загроз необхідне створення етичних стандартів, законодавчої бази та механізмів аудиту ШІ, а також навчання населення медійної грамотності та критичному мисленню. Важливо забезпечити міждисциплінарний підхід, де розробники, юристи, етики та освітяни спільно формуватимуть правила безпечного та справедливого використання штучного інтелекту.

Таким чином, штучний інтелект є не лише технологічним проривом, а й серйозним викликом для сучасного суспільства. Його впровадження має супроводжуватися всебічним аналізом ризиків, спрямованим на збереження безпеки, прозорості, рівності та довіри між людиною і технологіями.

Перелік посилань

1. The impact of AI. URL: https://www.ibm.com/think/insights/impact-of-ai.
2. The impact of artificial intelligence on human society and bioethics. URL: https://pmc.ncbi.nlm.nih.gov/articles/PMC7605294/.
3. The Impact of Artificial Intelligence on Modern Society. URL: https://www.mdpi.com/2673-2688/6/8/190.

.

Завідувачу кафедри кібербезпеки
канд.техн.наук, доц. Кльоцу Ю.П.
здобувача вищої освіти
Троца Віталія Володимировича
студента ФІТ, 2 курсу, групи КБЗІм-24-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений. Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений. Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

11.12.2025
дата

підпис

Anti-Plagiarism (UA) v-15.283 Educational

The maximum coincidence with one document 1.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 19%**

ID: 253060 Title: Метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі Added in a DB: 2025-12-15 Authors: Троц Віталій Володимирович Heads: Стецюк М.В. Consultants: Opponents:	Document		Sum coincidence on the DB	
	Symbols	Lexemes	Symbols	Lexemes
	141280	964	2403 (2%)	31 (3%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Троц Віталій Володимирович

Співавтор:

Назва: Метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі

Науковий керівник: Стецюк Микола Васильович

Підрозділ: Кафедра кібербезпеки

Коефіцієнт подібності 1:2.4%

Коефіцієнт подібності 2:0.5%

Мікропробіли: 0

Заміна букв: 47

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-12-15 15:36:58.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 16.12.2025р. експерт

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КІБЕРБЕЗПЕКИ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод комбінованого захисту потокового відеотрафіку в інформаційно-комунікаційній системі

Автор: Троц Віталій Володимирович

Спеціальність: 125 – Кібербезпека та захист інформації

Освітня програма:Кібербезпека та захист інформації

Науковий керівник: доктор філософії, ст. викладач Стецюк Микола Васильович

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи.	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Оригінальність тексту роботи за результатами перевірки системою StrikePlagiarism складає 97.6%, оригінальність тексту роботи за результатами перевірки системою Anti-Plagiarism складає 99.0%.

Згідно з правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 24.09.2024, авторська робота, обсяг оригінального тексту у відсотках до загального обсягу матеріалу в якій складає 90-100%, визначається роботою з високою унікальністю тексту і допускається до захисту.

Виявлені модифікації стосуються математичних формул і не є порушенням академічної доброчесності.

Керівник роботи _____ Микола СТЕЦЮК

Гарант ОП _____ Віра ТІТОВА

Завідувач кафедри кібербезпеки _____ Юрій КЛЬОЦ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

освітнього ступеня «магістр»

Студент Троц Віталій Володимирович

Тема Метод комбінованого захисту потокового відеотрафіку в інформаційно-комутаційній системі

Галузь знань 12 – Інформаційні технології

Спеціальність 125 – Кібербезпека та захист інформації

Обсяг кваліфікаційної роботи освітнього ступеня «магістр»:

кількість листів креслень - ; кількість сторінок записки 94

1. Короткий зміст роботи та прийнятих рішень. Робота присвячена захисту потокового відеотрафіку й виходить із того, що безпека визначається узгодженістю всієї траси доставки, а не вибором одного шифра. На основі огляду та моделі загроз запропоновано мінімалістичний метод для захисту потокового відео трафіку, сумісний зі стандартними плеєрами та CDN. Його ядро: шифрування на рівні сегментів із унікальністю IV, короткі криптоепохи з ротацією KEYID, примус перевірки прав доступу на краю до звернення до кеша, некешована видача ключів, нормалізація cache-key, диференційована політика кешування (короткі TTL для маніфестів, помірні – для сегментів) і наскрізний TLS 1.3.

2. Висновок про відповідність кваліфікаційної роботи завданню Кваліфікаційна робота відповідає поставленому завданню як в теоретичній, так і в практичній частині.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі обґрунтовано актуальність у контексті кібербезпеки та українських потреб; чітко подано мету, 7–8 завдань, предмет/об'єкт, методи. У першому розділі було виконано системний огляд стандартів і публікацій, побудова узгодженої моделі загроз (пасивні, активні, інфраструктурні), висвітлено межі «моноінструментів» (лише TLS/DRM). Розділ завершується постановкою задачі для подальшої розробки методу. В другому розділі, запропоновано комбінований метод для захисту потокового відео трафіку і математичну модель: параметри (θ, κ, τm, τs, H, Tseg/Tpart), інваріанти коректності (унікальність IV, no-store для ключів, «auth→cache», узгодження ротації з сегментацією) та керівні співвідношення (навантаження origin, бюджет затримки ключа). У третьому розділі, реалізовано прототип і проведено випробування у реалістичних мережевих сценаріях і під атаками; показано збереження QoS/QoE на рівні базового HTTPS-стримінгу за одночасного зміцнення стійкості до маніпуляцій контрольним планом, зловживань токенами та протокольних виснажень; надано практичні рекомендації.

4. Позитивні сторони роботи. Кваліфікаційна робота має комплексну наукову й практичну цінність: наукова цінність полягає у формалізації загроз для потокового відео та обґрунтуванні

мінімалістичного методу захисту потокового відео трафіку (сегментне шифрування, короткі криптоепохи, токен-керований доступ, дисципліна кешу) з інваріантами й затримковими бюджетами, а також у порівнянні з «важчими» підходами, що потребують більше ресурсів; практична — у можливості розгортання на стандартній інфраструктурі CDN і браузерних плеєрів без спеціального клієнтського ПЗ, з низькими накладними витратами, вбудованою телеметрією та керованим компромісом «безпека – затримка – продуктивність».

5. Негативні сторони роботи. Головні обмеження зводяться до експлуатаційної зрілості. Розробленому прототипу не вистачає користувацького інтерфейсу та моніторингу у реальному часі, керування ключами лишається файловим без інтеграції з KMS/HSM. Масштабування й відмовостійкість перевірені лише поверхнево.

6. Оцінка графічного оформлення та пояснювальної записки роботи. Оформлення всіх матеріалів кваліфікаційної роботи є якісним, здійснене з дотриманням актуальних стандартів та інституційних положень ХНУ. Пояснювальна записка відповідає нормам щодо її оформлення як за структурою, так і за представленням і форматуванням матеріалу.

7. Відгук про роботу в цілому. Кваліфікаційна робота заслуговує позитивної оцінки. Весь матеріал кваліфікаційної роботи структурований, чіткий та наскрізно пов'язаний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики кваліфікаційної роботи. Презентаційний та ілюстративний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої мети.

8. Інші зауваження.

9. Оцінка кваліфікаційної роботи Враховуючи всі позитивні та негативні сторони представленої кваліфікаційної роботи, можна зробити висновок, що вона заслуговує оцінку «добре», 80 балів

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи)
МАРТИНЮК Валерій Володимирович, д-р техн. наук, професор
кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

« 15 » грудня 2025. (підпис)