Check for updates

Journal of KIIT. Vol. 24, No. 4, pp. 73-84, Apr. 30, 2026. pISSN 1598-8619, eISSN 2093-7571
http://dx.doi.org/10.14801/jkiit.2026.24.4.73

온디바이스 AI 환경에서 보안 통신 프로토콜 및 암호화 구성에 따른 딥러닝 추론 성능 영향 분석

채주혁*[1], 권오빈*[2], 김진홍**, 안형태***

Performance Impact of Secure Communication Protocols and Cryptographic Configurations on On-Device AI Inference

Ju-Hyeok Chae*[1], Obin Kwon*[2], Jinhong Kim**, and HyeongTae Ahn***

본 연구 논문은 한국전자통신연구원 연구운영지원사업의 일환으로 수행되었음
[25ZD1170, 대경권 지역산업 기반 ICT 융합기술 고도화 지원사업(본부)]

요 약

온디바이스 AI와 사물인터넷(IoT) 기반 서비스가 산업 현장에 확산되면서, 현장에서 수집되는 영상·센서 데이터를 실시간으로 처리하면서도 안전하게 전송하기 위한 보안 통신의 중요성이 커지고 있다. 그러나 온디바이스 장치는 CPU·메모리 자원이 제한되어 있어, 보안 프로토콜과 암호화 알고리즘을 적용할 경우 딥러닝 추론과 통신 처리 간 자원 경합으로 인한 성능 저하가 발생할 수 있다. 본 논문은 온디바이스 AI 환경에서 대표적인 보안 프로토콜과 TLS 1.3 기반 암호화 알고리즘 조합에 따라 객체 탐지 모델의 추론 속도(FPS), 네트워크 지연(RTT), CPU·메모리 사용률이 어떻게 변화하는지를 정량적으로 분석한다. 이를 바탕으로 온디바이스 AI 시스템에서 요구되는 실시간 처리 성능과 보안 수준을 동시에 만족시키기 위한 보안 프로토콜 및 암호화 구성 선택 기준을 제시하고, 향후 온디바이스 보안 통신 구조 설계와 경량 보안 프로토콜 개발에 참고할 수 있는 기초 자료를 제공한다.

Abstract

As on-device AI and IoT services are increasingly deployed in industrial environments, edge devices are required to process video and sensor data in real time while ensuring secure data transmission. However, limited CPU and memory resources can lead to resource contention between deep-learning inference and secure communication processing, resulting in system performance degradation. This paper provides a quantitative analysis of how various combinations of secure communication protocols and TLS 1.3 cipher suites impact inference speed(FPS), network latency(RTT), and CPU/memory utilization in on-device AI environments. Based on these empirical results, we propose practical guidelines for selecting protocol and cryptographic configurations that balance real-time performance with security requirements. These findings offer foundational insights for designing future lightweight secure communication architectures for on-device AI.

Keywords

on-device AI, secure communication protocols, TLS 1.3, cryptographic algorithm, deep-learning inference performance

* 국립금오공과대학교 컴퓨터공학과 학사과정
- ORCID[1]: https://orcid.org/0009-0000-5807-457X
- ORCID[2]: https://orcid.org/0009-0001-4646-7410
** 한국전자통신연구원 대경권연구본부 선임연구원
- ORCID: https://orcid.org/0009-0000-9557-9606
*** 국립금오공과대학교 컴퓨터공학과 조교수(교신저자)
- ORCID: https://orcid.org/0000-0002-3390-3981

· Received: Nov. 26, 2025, Revised: Jan. 08, 2026, Accepted: Jan. 11, 2026
· Corresponding Author: HyeongTae Ahn
Dept. of Computer Engineering, Kumoh National Institute of Technology, 61 Daehak-ro (Yangho-dong), Gumi, Gyeongbuk, [39177] Korea
Tel.: +82-54-478-7537, Email: anten@kumoh.ac.kr

Ⅰ. 서 론

제4차 산업혁명 시대의 핵심 인프라인 사물인터넷(IoT)과 온디바이스 AI 기술은 현재 폭발적인 성장세를 보이고 있다. 정보통신산업진흥원(NIPA)의 보고서에 따르면, 글로벌 IoT 시장 규모는 2023년 1조 1,770억 달러에서 연평균 12.57% 성장하여 2028년에는 2조 2,270억 달러에 이를 것으로 전망된다[1]. 이에 따라 지능형 CCTV, 자율주행, 스마트팩토리 등 실시간성 및 보안성이 필수적인 분야로 그 적용 범위가 빠르게 확대되고 있다. 특히, 이러한 시스템에서 수집되는 민감 데이터와 제어 명령을 보호하기 위해 TLS와 같은 보안 프로토콜 적용은 필수적이다. 그러나 보안 프로토콜 적용에 따른 암호화 연산 및 통신 처리 오버헤드는 자원이 제한적인 온디바이스 환경에서 딥러닝 추론 작업과의 자원 경합을 발생시킨다[2]. 이로 인해 발생하는 성능 저하는 교통 제어 및 감시 시스템 등 안전과 직결된 서비스에서 제어 실패나 탐지 누락과 같은 중대한 사고의 원인이 될 수 있다. 따라서 자원이 제한된 환경에서도 딥러닝 추론의 실시간성을 확보하면서 보안 수준을 유지할 수 있는 시스템 구성이 요구된다. 이를 위해서는 특정 보안 프로토콜 선택을 넘어, 하드웨어 연산 특성과 부하 수준을 고려하여 보안 프로토콜 및 암호화 알고리즘 조합이 시스템 성능에 미치는 영향을 정량적으로 분석해야 한다.

기존 연구들은 온디바이스 환경의 제약을 극복하기 위해 다양한 시도를 지속해 왔으나, 몇 가지 한계를 지닌다. 예를 들어, 일부 연구는 MQTT나 CoAP 등 IoT 전용 프로토콜의 전송 효율 개선에 집중하거나[3], 특정 하드웨어 환경에서 암호화 알고리즘의 연산 속도를 개별적으로 비교하는 데 그쳤다[4]. 또한, 분석 범위가 주로 서버급 환경에서의 암호화 처리량 측정이나, IoT 단말의 네트워크 지연 및 전송률 변화 분석 수준에 머무는 경우가 많았다. 온디바이스 AI 환경에서는 보안 프로토콜 처리에 따른 CPU 부하가 추론 연산으로의 데이터 공급을 지연시켜 전체 성능을 저하시키는 자원 경합이 핵심적인 병목 요인임에도, 이러한 요소 간의 상호작용을 통합적으로 실증 분석한 연구는 여전히 부족한 상황이다. 이에 본 연구는 온디바이스 AI 환경에서 보안 통신이 딥러닝 추론 성능에 미치는 영향을 정량적으로 분석하고, 실시간성과 보안성을 동시에 만족하는 프로토콜 구성 기준 제시를 목표로 한다. 구체적으로, 하드웨어 가속 지원 특성이 상이한 Raspberry Pi 5와 Jetson Orin Nano 환경에서 보안 프로토콜 및 암호화 알고리즘 조합이 추론 속도, 네트워크 지연, 그리고 시스템 자원 사용률에 미치는 영향을 분석하여 실용적인 보안 구성 가이드라인을 제안한다.

Ⅱ. 관련 연구

온디바이스 AI 및 IoT 보안 통신 분야의 선행 연구들은 각각 의미 있는 성과를 제시하고 있으나, 대체로 단일 기술 요소 중심의 분석에 머무르고 있으며 온디바이스 환경에서 딥러닝 추론과 보안 프로토콜이 동시에 동작할 때 발생하는 복합적 성능 저하 문제를 충분히 설명하지 못한다. 표 1은 온디바이스 딥러닝 추론 및 IoT 보안 통신과 관련된 기존 연구들을 정리한 것이다. 구체적으로, 각 연구가 다룬 주요 목표, 분석에 활용된 환경/플랫폼, 적용한 보안 프로토콜, 그리고 평가 관점(Analysis focus)을 요약하여 제시한다.

표 1. 관련 연구 비교
Table 1. Related work comparison

Study	Environment	Security protocol	Analysis focus
Howard[2]	Mobile/embedded devices	None	Considers only model efficiency
Raza[3]	Resource-constrained IoT devices	DTLS	Evaluates DTLS alone
Wukkadada[4]	General IoT messaging systems	HTTP, MQTT	Focuses on network performance metrics only
Gueron[5]	x86-based server systems	AES-based encryption	Examines crypto performance only
Proposed method	Raspberry Pi, Jetson Orin Nano (ARM on-device platforms)	TLS, DTLS, HTTP/3, MQTT-TLS	Evaluates protocol impact on deep learning inference

먼저 A. G. Howard et al.[2]의 MobileNet 연구는 경량 합성곱 구조를 통해 모바일·임베디드 기기에서 실시간 추론을 가능하게 하며 온디바이스 최적화의 기반을 마련했다. 그러나 보안 통신 프로토콜 적용 시 발생하는 연산 자원 경쟁과 같은 외부 요인은 고려하지 않는다. S. Raza et al.[3]는 DTLS의 연결 지연 및 암호화 오버헤드를 분석하며 제약된 IoT 장치에서의 성능 저하 가능성을 제시하였지만, 분석 범위가 보안 프로토콜 단독 성능에 국한되어 있어 딥러닝 추론과의 상호 간섭 효과까지는 다루지 않았다. B. Wukkadada et al.[4]은 IoT 환경에서 HTTP와 MQTT의 지연·처리량 및 자원 사용량을 비교하여 애플리케이션 계층 프로토콜 선택이 네트워크 성능과 단말 자원 효율에 미치는 영향을 평가하였으나, TLS 기반 암호 채널 도입 시 기기 내부 CPU 점유율 증가와 딥러닝 추론 FPS 하락으로 이어지는 온디바이스 특성까지는 분석하지 않았다. 마지막으로 S. Gueron[5]의 AES-NI 기반 암호화 가속 연구는 암호 연산의 CPU 부담 완화 가능성을 보여주지만, 하드웨어 가속이 제한적인 ARM 기반 온디바이스 환경에서의 연산 경쟁 문제까지는 설명하지 않는다. 이처럼 기존 연구들은 온디바이스 추론, 보안 프로토콜, 암호화 연산을 개별적으로 평가하는 경우가 많으며, 실제 온디바이스 장치에서 딥러닝 추론과 보안 통신이 동시에 수행될 때 발생하는 복합적 성능 변화를 통합적으로 다룬 연구는 부족하다. 이에 본 연구는 Raspberry Pi 5와 NVIDIA Jetson Orin Nano 환경에서 TLS, DTLS, HTTP/3, MQTT-TLS 등 대표 보안 프로토콜을 적용한 뒤, 이들이 딥러닝 모델의 추론 성능(FPS, Frames Per Second), 네트워크 왕복 지연(RTT, Round Trip Time), CPU 및 메모리 사용률에 미치는 영향을 정량적으로 비교·분석한다. 이를 통해 온디바이스 AI 시스템에서 보안 프로토콜 선택이 실제 성능과 자원 효율성에 어떤 영향을 미치는지 규명하고, 서비스 특성별 적합한 프로토콜 선택 기준을 제시하고자 한다.

III. 실험 환경 및 측정 방법

3.1 실험 환경 구성

본 연구는 CPU 중심 실행 환경과 GPU 가속 실행 환경이라는 하드웨어 가속기 유무 및 연산 처리 특성이 상이한 두 가지 대표적인 온디바이스 AI 환경에서 실험을 수행한다. 모바일 SoC나 NPU 기반 장치 역시 온디바이스 AI 범주에 포함되나, 본 연구에서는 보안 구성에 따른 자원 점유 변화와 추론 성능 간의 상관관계를 보다 정밀하게 분석하고자 실험 제어 및 자원 모니터링이 용이한 두 플랫폼을 실험 대상으로 선정하였다. 그림 1(a)의 Raspberry Pi 5는 별도의 가속기 없이 CPU 연산에 의존하는 저전력 중심의 범용 IoT 디바이스 환경을 모사하기 위해 선정되었다. 반면, 그림 1(b)의 NVIDIA Jetson Orin Nano는 1,024개의 CUDA 코어와 32개의 Tensor 코어를 탑재하여 강력한 AI 추론 성능과 암호화 가속을 지원하는 고성능 온디바이스 AI 플랫폼이다. 두 기기 모두 동일한 네트워크 환경(Wi-Fi 5GHz)에서 원격 서버와 통신하며, 서버는 Nginx (HTTPS/HTTP/3), Mosquitto(MQTT), libcoap(DTLS)을 통해 다양한 프로토콜을 지원하도록 구성했다.

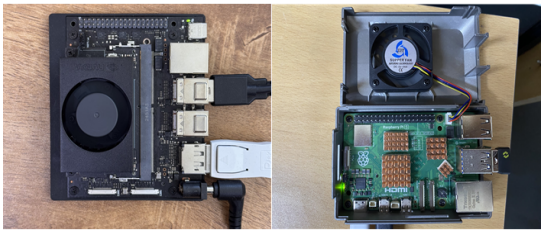

그림 1. 실험 장비
(a) Raspberry Pi 5 기기, (b) Jetson Orin Nano 기기
Fig. 1. Experimental devices
(a) Raspberry Pi 5 device, (b) Jetson Orin Nano device

표 2. 실험 하드웨어 및 소프트웨어 사양
Table 2. Specifications of experimental environment

Category	NVIDIA Jetson Orin Nano	Raspberry Pi 5
Processor	6-core ARM Cortex-A78AE v8.2 (1.5GHz)	Quad-core ARM Cortex-A76 (2.4GHz)
GPU	NVIDIA Ampere (1024 CUDA Cores)	VideoCore VII (No CUDA Support)
Memory	8GB LPDDR5	8GB LPDDR4X
OS	Ubuntu 20.04 LTS (JetPack 6.0)	Debian GNU/Linux 12(Bookworm)
AI model	YOLO11n (TensorRT Optimized)	YOLO11n (TFLite/PyTorch)
Network	Wi-Fi 5 (802.11ac)	Wi-Fi 5 (802.11ac)

표 2는 실험에 사용된 하드웨어 및 소프트웨어의 상세 사양을 나타낸다.

3.2 실험 변인 및 데이터셋

시스템 성능에 영향을 미치는 요인을 다각적으로 분석하기 위해 통신 프로토콜, 암호화 알고리즘, 그리고 입력 데이터(영상)를 독립 변인으로 설정했다.

3.2.1 통신 프로토콜

전송 계층의 특성(TCP vs. UDP)이 실시간성에 미치는 영향을 비교하기 위해 표 3과 같이 4종의 프로토콜을 선정했다. HTTPS[6]는 TCP를 기반으로 하며, TLS 1.3을 통해 기밀성과 무결성을 제공하는 가장 대표적인 보안 프로토콜이다. TLS 1.3은 초기 연결 시 최소 1-RTT가 필요한 구조를 가지므로 안정적이지만 상대적으로 초기 지연이 발생할 수 있다. 이에 비해 HTTP/3[7]는 UDP 위에서 동작하는 QUIC 프로토콜을 사용하며, 암호화된 핸드셰이크 절차를 QUIC 내부에 통합함으로써 짧은 연결 설정 시간과 높은 전송 효율을 구현한다. QUIC은 1-RTT 뿐 아니라 세션 재활용을 통한 0-RTT 연결도 지원하지만, 구현 복잡성으로 인해 경량 디바이스에서는 오버헤드가 증가할 가능성이 있다. MQTT[8]는 메시지 교환을 위한 경량 프로토콜로 설계되어 패킷 오버헤드가 작지만, 자체적으로 암호화를 포함하지 않기 때문에 TLS 적용이 필요하다. MQTT over TLS는 TCP 연결, TLS 핸드셰이크, MQTT CONNECT 메시지 교환이 순차적으로 이루어져 초기 설정 비용이 다른 프로토콜보다 클 수 있다.

표 3. 보안 통신 프로토콜 비교 및 선정 배경
Table 3. Comparison of secure communication protocols and selection rationale

Protocol	Transport layer	Selection reason
HTTPS	TCP	Baseline protocol for standard secure communication
HTTP/3	UDP	Evaluation of QUIC-based secure transport efficiency
MQTT	TCP	Assessment of IoT messaging protocol
DTLS	UDP	Comparison of UDP-based lightweight security protocol

한편, DTLS[9]는 TLS를 UDP 환경에 맞게 확장한 구조로 낮은 지연과 패킷 손실 내성을 제공하나, 비연결형 특성으로 인해 초기 검증 절차가 추가되어 네트워크 품질에 따라 연결 소요 시간이 변동될 수 있다.

3.2.2 암호화 알고리즘

하드웨어 가속 지원 여부에 따른 연산 효율성을 평가하기 위해 TLS 1.3의 주요 암호화 알고리즘 3종을 선정하였다. AES-128-GCM과 AES-256-GCM은 ARMv8 아키텍처의 암호화 확장(CE, Cryptographic Extensions) 명령어 세트와 Jetson Orin Nano의 전용 가속기를 통한 하드웨어 가속이 가능하다. 반면, ChaCha20-Poly1305는 하드웨어 가속기가 부재한 환경에서도 높은 성능을 발휘하도록 설계된 소프트웨어 최적화 기반의 스트림 암호이다.

3.2.3 입력 영상 데이터셋

실제 감시 환경의 다양성을 반영하고 딥러닝 모델의 연산 부하 수준을 조절하기 위해, MOT Challenge 플랫폼의 MOT17-05 시퀀스(그림 2)를 실험 데이터로 활용하였다. 해당 시퀀스는 640×480 해상도와 14 FPS로 기록된 총 837프레임의 도심 보행자 장면으로 구성된다. 보행자 탐지 중심의 시나리오는 추론 과정에서 지속적이고 일정한 연산 부하를 유발하므로, 보안 구성에 따른 성능 변화를 정밀하게 분석하기에 적합하다.

그림 2. MOT17-05 입력 영상 데이터셋
Fig. 2. MOT17-05 input image dataset

3.3 실험 절차 및 측정 방법

실험의 재현성과 정확성을 보장하기 위해 모든 과정은 자동화된 스크립트를 통해 제어되며, 각 실험 세션은 60초간 진행된다. Rate는 초당 발생하는 전송 요청의 빈도를 의미하며, Rate=1은 초당 1회 요청이 발생하는 저부하 환경을, Rate=100은 초당 100회의 요청이 발생하는 고부하 환경을 나타낸다. 또한 Size는 각 요청에 포함되는 메시지(페이로드)의 크기를 의미하며, Size=512B는 512바이트, Size=2048B는 2,048바이트의 패킷을 사용하여 실험을 수행했음을 의미한다.

동일 조건에서 10회 반복 측정 후 평균값을 산출하여 분석에 사용했다. 또한 실제 응용 환경과 유사한 부하를 생성하기 위해 3개의 스레드가 병렬로 동작한다. 네트워크 스레드는 지정된 프로토콜과 암호화 알고리즘으로 서버와 세션을 수립한 후, 설정된 전송률(Rate=1 또는 Rate=100)과 패킷 크기(512B 또는 2048B)로 더미 데이터를 전송하며 RTT를 실시간으로 측정한다. 딥러닝 모델은 YOLO 계열 중 경량화된 nano 버전인 YOLO11n을 객체 검출 모델로 사용하였으며, 입력 데이터에서 보행자 객체를 검출하도록 구성했다. YOLO11n은 실시간 처리를 목표로 설계된 1단계(one-stage) 모델로, 파라미터 수와 연산량은 작지만 일반적인 벤치마크에서 일정 수준 이상의 정확도를 유지할 수 있다. YOLO 추론 스레드는 지정된 영상을 프레임 단위로 로드하여 객체 탐지를 수행한다. 이 과정에서 GPU 또는 CPU에 연산 부하를 가중시키며, 초당 처리된 프레임 수(FPS)를 기록한다. 모니터링 스레드는 1초 간격으로 CPU 사용률, GPU 사용률, 메모리 점유율을 기록한다. 이때 CPU 사용률은 멀티코어 시스템의 특성을 반영하여 각 물리 코어의 사용량을 합산하는 방식으로 정의된다. 실험에 사용된 Raspberry Pi 5는 쿼드코어 프로세서를 탑재하여 이론적 최대치가 400%이며, Jetson Orin Nano는 6코어 프로세서를 탑재하여 최대 600%까지 측정될 수 있다. 실험 시 초기 5초간의 데이터는 시스템 안정화를 위해 결과 분석에서 제외했다.

IV. 실험 결과

4.1 Raspberry Pi 5 실험 결과

4.1.1 프로토콜별 딥러닝 추론 성능 비교

그림 3과 그림 4는 Raspberry Pi 5 환경에서 요청률과 패킷 크기 변화에 따른 프로토콜별 딥러닝 추론 성능(FPS)을 비교한 결과이다. 저부하 조건(Rate=1, 512B)의 그림 3에서는 HTTPS, DTLS, MQTT, HTTP/3가 모두 2.89~2.96 FPS 범위에서 동작했으며, 암호화 알고리즘 및 프로토콜 구성에 따른 유의미한 차이는 관측되지 않았다. 프로토콜 간 성능 편차 역시 최대 0.07 FPS 이내로 제한적이어서, 해당 조건에서는 네트워크 입출력 및 암호화 연산량이 전체 추론 파이프라인에서 차지하는 비중이 매우 작음을 확인했다. 고부하 조건(Rate=100, 2048B)의 그림 4에서도 전체 FPS는 2.79~2.93 수준으로, 저부하 대비 약 0.07 FPS(약 2~3%) 감소하는 수준에 그쳤다.

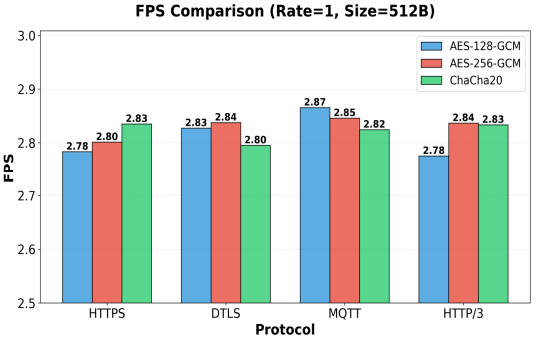

그림 3. Raspberry Pi 5의 FPS(Rate=1, Size=512B)
Fig. 3. FPS on Raspberry Pi 5(Rate=1, Size=512B)

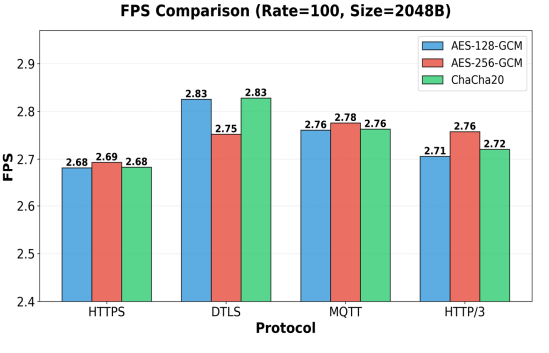

그림 4. Raspberry Pi 5의 FPS(Rate=100, Size=2048B)
Fig. 4. FPS on Raspberry Pi 5(Rate=100, Size=2048B)

DTLS는 2.89~2.93 FPS로 가장 높은 성능을, HTTPS는 2.79~2.83 FPS로 가장 낮은 성능을 보였으나, 프로토콜 간 최대 차이는 0.14 FPS로 기준값 대비 약 5% 수준에 불과했다. 이는 요청률과 패킷 크기를 크게 증가시키더라도 통신 계층의 추가 오버헤드는 딥러닝 추론 단계에서 유의미한 병목으로 작용하지 않으며, 온디바이스 환경에서 FPS는 여전히 모델 연산 비용에 의해 지배됨을 보여준다.

4.1.2 네트워크 지연 분석

저부하 조건(Rate=1, 512B)의 그림 5에서는 프로토콜 구조에 따른 지연 특성이 뚜렷하게 나타난다. HTTP/3는 세 가지 암호화 조합에서 약 9.54~9.69ms로 네 프로토콜 중 가장 낮은 RTT를 기록하였고, MQTT(AES-128/256-GCM)는 10.18~10.72ms 수준으로 그다음으로 낮은 지연을 보였다. HTTPS는 11.42~12.02ms 범위로 다소 높은 RTT를 보였으며, DTLS는 세 알고리즘 모두 12.88ms 수준으로 가장 큰 지연을 나타냈다. 특히 MQTT+ChaCha20 조합에서는 18.32ms로 다른 조합에 비해 큰 값이 관찰되었는데, 이는 MQTT의 구조적 한계라기보다는 ChaCha20 구현의 최적화 정도나 순간적인 자원 경쟁에 따른 일시적인 이상치로 판단된다. 고부하 조건(Rate=100, 2048B)의 그림 6에서는 MQTT는 AES-256-GCM에서 5.96ms, ChaCha20에서 6.42ms로 낮은 RTT를 유지하였고, AES-128-GCM에서도 8.59ms 수준을 보였다. HTTP/3는 ChaCha20에서 5.38ms, AES-128-GCM에서 6.53ms로 가장 낮은 RTT를 기록했지만, AES-256-GCM에서는 10.33ms로 다소 증가하는 양상을 보였다. HTTPS는 9.24~9.67ms 범위로 저부하 조건 대비 RTT가 소폭 감소하였으나 여전히 MQTT·HTTP/3보다는 높은 지연을 유지했고, DTLS는 12.17~13.08ms로 네 프로토콜 중 가장 큰 RTT를 보여 부하 증가에 가장 민감한 모습을 보였다.

전체적으로, FPS와 달리 RTT는 보안 프로토콜과 암호화 알고리즘에 더 민감하게 반응했다. 저지연이 요구되는 온디바이스 응용에서는 UDP 기반 QUIC을 사용하는 HTTP/3와 경량 메시징 구조의 MQTT가 상대적으로 낮은 RTT를 보였다. 한편 일부 조합에서는 저부하(Rate=1, 512B)보다 고부하(Rate=100, 2048B)에서 p50(중위값) RTT가 오히려 더 작게 측정되는 현상이 나타났다. 저부하에서는 요청 간격이 길어 CPU 전력 상태 전환, 캐시 미스, 컨텍스트 스위칭 등 초기화 비용이 반복적으로 발생하지만, 고부하에서는 CPU 클럭 상승과 캐시 상주 효과가 유지되어 개별 요청의 처리 비용이 감소하기 때문이다.

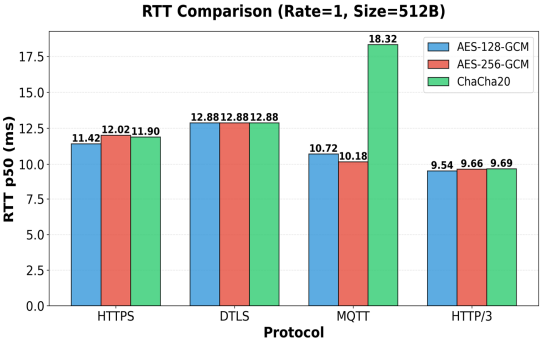

그림 5. Raspberry Pi 5의 RTT(Rate=1, Size=512B)
Fig. 5. RTT on Raspberry Pi 5(Rate=1, Size=512B)

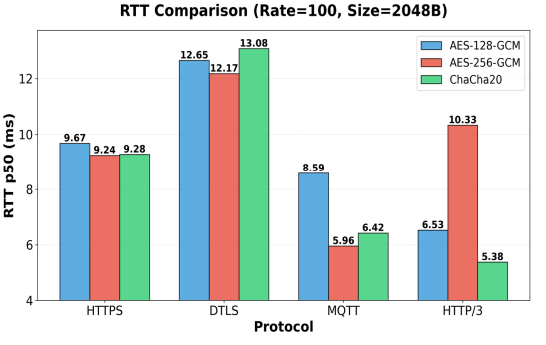

그림 6. Raspberry Pi 5의 RTT(Rate=100, Size=2048B)
Fig. 6. RTT on Raspberry Pi 5(Rate=100, Size=2048B)

4.1.3 암호화 알고리즘에 따른 성능 영향

암호화 알고리즘별 성능 차이를 살펴보면, AES-128-GCM, AES-256-GCM, ChaCha20 간 FPS 편차는 대부분 조건에서 0.05 FPS 이하로 매우 미미하며, DTLS의 일부 조합에서만 최대 0.08 FPS 수준의 차이가 나타난다. 이는 Raspberry Pi 5 환경에서는 암호화 방식의 선택이 딥러닝 추론 처리량에 거의 영향을 주지 않음을 의미한다. RTT 측면에서는 MQTT 저부하 조건에서 ChaCha20이 다른 알고리즘

대비 수 ms 높은 값을 보였고, HTTPS 고부하 조건에서는 AES-256-GCM이 다소 큰 RTT를 기록하는 등 일부 조합에서 제한적인 차이가 확인되었다. 이러한 차이는 ChaCha20의 소프트웨어 최적화 의존성, 알고리즘별 구조적 특성, 일시적인 CPU 자원 경쟁에 기인한 것으로 보이며, 전체적인 지연 특성을 좌우할 정도의 수준은 아니다. 특히 수 ms 이내의 차이는 Jetson Orin Nano의 극단적 꼬리 지연(Tail latency)과 구분된다. 따라서 전체 지연은 주로 프로토콜 특성에 좌우되며, 암호화 알고리즘은 연산 구조 및 최적화 수준에 따라 미세한 영향을 주는 보조 요인으로 판단된다.

4.1.4 시스템 자원 사용률 분석

표 4는 Raspberry Pi 5 환경에서 측정한 시스템 자원(CPU, GPU, Memory) 데이터를 분석한 결과이다. 모든 조합에서 CPU 사용률은 약 189~200% 범위에 분포하며, 이는 YOLO 추론 스레드와 보안 통신 처리가 동시에 수행되는 과정에서 대략 두 개의 코어가 지속적으로 포화되어 있음을 의미한다. 본 실험에서의 CPU 사용률은 단일 코어가 아닌 4개 코어를 가진 Raspberry Pi 5의 전체 시스템 사용률을 합산한 값으로, 이론적 최대치는 400%에 해당한다. 따라서 표에서 관찰되는 190% 내외의 값은 여러 코어에서 연산이 병렬로 수행된 결과로 100%를 초과하여 나타난 것이며, 단일 코어 기준의 과부하를 의미하는 것은 아니다. 메모리 사용률은 전반적으로 4.5~6.6% 수준으로 변화 폭이 작아, 본 실험 조건에서는 메모리가 주요 병목 요인으로 작용하지 않음을 확인할 수 있다. RTT 분석 결과, Rate=1 및 512B 조합에서는 모든 프로토콜의 p95가 약 13~47ms, p99가 수십 ms 수준으로 안정적인 수치를 기록했다. 반면, Rate=100 및 2048B의 고부하 환경에서는 여러 프로토콜의 꼬리 지연이 두드러지게 증가하는 양상을 보였다.

표 4. Raspberry Pi 5 환경에서의 프로토콜 및 암호화 알고리즘별 성능 측정 결과
Table 4. Performance measurement results by protocol and encryption algorithm on Raspberry Pi 5

Protocol	Cipher	Size	Rate	CPU(%)	Mem(%)	RTT(p95)	RTT(p99)
HTTPS	aesgcm	512	1	189.753	4.532	43.502	91.565
	aesgcm	2048	100	197.776	5.452	21.02	59.096
	aes256gcm	512	1	190.108	5.677	47.73	98.052
	aes256gcm	2048	100	199.653	5.87	18.558	54.038
	chacha20	512	1	191.018	5.935	45.869	123.636
	chacha20	2048	100	200.43	6.162	18.722	55.248
DTLS	aesgcm	512	1	192.72	5.95	21.296	37.107
	aesgcm	2048	100	192.822	6.272	13.247	13.299
	aes256gcm	512	1	192.695	6.34	24.903	40.386
	aes256gcm	2048	100	191.11	6.583	13.482	13.598
	chacha20	512	1	191.07	6.508	24.965	49.288
	chacha20	2048	100	192.982	6.354	13.165	13.173
MQTT	aesgcm	512	1	191.337	6.479	40.607	148.114
	aesgcm	2048	100	195.972	6.376	225.516	395.676
	aes256gcm	512	1	191.067	6.272	33.153	69.767
	aes256gcm	2048	100	196.103	6.576	109.641	313.947
	chacha20	512	1	190.857	6.615	33.203	143.463
	chacha20	2048	100	195.12	6.462	73.125	205.364
HTTP/3	aesgcm	512	1	189.191	6.472	26.209	59.087
	aesgcm	2048	100	193.335	6.543	17.904	50.104
	aes256gcm	512	1	190.861	6.502	34.25	61.503
	aes256gcm	2048	100	195.21	6.636	61.223	93.436
	chacha20	512	1	190.968	6.623	36.308	87.811
	chacha20	2048	100	194.125	6.569	13.541	31.485

특히 MQTT에서 AES-GCM을 사용한 2048B·Rate=100 조합은 p95가 225.516ms, p99가 395.676ms로 다른 조합 대비 매우 큰 지연 스파이크가 관측되었다. 이는 고부하 메시지 전송 조건에서 MQTT의 메시지 처리 구조와 YOLO 추론 스레드 간 CPU 스케줄링 경합이 특정 시점에 집중되며 꼬리 지연이 급격히 증가한 결과로 해석된다. 한편, 중위값(p50)에서는 DTLS가 네 프로토콜 중 가장 높은 RTT를 보였으나, p95·p99에서는 오히려 다른 프로토콜보다 낮은 수준을 유지했다. 이는 DTLS의 평균 지연은 다소 높으나, 극단적 지연이 발생하는 꼬리 구간이 짧아 전체 분포의 편차가 작고 상대적으로 안정적임을 의미한다.

4.2 Jetson Orin Nano 실험 결과

4.2.1 프로토콜별 딥러닝 추론 성능 비교

그림 7과 그림 8은 각각 저부하(Rate=1, 512B)와 고부하(Rate=100, 2048B) 조건에서 측정된 프로토콜별 딥러닝 추론 성능(FPS)을 나타낸다. 그림 7에서 확인할 수 있듯이, 데이터 전송량이 적은 저부하 조건에서는 모든 프로토콜이 평균 7.9~8.1 FPS 범위 내에서 동작하며 유의미한 성능 차이를 보이지 않았다. 이는 저부하 상황에서 네트워크 입출력 처리에 소모되는 CPU 사이클이 전체 시스템 부하에서 차지하는 비중이 미미하여, GPU가 주도하는 딥러닝 추론 파이프라인에 병목을 유발하지 않기 때문이다. 그러나 통신 부하가 증가한 그림 8의 결과를 살펴보면, 프로토콜 오버헤드에 따른 성능 차이가 명확히 드러났다. UDP 기반의 DTLS는 7.93~8.00 FPS를 기록하며 저부하 조건과 거의 유사한 성능을 유지했다. 이는 비연결형 특성상 연결 상태 유지 및 패킷 순서 보장에 필요한 CPU 오버헤드가 가장 적어, 딥러닝 추론을 위한 데이터 공급이 원활하게 이루어졌기 때문이다. TCP 기반의 MQTT 프로토콜은 7.61~7.69 FPS를 기록하며 DTLS에 준하는 우수한 효율성을 보였다. 이는 일반적으로 TCP가 UDP 대비 무겁다는 통념과 달리, MQTT의 경량 헤더 구조와 단순한 메시지 처리 방식이 고부하 상황에서도 CPU 부하를 효과적으로 낮게 유지했음을 시사한다. HTTP/3와 HTTPS는 각각 7.3 FPS 수준과 6.8 FPS 수준을 기록하며 상대적으로 낮은 성능을 보였다. HTTP/3는 UDP 기반임에도 불구하고 사용자 영역에서 암호화 및 패킷 처리가 수행됨에 따라, 잦은 컨텍스트 스위칭 비용이 발생하여 추론 스레드와의 자원 경합을 유발한 것으로 판단된다. 반면, HTTPS는 고부하 조건에서 6.7~6.8 FPS로 네 프로토콜 중 가장 낮은 성능을 보였다. 이는 TCP 기반 전송에서 필수적으로 수행되는 ACK 처리, 재전송 관리, 세그먼트 조립과 같은 커널 레벨 오버헤드가 TLS 암호화 연산과 결합되어 CPU 부담을 크게 증가시키기 때문이다. 실제로 동일 조건에서 HTTPS의 CPU 사용률은 약 112%까지 상승하여 시스템 내 주요 연산 자원을 점유하였고, 그 결과 추론 스레드가 확보할 수 있는 CPU 여유가 감소하면서 전체 딥러닝 추론 성능이 상대적으로 낮아진 것으로 해석된다.

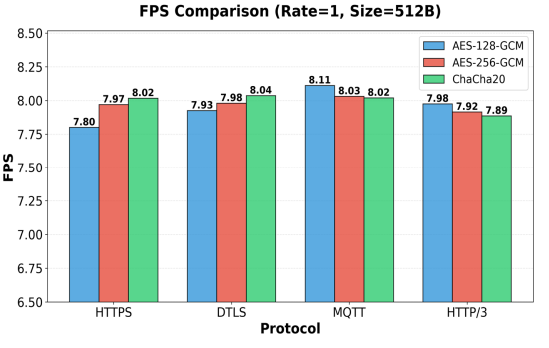

그림 7. Jetson Orin Nano의 FPS(Rate=1, Size=512B)
Fig. 7. FPS on Jetson Orin Nano(Rate=1, Size=512B)

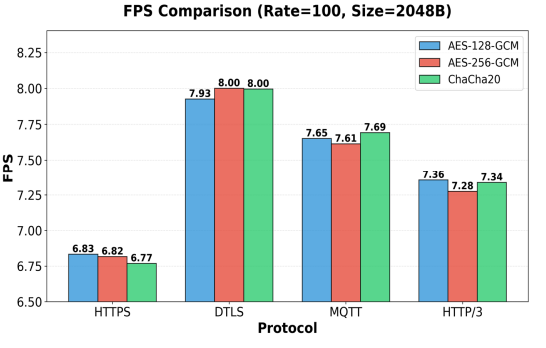

그림 8. Jetson Orin Nano의 FPS(Rate=100, Size=2048B)
Fig. 8. FPS on Jetson Orin Nano(Rate=100, Size=2048B)

4.2.2 네트워크 지연 분석

그림 9와 그림 10은 각각 저부하 및 고부하 조건에서의 네트워크 왕복 지연 시간 분포를 보여준다. 그림 9의 저부하 조건(Rate=1, Size=512B)에서는 경량 프로토콜의 이점이 두드러졌다. MQTT는 암호화 알고리즘과 무관하게 8.14~9.05ms의 가장 낮은 지연 시간을 기록했으며, HTTP/3가 9.18~12.11ms로 그 뒤를 이었다. 반면, HTTPS와 DTLS는 각각 15ms, 13ms 내외로 상대적으로 높은 지연을 보였다. 이는 MQTT의 단순한 패킷 구조가 단건 메시지 전송 시 오버헤드를 최소화하기 때문이다.

그림 10의 고부하 조건(Rate=100, Size=2048B)에서는 프로토콜과 암호화 알고리즘 조합에 따라 지연 양상이 크게 달라졌다. MQTT는 7.94~9.33ms 수준의 RTT를 유지하며 고부하 상황에서도 가장 안정적인 성능을 보였다. HTTP/3 또한 AES-GCM 적용 시 약 9.24ms로 낮은 지연을 유지하였지만, ChaCha20을 적용한 경우 p50 RTT가 181.47ms까지 증가하는 급격한 성능 저하가 발생했다. 이는 Jetson Orin Nano에서 AES-GCM이 ARMv8 암호화 확장을 통해 하드웨어적으로 처리되는 반면, ChaCha20은 소프트웨어 기반 연산에 의존하여 고부하 환경에서 CPU 자원 경합이 크게 발생하기 때문이다. 특히 QUIC은 TCP와 달리 사용자 공간에서 암호화와 전송 제어를 수행하므로, CPU 스케줄링 지연이 패킷 처리 지연으로 직접 이어져 RTT 변동성이 크게 확대된다. 이러한 자원 경합과 사용자 공간 처리 오버헤드가 누적되면서, HTTP/3+ChaCha20 조합에서 지연이 비정상적으로 증가한 것으로 해석된다.

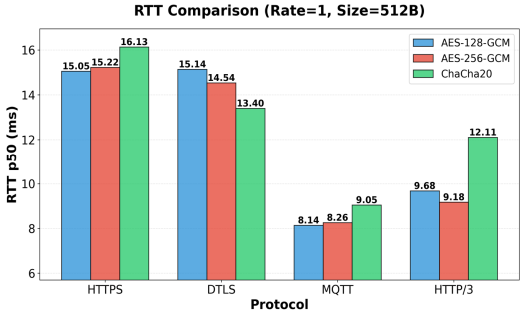

그림 9. Jetson Orin Nano의 RTT(Rate=1, Size=512B)
Fig. 9. RTT on Jetson Orin Nano(Rate=1, Size=512B)

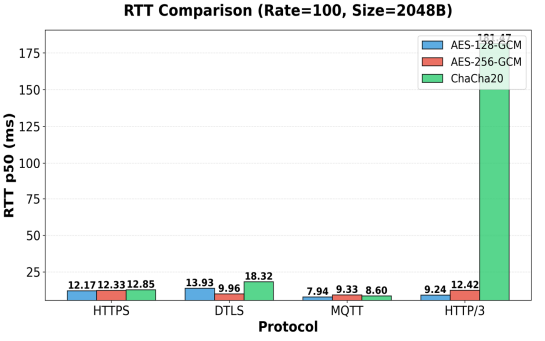

그림 10. Jetson Orin Nano의 RTT(Rate=100, Size=2048B)
Fig. 10. RTT on Jetson Orin Nano(Rate=100, Size=2048B)

4.2.3 암호화 알고리즘에 따른 성능 영향

암호화 알고리즘에 따른 성능 차이는 Jetson Orin Nano의 하드웨어 가속 기능과 프로토콜 처리 방식의 결합에 따라 달라진다. FPS 측면에서 AES-128-GCM, AES-256-GCM, ChaCha20-Poly1305의 성능 차이는 모든 프로토콜과 조건을 통틀어 최대 약 0.22 FPS 수준에 불과해, 딥러닝 추론 처리량에는 실질적으로 거의 영향을 주지 않았다. 그러나 지연 특성에서는 특정 조합에서 뚜렷한 차이가 나타났다. 특히 HTTP/3의 고부하 조건(2048B·Rate=100)에서는 ChaCha20의 p99 지연이 약 742ms까지 증가했지만, AES 계열은 수십 ms 수준에 머물러 꼬리 지연에서 큰 격차를 보였다. 이는 QUIC 계층의 패킷 처리와 암호화 연산이 모두 CPU에서 수행되는 구조에서, 소프트웨어 기반 알고리즘인 ChaCha20이 네트워크 스택과 YOLO 추론 스레드와의 자원 경합을 심화시킨 결과로 해석될 수 있다. 반면 AES-GCM은 ARM Cortex-A78AE가 제공하는 ARMv8 암호화 확장 명령어를 통해 하드웨어 가속 경로를 활용할 수 있어, 동일 조건에서도 지연 분포가 상대적으로 안정적이었다. 이러한 결과는 하드웨어 가속을 지원하는 온디바이스 환경에서 고부하 상황의 지연 안정성을 고려할 때, AES 계열 알고리즘이 ChaCha20 대비 효율적인 선택임을 시사한다.

4.2.4 시스템 자원 사용률 분석

Jetson Orin Nano 환경에서 측정한 시스템 자원

데이터를 분석한 결과, 프로토콜 간 자원 점유율의 절대적인 수치 차이는 크지 않았으나 자원 활용의 효율성 측면에서는 유의미한 패턴이 관찰되었다. 표 5의 저부하 조건에서는 모든 프로토콜이 CPU 102~105%, GPU 32~36% 범위를 유지하며 대등한 자원 사용량을 보였다. 이는 통신 부하가 낮을 때는 프로토콜의 오버헤드가 전체 시스템에 미치는 영향이 미미하여, 자원이 딥러닝 추론 작업에 온전히 집중될 수 있음을 시사한다. 그러나 고부하 조건에서는 프로토콜별 처리 방식 차이에 따른 자원 사용 특성이 뚜렷하게 나타났다. HTTPS는 CPU 사용률이 112%로 가장 높았으며, TCP 세그먼트 관리와 TLS 암호화 연산이 중첩되어 CPU 자원 소모가 크게 증가했다. GPU 사용률은 약 30% 수준으로 유지되어 GPU 측 병목은 관찰되지 않았다. DTLS는 CPU 사용률이 105~107% 범위에서 안정적으로 유지되었으며, 고부하 조건에서도 지연 분포가 큰 변동 없이 일정한 수준을 유지했다. 이는 UDP 기반 DTLS가 연결 유지와 세그먼트 조립에 필요한 부하가 상대적으로 적어, 꼬리 지연이 크게 증가하지 않는 특성을 보였음을 의미한다. MQTT는 조합에 따라 CPU 사용률이 66~109% 범위에서 크게 변동하였으며, 일부 조합에서는 CPU 사용률과 GPU 사용률이 동시에 낮아지는 현상이 관찰되었다. 이는 MQTT의 단일 이벤트 루프 구조가 특정 조건에서 GPU로의 연산 공급을 지속적으로 유지하지 못할 때 간헐적 idle 구간이 발생할 수 있음을 시사한다. 또한 ChaCha20 적용 시에는 p95·p99 구간에서 뚜렷한 지연 스파이크가 나타나, MQTT의 지연 특성이 암호화 알고리즘 선택에 민감하게 반응할 수 있음을 확인했다. HTTP/3는 AES-GCM 적용 시 자원 사용률과 지연 특성이 안정적이었으나, ChaCha20 적용 시 p95·p99 지연이 각각 589ms와 742ms로 급증했다. 이는 QUIC의 사용자 공간 기반 처리 경로가 소프트웨어 기반 암호화 부하와 결합되면서 스케줄링 지연이 누적된 결과로 분석된다.

표 5. Jetson Orin Nano 환경에서의 프로토콜 및 암호화 알고리즘별 성능 측정 결과
Table 5. Performance Measurement Results by Protocol and Encryption Algorithm on Jetson Orin Nano

Protocol	Cipher	Size	Rate	CPU(%)	GPU(%)	Mem(%)	RTT(p95)	RTT(p99)
HTTPS	aesgcm	512	1	101.182	32.847	12.495	48.304	193.92
	aesgcm	2048	100	112.701	30.37	13.026	29.088	70.985
	aes256gcm	512	1	102.34	32.854	13.137	103.009	140.676
	aes256gcm	2048	100	111.238	30.418	13.282	31.88	74.543
	chacha20	512	1	103.106	32.205	13.351	105.663	192.446
	chacha20	2048	100	111.46	31.402	13.438	38.575	97.448
DTLS	aesgcm	512	1	103.473	29.937	13.59	75.378	122.911
	aesgcm	2048	100	105.485	31.255	13.638	14.513	14.565
	aes256gcm	512	1	104.295	33.31	13.71	59.138	111.782
	aes256gcm	2048	100	106.875	34.05	13.84	10.065	10.074
	chacha20	512	1	105.532	30.599	13.862	43.721	101.076
	chacha20	2048	100	106.682	32.276	13.888	19.656	19.774
MQTT	aesgcm	512	1	103.879	30.659	13.888	26.757	101.495
	aesgcm	2048	100	66.011	18.246	8.39	23.067	45.511
	aes256gcm	512	1	92.224	31.036	12.512	52.784	124.051
	aes256gcm	2048	100	109.215	31.63	13.974	67.902	124.438
	chacha20	512	1	82.38	25.366	11.195	29.422	88.646
	chacha20	2048	100	109.7	32.442	14.006	130.08	234.665
HTTP/3	aesgcm	512	1	102.54	30.208	14.09	24.859	58.902
	aesgcm	2048	100	105.68	29.779	14.22	38.903	64.558
	aes256gcm	512	1	102.546	35.841	14.306	24.84	47.244
	aes256gcm	2048	100	105.868	34.131	14.367	58.809	83.602
	chacha20	512	1	102.008	34.125	14.388	45.902	109.75
	chacha20	2048	100	107.051	29.87	14.401	589.836	742.354

Ⅴ. 결론 및 향후 과제

본 논문에서는 온디바이스 장치에서 보안 프로토콜과 암호화 알고리즘 구성이 딥러닝 추론 성능과 네트워크 지연에 미치는 영향을 Raspberry Pi 5와 Jetson Orin Nano에서 비교·분석했다. 두 플랫폼 모두에서 저부하 환경에서는 AES-128-GCM, AES-256-GCM, ChaCha20 간 FPS 차이가 거의 없었고 프로토콜별 성능 편차 역시 제한적이었다. 그러나 고부하 조건에서는 프로토콜·암호화 조합에 따른 성능 차이가 뚜렷하게 나타났다. DTLS는 두 플랫폼에서 대체로 가장 높은 FPS를 보였다. 반면 HTTPS와 HTTP/3는 상대적으로 낮은 FPS를 기록하였으며, HTTP/3는 낮은 RTT를 제공했음에도 일부 조합에서는 지연 편차가 커지는 현상이 확인되었다. 또한 부하가 증가함에 따라 FPS가 전반적으로 하락하고 특정 조합에서 RTT의 불안정성이 나타나는 공통적인 경향이 두 플랫폼에서 모두 관찰되었다.

이러한 실험 결과를 토대로 온디바이스 AI 환경의 보안 프로토콜 및 암호화 알고리즘 선택 기준을 다음과 같이 제시한다. 첫째, 고부하 영상 처리나 로봇 제어와 같이 FPS 유지와 지연 안정성이 핵심인 응용은 UDP 기반 DTLS가 높은 FPS와 안정적인 지연 특성을 보이므로 우선 고려된다. 둘째, 제어 명령 전송 등 네트워크 왕복 지연이 핵심 성능 요소인 환경은 HTTP/3가 상대적으로 낮은 RTT를 기록했으며, 지연 편차를 최소화하려면 AES-GCM 구성이 ChaCha20보다 유리하다. 셋째, 저부하의 센서 데이터 보고나 주기적 텔레메트리 전송과 같은 경량 IoT 환경은 MQTT-TLS가 적은 자원 소모로 안정적 성능을 보여 실용적인 대안이 된다. 넷째, 웹 기반 서비스나 외부 시스템 연동 및 표준 호환성이 필수적인 환경은 HTTPS와 AES-GCM 조합이 가장 적합하다.

향후 연구에서는 다양한 부하 조건에서 FPS, RTT, 꼬리 지연의 변화 양상을 더 세밀하게 분석하여 보안 통신 오버헤드가 온디바이스 추론 성능에 미치는 영향을 보다 정밀하게 규명하고자 한다. 또한 하드웨어 플랫폼과 보안 프로토콜의 범위를 확대하고, 경량 객체 탐지 모델뿐만 아니라 대규모 언어 모델 등 파라미터 규모가 큰 모델을 포함한 비교 분석을 통해, 장치 및 프로토콜 특성에 따른 성능 차이를 종합적으로 검토함으로써 온디바이스 AI 환경에서 적용 가능한 보다 일반적인 보안 통신 기준을 도출할 수 있을 것으로 기대된다.

References

[1] NIPA, "Global ICT Portal – ICT Market Trends by Product (January 2023 – December 2023): Internet Product Trends", National IT Industry Promotion Agency, Jan. 2024.

[2] A. G. Howard, M. Zhu, B. Chen, K. A. Jarrett, S. Adam, R. Balan, and A. Y. Hannun, "MobileNets: Efficient convolutional neural networks for mobile vision applications", arXiv preprint arXiv:1704.04861, Apr. 2017. https://doi.org/10.48550/arXiv.1704.04861.

[3] S. Raza, H. Shafagh, K. Hewage, R. Hummen, and T. Voigt, "Lithe: Lightweight secure CoAP for the Internet of Things", IEEE Sensors Journal, Vol. 13, No. 10, pp. 3711-3720, Oct. 2013. https://doi.org/10.1109/JSEN.2013.2277656.

[4] B. Wukkadada, K. Wankhede, R. Nambiar, and A. Nair, "Comparison with HTTP and MQTT in Internet of Things (IoT)", Proc. Int. Conf. on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, pp. 249-253, Jul. 2018. https://doi.org/10.1109/ICIRCA.2018.8597401.

[5] S. Gueron, "Intel's New AES Instructions for Enhanced Performance and Security", Fast Software Encryption (FSE 2009), Vol. 5665, pp. 51-66, 2009. https://doi.org/10.1007/978-3-642-03317-9_4.

[6] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, IETF, Aug. 2018. https://doi.org/10.17487/RFC8446.

[7] M. Bishop, "Hypertext Transfer Protocol Version 3 (HTTP/3)", RFC 9114, IETF, Jun. 2022. https://doi.org/10.17487/RFC9114.

[8] A. Banks and R. Gupta, "MQTT Version 3.1.1", OASIS Standard, Dec. 2015. https://docs.oasis-open.

org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. [accessed: Jan. 02, 2026]

[9] E. Rescorla and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.2", RFC 6347, IETF, Jan. 2012. https://doi.org/10.17487/RFC6347.

저자소개

채 주 혁 (Ju-Hyeok Chae)

2021년 3월 ~ 현재 :
국립금오공과대학교
컴퓨터공학전공 학사과정
관심분야 : 웹 시스템, 서버

권 오 빈 (Obin Kwon)

2021년 3월 ~ 현재 :
국립금오공과대학교
컴퓨터공학전공 학사과정
관심분야 : 웹 시스템, 서버

김 진 홍 (Jinhong Kim)

2010년 2월 : 경북대학교
컴퓨터공학과(학사)
2013년 2월 : 경북대학교
전기전자컴퓨터학부(석사)
2013년 1월 ~ 현재 :
한국전자통신연구원 선임연구원
관심분야 : IoT, AI 엣지 컴퓨팅

안 형 태 (HyeongTae Ahn)

2020년 2월 : 포항공과대학교
컴퓨터공학과(공학박사)
2020년 3월 ~ 2021년 8월 :
한국전자통신연구원 연구원
2021년 9월 ~ 현재 :
국립금오공과대학교
컴퓨터공학과 조교수
관심분야 : 무선랜, 지능형 IoT, 스마트팜, 빅데이터