PRIVATE CLOUD COMPUTING IN THE CONTEXT OF AI

Крайнік М.В.[1], Письменний І.О.[2]

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

[1] krainik.mykyta@lll.kpi.ua, [2] pysmennyi.ihor@lll.kpi.ua [0000-0001-7648-2593]

У дослідженні розглядається архітектура приватних хмарних обчислень у контексті штучного інтелекту з локальним маршрутизатором запитів на пристрої для визначення складності обробки запиту та релевантності його виконання у хмарі. Використано аналіз публічних технічних матеріалів, аналіз існуючих рішень та архітектурне моделювання системи. Отримано узагальнену архітектуру з локальним маршрутизатором, IP-blinding ретранслятором, анонімними токенами та захищеними середовищами виконання (TEE). Її наукова новизна полягає в уніфікованому відкритому підході, який може стати основою подальших практичних систем із підвищеними вимогами до приватності.

Ключові слова: приватні хмарні обчислення, ШІ, TEE, анонімні токени, гібридна архітектура "пристрій-хмара".

1. ВСТУП

Зважаючи на стрімке поширення генеративних моделей, які все частіше працюють з надчутливими персональними та корпоративними даними, дослідження архітектур для проведення приватних ШІ-обчислень стає все актуальнішим. Класична модель "обробка усього в хмарі" більше не задовольняє вимоги до безпеки та не відповідає регуляціям, адже зростає тиск з боку законодавства (GDPR, європейський ШІ-акт) й очікувань користувачів щодо реальної, а не декларативної приватності. Паралельно відбувається технологічний зсув до поєднання моделей на локальних пристроях і в захищених хмарних середовищах, що так чи інакше спричинює розвиток безпекових практик у роботі з ШІ моделями, включно з використанням вже існуючих підходів і знань у приватних хмарних обчисленнях.

2. ЩО ТАКЕ PRIVATE CLOUD COMPUTING? ЙОГО ВИКОРИСТАННЯ З ШІ

Private Cloud Computing у загальному розумінні – це модель хмарних обчислень, у якій обчислювальні ресурси, сховище та мережа виділяються для однієї організації або вузького кола користувачів і працюють у контрольованому, ізольованому середовищі [1]. На відміну від публічних хмар, де інфраструктура спільна для багатьох клієнтів, приватна хмара дозволяє краще контролювати розміщення даних, політики безпеки, відповідність регуляціям та інтеграцію з внутрішніми системами [1]. У контексті сучасних систем штучного інтелекту поняття приватної хмари доповнюється вимогою до конфіденційних обчислень: дані мають бути захищені не лише "на диску" та "у мережі", а й під час самого виконання моделей. Саме тому з'являються концепції на кшталт Private AI Compute або Private Cloud Compute, де хмарна інфраструктура доповнюється апаратно захищеними середовищами виконання (TEE), анонімізацією ідентичності користувача та обмеженням будь-якого доступу інфраструктурних операторів до вмісту запитів. У таких системах генеративні та аналітичні

моделі ШІ можуть працювати з чутливими персональними або корпоративними даними, фактично наближаючись за рівнем приватності до локальної обробки на пристрої. Водночас приватна хмара забезпечує еластичність і масштабованість, необхідні для великих моделей, які не можуть бути розгорнуті безпосередньо на кінцевих пристроях. Для користувача це відкриває можливість отримувати набагато вищий рівень якості відповідей від ШІ, не жертвуючи контролем над даними та відповідністю юридичним вимогам. Для розробників і архітекторів це означає потребу в спеціалізованих архітектурних патернах, що поєднують традиційну хмарну інфраструктуру, конфіденційні обчислення та гібридну модель локального/хмарного інференсу.

3. ОПИС АРХІТЕКТУРИ

Як було згадано раніше, архітектура поєднує локальне виконання на пристрої користувача та захищену хмарну інфраструктуру, побудовану за принципами приватних віддалених обчислень.

У центрі взаємодії на стороні клієнта знаходиться локальний ШІ-маршрутизатор, який для кожного запиту визначає, чи достатньо можливостей вбудованої моделі на пристрої, чи потрібен виклик хмарних потужностей. Якщо запит може бути оброблений локально, він ніколи не покидає пристрій і виконується у захищеному середовищі локальної моделі.

Якщо ж потрібна хмара, роутер спершу намагається встановити захищене з'єднання з хмарою з проведенням атестації серверу, щоб впевнитись у його автентичності. У разі якщо ця перевірка провалюється, то з'єднання не встановлюється та користувач отримує помилку, або ми намагаємось обробити запит локально. У позитивному випадку ми встановлюємо з'єднання та спершу звертаємось до IP-blinding ретрансляторf, щоб приховати нашу реальну IP-адресу. Надалі такого роду атестація, але вже взаємна, сервісів проводиться для кожної спроби створити з'єднання між будь-якими компонентами у хмарі задля того, щоб гарантувати автентичність всіх модулів та уникнути випадкового витоку даних у неперевірені вузли, що можуть виконувати логіку, що не пройшла аудиту і/або не використовувати механізми для шифрування даних у пам'яті та ізольовані на рівні інфраструктури віртуальні машини.

Після IP-blinding ретранслятора, запит користувача потрапляє до сервісу обміну токенів, щоб замінити звичайний автентифікаційний токен на анонімний, щоб при обробці запиту вже на стороні наступних сервісів ми ніяк не викривали, чиї дані зараз знаходяться в роботі.

Після ретранслятора запит потрапляє до шлюзу, де встановлюється захищена сесія (з віддаленою атестацією сервера) і спершу перевіряється дійсність раніше випущеного анонімного токена та відразу після цього відбувається отримання списку доступних сервісів для проведення потрібних обчислень. Шлюз фактично виконує роль єдиної точки входу в систему, здійснює авторизацію та розподіляє трафік між кількома сервісами інференсу; взаємодія між самим шлюзом і внутрішніми сервісами відбувається через окреме захищене з'єднання (наприклад, mTLS/ALTS), враховуючи вже вищезгадану взаємну атестацію модулів.

Кожен сервіс інференсу запускає ШІ-модель усередині апаратно та програмно ізольованого середовища (TEE або конфіденційна віртуальна машина в залежності від імплементації та наявності потрібної інфраструктури), де дані завжди зберігаються в пам'яті в зашифрованому вигляді й недоступні ні для гіпервізора (так званий монітор віртуальних машин), ні для адміністраторів навіть у випадку так званих "broken glass" ситуацій, коли відбувається критична поломка в сервісі.

У результаті після інференсу запит та дані користувача потрапляють до кінцевої моделі ШІ, яка буде здійснювати обробку та поверне результат, який пройде той самий шлях у зашифрованому вигляді через всі раніше згадані вузли системи й повернеться до користувача.

Окремо варто зазначити, що також для більшої безпеки для кожного сервісу наявні правила вхідного та вихідного трафіку, які дозволяють загалом контролювати те, звідки сервіси можуть отримувати дані на вхід та що й куди вони можуть повертати як результат/до яких сервісів можуть звертатись.

Для простішого сприйняття концепту архітектури, побудуємо архітектурну діаграму, яка відобразить всі базові компоненти, що були описані раніше. Вона наведена на рисунку 1.

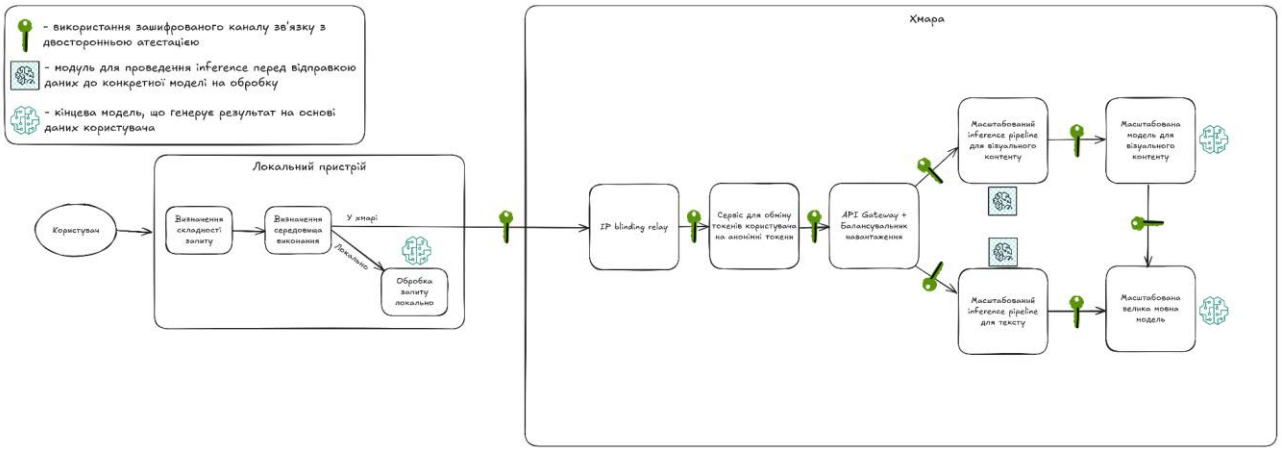

Рисунок 1. Архітектурна діаграма для системи приватних хмарних ШІ обчислень

Тепер розглянемо більш детально окремі частини нашої архітектури.

3.1. Локальний пристрій та роутер запитів

Раніше було згадано про основні дії, що виконуються на користувацькому пристрої, як-от смартфоні. Але також варто окреслити окремі важливі деталі роботи тих чи інших його частин.

Загалом локальний роутер запитів може враховувати велику кількість параметрів під час прийняття рішень щодо того, де потрібно виконати запит користувача, але серед них є звичайно базові, які мають вищий пріоритет і дозволяють отримати більш точну оцінку.

Одним з таких факторів є навантаження та енергоресурси пристрою: якщо пристрій вже знаходиться під значним навантаженням або його акумулятор має низький рівень заряду, то потенційно запит, особливо важкий, потрібно виконати в хмарі, а не локально.

Іншим важливим параметром може бути налаштування приватності користувача: якщо дані надчутливі і користувач заборонив їх надсилання, навіть ціною точності – роутер залишить обробку локально, або взагалі відмовиться виконувати запит у хмарі.

Також звичайно потрібно зважати на складність самого запиту: якщо він є доволі важким, що локальна модель або не зможе його повноцінно обробити через свої малі розміри, або буде це робити надзвичайно довго. Це наприклад, може стосуватись мультимодальних запитів, які включають у себе і текст, і візуальний контент, і навіть аудіо. У такому випадку запит варто надсилати в хмару, звичайно зважаючи на інші вищеописані фактори.

Говорячи про локальне виконання запитів користувача за допомогою ШІ, потрібно сказати, що ця обробка також повинна відбуватись в ізольованому середовищі. Наприклад, на Android, починаючи з 12 версії, доступний такий званий Private Compute Core, який

дозволяє ізолювати ШІ-функції або інші, що потребують високої конфіденційності від решти системи подібно до того, як це робить TEE [2].

Для локального інференсу використовуються оптимізовані моделі: як правило, це так звані “TinyML” або “edge AI” моделі, що працюють через фреймворки на кшталт TensorFlow Lite, ONNX Runtime, PyTorch Mobile, Core ML тощо – залежно від платформи [3]. Ці інструменти дозволяють виконувати ШІ моделі напряму на CPU/GPU/NPU пристрою, часто в поєднанні з апаратними засобами безпеки (наприклад, TrustZone або Secure Enclave у телефонах) для захисту пам’яті.

Таким чином, пристрій користувача з роутером забезпечує першу лінію приватності: максимум даних не покидає пристрій. Але коли потрібна допомога хмари, він передає запит так, щоб і надалі зберегти приватність через механізми, що були описані раніше.

3.2. IP-blinding ретранслятора, анонімізація токенів

Раніше вже було згадано про IP-blinding ретранслятора. Фактично цей механізм унеможливлює прив’язування запитів до конкретного пристрою чи геолокації за мережевими даними і суттєво ускладнює навіть потенційні атаки (наприклад, DDoS на конкретного користувача стає неможливим, а спроби відстежити активність – марними).

Для такого роду ретрансляторів можуть використовувати сторонні сервіси, що виступають у ролі проксі. Але загалом сам механізм часто працює за допомогою так званого Oblivious HTTP Proxy (OHTTP), коли дані користувача подвійно шифруються: зовнішній шар знімає ретранслятор, не бачачи самих даних, а внутрішній шар зашифровано для кінцевого сервера [4]. Головний тут фактор полягає ж у тому, що ретранслятор має бути розташований поза інфраструктурою основного хмарного провайдера, щоб той не міг ідентифікувати, звідки саме прийшов запит користувача [4].

Іншим важливим кроком тут є анонімізація токенів, коли ми обмінюємо токени автентифікації користувача на ті, які так чи інакше від нього відв’язані, але в той же час підтверджують його право на запит. Загалом існує стандарт Privacy Pass, що описує цей механізм та те, як відбувається підтвердження валідності токену [5].

3.3. Віддалена та взаємна атестації

Іншим ключовим механізмом організації довіри в архітектурах такого типу є віддалена атестація сервісів, що працюють у довіреному середовищі виконання (TEE). TEE – це апаратно захищене оточення з ізольованою пам’яттю та мінімальним обсягом довіреного коду, яке гарантує конфіденційність і цілісність виконання навіть за компрометації операційної системи чи гіпервізора [6]. У процесі атестації таке середовище формує криптографічно підписаний звіт, що містить хеш коду, що на ньому працює та параметри конфігурації; віддалений учасник порівнює їх із еталонними значеннями та, у разі збігу, приймає рішення довіряти цьому вузлу [6]. Таким чином клієнт (або інший сервіс) не просто вірить сертифікату, який надає сервіс, а перевіряє, що на іншому боці дійсно запущено саме той код і в саме такому захищеному режимі, які очікується.

У розподілених системах, де кілька сервісів обмінюються чутливими даними, часто використовується взаємна атестація: кожна сторона перевіряє, що інший вузол також працює в належному TEE з коректним кодом. Після успішної (взаємної) атестації сторони узгоджують ключі шифрування й організовують захищений канал поверх стандартних протоколів. Типовим підходом є поєднання атестації з mutual TLS (mTLS): у mTLS обидві сторони з’єднання мають сертифікати й підтверджують володіння відповідними приватними ключами; це дає змогу аутентифікувати як сервер, так і клієнта й одночасно шифрувати трафік [7].

Часто в цьому контексті використовується Apache Teaclave – фреймворк для організації безпечних обчислень, який працює поверх TEE (Intel SGX) і використовує віддалену атестацію для встановлення довірених каналів зв'язку між клієнтами та обчислювальними вузлами[8]. У Teaclave клієнт може перевірити, що його код або функція виконуються в коректно атестованому enclave, після чого встановлюється зашифрований канал для передачі вхідних даних і отримання результатів [7]. На рівні мережі така взаємодія може бути реалізована у вигляді mTLS-з'єднання, де сертифікати пов'язані з атестованим станом enclave, тож будь-яка спроба замінити код або покинути TEE під час виконання призведе до зміни вимірів і, відповідно, до того, що атестація провалиться [6, 7].

4. ВИСНОВКИ

У цій статті було розглянуто архітектуру приватних хмарних обчислень, що забезпечує достатній рівень конфіденційності даних користувача при їх передачі та обробці ШІ. Архітектура поєднує локальні можливості пристрою користувача із захищеною хмарною інфраструктурою на базі TEE, анонімних токенів та IP-blinding. Ми також окреслили основні компоненти такої системи, шлях проходження запиту й механізми атестації, які дозволяють підтвердити коректність та захищеність середовища виконання. Особлива увага була приділена тому, як приватні хмарні обчислення дозволяють використовувати потужні моделі ШІ, не порушуючи конфіденційність і вимоги регуляцій, що робить цю тему критично важливою в умовах стрімкого розвитку генеративного ШІ. Загалом ж приватні хмарні обчислення є не лише технічною опцією, а й необхідною умовою для довіри користувачів та організацій до інтелектуальних сервісів.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What is a Private Cloud? Amazon Web Services. URL: https://aws.amazon.com/what-is/private-cloud/ (дата звернення: 18.11.2025).
2. Frey S. Introducing Android's Private Compute Services. Google Online Security Blog, 09.09.2021. URL: https://security.googleblog.com/2021/09/introducing-androids-private-compute.html (дата звернення: 18.11.2025).
3. Krishnamoorthy M. V. Edge AI: TensorFlow Lite vs. ONNX Runtime vs. PyTorch Mobile. DZone, 03.06.2025. URL: https://dzone.com/articles/edge-ai-tensorflow-lite-vs-onnx-runtime-vs-pytorch (дата звернення: 19.11.2025).
4. Wood C., Hoyland J. Stronger than a promise: proving Oblivious HTTP privacy properties. The Cloudflare Blog, 27.10.2022. URL: https://blog.cloudflare.com/stronger-than-a-promise-proving-oblivious-http-privacy-properties/ (дата звернення: 19.11.2025).
5. Privacy Pass (privacypass). About. IETF Datatracker. URL: https://datatracker.ietf.org/wg/privacypass/about/ (дата звернення: 19.11.2025).
6. Sabt M., Achemlal M., Bouabdallah A. Trusted Execution Environment: What It is, and What It is Not. In: 2015 IEEE Trustcom/BigDataSE/ISPA. 2015. URL: https://hal.science/hal-01246364/file/trustcom_2015_tee_what_it_is_what_it_is_not.pdf (дата звернення: 20.11.2025).
7. What is mutual TLS (mTLS)? Cloudflare Learning Center. URL: https://www.cloudflare.com/learning/access-management/what-is-mutual-tls/ (дата звернення: 20.11.2025).
8. Sun M. Teaclave: A Universal Secure Computing Platform. ApacheCon 2020, 30.09.2020. URL: https://mssun.me/assets/teaclave.pdf (дата звернення: 20.11.2025).