МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

МОСЄВНІНА АЛІНА СЕРГІЇВНА

Допускається до захисту:
В.о. завідувача кафедри
прикладної математики та
кібербезпеки, д-р філософії з
математики,
_____Луценко А. В.
«__»_____20__р.

ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ ТА ЦІЛІСНОСТІ ДЗВІНКІВ НА ГАРЯЧІ ЛІНІЇ СИСТЕМИ «SMART CITY»

Спеціальність 125 Кібербезпека

Кваліфікаційна (бакалаврська) робота

Науковий керівник:
Загоруйко Л. В.,
к.т.н., доцент, доцент кафедри
прикладної математики та кібербезпеки

_____
(підпис)

Оцінка : ___/ ___ / _____
(бали/за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця – 2025

АНОТАЦІЯ

Мосєвніна А.С. Забезпечення конфіденційності та цілісності дзвінків на гарячі лінії системи Smart City. Спеціальність 125 «Кібербезпека». Донецький національний університет імені Василя Стуса, Вінниця, 2025.

У кваліфікаційній (бакалаврській) роботі досліджено методи забезпечення конфіденційності та цілісності голосових дзвінків, що надходять на гарячі лінії системи «Smart City». Здійснено аналіз загроз, пов'язаних із обробкою аудіоданих в IoT-інфраструктурах, запропоновано модель захисту аудіо з використанням симетричного шифрування AES-256 у режимі CBC та перевірки цілісності на основі цифрового підпису ECDSA. Реалізовано прототип системи в середовищі Python із застосуванням бібліотек PyCryptodome та PyDub, що дозволяє на практиці перевірити ефективність обраного підходу.

Ключові слова: конфіденційність, цілісність, гаряча лінія, Smart City, AES, ECDSA, аудіофайл, шифрування, цифровий підпис.

56 с., 4 рис., 3 табл., 26 джерел

ANNOTATION

Mosievnina A.S. Ensuring Confidentiality and Integrity of Calls to the Smart City Hotline. Specialty 125 "Cybersecurity." Vasyl Stus Donetsk National University, Vinnytsia, 2025.

The bachelor's thesis explores methods for ensuring the confidentiality and integrity of voice calls received by Smart City hotline systems. The study analyzes threats related to audio data processing in IoT infrastructures and proposes a post-recording protection model using AES-256 symmetric encryption in CBC mode along with integrity verification based on the ECDSA digital signature. A system prototype was implemented in Python using the PyCryptodome and PyDub libraries, enabling practical validation of the proposed approach.

Keywords: confidentiality, integrity, hotline, Smart City, AES, ECDSA, audio file, encryption, digital signature.

56 pp., 4 figures, 3 tables, 26 sourse

ЗМІСТ

ВСТУП 4
РОЗДІЛ 1 ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ ТА ЦІЛІСНОСТІ У СИСТЕМАХ IoT Smart City 6
1.1 Опис IoT-системи «Smart City» та її компоненти. 6
1.2 Поняття конфіденційності та цілісності в IoT-системах Smart City. 11
1.3 Актуальні виклики та загрози конфіденційності й цілісності в Smart City. 13
1.4 Методи забезпечення конфіденційності та цілісності: криптографічні підходи. 18
1.5 Висновки до розділу 1. 24
РОЗДІЛ 2 АНАЛІЗ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАСОБІВ ЗАХИСТУ ГОЛОСОВИХ ВИКЛИКІВ У СИСТЕМАХ SMART CITY 25
2.1 Порівняльний аналіз ефективності протоколів захисту голосових викликів. 25
2.2 Оцінка ефективності механізмів забезпечення цілісності та автентифікації голосових викликів. 35
2.3 Висновки до розділу 2. 40
РОЗДІЛ 3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ 42
3.1 Вибір середовища розробки та інструментів. 42
3.2 Реалізація алгоритму шифрування дзвінків. 43
3.3 Перевірка цільності переданих даних. 46
3.4 Висновки до розділу 3. 48
ВИСНОВКИ 50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 51
ДОДАТОК А 55

ВСТУП

Актуальність роботи. Сучасні тенденції розвитку технологій призводять до активного впровадження концепції «Smart City» - розумного міста, що поєднує різноманітні IoT-рішення для підвищення комфорту та безпеки громадян. Одним із важливих елементів Smart City є гарячі лінії - служби оперативного зв'язку мешканців із міськими органами та екстреними службами. По телефонних та VoIP-каналах гарячих ліній передається чутлива інформація, включно з персональними даними громадян, повідомленнями про надзвичайні ситуації, правопорушення чи інші критичні події. Водночас стрімке зростання кількості під'єднаних пристроїв і обсягу даних, що передаються в міських мережах, породжує нові виклики кібербезпеці.

Конфіденційність таких викликів може опинитися під загрозою через несанкціоноване прослуховування або витік даних, а цілісність переданої інформації - через можливі спотворення чи навмисну зміну контенту повідомлень. Порушення конфіденційності телефонної розмови на гарячій лінії здатне призвести до розголошення особистої чи службової таємниці, тоді як втрата цілісності даних може мати серйозні наслідки: від неправильної реакції служб на інцидент до втрати довіри громадян до міських інформаційних систем. Таким чином, забезпечення конфіденційності та цілісності дзвінків у системі «Smart City» є надзвичайно актуальним завданням у сфері кібербезпеки, від вирішення якого залежить захищеність критичної інформаційної інфраструктури міста і безпека його мешканців.

Мета роботи - підвищення безпеки голосових викликів в системах VoIP, гарячих лініях та "розумної" міської інфраструктури шляхом аналізу актуальних загроз і розробки заходів захисту конфіденційності та цілісності голосових даних.

Для досягнення поставленої мети потрібно виконати наступні *завдання*:

1. Дослідити існуючі загрози та вразливості, що можуть призвести до порушення конфіденційності або цілісності даних під час голосових викликів на гарячі лінії;

2. Порівняти сучасні методи забезпечення конфіденційності голосових даних (методи шифрування каналів зв’язку, захищені протоколи VoIP тощо) та методи забезпечення цілісності (хешування, механізми цифрового підпису, автентифікація повідомлень);
3. Проаналізувати існуючі протоколи і рішення для захищеного передавання голосу (наприклад, протоколи шифрування VoIP, системи шифрованого зв’язку) та оцінити їх придатність для використання в інфраструктурі Smart City;
4. Розробити програмну модель на Python для симуляції дзвінка на гарячу лінію із реалізацією шифрування аудіоданих та накладання цифрового підпису для перевірки цілісності.

Об’єкт дослідження - процес передачі голосових викликів у інформаційно-комунікаційній системі «Smart City», зокрема взаємодія абонента з гарячою лінією міських служб.

Предмет дослідження - методи і засоби криптографічного захисту, що забезпечують конфіденційність та цілісність інформації під час голосових дзвінків (викликів) на гарячі лінії Smart City, а також ефективність їх застосування в даній системі.

РОЗДІЛ 1 ЗАБЕЗПЕЧЕННЯ КОНФІДЕНЦІЙНОСТІ ТА ЦІЛІСНОСТІ У СИСТЕМАХ IoT Smart City

1.1 Опис IoT-системи «Smart City» та її компоненти.

Смарт-місто (Smart City) - це концепція, що передбачає інтеграцію різноманітних інформаційно-комунікаційних технологій, включно з IoT (Internet of Things - Інтернетом речей), для ефективного управління міською інфраструктурою та надання якісних послуг мешканцям. Основна мета впровадження концепції «розумного міста» - оптимізувати функціонування міських служб, підвищити якість життя громадян і забезпечити сталий розвиток міста за допомогою технологій збору та аналізу даних[1]. **Інтернет речей** відіграє ключову роль у Smart City, оскільки завдяки системі взаємопов'язаних датчиків і пристроїв місто може збирати великі обсяги інформації про всі аспекти свого функціонування, передавати ці дані для централізованої обробки та використовувати отримані результати для підтримки прийняття рішень. Завдяки IoT відбувається **перетворення фізичного міського середовища на кібер-фізичну систему**, в якій дані від різних об'єктів (транспорту, будівель, датчиків довкілля тощо) збираються в режимі реального часу та автоматично опрацьовуються для підвищення ефективності міського управління[2].

IoT-екосистема розумного міста складається з низки взаємопов'язаних компонентів, які спільно забезпечують збір, передачу, зберігання, обробку даних та надання корисної інформації кінцевим користувачам. На рисунку 1.1 відображено основні складові такої системи[3]:

- **Апаратні пристрої та датчики.** Фізичний рівень IoT-системи утворюють різноманітні датчики, сенсори та виконавчі пристрої, розгорнуті по місту. Вони слугують «очима й вухами» Smart City, збираючи інформацію про навколишнє середовище та інфраструктуру: наприклад, показники температури і вологості повітря, рівень шуму, інтенсивність руху транспорту, споживання електроенергії тощо. Також до цієї категорії можна віднести **пристрої громадян** - смартфони, розумні годинники, автомобільні трекери - які генерують дані про

активність мешканців міста. Зібрана датчиками інформація оцифровується і підготовлюється для передачі далі по системі. Прикладами обладнання є екологічні сенсори, CCTV-камери спостереження, датчики руху, «розумні» лічильники та інші IoT-пристрої, встановлені на міських об’єктах[4].

- **Комунікаційна мережа.** Для доставки даних від датчиків до центрів обробки використовується мережна інфраструктура - «нервова система» IoT. Вона забезпечує **зв’язок** між мільйонами пристроїв в місті в режимі реального часу. Залежно від вимог, застосовуються різні види мереж: дротові (оптоволоконні лінії, Ethernet) і бездротові. У міському IoT-середовищі широко використовують бездротові технології: Wi-Fi для локального збору даних, стільникові мережі 4G/5G для високошвидкісної передачі, а також спеціалізовані протоколи далекого радіусу дії та низького енергоспоживання - **LoRaWAN, NB-IoT, Sigfox** та інші. Надійна мережа зв’язку з достатньою пропускною здатністю і низькими затримками є критичною для IoT-системи, оскільки від неї залежить актуальність і повнота переданої інформації.
- **Хмарні обчислення та серверна інфраструктура.** На вищому рівні архітектури знаходяться центри обробки даних (ЦОД) та хмарні платформи, що виконують роль «мозку» Smart City. Саме сюди стікається вся інформація, зібрана від польових пристроїв через мережу. **Хмарні обчислення** забезпечують централізовані ресурси для зберігання великих масивів даних і проведення складних обчислень. У хмарному середовищі розміщуються бази даних, сервери додатків, аналітичні модулі та інші сервіси, які обробляють потоки інформації від IoT-пристроїв. Використання хмари дозволяє масштабувати систему під зростаючу кількість пристроїв і даних, а також надає доступ до даних із будь-якого місця в режимі 24/7.

- **Великі дані та сховища.** Обсяг, швидкість надходження і різнорідність міських даних характеризують їх як **великі дані** (Big Data). IoT-система Smart City генерує потоки інформації, що можуть сягати терабайтів щодоби (телеметрія датчиків, відеопотоки з камер, логи від пристроїв, записи дзвінків тощо). Для ефективної роботи з такими масивами застосовуються спеціалізовані сховища даних та платформи Big Data (розподілені файлові системи, NoSQL/SQL бази даних, технології типу Hadoop, Spark). Вони забезпечують надійне збереження історичних даних, швидкий пошук та можливість паралельної обробки. Великі дані слугують «паливом» для аналітики: накопичені відомості можна використовувати для виявлення довгострокових тенденцій, трендів та прихованих закономірностей у житті міста. Наприклад, аналіз великих даних може показати завантаженість доріг у різний час доби, допомогти оптимізувати розклад громадського транспорту або спрогнозувати споживання електроенергії в пікові години.
- **Аналітичні сервіси та додатки.** Це програмні компоненти, які виконують **аналіз даних** та реалізують прикладні функції Smart City. Отримавши дані в хмарі, аналітичні модулі застосовують алгоритми обробки - від простих правил і фільтрів до штучного інтелекту та машинного навчання - щоб перетворити «сирі» дані на корисну інформацію або дії. Наприклад, аналітичний сервіс може обробляти показники датчиків погоди і видавати прогноз smog-ризику, або аналізувати звернення громадян і класифікувати їх за пріоритетністю. До цієї категорії входять системи моніторингу та оповіщення (наприклад, автоматичне сповіщення комунальних служб про аварію), платформи підтримки прийняття рішень (дашборди з показниками ефективності міста), а також **додатки** для кінцевих користувачів - сервіси «розумного» міста. Застосунки можуть охоплювати управління транспортом (смарт-світлофори, система керування трафіком на основі даних), безпеку (системи відеоаналітики для виявлення надзвичайних

ситуацій), комунальні послуги (розумне освітлення, оптимізація споживання ресурсів), охорону здоров'я, освіту, тощо. Саме на рівні додатків реалізуються цільові **сервіси Smart City** - наприклад, система «розумного» паркування, яка на основі даних датчиків показує водіям вільні місця через мобільний додаток, або система громадської безпеки, що аналізує відеопотоки для швидкого виявлення інцидентів.

- **Інтерфейси користувача.** Щоб результати роботи IoT-системи були корисними, вони мають бути наочно представлені та доступні людям. Інтерфейси користувача - це ті точки взаємодії, через які дані та аналітика доходять до міських операторів, управлінців чи самих мешканців. До них належать веб-портали і **дашборди** диспетчерських центрів, мобільні додатки для мешканців, інформаційні кіоски, а також традиційні засоби зв'язку - **гарячі лінії** міських служб. Через інтерфейси здійснюється не лише візуалізація даних, але й **зворотний зв'язок**: міські оператори можуть надсилати команди на IoT-пристрої (наприклад, змінити режими світлофорів), а громадяни - взаємодіяти із системою (надавати дані або запити). Зручність і простота інтерфейсів є важливою, адже від цього залежить залученість користувачів і оперативність реагування на аналітичні висновки.

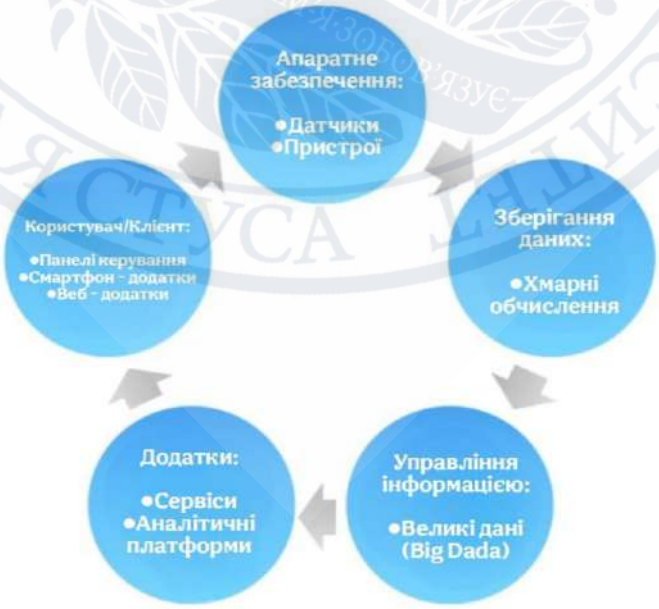

Рис. 1.1. Спрощена схема архітектури IoT в системі Smart City (від збору даних до прийняття рішень).

Перелічені компоненти утворюють єдину багаторівневу **архітектуру IoT-системи Smart City**, де кожен рівень виконує свою роль і передає естафету наступному. Логіка взаємодії така: *датчики* та пристрої на фізичному рівні сприймають дані навколишнього середовища і через *мережі зв'язку* транслюють ці дані до *центральної платформи* (хмарного центру). У хмарі відбувається накопичення інформації у *сховищах даних* та її первинна обробка. Потім *аналітичні сервіси* здійснюють глибший аналіз, виявляють значущі події, патерни або аномалії. На основі цього *додатки* генерують певні вихідні дані: повідомлення, рекомендації, команди для пристроїв. Ці результати через відповідні *інтерфейси* доводяться до відома людей (операторів або громадян) або ж автоматично надсилаються назад на виконання *виконавчим пристроям*. Таким чином забезпечується циклічний процес: **збір даних → передача → зберігання та обробка → аналіз → дія/відображення**, що і реалізує функціональність «розумного» міста. Важливо, що всі компоненти повинні працювати узгоджено: якщо випаде хоча б одна ланка (скажімо, мережа зв'язку дасть збій або не буде доступу до даних), то вся система не зможе повноцінно функціонувати[5].

Описана IoT-інфраструктура Smart City може підтримувати різноманітні міські сервіси, зокрема і систему **гарячих ліній** - телефони довіри, кол-центри міських служб, екстрені номера. У контексті голосових викликів компоненти IoT-системи працюють наступним чином: **інтерфейсом користувача** тут виступає телефонний зв'язок, через який мешканець передає голосове повідомлення про проблему або запит. Цей дзвінок через телекомунікаційну *мережу* (традиційну телефонну або мережу VoIP) надходить до централізованого **кол-центру** міста, що можна розглядати як частину хмарної платформи Smart City. Розмова може автоматично записуватися і зберігатися у базі даних. Далі спеціальні *аналітичні сервіси* можуть обробити отриману інформацію: наприклад, система розпізнавання мовлення перетворює голос на текст, аналізує ключові слова для визначення характеру звернення (аварія,

консультація, скарга тощо) і пріоритетності реагування. Отримані дані передаються відповідним міським службам або диспетчерам (через їх робочі *дашборди*), які вже приймають рішення та діють - направляють бригаду на місце події, дають відповідь заявнику, фіксують проблему в міській системі управління. Таким чином, **гаряча лінія** інтегрується в загальну IoT-систему: телефон є ще одним сенсором (що генерує дані про події), мережа зв'язку доставляє ці дані, хмарна платформа зберігає та аналізує звернення, а результати (реакція міста) доводяться до користувача. Усе це повинно відбуватися швидко та надійно, особливо коли йдеться про екстрені виклики. Тому забезпечення **конфіденційності та цілісності** таких голосових даних є критично важливим завданням. Зокрема, необхідно гарантувати, що канал зв'язку захищений від несанкціонованого перехоплення, записи розмов зберігаються шифрованими, а доступ до них мають лише уповноважені особи. Реалізація відповідних заходів безпеки (автентифікація викликів, шифрування VoIP-трафіку, контроль доступу до баз даних звернень тощо) дозволяє захистити конфіденційність інформації і зберегти цілісність даних гарячої лінії у системі Smart City.

Отже, IoT-система Smart City - це багатокомпонентна структура, що охоплює усе: від датчиків на міських об'єктах до хмарних серверів та користувацьких застосунків[6]. Вона дає місту можливість *перетворювати дані на дії*: збирати інформацію з фізичного світу, інтелектуально її опрацьовувати і застосовувати для покращення міських сервісів. У випадку систем гарячих ліній, IoT-підхід забезпечує більш оперативне і скоординоване реагування на звернення громадян, але водночас вимагає особливої уваги до захисту конфіденційності та цілісності переданої голосової інформації.

1.2 Поняття конфіденційності та цілісності в IoT-системах Smart City.

Конфіденційність в контексті IoT-систем означає властивість інформації залишатися недоступною для неавторизованих осіб. Лише відправник та призначений отримувач повинні мати доступ до змісту переданих даних. Інакше кажучи, сторонні особи не повинні дізнатися ні змісту, ні факту передачі повідомлення[7]. **Цілісність** означає збереження незмінності та достовірності

інформації під час її передавання та зберігання. Всі отримувачі мають отримати саме той зміст повідомлення, який був відправлений, без перекручень чи підроблень. Порушення цілісності - це будь-яка несанкціонована зміна даних, яку можливо виявити та перевірити[8]. У сфері інформаційної безпеки конфіденційність та цілісність (поряд із доступністю) входять до класичної тріади **CIA** (Confidentiality, Integrity, Availability) - основних цілей захисту інформації. Зокрема, для розумних міських систем (Smart City), що базуються на технологіях Інтернету речей, забезпечення конфіденційності та цілісності даних є фундаментальним завданням кібербезпеки.

У середовищі Smart City **голосові виклики на гарячі лінії** (служби екстреної допомоги, міські довідкові служби тощо) є критично важливими сервісами, які покладаються на IoT-інфраструктуру зв'язку. Конфіденційність і цілісність голосового трафіку тут мають особливе значення, адже під час дзвінків може передаватися чутлива інформація: персональні дані громадян, відомості про надзвичайні ситуації, конфіденційні звернення тощо. Порушення конфіденційності таких викликів, наприклад через **несанкціоноване прослуховування**, призводить до витоку приватної інформації. Зловмисники, перехопивши розмову, можуть отримати "скарбницю" цінних відомостей - від фінансових даних до медичної інформації. Це не тільки порушує права на приватність, але й може поставити під загрозу безпеку мешканців (наприклад, якщо розкрита інформація про місцеперебування або стан здоров'я заявника). Цілісність голосових викликів так само критична: уявімо, що дані викривлені або підроблені під час передавання. В такому разі існує ризик невірного реагування служб - навіть найменше спотворення аудіо може змінити суть повідомлення. Наприклад, якщо зловмисник втрутився в канал зв'язку і **змінив зміст голосового повідомлення**, це може призвести до хибних дій екстрених служб або дезінформації відповідальних осіб. Таким чином, довіра до цілісності переданої по телефону інформації прямо впливає на ефективність та безпеку міських голосових сервісів.

1.3 Актуальні виклики та загрози конфіденційності й цілісності в Smart City.

У сучасних системах **Smart City** гарячі лінії та інші служби голосового зв'язку все частіше використовують технології **VoIP (Voice over IP)**. Це дозволяє інтегрувати телефонні виклики з інфраструктурою міста, але водночас ставить нові виклики щодо **конфіденційності та цілісності** цих голосових комунікацій (рис. 1.3) [9]. Згідно з дослідженнями, протокол SIP, який зазвичай використовується в VoIP, є вразливим до низки атак (перехоплення, підроблення повідомлень, тощо), тому потребує впровадження механізмів автентифікації, шифрування та контролю цілісності для захисту викликів[10].

1. Прослуховування викликів (Eavesdropping).

Однією з найбільш очевидних загроз конфіденційності є **несанкціоноване прослуховування** телефонних розмов[11]. У традиційних аналогових мережах цьому протидіяли фізичними засобами, тоді як у VoIP-зв'язку зловмисник може перехоплювати пакетизовані голосові дані через комп'ютерну мережу. Якщо з'єднання не зашифроване, отримати доступ до змісту розмови відносно просто - достатньо запустити на локальній мережі снифер трафіку. Практичні перевірки показали, що значна частина VoIP-викликів є вразливою: **7 з 10 дзвінків** можна перехопити через мережу, оскільки багато кол-центрів не використовують шифрування і мають слабкі налаштування безпеки. Зловмисники можуть отримати таким чином конфіденційну інформацію - наприклад, особисті дані громадян або паролі, продиктовані голосом. Відомі випадки, коли через прослуховування VoIP-з'єднань зловмисники викрадали тональні сигнали (DTMF) для доступу до банківських сервісів або іншу чутливу інформацію. Таким чином, прослуховування безпосередньо компрометує конфіденційність гарячої лінії, перетворюючи приватну розмову на надбання третіх осіб. Для протидії цій загрозі рекомендується шифрувати трафік (використовувати **SRTP** для голосу, **TLS** для сигналізації SIP), сегментувати мережі голосового трафіку та впроваджувати системи виявлення вторгнень, які можуть сповістити про підозрілу активність перехоплення.

2. Підміна та вставка голосових даних (Replay та Injection).

До загроз цілісності голосових викликів відносяться атаки, пов'язані з **підміною або повторним відтворенням переданих даних**. Атака типу **replay** (повторне відтворення) полягає в тому, що зловмисник записує фрагмент розмови або автентифікаційний голосовий код, а згодом відтворює його, видаючи за оригінал. Наприклад, якщо гаряча лінія використовує голосову автентифікацію клієнта, нападник може записати фразу-пароль користувача і пізніше відтворити її, щоб обійти перевірку. **Ін'єкція (injection)** - це впровадження фальшивих даних у сеанс зв'язку. В контексті VoIP це може означати додавання сторонніх голосових пакетів у потік RTP або внесення несанкціонованих SIP-повідомлень. У результаті нападник здатен змінити зміст дзвінка (наприклад, вставити хибне повідомлення) або порушити нормальний хід сеансу. Базовий метод реалізації таких атак - стати **"людиною посередині" (Man-in-the-Middle)** між двома сторонами, перехопивши їхній трафік, і потім **модифікувати або вставляти** свої дані. Зокрема, можливе навмисне **переривання сеансу**: нападник, перебуваючи в мережі, може надіслати підроблене SIP-повідомлення **BYE** від імені одного з співрозмовників, примусово завершивши дзвінок. Інший приклад - атака на маршрутизацію виклику: зловмисник перехоплює службове повідомлення на кшталт 302 Moved Temporarily і підміняє у ньому адресу на свою, перенаправляючи тим самим дзвінок через себе (вводить себе як транзитний вузол). Це дозволяє перехопити розмову або навіть непомітно змінювати переданий звук. Реальний кейс підробки голосових даних стався у 2019 році, коли шахраї використали штучний інтелект для **клонування голосу** директора компанії та подзвонили його підлеглому з проханням здійснити терміновий грошовий переказ - голос звучав настільки правдоподібно, що обман вдався. Хоча той випадок стосувався фінансового шахрайства, суть аналогічна - технологія дозволила **імітувати чужий голос**, підриваючи довіру до цілісності голосового каналу. Для захисту від атак replay/injection необхідно впроваджувати контроль цілісності трафіку (наприклад, перевірку хешів повідомлень, нумерацію RTP-пакетів з виявленням

дублікатів), використовувати **захищені протоколи (SIPS, SRTP)**, а також механізми **шифрування та цифрового підпису** сигналізації, щоб сторонній не зміг непомітно внести зміни.

3. Атаки на мережевому рівні (DoS, ARP Spoofing).

Атаки відмови в обслуговуванні (DoS) та інші мережеві атаки становлять загрозу як для цілісності, так і для доступності голосових сервісів Smart City. Зокрема, атака типу **Telephony Denial of Service (TDoS)** спрямована на перевантаження лінії чи серверів, що обробляють дзвінки. Зловмисники можуть автоматично генерувати тисячі фальшивих викликів або сигналів, щоб зайняти всі доступні лінії гарячої лінії, не даючи реальним абонентам додзвонитися. У пресі описано інциденти, коли **шпиталі піддавалися шантажу через TDoS-атаки**: так, у березні 2013 року невідомий зловмисник погрожував “покласти” телефони приймального відділення лікарні, вимагаючи викуп, і після відмови протягом двох днів дійсно блокував всі 6 телефонних ліній швидкої допомоги, генеруючи масовий трафік дзвінків. На щастя, ніхто не постраждав, але цей випадок висвітлив вразливість екстрених служб до VoIP-атак та можливі наслідки для життя і здоров’я. Окрім навмисного вимкнення служб, DoS-атака може також слугувати ширмою - поки мережеві ресурси перевантажені, зловмисник може паралельно здійснювати інші атаки (наприклад, спроби проникнення) непоміченим.

Ще однією поширеною мережевою загрозою є **ARP-spoofing** (підроблення ARP)[12]. Ця атака належить до класичних методів реалізації **Man-in-the-Middle** у локальних мережах. Протокол ARP, що відповідає за співставлення IP-адрес MAC-адресам у мережі Ethernet, за своєю природою не передбачає аутентифікації запитів. Тому зловмисник у тій самій мережі може розсилати підроблені ARP-відповіді, прив’язуючи свою MAC-адресу до IP-адреси шлюзу (маршрутизатора) або потрібного вузла. В результаті інші вузли мережі починають помилково надсилати трафік через машину нападника. Таким чином, атакувальник одночасно прикидається для маршрутизатора - клієнтом, а для клієнта - маршрутизатором, ставлячи себе посередині каналу зв’язку. Це

відкриває можливості для **перехоплення або модифікації** всіх даних, що проходять між жертвою та мережею, у тому числі і голосових пакетів VoIP.

У контексті гарячих ліній Smart City, успішна атака ARP-spoofing дозволила б перехопити розмови, що проходять через внутрішню мережу організації (наприклад, кол-центру). Об'єднавши ARP-spoofing з аналізом трафіку, зловмисник може витягнути з мережевих пакетів аудіопотік і відновити розмову у зрозумілій формі. Більше того, такий **MITM-напад** дає можливість активно втрутитися - наприклад, впровадити пакет з командою завершення сеансу або спотворити окремі фрагменти голосу. Атаки на мережевому рівні нерідко є передумовою для реалізації більш складних загроз на рівні додатків: отримавши привілейоване становище в мережі, зловмисник може легше скористатися вразливостями протоколів вищого рівня.

4. Уразливості протоколів SIP/VoIP.

Протоколи, що лежать в основі VoIP-телефонії (SIP для сигналізації та RTP для передачі голосу), мають низку вразливостей, які можуть бути експлуатовані зловмисниками для порушення конфіденційності чи цілісності дзвінків. Багато з цих уразливостей пов'язані з тим, що початково SIP проектувався як гнучкий відкритий протокол, орієнтований більше на функціональність, ніж на безпеку. Наприклад, якщо не використовується шифрування (**SIP over TLS**) та надійна автентифікація, SIP-повідомлення можна перехопити або підробити. Зловмисник, знаючи адреси SIP-серверів і номерні плани, може здійснити **реєстраційне викрадення (registration hijacking)** - відправити на сервер підроблений запит реєстрації з ідентифікатором жертви, щоб перевести її виклики на свій пристрій. Інша загроза - **спуфінг ідентифікатора виклику**: атакувальник може підробити поле **Caller ID** в SIP-повідомленні, видаючи себе за інший абонент (наприклад, за офіційний номер міської служби), що підриває довіру до автентичності дзвінка. Уразливості в реалізації SIP-серверів також відкривають шлях до атак: зокрема, у 2020 році виявлено, що понад **1200 організацій** по всьому світу були скомпрометовані через відомі баги в їхніх VoIP-системах - атакери отримали несанкціонований доступ до облікових

записів IP-телефонії. Основна мета цієї кампанії полягала у телефонному шахрайстві (дзвінки на преміум-номери для отримання прибутку), але дослідники підкреслюють, що з таким доступом злочинці могли **слухати приватні розмови** чи навіть використовувати скомпрометовані вузли як плацдарм для подальших атак. Це реальний приклад того, як **невиправлені вразливості** (відсутність оновлень безпеки) в SIP-пристроях призводять до масового порушення конфіденційності.

Іншим слабким місцем є **відсутність шифрування RTP**. Якщо аудіопотік (RTP) передається у відкритому вигляді, перехопивши пакетний трафік, зловмисник може його зберегти і **відтворити запис розмови**. Також можливі атаки типу **buffer overflow** та інші помилки в програмному забезпеченні IP-АТС або SIP-клієнтів, що дозволяють виконати довільний код. Це може дати атакеру повний контроль над вузлом і, відповідно, можливість прослуховувати всі дзвінки, спрямовувати їх через підставні сервери, змінювати сценарії виклику тощо.

5. Додаткові фактори ризику.

Окрім суто технічних атак, значну роль у виникненні загроз відіграють також **людський фактор, застаріле обладнання та хиби конфігурації** систем гарячих ліній. Людський фактор може проявлятися як свідомі зловживання (наприклад, випадки, коли співробітники кол-центрів **продавали записи розмов або повідомляли конфіденційну інформацію** стороннім за винагороду) або ненавмисні помилки (неуважне поводження з паролями адміністрування SIP-систем, підключення незахищених пристроїв до мережі тощо). Застаріле обладнання та програмне забезпечення, яке не підтримує сучасні протоколи шифрування, стає легкою мішенню - наприклад, старі IP-АТС можуть не мати підтримки TLS/SRTP, що робить весь трафік відкритим для перехоплення. Неправильна або недбала конфігурація теж суттєво підвищує вразливість: залишені за замовчуванням паролі SIP-акаунтів, відсутність сегрегації прав для операторів, виключене шифрування “для зручності” адміністрування - усе це

проломи, через які навіть несильно технічно підкований зловмисник може проникнути в систему.

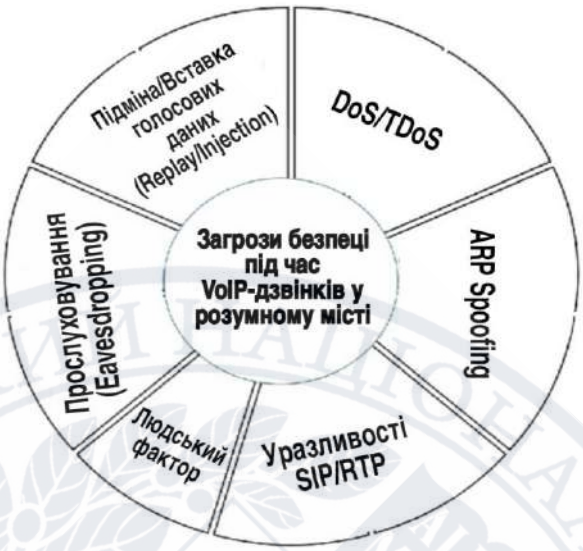

Рис. 1.3. Типи атак на гарячі лінії Smart City

Отже, **забезпечення конфіденційності й цілісності дзвінків на гарячі лінії Smart City** вимагає комплексного підходу. Необхідно врахувати і нейтралізувати різноманітні загрози: від прослуховування розмов та підміни даних до мережевих атак та експлуатації вразливостей протоколів. Реальні інциденти й дослідження підтверджують актуальність цих загроз, тому при проектуванні й впровадженні системи міських гарячих ліній слід закладати сучасні механізми криптографічного захисту, мережевого контролю та аудиту безпеки, а також здійснювати постійний моніторинг і навчання персоналу з питань кібергігієни.

1.4 Методи забезпечення конфіденційності та цілісності: криптографічні підходи.

Голосові дзвінки на «гарячі лінії» Smart City зазвичай реалізуються засобами VoIP (Voice over IP), що дозволяє інтегрувати телефонію в IT-інфраструктуру міста. Однак передавання сигналізації і аудіопотоків у VoIP за замовчуванням не шифрується, тому конфіденційна інформація (ідентифікатори абонентів, зміст розмови тощо) може бути перехоплена або змінена зловмисником[13]. Для захисту конфіденційності та цілісності таких викликів

застосовуються криптографічні методи шифрування та автентифікації даних на різних рівнях VoIP-зв’язку. Основні протоколи і технології, що забезпечують захист VoIP-викликів, включають: захищений сигнальний протокол **SIP over TLS**, шифрування медіапотоку за допомогою **SRTP**, протокол узгодження ключів **ZRTP**, а також використання **цифрових підписів** і **хеш-функцій** для перевірки цілісності.

Захист SIP-сигналізації (TLS). Протокол SIP (Session Initiation Protocol) відповідає за встановлення, зміну та завершення сеансів виклику. Стандартний SIP (порт 5060) передає дані у відкритому вигляді, через що можливе перехоплення даних виклику або навіть несанкціонована зміна маршрутизації[14]. Для забезпечення конфіденційності та цілісності сигнальних повідомлень застосовується шифрування на транспортному рівні - протокол TLS (Transport Layer Security). SIP over TLS шифрує весь трафік SIP між клієнтом і сервером (або між вузлами SIP) подібно до того, як HTTPS шифрує HTTP-трафік. Це унеможливлює пасивне прослуховування сигналізації та утруднює її підробку. TLS забезпечує аутентифікацію сервера (і за потреби клієнта) за допомогою цифрових сертифікатів X.509, підписаних центром сертифікації, тим самим гарантує цілісність повідомлень SIP і довіреність сторін з’єднання. Встановлення TLS-з’єднання включає криптографічне рукостискання з обміном ключами шифрування і параметрами шифрів, після чого вся сигнальна інформація передається в зашифрованому вигляді. В результаті, такі дані як номери виклику, ідентифікатори абонентів, команди протоколу (INVITE, BYE тощо) захищені від перегляду чи модифікації. Важливо, що для повноцінного захисту голосового трафіку протокол TLS для SIP слід використовувати у поєднанні з шифруванням медіапотоку, інакше зловмисник міг би отримати ключі шифрування аудіо при обміні ними через нешифрований SIP. Стандарт SIP передбачає використання URI формату **sips:** для позначення сеансів, які обов’язково проходять через TLS-шифрування.

Шифрування медіапотоку (SRTP). Для передавання власне голосу в системах VoIP використовується протокол RTP (Real-Time Transport Protocol),

що транспортує аудіо- та відеопакети. RTP не має засобів шифрування або контролю цілісності, тому атаки перехоплення (прослуховування) чи заміни RTP-пакетів становлять реальну загрозу[15]. Для захисту медіапотоку розроблено протокол **Secure RTP (SRTP)** - профіль RTP, що забезпечує шифрування даних, перевірку цілісності повідомлень та захист від повторного відтворення пакетів.

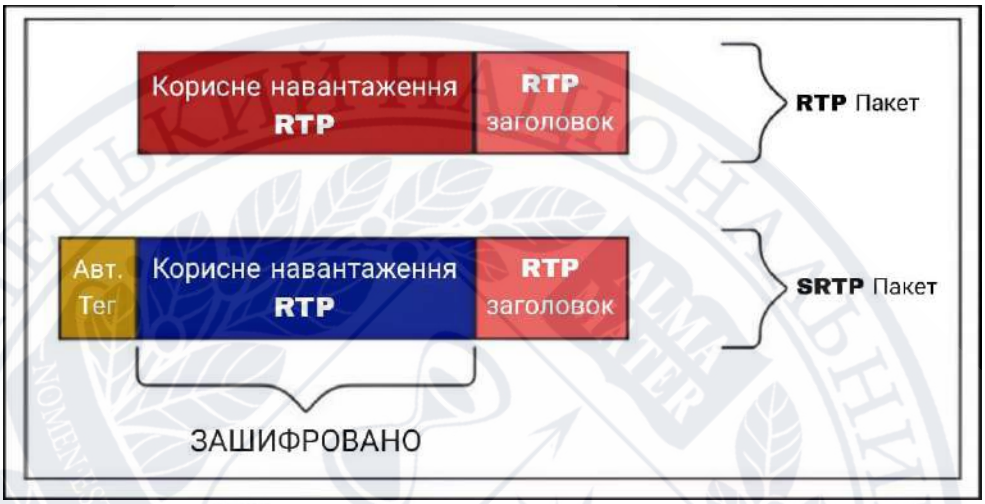

Рис. 1.4. Структура RTP vs SRTP-пакету

Звичайний RTP-пакет містить заголовок і незашифрований аудіо-потік. У SRTP-пакеті корисні дані зашифровані, а додається спеціальний автентифікаційний тег для перевірки цілісності. Заголовок залишається відкритим для маршрутизації[16].

Типово в SRTP використовується стійкий симетричний алгоритм AES для шифрування аудіопотоку (зі 128-бітним ключем у режимі лічильника CTR) та алгоритм HMAC-SHA1 для обчислення коду автентичності повідомлення (80-бітного тега). Це гарантує, що навіть отримавши зашифрований SRTP-потік, зловмисник не зможе відтворити аудіо (через відсутність ключа) і не зможе непомітно модифікувати пакети - будь-яка зміна призведе до невідповідності HMAC і відкидання пакета отримувачем. Протокол SRTP також включає лічильник номерів пакетів і механізм перевірки повторів, що захищає від **атаки повторного відтворення** (replay attack), коли старі захоплені пакети надсилаються повторно.

Важливим аспектом SRTP є розподіл ключів шифрування між сторонами виклику. SRTP сам по собі визначає формат зашифрованих пакунків, але не описує, як саме абоненти домовляються про секретні ключі. Існує кілька підходів до узгодження ключів для SRTP: 1) обмін ключами через сигналізацію SIP (наприклад, протокол SDES - Security Descriptions for SDP, коли ключі вставляються в повідомлення SIP/SDP); 2) переговорювання ключів через захищений канал TLS (наприклад, у разі встановлення сеансу за допомогою SIP TLS сервер може згенерувати та передати ключі SRTP в зашифрованому вигляді); 3) використання окремих протоколів узгодження, таких як ZRTP або DTLS-SRTP. У будь-якому випадку, забезпечення конфіденційності ключового обміну є критично важливим - якщо зловмисник перехопить сигнальний трафік і дістане з нього сесійному ключ SRTP, він зможе розшифрувати весь аудіопотік. Саме тому найпоширеніша схема - це комплексне використання SIP over TLS для захисту сигналізації та SRTP для шифрування медіа. Практична реалізація SRTP передбачає узгодження підтримки шифрування у процесі встановлення виклику: в заголовках SDP протоколу SIP сторони обмінюються інформацією про підтримувані алгоритми шифрування і ключі сесії (якщо використовується SDES), або ж вказують бажання використати інший метод (наприклад, ZRTP). Якщо обидва абоненти і сервер підтримують SRTP, сеанс переходить у захищений режим: аудіо шифрується на стороні відправника і розшифровується отримувачем, перевіряючи MAC кожного пакету.

Протокол ZRTP - узгодження ключів “на льоту”. Одним із сучасних підходів до забезпечення повної конфіденційності голосових викликів є використання протоколу **ZRTP** для встановлення спільного секрету безпосередньо між кінцевими пристроями. ZRTP - це протокол узгодження ключів у медіапотоці, який працює поверх RTP/SRTP і не потребує попередньо виданих сертифікатів чи серверів для ключового обміну. Суть ZRTP полягає в тому, що одразу після встановлення сеансу (на етапі початку RTP-стриму) двоє абонентів виконують обмін ключовою інформацією за допомогою алгоритму Діффі-Геллмана (DH) прямо в каналі RTP. Використовується DH–обмін, тобто

генеруються випадкові пари ключів для кожного виклику, що забезпечує **протокол стійкий до компрометації ключів** - навіть якщо зловмисник пізніше отримає якісь довгострокові ключі, він не зможе розшифрувати вже перехоплені розмови. За результатами обміну DH обидва кінці отримують спільний секретний ключ, який далі застосовується для шифрування SRTP-потоку. Таким чином, ZRTP створює захищений канал передачі голосу безпосередньо між користувачами, не покладаючись на довіру до SIP-сервера чи третьої сторони[17].

Для перевірки справжності встановленого ключа в ZRTP використовується механізм підтвердження відсутності атаки «людина посередині». Оскільки протокол DH сам по собі не автентифікує сторони, теоретично зловмисник може здійснити атаку MitM і непомітно підмінити ключі. ZRTP протидіє цьому через використання **Short Authentication String (SAS)** - короткого рядка для аутентифікації, який обчислюється з параметрів DH. SAS являє собою кілька символів (наприклад, два випадкових слова), які відображаються на обох кінцях після встановлення захищеного каналу. Абоненти мають голосом порівняти ці рядки між собою; якщо вони співпадають - отже, обидві сторони отримали один і той самий спільний ключ. Неспівпадіння SAS свідчить про можливу атаку «посередника», і такий виклик вважається скомпрометованим. Зазвичай інтерфейс користувача сигналізує про успішне шифрування виклику та пропонує підтвердити SAS. Важливо, що на відміну від традиційного SRTP через SDES, схема ZRTP забезпечує **кінець-до-кінця шифрування**: навіть сервер VoIP, через який проходить дзвінок, не має доступу до секретних ключів і не може розшифрувати аудіо. Це підвищує рівень конфіденційності, хоча й може обмежити можливості сервера (наприклад, запис розмови або прослуховування дзвінків техпідтримкою стає неможливим без участі клієнта). В інфраструктурі Smart City протокол ZRTP доцільно застосовувати для найбільш критичних «гарячих ліній», де потрібне максимальне забезпечення приватності звернень громадян, оскільки він гарантує захист навіть у випадку компрометації серверів чи сертифікатів.

Цифрові підписи і хеш-функції. Додатковими криптографічними засобами забезпечення цілісності та автентичності даних є використання механізмів цифрового підпису та криптографічних геш-функцій. **Цифровий підпис** дозволяє підтвердити справжність та цілісність повідомлення: відправник обчислює хеш від даних і шифрує його своїм закритим ключем, утворюючи підпис, який отримувач може перевірити відкритим ключем відправника. У контексті VoIP-телефонії цифрові підписи можуть застосовуватися до сигнальних повідомлень або до журналів викликів. Так само, цифрові сертифікати, задіяні в TLS, фактично є носіями цифрового підпису від центру сертифікації, який засвідчує справжність відкритого ключа сервера. **Хеш-функції** використовуються як складова частина описаних протоколів: вони забезпечують отримання стислого відбитку повідомлення для подальшої перевірки цілісності. Наприклад, в SRTP для обчислення MAC-тегу використовується алгоритм HMAC на базі SHA-1. У ZRTP обчислення короткого рядка SAS також ґрунтується на хешуванні параметрів сеансу. В цілому, стійкі геш-функції гарантують виявлення найменшої модифікації даних: при будь-якій зміні пакету його хеш не співпаде з очікуваним, і система виявить порушення цілісності. Таким чином, поєднання цифрових підписів і хеш-функцій дає можливість контролю цілісності як сигнальних, так і медіа даних у системі VoIP.

Застосування перелічених криптографічних заходів дозволяє забезпечити високий рівень безпеки голосових викликів у інфраструктурі «розумного міста». Шифрування сигнального трафіку (TLS) унеможливлює отримання зловмисником відомостей про виклик або перехоплення керування сеансом. Шифрування голосового потоку (SRTP) гарантує, що розмова залишиться конфіденційною, а її цілісність не буде порушеною по дорозі до отримувача. Додаткові протоколи на кшталт ZRTP забезпечують захист «в кінцях» і незалежність від проміжних вузлів, що особливо актуально для розподілених систем Smart City. Цифрові підписи та хешування доповнюють ці протоколи, дозволяючи верифікувати достовірність даних і протидіяти спробам непомітної модифікації інформації. На практиці, для побудови захищеної «гарячої лінії»

доцільно комбінувати ці підходи: **SIP-TLS + SRTP** (для більшості стандартних викликів) або **ZRTP + SRTP** (для сценаріїв, де потрібне наскрізне шифрування), з використанням сучасних алгоритмів (AES-256, SHA-256, RSA/ECDSA) та належним управлінням ключами. Розгорнувши такі засоби криптографічного захисту, міста можуть гарантувати громадянам конфіденційність звернень на критичні служби та цілісність переданої інформації, що є необхідною передумовою довіри до сервісів Smart City.

1.5 Висновки до розділу 1.

У першому розділі сформовано теоретичну основу для подальшого дослідження: визначено ключові терміни «конфіденційність» та «цілісність» в контексті IoT та Smart City, проаналізовано спектр загроз цим властивостям, а також детально розглянуто сучасні криптографічні підходи, що дозволяють ці загрози мінімізувати. Зроблено висновок, що для забезпечення безпеки «розумних» міст необхідно застосовувати багатошаровий підхід, поєднуючи різні криптографічні методи. Реалізація захищених протоколів зв’язку, шифрування даних та механізмів перевірки цілісності повинна стати невід’ємною частиною архітектури Smart City. Це створює основу довіри до цифрової інфраструктури міста та забезпечує стійкість до кібернетичних впливів, що надалі дозволить успішно впроваджувати інноваційні сервіси на благо громадян.

РОЗДІЛ 2 АНАЛІЗ ТА ОЦІНКА ЕФЕКТИВНОСТІ ЗАСОБІВ ЗАХИСТУ ГОЛОСОВИХ ВИКЛИКІВ У СИСТЕМАХ SMART CITY

2.1 Порівняльний аналіз ефективності протоколів захисту голосових викликів.

Для захисту голосових викликів на міських «гарячих лініях» важливо врахувати кілька критеріїв безпеки. Основними є конфіденційність (шифрування змісту розмови та сигнальних даних), цілісність та автентичність (запобігання непомітній модифікації даних), а також практичні аспекти: затримка і якість зв'язку, обчислювальні витрати та сумісність з VoIP-системами. Нижче проведено порівняння найпоширеніших механізмів: протоколу SIP over TLS, протоколу SRTP і протоколу ZRTP для захисту VoIP.

1. **Конфіденційність і приватність передачі даних.**

Конфіденційність означає, що вміст розмови та пов'язані дані не можуть прочитати сторонні особи. Розглянемо, як кожен з механізмів забезпечує шифрування і приватність:

- **SIP over TLS:** Шифрує сигнальний трафік SIP між клієнтом і сервером за допомогою TLS. Це захищає від прослуховування інформацію про дзвінок (номери, адреси, параметри сесії тощо). Однак TLS забезпечує шифрування лише на транспортному рівні (між кожною парою «клієнт–сервер» або «сервер–сервер»), тобто не є наскрізним від абонента до абонента. Медійні дані (голос) через один TLS не шифруються - для цього потрібен окремий протокол (SRTP тощо). Втім, TLS критично необхідний, якщо для обміну ключами SRTP використовується SIP/SDP: без TLS ключі шифрування голосу можуть передаватися у відкритому вигляді і бути перехоплені зловмисником. Таким чином, TLS суттєво підвищує конфіденційність сигнального обміну і опосередковано захищає медіа-канал, приховуючи ключі шифрування, що передаються по SIP.
- **SRTP:** Забезпечує конфіденційність медіа-потоку (голосу або відео). Він шифрує власне аудіо/відео, що передається по мережі, за

допомогою симетричного алгоритму (наприклад, AES). В результаті, навіть якщо зловмисник перехопить RTP-пакети, їхній корисний вміст (мова) буде зашифрований і виглядатиме як випадковий набір бітів[18]. Важливо, що SRTP шифрує лише payload (корисне навантаження) RTP-пакету, не зачіпаючи заголовки, які потрібні для маршрутизації. Це схоже на відправлення листа в запечатаному конверті: сама посилка (голос) прихована, хоча на конверті все ще видно службову інформацію (адреси, мітки часу). Наприклад, у відкритому RTP дзвінку звук можна реконструювати через Wireshark, а при використанні SRTP - перехоплений аудіо потік звучатиме як шум (шифрований потік не має смислу без ключа). SRTP гарантує конфіденційність розмови: лише сторони, що володіють секретним ключем сеансу, можуть розшифрувати аудіо. Це особливо важливо для захисту приватності громадян, які повідомляють чутливі дані на гарячій лінії.

- **ZRTP:** Це протокол узгодження ключів шифрування по медіа-каналу (RTP) між двома кінцевими вузлами[19]. ZRTP забезпечує конфіденційність голосового трафіку, оскільки дозволяє двом телефонам встановити спільний секретний ключ без передачі його через сервери або відкриті канали. Після узгодження ключа медіа передається по SRTP, тож контент розмови на всьому шляху мережі зашифрований. Перевага ZRTP - відсутність залежності від інфраструктури PKI чи серверів: навіть якщо SIP-сервер компрометовано, він не має доступу до ключів розмови. Таким чином, ZRTP дає **наскрізну** конфіденційність для голосу.

Отже, максимальну конфіденційність контенту дає використання SRTP для голосу, особливо у поєднанні з наскрізним обміном ключами (ZRTP) - тоді ні сигналізація, ні медіа не можуть бути прочитані в мережі. SIP over TLS доповнює SRTP, шифруючи службову інформацію та захищаючи передачу ключів.

2. **Цілісність даних та автентичність абонентів.**

Цілісність означає захист від непомітної модифікації даних, а **автентичність** - можливість підтвердити особу або джерело даних. У випадку голосових викликів це зводиться до того, щоб пакетами голосу чи сигналізації не можна було таємно маніпулювати (вставляти, змінювати або підміняти), а сторони виклику могли переконатися, що з'єднуються саме одна з одною, а не з нападником-посередником. Порівняння механізмів за цими аспектами:

- **SIP over TLS:** TLS забезпечує цілісність сигнальних повідомлень - будь-яка зміна SIP-повідомлення в транзиті буде виявлена, оскільки шифрування TLS включає контрольну суму (HMAC) на зашифровані дані. Автентичність сервера також гарантується сертифікатом: клієнт перевіряє сертифікат SIP-сервера за допомогою довіреного центру сертифікації. Таким чином, TLS запобігає атакам типу «man-in-the-middle» на рівні сигналізації (за умови довіри до сертифіката). Проте автентифікація **клієнта** через TLS зазвичай не проводиться, натомість клієнт автентифікується по паролю (HTTP Digest) вже всередині захищеного каналу. Отже, TLS надає цілісність даних і односторонню автентичність сервера, а автентичність клієнта покладається на інші механізми (паролі, токени).
- **SRTP:** Запроваджує спеціальний аутентифікаційний код для кожного RTP-пакету. Окрім шифрування, до пакету додається хеш-код (автентифікаційний тег), який розраховується з незашифрованого змісту пакету і секретного ключа. За замовчуванням SRTP використовує HMAC-SHA1 (або інші алгоритми) для цього тега. На приймаючій стороні тег перевіряється - якщо пакет було змінено в дорозі, перевірка провалиться. Таким чином, **SRTP забезпечує цілісність медіа-даних** і захищає від їхньої непомітної підробки або вставлення чужих пакетів[20].
- **ZRTP:** Сам по собі ZRTP забезпечує автентичність обміну ключами. Він використовує метод перевірки на відсутність «людини посередині» - обчислення та голосове порівняння короткого **коду перевірки (SAS)**.

Під час рукостискання ZRTP сторони обмінюються хешами своїх DH-ключів і в кінці отримують короткий рядок (наприклад, 4 слова), який користувачі можуть звірити голосом. Якщо рядки співпали, це підтверджує, що ніхто не підміняв ключ (інакше б SAS не збігся). Таким чином, при належній перевірці SAS ZRTP забезпечує **наскрізну автентичність** з'єднання (немає прихованого посередника). Цілісність повідомлень ZRTP також захищена хешами: кожне повідомлення протоколу містить хеш попереднього, утворюючи «ланцюжок», який не дасть вставити фальшиве повідомлення без виявлення.

Усі розглянуті засоби здатні забезпечити цілісність. TLS дає цілісність сигналізації і автентифікує сервер, SRTP (з HMAC) - цілісність голосових пакетів, ZRTP - перевірку відсутності MITM на етапі узгодження ключа. Для **«гарячої лінії»** важливо запобігти спотворенню або підробці повідомлень (наприклад, щоб нападник не перенаправив виклик або не вставив фальшиві команди). Оптимальним є комбінування: TLS + SRTP дають хорошу цілісність на транспортному рівні, а для додаткової впевненості в кінцевих вузлах можна додати перевірку SAS (ZRTP) або впровадити підписування критично важливих даних.

Табл. 2.1.1 - Порівняльна таблиця, що співставляє SIP over TLS, SRTP, ZRTP (конфідненційність; цілісність та автентичність).

Механізм	**Конфіденційність**	**Цілісність**	**Автентичність**
SIP over TLS	Шифрується сигнальний трафік (повністю вся SIP-повідомлення між вузлами); приховує зміст сигналізації (номери, заголовки SIP) від сторонніх.	Контролює цілісність через HMAC у TLS-записах; будь-яка зміна SIP-пакету виявляється і з'єднання розривається.	Автентифікує сервер за сертифікатом (запобігає спуфінгу серверів); клієнт опціонально за сертифікатом або паролем.
SRTP	Шифрує аудіо/відео AES-алгоритмом; голос передається як зашифрований потік (недоступний для прослуховування).	Додає до кожного RTP-пакета MAC-тег (HMAC-SHA1); гарантує, що аудіопакет не змінено (і захист від повторних атак).	Автентичність джерела голосу забезпечується знанням спільного ключа (пакет з правильним HMAC походить від свого); окрема ідентифікація особи не надається протоколом.
ZRTP	Узгоджує секретний ключ для SRTP через	Перевіряє цілісність обміну ключами (хеш-	Автентифікує кінцеві точки через Short Authentication

	обмін Diffie-Hellman; забезпечує end-to-end шифрування медіа без передачі ключа в мережі.	зв’язування повідомлень + підтвердження SAS); гарантує, що спільний ключ отримано без підробки.	String (SAS), який користувачі звіряють вголос; відсутність розбіжностей SAS підтверджує, що ніхто не підміняє співрозмовника

3. Вплив на затримку та якість зв’язку.

Будь-яка криптографія додає певну обробку даних, що може спричинити затримки або додаткове навантаження на мережу. Тут порівнюється, як кожен механізм впливає на **час встановлення дзвінка**, **затримку передачі голосу**, **джиттер** та загальну якість.

- **SIP over TLS:** Основний вплив TLS - на стадії встановлення з’єднання. При першому дзвінку між клієнтом і сервером має відбутися TLS-рукостискання: обмін сертифікатами, шифрувальними наборами, генерування спільного секрету. Цей процес займає кілька додаткових мережевих раундів. У гіршому випадку (TLS 1.2) це може додати сотні мілісекунд до встановлення дзвінка через інтернет. Проте новіша версія TLS 1.3 скоротила кількість раундів. Після встановлення сесії накладні витрати TLS на сам трафік SIP мінімальні: сигналізаційні пакети невеликі за розміром, шифрування/дешифрування їх займає частки мілісекунди. На якість голосу під час розмови TLS не впливає, оскільки сам голос йде окремо (по RTP/SRTP). Отже, **вплив TLS на затримку** - це трохи довший набір дзвінка; у розмові ж різниці немає, якщо з’єднання вже встановлене[21].
- **SRTP:** Шифрування медіа за допомогою SRTP додасть лише незначне накладення на трафік. Сервер або телефон просто додають криптообробку (AES і HMAC) до кожного пакета RTP. Дослідження показують, що це призводить до ~2% збільшення затримки пакетів у порівнянні з незашифрованим RTP. В реальному часі такі кількапроцентні затримки практично непомітні для користувача, тому якість голосу зазвичай не погіршується. Наприклад, при передачі голосу по SRTP до Ethernet додається близько десятків байтів служби, що не

впливає на загальний зсув у часі. Таким чином, затримка і джиттер залишаються на практично такому ж рівні, що й без захисту[22].

- **ZRTP:** Протокол ZRTP виконується при встановленні медіа-сесії, додаючи декілька службових обмінів пакетами. Ці пакети передаються лише раз на початку дзвінка. Затримка, поки триває рукостискання ZRTP, зазвичай становить кілька сотень мілісекунд або менше (залежить від мережевої затримки між абонентами). Цей процес часто відбувається паралельно з встановленням SIP-сеансу або одразу після нього, тому користувач майже не помітить затримки: виклик може з'єднатися і одразу піде голос (протягом менш ніж секунди). Таким чином, **ZRTP додає лише сигналізаційну затримку на початку дзвінка**. На перебіг самої розмови (після встановлення ключів) ZRTP не впливає, оскільки далі працює SRTP.

Протоколи TLS, SRTP, ZRTP спроектовано таким чином, щоб **не погіршувати якість голосового зв'язку**. Таким чином, можна зробити висновок, що **безпека VoIP-сесій досягається без суттєвої шкоди для якості** за умови правильного налаштування. Важливо лише уникати надлишкових операцій (наприклад, не підписувати кожен пакет, де достатньо HMAC) та використовувати оптимізовані бібліотеки шифрування.

4. Обчислювальні витрати та масштабованість.

У цьому розділі порівнюються **обчислювальні витрати** (CPU, пам'ять) при використанні розглянутих механізмів, а також їх вплив на масштабованість системи (кількість одночасних викликів, яке може обслуговувати обладнання).

- **SIP over TLS:** Криптообмін TLS вимагає сильних операцій з відкритими ключами (RSA/ECC) під час встановлення з'єднання. RSA-операції під час хендшейку є ресурсоємними, що і зумовлює вищу затримку. У той же час симетричне шифрування (AES) у TLS-сесії досить легке і впливає незначно на CPU. Отже, перший виклик (новий хендшейк) може «вантажити» процесор, а повторні дзвінки за тими ж сесіями - майже так, як незахищені, завдяки повторному використанню

ключів. З масштабованістю пов’язаний масштаб TLS-сертифікатів та держконекцій: щоб обслуговувати тисячі одночасних дзвінків, серверу потрібна достатня потужність CPU та пам’яті для SSL-контекстів. Для зниження навантаження використовують постійні TCP/TLS-сесії, що істотно зменшує накладні витрати на RSA[23].

- **SRTP:** Перетворення медіапакетів зазвичай виконується за допомогою AES з хешем (HMAC-SHA1). За сучасного обладнання AES-приставки забезпечують апаратне прискорення, тому CPU-затрати на потік аудіо невеликі. Відповідно, навантаження лінійно зростає з кількістю викликів, але кожен виклик додає лише малий відсоток витрат CPU. При використанні SDES-ключів компроміс полягає в тому, що ключі треба передавати по SIP (цілеспрямовано чи в зашифрованому вигляді); при DTLS ключі обмінюються через додатковий протокол Datagram-TLS (що вимагає окремого хендшейку DTLS, але все одно дешевшого за RSA-хендшейк SIP). Загалом, SRTP відрізняється низькими криптовитратами і добре масштабується: він впроваджується «легковаговими» способом без створення масштабованої інфраструктури, особливо якщо апаратно прискорений AES.
- **ZRTP:** В протоколі ZRTP найважчий крок - це обчислення спільного секрету через алгоритм Диффі-Геллмана (DH) або еквівалент. Обчислення DH виконується один раз на виклик на кожному кінці. На сучасному процесорі це десятки мілісекунд роботи. Для одного виклику це не відчутно, але якщо шлюз чи сервер мав би виконувати ZRTP для тисяч дзвінків одночасно, це могло б бути навантаженням. Втім, **ZRTP зазвичай реалізується на кінцевих вузлах**, тобто навантажує лише телефони абонентів. Після встановлення ключів ZRTP більше не споживає CPU, окрім як передає кілька повідомлень. Все шифрування/дешифрування далі робить SRTP, навантаження якого низьке. Тому **обчислювальні витрати ZRTP невеликі на стороні клієнта**, і нульові на стороні інфраструктури (серверів), якщо ті взагалі

не залучені. Можна зазначити, що ZRTP не потребує зберігання сертифікатів чи складних розрахунків довірчих ланцюжків – це також спрощує «криптографічну вагу» рішення.

З точки зору продуктивності, **найбільш «дорогим» є TLS**, особливо на етапі встановлення сесій і при великій кількості одночасних з'єднань. Тим не менш, оптимізація значно пом'якшує цю проблему. **SRTP та HMAC** - дуже ефективні і майже прозорі для CPU, їх можна використовувати навіть на слабких пристроях без побоювань за продуктивність. **ZRTP** додає мінімальне навантаження на клієнт і не навантажує сервери взагалі, що плюс з точки зору масштабованості. Таким чином, з точки зору інфраструктури Smart City: **використання TLS + SRTP цілком сумісне з вимогами продуктивності**, потрібно лише передбачити дещо вищі вимоги до серверів. Експерти рекомендують за замовчуванням вмикати TLS у VoIP-системах, адже його переваги переважують витрати.

5. **Сумісність з VoIP-системами.**

Сумісність означає, наскільки легко даний механізм інтегрувати в існуючі VoIP-середовища, чи підтримують його стандартні телефони, сервери, та чи потрібні додаткові компоненти. Оцінка протоколів:

- **SIP over TLS:** Широко підтримується практично всіма сучасними SIP-серверами і багатьма SIP-клієнтами. Проблеми сумісності можуть виникнути з дуже старим обладнанням або простими SIP-клієнтами, що підтримують лише UDP/TCP. Деякі провайдери VoIP (SIP-транкінгу) досі можуть працювати без TLS, проте багато хто вже надає опцію TLS. Отже, **сумісність TLS висока**.
- **SRTP:** Визначений IETF профіль для RTP (RFC 3711), тому підтримується майже усіма сучасними VoIP-клієнтами й серверами. Наприклад, усі браузери з WebRTC використовують SRTP для передачі аудіо/відео. Підприємницькі PBX часто мають опції увімкнення SRTP (зазвичай в поєднанні з TLS на SIP для безпечної передачі ключів). Є кілька способів обміну ключами (SDES у SDP, DTLS-SRTP, ключі

заздалегідь налаштовані), але самі пакети даних у SRTP майже стандартизовані. Отже SRTP має надзвичайно широку підтримку: для більшості VoIP-систем достатньо лише налаштувати шифрування медіа, без додаткового «зовнішнього» протоколу[24].

- **ZRTP:** Як окремий протокол, ZRTP нечасто присутній у готових рішеннях: його підтримує низка клієнтів, але він рідше зустрічається в комерційних SIP-АТС чи провайдерах. Перевагою є end-to-end захист без PKI, але в практиці це означає складність конфігурування та обмежену сумісність. Більшість бізнес-систем не використовують ZRTP «з коробки» - принаймні не без спеціальних доповнень. Таким чином, сумісність ZRTP оцінюється як обмежена: для використання потрібне спеціалізоване програмне забезпечення (телефони чи додатки, що підтримують ZRTP), а також узгодженість конфігурацій з обох кінців зв’язку.

З точки зору впровадження в існуючі системи, **TLS і SRTP є найбільш сумісними і зрілими технологіями** - їх підтримує більшість виробників і відкритих рішень, вони стандартизовані та задокументовані у посібниках з безпеки VoIP. Багато постачальників послуг уже пропонують TLS/SRTP як опцію для захисту дзвінків. **ZRTP** забезпечує сильнішу наскрізну безпеку, але страждає на меншу сумісність - перед впровадженням треба пересвідчитись, що і телефон абонента, і програма оператора лінії підтримують його; інакше він просто не спрацює. Для інфраструктури «гарячої лінії» раціонально обрати рішення на основі стандартних технологій: TLS для сигналізації, SRTP для аудіо. Це гарантує, що і комерційні IP-телефони, і софтфони легко інтегруються. Якщо потрібен максимальний захист і є можливість контролю за клієнтським ПЗ, можна розглянути використання ZRTP. Але для дзвінків «громадянин – оператор» імовірно доведеться задовольнитися TLS+SRTP, що вже значно підвищує безпеку порівняно з незашифрованим зв’язком.

Табл. 2.1.2 - Порівняльна таблиця, що співставляє SIP over TLS, SRTP, ZRTP (вплив на затримку та якість зв'язку; обчислювальні витрати та масштабованість; сумісність з VoIP-системами).

Механізм	**Вплив на затримку**	**Обчислювальні витрати**	**Сумісність з VoIP**
SIP over TLS	Невелике збільшення часу встановлення дзвінка (handshake); мінімальний вплив на затримку голосу.	Високі на етапі встановлення з'єднання (операції RSA), проте після отримання ключа AES-шифрування незначно навантажує CPU. Масштабування за кількістю з'єднань потребує потужних ресурсів для TLS-сесій.	Широко підтримується стандартними SIP-серверами та телефонами, але потребує налаштування сертифікатів на всіх кінцях.
SRTP	Лише незначне підвищення затримки (~2%), тому голосова якість практично не змінюється.	Низькі криптовитрати (швидке AES-шифрування, апаратне прискорення можливе). Витрати зростають лінійно з кількістю дзвінків.	Дуже широко підтримується; є стандартизовані методи обміну ключами (SDES, DTLS).
ZRTP	Додає ~5 пакетів (ланцюжок DH) на початку виклику - затримка <0.5 с; надалі не впливає. SAS-звірка може додати кілька секунд (якщо виконується вручну).	Середні: одна операція DH/ECDH на виклик (несуттєво для CPU); сервери не навантажує.	Середня/Низька: підтримується обмеженим числом клієнтів; стандартні IP-телефони переважно не мають ZRTP. Серверам підтримка не потрібна.

На основі аналізу можна зробити такі висновок: SIP over TLS + SRTP на сьогодні є оптимальним рішенням для більшості випадків: вони забезпечують конфіденційність і цілісність на достатньо високому рівні, підтримуються типовим обладнанням і не викликають суттєвих проблем з якістю зв'язку. Для захисту «гарячої лінії» у Smart City ця зв'язка закриває основні ризики - дзвінок шифрується і на рівні сигналізації, і на рівні медіа, що унеможливлює перехоплення або підробку трафіку на шляху від абонента до call-центру. Експерти наголошують, що TLS для VoIP слід ввімкнути за замовчуванням, оскільки ризики від його невикористання значні (крадіжка паролів SIP, підробка викликів, витік аудіо через перехоплені ключі тощо), а вплив на систему мінімальний.

2.2 Оцінка ефективності механізмів забезпечення цілісності та автентифікації голосових викликів.

Захист голосових викликів у системах «Smart City» потребує гарантування того, що передані аудіодані не були змінені під час передачі, а також підтвердження джерела виклику. Для цього використовують криптографічні механізми контролю цілісності та автентичності повідомлень. Зокрема, хеш-функції (SHA-256, SHA-3) забезпечують **перевірку цілісності** даних, MAC-коди (HMAC-SHA1, HMAC-SHA256) - поєднану цілісність і автентифікацію на основі секретного ключа, а цифрові підписи (RSA-2048, ECDSA-P256) - **неспростовне підтвердження джерела повідомлення** та захист від підробки. Нижче проведено порівняльний аналіз цих методів у контексті VoIP-сесій Smart City за наведеними критеріями.

1. Геш-функції для перевірки цілісності голосових даних.

Криптографічні геш-функції формують унікальний «відбиток» (хеш) від даних голосового виклику, за яким можна перевірити цілісність. У контексті VoIP це може бути застосовано, наприклад, для контрольних сум RTP-пакетів або для пост-аналізу запису виклику. Якщо отриманий хеш співпадає з очікуваним, дані не були змінені під час передачі або зберігання. Сучасні геш-алгоритми сімейства SHA-2 (SHA-256/384/512) та SHA-3 забезпечують високий рівень криптостійкості (ймовірність випадкового збігу надзвичайно мала, колізій практично недосяжно при сучасних обчислювальних можливостях). Наприклад, SHA-256 формує 256-бітний підпис даних, який стійкий до підробки - будь-яка найменша зміна голосового пакету призведе до зовсім іншого значення хешу, що легко виявити при перевірці. SHA-3 є альтернативою з іншою криптографічною конструкцією і порівнянним рівнем безпеки. За оцінками, на типових CPU SHA-256 дещо швидший, ніж SHA-3, хоча в певних апаратних реалізаціях SHA-3 може мати переваги. В цілому, обидва алгоритми придатні для контролю цілісності голосових даних і значно перевершують за надійністю застарілі хеші[25].

Використання в Smart City: Геш-функції самі по собі забезпечують контроль цілісності, але не гарантують автентичності джерела. Якщо зловмисник може перехопити і змінити голосові пакети, він може просто обчислити новий хеш і підмінити його - одержувач побачить коректний хеш і не помітить підміни. Тому в реальному часі геш-функції застосовуються або як допоміжний механізм, або для офлайн-перевірки цілісності. В ситуації дзвінків на гарячі лінії міста, гешування може використовуватися для отримання контрольної суми всього сеансу або окремих фрагментів мовлення - надалі цей хеш може бути підписаний цифровим підписом відповідного вузла, щоб гарантувати незмінність запису та його походження. Такий підхід забезпечує невідмовність, оскільки ніхто, окрім власника секретного ключа, не міг згенерувати підпис до хешу даного виклику.

Продуктивність: Обчислення сучасних геш-функцій є доволі швидким і ефективним, особливо з огляду на апаратні оптимізації. Для потоку голосових даних накладні витрати на обчислення SHA-256 чи SHA-3 для кожного RTP-пакету мінімальні, тож це не створює відчутної затримки. Проте, без додаткового механізму автентифікації просте передавання хешів не захищає від цілеспрямованої підміни.

2. Цифрові підписи для автентифікації джерела виклику.

Цифровий підпис дозволяє перевірити, що голосовий потік або сигналізація походять саме від довіреного джерела. Для системи Smart City це означає, що вузол гарячої лінії може впевнитися, що дзвінок надійшов від легітимного пристрою/користувача, а не від підробленого абонента. Популярні алгоритми цифрового підпису мають різні характеристики. **RSA-2048** - класичний підхід на основі факторизації, що дає ~112 біт безпеки; довжина підпису ~256 байт. **ECDSA-P256** (еліптичні криві, ключ 256 біт) забезпечує подібний рівень безпеки, але підпис коротший (~64 байти) і генерація/перевірка, як правило, швидші за RSA. У сучасних системах спостерігається тенденція переходу до алгоритмів на основі еліптичних кривих через їх підвищену ефективність при еквівалентній стійкості. Дійсно,

дослідження показують, що заміна RSA на ECDSA радикально підвищує продуктивність системи автентифікації викликів[26]. На **рис. 2.2** наведено порівняння продуктивності RSA та ECDSA при підписуванні викликів: видно, що ECDSA дозволяє обробити в кілька разів більше з'єднань одночасно при тій самій гарантії безпеки.

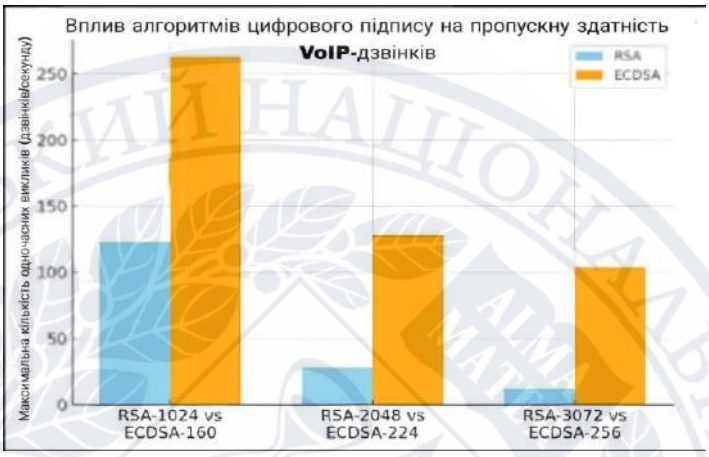

Рис. 2.2 - Максимальна пропускна здатність викликів при використанні цифрових підписів RSA vs ECDSA для автентифікації SIP-викликів.

Як видно, використання RSA-підпису на кожне повідомлення виклика суттєво обмежує швидкодію системи (навантаження на CPU зростає). NIST також попереджає, що протоколи, які підписують **кожен** пакет або повідомлення голосового сеансу, можуть мати критичний вплив на продуктивність реального часу. Тому в реальних VoIP-системах цифрові підписи застосовуються обмежено - переважно на етапах встановлення з’єднання або обміну ключами, де вони необхідні для початкової автентифікації, після чого для захисту самих медіаданих використовується швидший симетричний захист.

Сумісність з VoIP: Стандартні протоколи IP-телефонії вже передбачають способи інтеграції цифрових підписів. У SIP присутній механізм Identity header, де до запиту виклику додається маркер з цифровим підписом, який підтверджує номер та ім’я викликача, підписаний оператором походження; на боці завершення виклику цей підпис перевіряється по сертифікату в рамках програми STIR/SHAKEN для боротьби з підробкою номерів. Таким чином, **цифровий**

підпис ефективно вирішує проблему автентифікації джерела виклику і захищає від спуфінгу (на рівні сигналізації). В середовищі Smart City, де можливі виклики від різних служб або підсистем, використання сертифікатів і підписів в загальній інфраструктурі дозволяє побудувати довірчу мережу: кожен вузол має свій ключ і сертифікат, яким підписує свої повідомлення, забезпечуючи взаємну довіру.

Затримка: Операції підпису/перевірки за допомогою RSA-2048 чи ECDSA-P256 займають порядку кількох мілісекунд на сучасних CPU (в залежності від оптимізації). Це прийнятно для ініціалізації дзвінка, яка триває десятки мс, але **неприйнятно для кожного аудіо-пакета** (RTP-пакети надходять кожні ~20 мс). Як наслідок, пряма реалізація «підпис кожного пакету» створила б значні затримки і джиттер, порушивши реальний час передачі голосу. Тому на практиці підпис можуть ставити, наприклад, не на кожен RTP-пакет, а на групи пакетів або на контрольну суму всього потоку. Один із варіантів - генерувати цифровий підпис після завершення сеансу на фінальний хеш всього аудіопотоку і зберігати його для доказу цілісності запису.

3. MAC-коди для захисту від підміни повідомлень.

Код автентифікації повідомлення (Message Authentication Code) - це перевірочне значення, яке обчислюється на основі повідомлення та секретного ключа, спільного між відправником і одержувачем. У контексті голосових викликів найпоширенішим є **HMAC (Keyed-Hash MAC)**, зокрема на основі SHA-1 або SHA-256. MAC забезпечує як **цілісність** (будь-яка зміна в даних призводить до невідповідності MAC-коду при перевірці), так і **автентичність** для сторін, що володіють ключем (ніхто інший не може сформувати правильний код без знання секретного ключа). На відміну від цифрового підпису, MAC є симетричним механізмом: обидві сторони мають спільний секрет. Це означає, що MAC не надає невідмовності, але для захисту від зовнішніх зловмисників він дуже ефективний.

Використання в VoIP: Протокол **SRTP (Secure RTP)**, який є стандартом шифрування і автентифікації медіа-трафіку в IP-телефонії, передбачає саме використання HMAC для забезпечення цілісності кожного RTP-пакету.

Стандартним вибором є HMAC-SHA1 з 160-бітним виходом, який **трасується до 80 біт** для скорочення службових даних (80 біт достатньо для захисту, ризик підбору ключа чи колізії мінімальний). Секретний ключ HMAC встановлюється для сесії за допомогою протоколів обміну (наприклад, через SDES, DTLS-SRTP або ZRTP). Таким чином, кожен RTP-пакет несе 80-бітний MAC-тег, що дозволяє отримувачу перевірити, що пакет дійсно від легітимного відправника і не був змінений.

Продуктивність: MAC на основі геш-функцій є значно легшим обчислювально, ніж цифровий підпис. Обчислення HMAC-SHA1 для невеликого пакету виконується дуже швидко (симетричні криптооперацї на порядок швидші за асиметричні). Тому для реального часу MAC є оптимальним вибором: він дозволяє перевіряти кожен пакет без помітної затримки. На відміну від цифрового підпису, MAC не потребує важких розрахунків з великими числами; він спирається на ті ж геш-алгоритми (SHA-1, SHA-256), але доповнює їх секретним ключем.

Сумісність і реалізація: MAC-примітиви легко інтегруються в існуючі VoIP-протоколи. У випадку SRTP - передбачене поле для автентифікаційного тега. У сигнальних протоколах теж можливе використання MAC для автентифікації повідомлень, але зазвичай там віддають перевагу асиметричним методам або TLS. У системі “розумного міста” можна уявити, що внутрішні вузли (наприклад, шлюзи, сервери) мають спільні ключі для швидкої перевірки справжності службових повідомлень. Проте для масштабованості краще кожну сесію захищати окремим MAC-ключем, отриманим в ході встановлення виклику.

Кожен з описаних методів має свої сильні та слабкі сторони в контексті голосових викликів. Підсумуємо їх порівняння за ключовими критеріями у **Таблиці 2.2.**

Табл. 2.2 - Порівняння механізмів забезпечення цілісності та автентифікації голосових викликів

Критерій	**Хеш-функція (SHA-256, SHA-3)**	**MAC HMAC (HMAC-SHA1, HMAC-SHA256)**	**Цифровий підпис (RSA-2048, ECDSA-P256)**

Захист від MITM-атак	Відсутній (немає ключа, MITM може перерахувати хеш)	Забезпечується при секретному ключі (зміна даних виявляється)	Повний: без приватного ключа підробити підпис неможливо
Захист від replay-атак	Немає (повтор пакету дасть той самий хеш)	Потрібні додаткові механізми (номер пакета, «вікно повторів» SRTP)	Підпис з таймстемпом/номером ускладнює повтор; в іншому разі потрібна перевірка цілого сеансу
Захист від підміни аудіо	Немає автентифікації (MITM може змінити аудіо та змінити хеш)	Зміна виявиться по некоректному MAC-тегу (підміна виявляється)	Підміна призведе до невірної перевірки підпису (захищає цілісність)
Витрати ресурсів (затримка)	Дуже низькі (кілька десятків мкс на пакет)	Невеликі (помітні, але сотні мкс на пакет)	Високі при підписі (мс на пакет або підпис інтервалу); перевірка теж повільніша ніж HMAC
Сумісність з SRTP/SIP TLS/ZRTP	Не використовується напряму (лише у складі HMAC або сертифікатів)	Підтримується SRTP (HMAC-SHA1 за замовчуванням); сумісний з SIP-TLS (HMAC як MAC); ZRTP – частково через SAS	SRTP/SIP TLS: сертифікати (RSA/ECDSA) для ключів; **пряма підпис** не стандартизована

2.3 Висновки до розділу 2.

У другому розділі проведено глибокий аналіз сучасних механізмів забезпечення конфіденційності, цілісності та автентичності голосових викликів у системах Smart City. Зокрема, було здійснено порівняльний аналіз ефективності найпоширеніших протоколів і технологій захисту VoIP-зв'язку: SIP over TLS, SRTP та ZRTP, а також механізмів забезпечення цілісності й автентифікації голосових даних.

Було встановлено, що з точки зору забезпечення конфіденційності найбільш надійним підходом є поєднання SIP over TLS (для шифрування сигнальних даних) та SRTP (для шифрування аудіо-даних). Протокол ZRTP має значну перевагу в забезпеченні наскрізного шифрування і конфіденційності завдяки можливості прямого обміну ключами між кінцевими користувачами без використання централізованої інфраструктури сертифікації. Втім, широке впровадження ZRTP може бути ускладнене через обмежену сумісність зі

стандартними VoIP-системами та необхідність спеціалізованих програмних клієнтів.

Аналіз механізмів цілісності та автентифікації показав, що використання протоколу SRTP, який застосовує HMAC-коди для перевірки цілісності кожного аудіо-пакету, є оптимальним рішенням для захисту голосового трафіку в реальному часі. Хоча цифрові підписи (RSA-2048, ECDSA-P256) надають більш високий рівень захисту від підміни даних та забезпечують автентичність джерела, їх застосування до кожного аудіо-пакета не є практичним через значні затримки та високі вимоги до ресурсів. Тому доцільним є їхнє застосування лише на ключових етапах встановлення з’єднання або для довгострокового зберігання записів викликів з підтвердженням їх цілісності.

РОЗДІЛ 3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО РІШЕННЯ

3.1 Вибір середовища розробки та інструментів.

Для реалізації системи обрано мову програмування Python. Це зумовлено її високим рівнем абстракції, читабельністю та наявністю численних бібліотек для різноманітних завдань. Як відзначають дослідники, «Python із широким спектром бібліотек забезпечує потужну платформу для реалізації й удосконалення аудіообробних технологій». Завдяки простоті та адаптивності Python дозволяє швидко прототипувати рішення, включаючи криптографічні операції й обробку звуку. Зокрема, у середовищі Python доступні реалізації стандартних алгоритмів шифрування (AES, RSA, DSA тощо) і функцій цифрових підписів. Це дає можливість ефективно реалізувати функції конфіденційності та цілісності даних у дзвінках гарячих ліній. Також підкреслюється, що «Python з його широкою підтримкою бібліотек і простотою використання є ідеальною платформою для реалізації алгоритму AES, що забезпечує надійне та ефективне шифрування аудіоданих». Таким чином, вибір Python обґрунтований його зручністю розробки, кросплатформеністю та готовими рішеннями для криптографії та обробки аудіо.

PyCryptodome - багатофункціональна криптографічна бібліотека для Python, яка підтримує алгоритми симетричного шифрування, гешування та цифрових підписів. У прототипі вона використовується для генерації випадкових ключів, шифрування даних за алгоритмом AES-256 у режимі CBC, обчислення геш-функції SHA-256 та створення/перевірки цифрових підписів ECDSA. PyCryptodome є форком PyCrypto з покращеним забезпеченням безпеки і широко застосовується у дослідницьких проектах (зокрема, для реалізації стандартних криптопрактимів).

PyDub - зручна бібліотека для роботи з аудіофайлами формату WAV/MP3. Вона спрощує завантаження, конвертацію, розділення та збереження аудіо через простий інтерфейс Python. У прототипі PyDub дозволяє читати голосові записи у бажаному форматі, за потреби конвертувати їх між WAV та MP3, розбивати на частини, а також зберігати після обробки. Використання PyDub прискорює

розробку обробки аудіо та взаємодію з форматом файлів без необхідності вручну працювати з бітовими потоками аудіо.

Програмна модель реалізації включає:

- Зчитування та попередню обробку аудіофайлу (за допомогою PyDub).
- Шифрування аудіопотоку симетричним алгоритмом AES-CBC з ініціалізаційним вектором.
- Генерацію та накладення електронного цифрового підпису (алгоритм ECDSA на базі кривої ECC) для захисту цілісності.
- Обчислення контрольної суми SHA-256 для перевірки цілісності переданих даних.
- Збереження зашифрованого аудіо та відповідних метаданих (ключів, підпису, IV) з подальшим відновленням і розшифруванням на приймальному боці.

Застосування зазначених інструментів та середовища розробки дозволяє забезпечити необхідний рівень захис ту конфіденційності та цілісності голосових даних, що є критично важливим у контексті роботи гарячих ліній Smart City.

3.2 Реалізація алгоритму шифрування дзвінків.

Для захисту конфіденційності записів дзвінків на гарячі лінії системи "Smart City" розроблено програму, що використовує симетричне шифрування та цифровий підпис. На етапі шифрування аудіодані дзвінка перетворюються у зашифрований вигляд за допомогою алгоритму AES (Advanced Encryption Standard) у режимі CBC (Cipher Block Chaining). AES - це сучасний симетричний блоковий алгоритм шифрування, який при довжині ключа 256 біт (AES-256) забезпечує високий рівень криптостійкості. Режим CBC забезпечує додаткову стійкість шляхом змішування кожного блоку відкритих даних з попереднім блоком шифрованих, використовуючи початковий вектор. Перед шифруванням генерується випадковий IV (Initialization Vector, ініціалізаційний вектор) -

випадкова послідовність 16 байтів, яка забезпечує унікальність шифрування кожного дзвінка навіть при однаковому ключі і даних.

Першим кроком є імпорт необхідних бібліотек та завантаження аудіоданих дзвінка. Програма написана мовою Python з використанням бібліотеки PyCryptodome та PyDub для роботи з аудіофайлами. Нижче наведено фрагмент коду, що імпортує бібліотеки та зчитує аудіофайл дзвінка у форматі WAV:

```
# Імпорт бібліотек
from Crypto.Cipher import AES # AES для симетричного шифрування
from Crypto.Random import get_random_bytes # Генерація криптографічно стійких випадкових байтів
from Crypto.Hash import SHA256 # SHA-256 для хешування даних
from Crypto.PublicKey import ECC # ECC для генерації ключів ECDSA (еліптична крива)
from Crypto.Signature import DSS # DSS (Digital Signature Standard) для підпису/перевірки ECDSA
from Crypto.Util.Padding import pad, unpad # Допоміжні функції для доповнення (padding)
from pydub import AudioSegment # PyDub для роботи з аудіофайлами

# Завантаження аудіофайлу WAV за допомогою PyDub (WAV)
audio = AudioSegment.from_wav(r'C:\Users\Hp\Downloads\call_101_0.wav')

# Конвертація аудіо у сирі байти для обробки
audio_bytes = audio.raw_data
```

Далі генерується секретний ключ для AES та ініціалізаційний вектор. Вибирається довжина ключа 256 біт (32 байти) для забезпечення високої стійкості шифру. Генерація здійснюється за допомогою криптографічно стійкого генератора **get_random_bytes**. Після цього дані дзвінка шифруються алгоритмом AES-256 у режимі CBC. Оскільки AES є блочним шифром з розміром блоку 128 біт (16 байтів), аудіодані доповнюються до кратної 16 байтам довжини (padding). Доповнення забезпечує коректне шифрування навіть якщо довжина даних не кратна розміру блоку. Фрагмент коду нижче реалізує генерацію ключових параметрів та шифрування аудіобайтового потоку:

```
# Генерація 256-бітового ключа AES та 128-бітового IV
key = get_random_bytes(32)  # 32 байти = 256 біт для ключа AES-256
iv = get_random_bytes(16)   # 16 байтів = 128 біт для IV (ініціалізаційного вектора)

# Налаштування шифра AES в режимі CBC з згенерованими ключем та IV
cipher = AES.new(key, AES.MODE_CBC, iv)
```

```
# Доповнення аудіоданих до кратного 16 байт і шифрування
padded_audio = pad(audio_bytes, AES.block_size) # AES.block_size = 16 байт
encrypted_audio = cipher.encrypt(padded_audio) # Отримуємо зашифровані дані дзвінка
```

На цьому етапі конфіденційність забезпечена, але необхідно потурбуватися про цілісність даних. Щоб мати можливість перевірити, чи не були зашифровані дані змінені або пошкоджені, у програмі реалізовано створення цифрового підпису. Для цього використовується алгоритм цифрового підпису на еліптичних кривих ECDSA. Спершу від зашифрованих даних обчислюється криптографічний геш алгоритмом SHA-256. Після отримання гешу генерується пара ключів ECDSA: приватний ключ (для підпису) та відповідний публічний ключ (для перевірки). Приватний ключ залишається в пам’яті (його необхідно берегти у таємниці), а публічний ключ можна розповсюджувати для перевірки підпису. Далі приватним ключем підписується раніше обчислений геш зашифрованих даних, формуючи цифровий підпис. Наведемо відповідний фрагмент коду:

```
# Створення SHA-256 хеш зашифрованих даних
hash_obj = SHA256.new(encrypted_audio) # Обчислення дайджеста SHA-256 для шифротексту

# Генерація пари ключів для ECDSA (еліптична крива P-256)
private_key = ECC.generate(curve='P-256') # Генерація приватного ключа (ключ підпису)
public_key = private_key.public_key() # Отримання відповідного публічного ключа

# Формування цифрового підпису ECDSA на основі гешу
signer = DSS.new(private_key, 'fips-186-3') # Ініціалізація об’єкта підписувача (DSS, стандарт FIPS 186-3)
signature = signer.sign(hash_obj) # Створення цифрового підпису (ECDSA) для хешу даних
```

Останнім кроком в процесі шифрування є збереження результатів у файли. Зашифрований аудіо-вміст дзвінка, цифровий підпис та відкритий ключ зберігаються на диск. Це потрібно, щоб у подальшому можна було передати або відправити їх отримувачу та здійснити перевірку і розшифровку. Ініціалізаційний вектор IV також зберігається, оскільки він необхідний для розшифрування (IV не є секретним, але повинен бути достеменно відомим одержувачу). Приватний ECDSA-ключ не зберігається у файл і залишається відомим лише відправнику. Симетричний AES-ключ у даному прикладі теж

залишається в пам’яті програми; в реальній системі його слід передати захищеним шляхом отримувачу або зберігати в безпечному сховищі. Нижче наведено фрагмент коду, який демонструє збереження зашифрованих даних та пов’язаних артефактів, а також фінальні повідомлення програми:

```
# Збереження зашифрованих даних та підпису у файли
with open(r'C:\Users\Hp\Documents\encrypted_audio.bin', 'wb') as f:
    f.write(encrypted_audio) # Збереження зашифрованого аудіопотоку

with open(r'C:\Users\Hp\Documents\signature.sig', 'wb') as f:
    f.write(signature) # Збереження підпису (бінарний формат)

with open(r'C:\Users\Hp\Documents\public_key.pem', 'wt') as f:
    f.write(public_key.export_key(format='PEM')) # Збереження відкритого ключа у форматі
PEM

with open(r'C:\Users\Hp\Documents\iv.bin', 'wb') as f:
    f.write(iv) # Збереження IV (ініціалізаційний вектор)
```

3.3 Перевірка цільності переданих даних.

Після того як дані дзвінка було зашифровано і необхідні файли сформовано, сторона-отримувач або відповідний модуль системи "Smart City" повинен перевірити цілісність отриманої інформації перед розшифровкою. Перш за все, необхідно завантажити раніше збережені файли: зашифрований вміст дзвінка, цифровий підпис та відкритий ключ. У нашому прикладі ці файли були збережені у каталозі **C:\Users\Hp\Documents\.** Окрім того, завантажується IV, потрібний для розшифрування AES-CBC. Нижче наводиться код для зчитування файлів і ініціалізації даних для перевірки та дешифрування:

```
# Перевірка цілісності
# Зчитування зашифрованих даних, підпису, відкритого ключа
with open(r'C:\Users\Hp\Documents\encrypted_audio.bin', 'rb') as f:
    encrypted_audio_check = f.read() # Завантаження шифротексту дзвінка

with open(r'C:\Users\Hp\Documents\signature.sig', 'rb') as f:
    signature_check = f.read() # Завантаження підпису ECDSA

with open(r'C:\Users\Hp\Documents\public_key.pem', 'rt') as f:
    public_key_check = ECC.import_key(f.read()) # Імпорт відкритого ключа із PEM-файлу
```

Для перевірки підпису повторно обчислюється геш SHA-256 від отриманого шифротексту і порівнюється з цифровим підписом за допомогою публічного ключа. Якщо при підписанні використовувався той самий відкритий ключ (а приватний ключ залишався в таємниці), успішна перевірка означатиме, що дані не змінилися. У разі, якщо зашифровані дані було пошкоджено або підпис підроблено, перевірка не пройде (метод verify згенерує виключення). Наведемо код перевірки цілісності:

```
# Обчислюємо SHA-256 хеш для перевірки
hash_check = SHA256.new(encrypted_audio_check)

# Перевірка цифрового підпису за допомогою відкритого ключа
verifier = DSS.new(public_key_check, 'fips-186-3') # Ініціалізація об’єкту перевірки підпису
try:
    verifier.verify(hash_check, signature_check)
    print("Підпис дійсний, цілісність підтверджено!")
except ValueError:
    print("Підпис не дійсний, цілісність порушена!")
```

Якщо цифровий підпис підтвердив цілісність, можна переходити до розшифрування аудіоданих дзвінка. Розшифрування виконується алгоритмом AES-256 у режимі CBC з використанням того самого секретного ключа і IV, що застосовувалися при шифруванні. У даній реалізації ключ key та ініціалізаційний вектор iv збережені в оперативній пам'яті програми з етапу шифрування. Якщо б перевірка та розшифрування виконувалися окремим модулем, необхідно було б передати або завантажити секретний ключ з безпечного джерела. Після розшифрування отримуються початкові доповнені байти аудіо, з яких видаляється доданий паддінг (відновлюємо оригінальну довжину даних). Далі байти аудіо конвертуються назад у аудіоформат WAV і зберігаються у файл, що відтворює початковий дзвінок. Наведемо фінальний фрагмент коду, який виконує розшифрування та відновлення аудіо:

```
# Розшифрування даних AES-256-CBC за допомогою ключа та IV
cipher_dec = AES.new(key, AES.MODE_CBC, iv) # Ініціалізація об’єкту AES для розшифрування
decrypted_padded = cipher_dec.decrypt(encrypted_audio_check) # Розшифрування байтів (ще містять паддінг)
```

```
decrypted_audio_bytes = unpad(decrypted_padded, AES.block_size) # Видалення паддінгу, отримання оригінальних аудіо-байтів

# Створюємо аудіо з розшифрованих байтів та зберігаємо
recovered_audio = AudioSegment(
    data=decrypted_audio_bytes,
    sample_width=audio.sample_width,
    frame_rate=audio.frame_rate,
    channels=audio.channels
)
recovered_audio.export('recovered_call.wav', format='wav')
print("Розшифрування завершено. Аудіо відновлено у файлі recovered_call.wav.")
```

Таким чином, розроблена програма успішно шифрує конфіденційні дані дзвінка та підписує їх для забезпечення цілісності, а потім перевіряє підпис на одержаному боці і розшифровує дані назад в аудіо. Запропонована схема гарантує, що запис дзвінка на гарячу лінію не зможе бути прочитаний без відповідного ключа (конфіденційність), і що будь-яка модифікація шифрованого файлу буде виявлена під час перевірки підпису (цілісність). Це особливо важливо для системи "Smart City", де захист даних мешканців і гарантія їх незмінності є критичною.

3.4 Висновки до розділу 3.

У третьому розділі розроблено та реалізовано прототип програмного забезпечення для захисту голосових викликів на гарячі лінії системи «Smart City». Запропоновано алгоритм, у якому аудіодані дзвінків шифруються за допомогою симетричного алгоритму AES-256 у режимі CBC, що гарантує високий рівень конфіденційності. Одночасно реалізовано механізм електронного цифрового підпису на основі ECDSA (еліптична крива P-256) із геш-функцією SHA-256 для забезпечення цілісності даних. Після генерації симетричного ключа та ініціалізаційного вектора, аудіофайл шифрується, а отриманий шифротекст підписується приватним ключем ЕЦП. На прийомній стороні здійснюється перевірка цифрового підпису відкритим ключем, у разі успішної валідації дані розшифровуються до початкового формату аудіо.

Таким чином, розроблена програма реалізує повний захист голосових викликів: запис дзвінка залишається недоступним стороннім без відповідного ключа шифрування, а будь-яка несанкціонована зміна зашифрованих даних

автоматично виявляється під час перевірки цифрового підпису. Використані алгоритми AES-256 та ECDSA відповідають сучасним стандартам криптографічного захисту інформації, що забезпечує відповідність рішення актуальним вимогам кібербезпеки. Розроблений прототип має високу практичну цінність: його можна інтегрувати в інфраструктуру системи «Smart City» для захищеного зберігання та передачі записів голосових викликів.

ВИСНОВКИ

У роботі проведено аналіз існуючих загроз щодо конфіденційності та цілісності голосових дзвінків на гарячі лінії системи «Smart City». Проаналізовано сучасні протоколи та механізми захисту, виокремлено та охарактеризовано основні ризики. Встановлено, що для захисту конфіденційності необхідно застосовувати сучасні алгоритми симетричного шифрування (наприклад AES) з регулярною зміною ключів, а для забезпечення цілісності - криптографічні хеш-функції (SHA-256) чи механізми HMAC для автентифікації голосових пакетів. Такий підхід дозволяє суттєво знизити вразливість каналів зв’язку.

У практичній частині роботи реалізовано програмне рішення, яке забезпечує шифрування та перевірку цілісності голосових даних. Розроблена програма, реалізована мовою Python, успішно опрацьовує вхідні аудіофайли, виконує шифрування та перевірку даних, забезпечуючи відновлення голосового сигналу без втрат якості.

Таким чином, проведене дослідження і створений програмний засіб доводять ефективність застосування криптографічних методів захисту у системі «Smart City» для забезпечення безпечних голосових комунікацій. Отримані результати свідчать, що підвищення конфіденційності та цілісності дзвінків на гарячі лінії є критично важливим аспектом безпеки міської інформаційної інфраструктури. Запропонований підхід підвищує надійність і стійкість каналів зв’язку, сприяє захисту персональних і службових даних користувачів. Забезпечення захисту таких сервісів сприяє зміцненню довіри громади до технологій Smart City та підвищенню оперативності реагування у кризових ситуаціях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Tai-hoon Kim, Carlos Ramos , Sabah Mohammed. «Smart City and IoT», 2017. URL:https://www.sciencedirect.com/science/article/abs/pii/S0167739X17305253 (дата звернення: 08.04.2024).

2. Fadi AI-Turjman, Muhammad Imran. «IoT Technologies in Smart Cities: From sensors to big data, security and trust», 2020 (дата звернення: 08.04.2024).

3. Ahmed Samy Nassar, Ahmed Hossam Montasser, Nashwa Abdelbak. «A Survey on Smart Cities' IoT», 2018. URL: https://www.researchgate.net/publication/319404288_A_Survey_on_Smart_Cities'_IoT (дата звернення: 10.04.2024).

4. Journal of Sensor and Actuator Networks (JSAN). «Smart City: The Different Uses of IoT Sensors», 2022. URL: https://www.researchgate.net/publication/363754223_Smart_City_The_Different_Uses_of_IoT_Sensors (дата звернення: 11.04.2024).

5. Seloua Haddaoui, Salim Chikhi, Miles Badreddine. «The IoT Ecosystem: Components, Architecture, Communication Technologies, and Protocols», 2022. URL: https://www.researchgate.net/publication/364331685_The_IoT_Ecosystem_Components_Architecture_Communication_Technologies_and_Protocols (дата звернення: 11.04.2024).

6. Aditya Gaura, Bryan Scotneya, Gerard Parra, Sally McCleana. «Smart City Architecture and its Applications based on IoT», 2015. URL: https://www.researchgate.net/publication/278726450_Smart_City_Architecture_and_its_Applications_Based_on_IoT (дата звернення: 12.04.2024).

7. Lo'ai Tawalbeh , Fadi Muheidat, Mais Tawalbeh, Muhannad Quwaider. «IoT Privacy and Security: Challenges and Solutions», 2020. URL: https://www.mdpi.com/2076-3417/10/12/4102 (дата звернення: 12.04.2024).

8. Gift Matsemela, Suvendi Rimer, Khmaies Ouahada, Richard Ndjiongue, Zinhle Mngomezulu. «Internet of things data integrity», 2017 (дата звернення: 12.04.2024).

9. Onais Ahmad, Gautami Tripathi, Farheen Siddiqui, Mohammad Afshar Alam, Mohd Abdul Ahad, Mohd Majid Akhtar, Gabriella Casalino. «BAuth-ZKP - A Blockchain-Based Multi-Factor Authentication Mechanism for Securing Smart Cities», 2023. URL: https://www.researchgate.net/publication/369008077_BAuth-ZKP-A_Blockchain-Based_Multi-Factor_Authentication_Mechanism_for_Securing_Smart_Cities (дата звернення: 13.04.2024).

10. Osama Younes, Umar Albalawi. «Securing Session Initiation Protocol», 2022. URL: https://www.mdpi.com/1424-8220/22/23/9103 (дата звернення: 17.04.2024).

11. «Attacks and Defenses Against Voice over IP (VoIP)», 2015. URL: https://www.cs.tufts.edu/comp/116/archive/fall2015/sshaidani.pdf (дата звернення: 19.04.2024).

12. Ghazi Al Sukkar1, Ramzi Saifan, Sufian Khwaldeh, Mahmoud Maqableh, Iyad Jafar. «Address Resolution Protocol (ARP): Spoofing Attack and Proposed Defense», 2016. URL: https://www.researchgate.net/publication/305336234_Address_Resolution_Protocol_ARP_Spoofing_Attack_and_Proposed_Defense (дата звернення: 17.04.2024).

13. Christoforos Ntantogian, Eleni Veroni, Georgios Karopoulos, Christos Xenakis. «A Survey of Voice and Communication Protection Solutions Against Wiretapping», 2019 (дата звернення: 19.04.2024).

14. Gideon Areo. «TLS-Enabled SIP Server Security: A Comprehensive Performance Evaluation», 2024. URL: https://www.researchgate.net/publication/388671385_TLS-Enabled_SIP_Server_Security_A_Comprehensive_Performance_Evaluation (дата звернення: 19.04.2024).

15. Dr. Ritesh Sadiwala. «Analysis of Security Threats of VoIP Systems», 2018. URL: https://www.academia.edu/37819034/Analysis_of_Security_Threats_of_VoIP_Systems (дата звернення: 21.04.2024).

16. Matt Coser. «SRTP and You: A Deep Dive into Encrypted VoIP Communications», 2022 (дата звернення: 21.04.2024).

17. Dominik Schürmann, Fabian Kabus, Gregor Hildermeier, Lars Wolf. «Wiretapping End-to-End Encrypted VoIP Calls: Real-World Attacks on ZRTP», 2017. URL: https://petsymposium.org/popets/2017/popets-2017-0025.pdf (дата звернення: 21.04.2024).

18. Dimitrios Alvanos, Konstantinos Limniotis, Stavros Stavrou. «On the Cryptographic Features of a VoIP Service», 2018. URL: https://www.mdpi.com/2410-387X/2/1/3 (дата звернення: 02.05.2024).

19. Riccardo Bresciani, Andrew Butterfield. « ProVerif Analysis of the ZRTP Protocol». URL: https://www.researchgate.net/publication/229051596_ProVerif_Analysis_of_the_ZRTP_Protocol (дата звернення: 03.05.2024).

20. Levi Perigo, Rahil Gandotra, Dewang Gedia, Moiz Hussain, Praniti Gupta, Shirin Bano, Vineet Kulkarni. «VoIP SECURITY: A PERFORMANCE AND COST-BENEFIT ANALYSIS», 2020. URL: https://it-in-industry.com/itii_papers/2020/8220itii04.pdf (дата звернення: 03.05.2024).

21. Abdullah Alhayajneh, Alessandro Baccarini, Thaier Hayajneh . «Quality of Service Analysis of VoIP Services», 2018. URL: https://www.researchgate.net/publication/335199569_Quality_of_Service_Analysis_of_VoIP_Services (дата звернення: 04.05.2024).

22. Tobiloba Kollawole Adenekan. «Impact of Security Protocols on Wireless VoIP Call Quality and Latency», 2024. URL: https://www.researchgate.net/publication/388653823_Impact_of_Security_Protocols_on_Wireless_VoIP_Call_Quality_and_Latency (дата звернення: 04.05.2024).

23. Charles Shen, Erich M. Nahum, Henning Schulzrinne, Charles P. Wright. «The Impact of TLS on SIP Server Performance: Measurement and Modeling». URL: https://www.researchgate.net/publication/260353578_The_Impact_of_TLS_on_SIP_Server_Performance_Measurement_and_Modeling (дата звернення: 05.05.2024).

24. Regis J., Jr (Bud) Bates. «Securing VoIP: Keeping Your VoIP Network Safe», 2015 (дата звернення: 06.05.2024).

25. Oleksandr Kuznetsov, Oleksandr Peliukh, Nikolay Poluyanenko, Serhii Bohucharskyi, Ievgeniia Kolovanova. «Comparative Analysis of Cryptographic Hash Functions in Blockchain Systems», 2023. URL: https://ceur-ws.org/Vol-3550/paper7.pdf (дата звернення: 07.05.2024).

26. Dhanashree Toradmalle, Rohan Singh, Het Shastri, Nikita Naik, Vishal Panchidi. «Prominence Of ECDSA Over RSA Digital Signature Algorithm», 2018 (дата звернення: 07.05.2024).

ДОДАТОК А

Лістинг програмного коду.

```
# Імпорт бібліотек
from Crypto.Cipher import AES # AES для симетричного шифрування
from Crypto.Random import get_random_bytes # Генерація криптографічно стійких випадкових байтів
from Crypto.Hash import SHA256 # SHA-256 для хешування даних
from Crypto.PublicKey import ECC # ECC для генерації ключів ECDSA (еліптична крива)
from Crypto.Signature import DSS # DSS (Digital Signature Standard) для підпису/перевірки ECDSA
from Crypto.Util.Padding import pad, unpad # Допоміжні функції для доповнення (padding)
from pydub import AudioSegment # PyDub для роботи з аудіофайлами

# Завантаження аудіофайлу WAV за допомогою PyDub (WAV)
audio = AudioSegment.from_wav(r'C:\Users\Hp\Downloads\call_101_0.wav')

# Конвертація аудіо у сирі байти для обробки
audio_bytes = audio.raw_data


# Генерація 256-бітового ключа AES та 128-бітового IV
key = get_random_bytes(32)  # 32 байти = 256 біт для ключа AES-256
iv = get_random_bytes(16)   # 16 байтів = 128 біт для IV (ініціалізаційного вектора)

# Налаштування шифра AES в режимі CBC з згенерованими ключем та IV
cipher = AES.new(key, AES.MODE_CBC, iv)

# Доповнення аудіоданих до кратного 16 байт і шифрування
padded_audio = pad(audio_bytes, AES.block_size) # AES.block_size = 16 байт
encrypted_audio = cipher.encrypt(padded_audio) # Отримуємо зашифровані дані дзвінка


# Створення SHA-256 хеш зашифрованих даних
hash_obj = SHA256.new(encrypted_audio) # Обчислення дайджеста SHA-256 для шифротексту

# Генерація пари ключів для ECDSA (еліптична крива P-256)
private_key = ECC.generate(curve='P-256') # Генерація приватного ключа (ключ підпису)
public_key = private_key.public_key() # Отримання відповідного публічного ключа

# Формування цифрового підпису ECDSA на основі гешу
signer = DSS.new(private_key, 'fips-186-3') # Ініціалізація об’єкта підписувача (DSS, стандарт FIPS 186-3)
signature = signer.sign(hash_obj) # Створення цифрового підпису (ECDSA) для хешу даних


# Збереження зашифрованих даних та підпису у файли
with open(r'C:\Users\Hp\Documents\encrypted_audio.bin', 'wb') as f:
    f.write(encrypted_audio) # Збереження зашифрованого аудіопотоку

with open(r'C:\Users\Hp\Documents\signature.sig', 'wb') as f:
    f.write(signature) # Збереження підпису (бінарний формат)
```

```
with open(r'C:\Users\Hp\Documents\public_key.pem', 'wt') as f:
    f.write(public_key.export_key(format='PEM')) # Збереження відкритого ключа у форматі PEM

with open(r'C:\Users\Hp\Documents\iv.bin', 'wb') as f:
    f.write(iv) # Збереження IV (ініціалізаційний вектор)


# Перевірка цілісності
# Зчитування зашифрованих даних, підпису, відкритого ключа
with open(r'C:\Users\Hp\Documents\encrypted_audio.bin', 'rb') as f:
    encrypted_audio_check = f.read() # Завантаження шифротексту дзвінка

with open(r'C:\Users\Hp\Documents\signature.sig', 'rb') as f:
    signature_check = f.read() # Завантаження підпису ECDSA

with open(r'C:\Users\Hp\Documents\public_key.pem', 'rt') as f:
    public_key_check = ECC.import_key(f.read()) # Імпорт відкритого ключа із PEM-файлу


# Обчислюємо SHA-256 хеш для перевірки
hash_check = SHA256.new(encrypted_audio_check)

# Перевірка цифрового підпису за допомогою відкритого ключа
verifier = DSS.new(public_key_check, 'fips-186-3') # Ініціалізація об’єкту перевірки підпису
try:
    verifier.verify(hash_check, signature_check)
    print("Підпис дійсний, цілісність підтверджено!")
except ValueError:
    print("Підпис не дійсний, цілісність порушена!")


# Розшифрування даних AES-256-CBC за допомогою ключа та IV
cipher_dec = AES.new(key, AES.MODE_CBC, iv) # Ініціалізація об’єкту AES для розшифрування
decrypted_padded = cipher_dec.decrypt(encrypted_audio_check) # Розшифрування байтів (ще містять паддінг)
decrypted_audio_bytes = unpad(decrypted_padded, AES.block_size) # Видалення паддінгу, отримання оригінальних аудіо-байтів

# Створюємо аудіо з розшифрованих байтів та зберігаємо
recovered_audio = AudioSegment(
    data=decrypted_audio_bytes,
    sample_width=audio.sample_width,
    frame_rate=audio.frame_rate,
    channels=audio.channels
)
recovered_audio.export('recovered_call.wav', format='wav')
print("Розшифрування завершено. Аудіо відновлено у файлі recovered_call.wav.")
```