SemantJournals

American Journal of Technology Advancement 2997-9382, Vol 03, No. 10, 2026

Article

SDN-Controlled Adaptive Authenticated Encryption for Resource-Constrained IoT Networks: A Multi-Load Experimental Study

Hadi Hussein Madhi[1], Ali Mohammed Al-Saedi[2], Mustafa Challoob Albazoone[3]

[1]Department of Basic Science, College of Nursing, University of Misan, Iraq, hadihm8@uomisan.edu.iq

[2]IQ.Computer Science, University of Misan, Iraq, ali.m.a@uomisan.edu.iq

[3]IQ.Computer Science, University of Misan, Iraq, mustafa.h@uomisan.edu.iq

Abstract: Resource-constrained Internet of Things (IoT) deployments require confidentiality and integrity protection, although the most suitable authenticated encryption policy may differ according to traffic load and device pressure. The above study describes the working of an SDN controlled adaptive policy to select a suitable authenticated encryption configuration from a selection of packet encryption algorithms namely AES-128-GCM, AES-256-GCM and ChaCha20 Poly1305. This is a Linux/WSL2, Mininet, Open vSwitch, FAUCET controller, MQTT, and Python implementation. A full factor experiment involving three loads, three counts of devices, three sizes of payload, four policies and 30 repetitions yielded 3,240 completed runs. All of the policies achieved delivery of the 756,000 planned messages with zero aggregate loss, zero authentications failures and zero selection errors. For the entire matrix, the end-to-end median latency for the adaptive policy was 12.015 ms as compared to 12.092 ms for AES-128-GCM, 12.180 ms for AES-256-GCM and 12.027 ms for ChaCha20-Poly1305. The adaptive policy matched the behavior of its chosen fixed baseline at low and medium loads. Under high load, it was shown that the median latency vs AES-128-GCM was reduced by 2.118%. However, this impact was not statistically significant. According to the findings, relying on their own laboratory SDN policy allows for authentic delivery of data while explicitly permitting encryption choice depending on circumstances. We argue that our contribution is a repeatable framework for involving policy-management and evaluation, not that one cipher dominates all workloads.

Keywords: software-defined networking, Internet of Things, authenticated encryption, adaptive security policy, AES-GCM, ChaCha20-Poly1305

This is an open-access article under the CC-BY 4.0 license

1. Introduction

Deployments of Internet of Things increasingly connect sensors, gateways, and control applications over shared networks. These things usually do not have much processing capability or memory capacity. But they exchange data which needs confidentiality as well as integrity assurance. Because it provides encryption as well as tamper detection, authenticated encryption is a useful baseline. The problem in operation is that a fixed policy is chosen before deployment but load, device population and payload size can change after deployment. This study looks at whether SDN can coordinate that choice at the network layer whilst keeping the cryptographic operation at the

endpoint. [1], [2], [3].

AES-GCM hardware support is widely available across multiple platforms, unlike ChaCha20-Poly1305, which is often used in fallback situations where AES acceleration is unavailable. No single property is sufficient for identifying the optimal deployment policy for a constrained IoT network. In any given traffic scenario, a useful policy must be assessed, and it must also maintain authenticated delivery in the decision process.

As a result, the current study compares fixed AES-128-GCM, fixed AES-256-GCM, fixed ChaCha20-Poly1305, and an SDN-controlled adaptive policy under identical experiment matrix.

[4], [5], [6].

This research seeks to address three questions: Does the adaptive policy still guarantee reliably authenticated MQTT delivery? What are the impacts of load, number of devices, payload size, and combination of policy on the following metrics: Latency, Throughput, CPU Use, Memory Use, and an estimated proxy for energy use? Does the adaptive policy give a practically useful trade-off when compared with the fixed policy it selects for each load class? The research presents a clear policy rule, a complete factorial experimental design with 3,240 repetitions, and a breakdown that separates statistically significant differences from differences too small to be operationally significant.

The project focuses on application protection and policy selection of the data-plane. It does not suggest a new cipher, substitute protocol key management, and does not treat the SDN controller as an encryption endpoint. The controller gets the operating class that the publisher configured and applies a policy choice; the publisher and subscriber continue to perform authenticated encryption and verification. This separation makes sure the design stays compatible with endpoint security boundaries. It also makes it possible to tell apart a failed, delayed controller decision from an indistinguishable cryptographic failure. The question becomes operational: is it possible for central policy management to select among known AEAD algorithms without impairing reliable delivery to IoT?

This scope also prevents overclaiming. Improvements that are workload-specific should not be generalized to all processors or all IoT settings. Consequently, the experiment separately records the overall outcomes, matched baseline outcomes, and the most demanding circumstance. Thus, this paper treats the adaptive policy as a network-management mechanism to be proven useful under stated conditions, and not as a cryptographic replacement for any of the standardised algorithms selected.

1.1 Related Works

1.1.1 SDN-Enabled IoT Security

The control decisions and forwarding behavior can be decoupled if SDN is introduced. It allows the controller to have a view of the entire network. The view will encapsulate host, flows and policy state. Researchers have previously utilized this ability in designing privacy-aware IoT architectures, performing evidence collection and fog security, managing traffic, and thwarting attacks. While these studies prove the usefulness of centralized policy control, they typically deal with routing, monitoring, or attack response problems rather than picking an authenticated-encryption policy as operating conditions change. [7], [8], [9], [10], [11], [12].

According to survey evidence, IoT security decisions need to be made in light of heterogeneous endpoints and resource constraints. Reshi and Sholla discuss broader security and privacy challenges of IoT systems, while Ohri and Neogi identify security problems and solutions of SDN deployments. The pressures call for putting the policy coordination in the control plane of the SDN rather than forcing every constrained device to make an independent system-level choice. [13], [14].

Cryptographic Processing isn't cheap just because of SDN. In this context, visibility and consistency of policies have value. A controller can recognize the traffic class identify, captures the current working context, and orchestrates that the similar rule applies for comparable flows. The enpoint still has the encryption and verification cost This difference is important because the present study investigates the end-to-end impact of a controller decision, not the effect of control plane on endpoint behavior. Healthcare-focused SDN-fog research also indicates that security architecture,

deployment restrictions, and application workload must be jointly optimized. [15], [2].

To secure IoT with SDN, it is necessary to perform more than installing encryption at the endpoint and enforcing static forwarding rules at the network. A variety of devices with different processing capacities, message rates, communication patterns, and security requirements coexist in IoT environments. A policy that works for a lightly loaded sensor network can introduce delay as the number of active nodes is increased or as the message payload is increased. The security mechanism thus should be treated as an operating environment and not as a fixed configuration to be used for every conditions. As SDN can gather or receive information about the state of the network, apply predetermined rules, and coordinate traffic handling, it is suitable as the management layer since not much control function is imposed on each constraint device [9],[11],[13],[14].

Nonetheless, central control needs to be limited. When sending packets, the controller should not read the application plaintext, substitute the endpoint authentication, or make unauthenticated cryptographic choices. This frequently uncontrolled application adjusts operations according to activity load, the number of active machines, the size of the payload in the network. The IoT communication workflow can then apply the selected policy and message level, confidentiality and integrity is still provided by authenticated encryption. This separation maintains the spirit of authenticated encryption, while facilitating the inspection, repetition, and assessment of the controller's decision [2], [15].

The gap in the relevant literature is also addressed by this viewpoint. Most existing SDN-IoT studies typically focus on issues like routing, access control, attack detection, resource management, or general security advantages. The works establish the usefulness of SDN systems in heterogeneous IoT environments but do not always consider how the choice of an authenticated-encryption policy should vary with operating conditions. It may be appropriate to use AES-128-GCM, AES-256-GCM, or ChaCha20-Poly1305 fixed to a particular workload. But no one fixed choice can be assumed to provide the same balance of cryptographic strength, latency, and processing demand over every workload. The current study thus assesses a controller managed adaptive policy along with three fixed AEAD baselines. This document investigates whether or not the policy-selection process can facilitate reliable authenticated communication in response to changing load, device population and payload size. This design keeps contributions focused: it is not about whether SDN replaces cryptographic protection or whether one cipher is simply superior to another [2], [9], [11], [13]–[15].

1.1.2 Lightweight Authenticated Encryption

Research on lightweight cryptography indicates that the implementation cost relies on the primitive and target platform. Low-end microcontroller optimization for AES-GCM and implementation studies versus CPU families and high-performance multiplier designs. Lightweight authenticated-encryption modes and primitives have been developed, such as AES-LBBB and Ascon. Although load forecasting impacts the changes made to traffic policies, the changes are not necessarily made according to a controller rule. [16], [5], [17], [18], [19], [20].

Recent reviews still focus on the trade-off among security strength, execution cost, memory footprint, interoperability, application requirements. Developing compression-encryption approaches, 8-bit security for the Internet of Things, post-quantum alternatives, and health IoT protection enlarges the design space. The present study is purposefully narrower: it uses mature standardized AEAD algorithms and tests when an SDN policy should select them in a controlled manner within a MQTT deployment. [21]- [27].

This comparison is deliberately confined to algorithms that are well known to systems implementers. AES-GCM is a widely used authenticated encryption based on AES with two key sizes and, while ChaCha20-Poly1305 is a standardized stream-cipher-based alternative. This avoids the complication of the study introducing new primitive design and keeps the policy question clear: under a common MQTT and SDN testbed, does a load-aware selection rule behave as intended relative to fixed configurations? The response relates to measures not to general statements that one

group of families is always lighter than the other.

1.1.3 Gap and Position of the Current Study

The closest recent trend is context-centric encryption for SDN powered smart-home IoT. The work provides evidence that encryption can be affected by context, a deployment claim still requires a clear rule, reproducible testing conditions, fixed baselines, and results over multiple stressors. This study implements the controller such that AES-256-GCM is selected by the controller at low and medium load, while AES-128-GCM is selected a high load; ChaCha20-Poly1305 remains a candidate when endpoint CPU or memory pressure is the governing constraint. The policy has been designed to be simple, so that we can test its operational effect, rather than an optimizer that is not transparent. [3], [28], [29].

The summaries of the studies most directly used to situate the work is provided in Table 1. Among the earlier evidence is an IoT security survey, an SDN security survey, and a lightweight-aes implementation as three papers published in the International Journal of Computing and Digital Systems. The evidence they present shows that the journal has relevant IoT, SDN, and lightweight-cryptography work. The present study differs in evaluating controller-managed selection over the entire performance matrix, unlike previous works which either evaluate a single cipher implementation or provide a survey.

Table 1. Comparison of related studies and the current study

Reference	Study setting	Encryption or security focus	Adaptation or SDN role	Scope relative to the current study
[4]	Lightweight IoT devices	Authenticated-encryption comparison	None	Compares primitives; no controller policy or load matrix.
[16]	Processor implementation	AES-GCM	None	Implementation study; no IoT deployment or adaptive selection.
[7]	Smart-city IoT	Privacy-preserving SDN method	Context-aware SDN	Uses SDN context but does not compare AEAD policies.
[8]	SDN-IoT	Blockchain-enabled forensics	SDN architecture	Forensics focus; no encryption-policy selection.
[5]	IoT CPU families	AES-GCM	None	Platform comparison; no controller decision.
[9]	IoT networks	SDN security framework	SDN enforcement	No AEAD evaluation or policy-selection rule.
[17]	Low-end microcontrollers	AES-GCM	None	Endpoint optimization; fixed cipher.
[19]	Constrained systems	Ascon AEAD	None	Primitive design; not SDN managed.
[1]	Resource-constrained IoT	Lightweight cryptography review	None	Identifies resource trade-offs; no deployed selection.
[6]	TLS 1.3	ChaCha20-Poly1305	Protocol setting	Not an IoT controller policy.
[13]	IoT	Security survey	None	Identifies IoT security challenges; no AEAD selection.
[14]	SDN	Security survey	SDN survey	Identifies SDN security challenges; no AEAD selection.

Reference	Study setting	Encryption or security focus	Adaptation or SDN role	Scope relative to the current study
[30]	IoT hardware	Lightweight AES FPGA implementation	None	Fixed-cipher implementation.
Current study	Mininet, OVS, FAUCET, and MQTT	AES-128-GCM, AES-256-GCM, and ChaCha20-Poly1305	SDN adaptive rule	Four policies; 3,240 runs; delivery, latency, throughput, CPU, and memory measures.

2. Materials and Methods

2.1 Research Design

The study uses a controlled factorial experiment. Each The run transmits MQTT messages through a Mininet topology while the employed AEAD policy safeguards application data. The Open vSwitch is used to build the forwarding plane and the FAUCET controller serves as the point of centralized visibility and policy-management. The architecture separates forwarding, monitoring, and policy selection operations logically, so that experiments can be reproduced, and fixed and adaptive policies can be compared in identical circumstances. [31], [32], [33].

A run starts by setting the four experimental factors. The fixed or adaptive policy driven by the controller is applied, the publisher creates the MQTT traffic as configured, the subscriber checks the authentication tag and records the delivery, and the timings and process measures are collected by the runtime. The run gets cleaned up for the next condition in line. This sequence prohibits a condition from intentionally selecting a preceding topology state or policy. All four policies have the same lifecycle, so it is not an artifact of a different traffic procedure but the policy label which is the variable of interest.

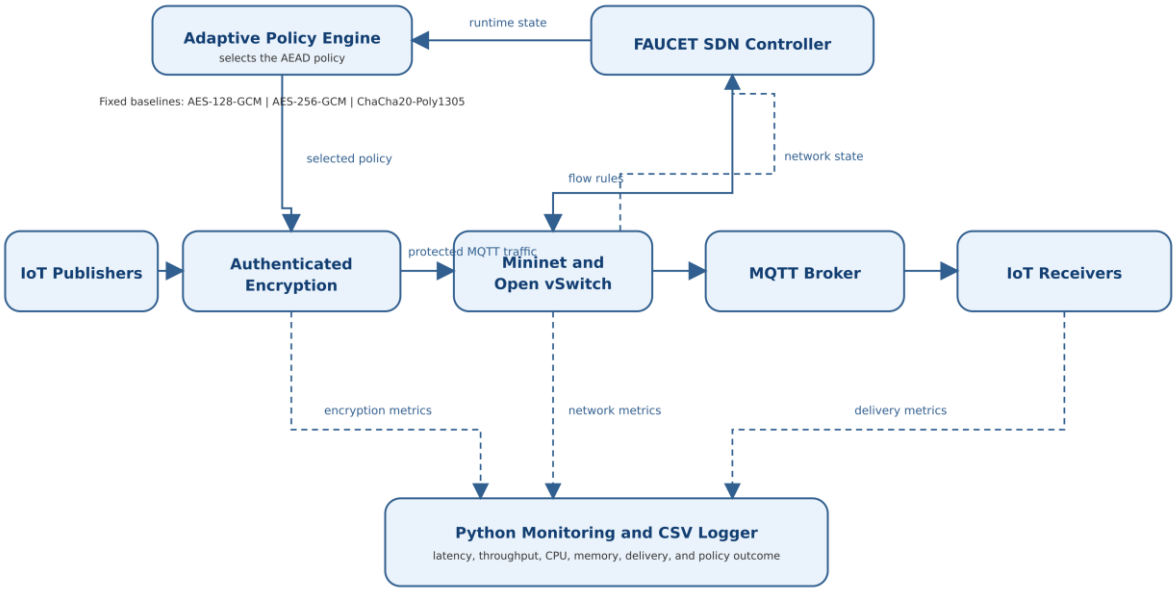


Figure 1. SDN-controlled adaptive authenticated-encryption architecture.

2.2 Testbed and Policy Rules

The testbed was executed on Linux through WSL2 with Mininet, Open vSwitch, FAUCET, MQTT, and Python. For each run, the application logic logged the encryption time, decryption time, end-to-

end latency, throughput, process CPU use, resident memory, and an estimated energy proxy. The proxy is kept as a comparative measure of software and is not understood as a measure of physical power. The controller chooses the policy pre-execution based on the designated load class. In turn, MQTT delivers the publish/subscribe transport. [34], [35], [33].

The fixed configurations utilize AES-128-GCM and AES-256-GCM and ChaCha20-Poly1305 during the entirety of executions. The adaptive configuration selects AES-256-GCM (for low/medium load classes) and AES-128-GCM (for high load class) at the same endpoint implementation. This design renders the adaptive outcome comprehensible. A matched comparison questions whether the configuration controlled by the controller differs from the fixed configuration it would otherwise use. As a representative non-AES AEAD option, ChaCha20-Poly1305 continues to be a benchmark and one that may be preferred as a future controller observes endpoint resource pressure directly.

Table 2. Encryption policies used in the experiment

Policy	Selection rule	Purpose
AES-128-GCM	Fixed baseline	Lower-key-size AES-GCM baseline.
AES-256-GCM	Fixed baseline	Higher-key-size AES-GCM baseline.
ChaCha20-Poly1305	Fixed baseline	Non-AES standardized AEAD baseline.
Adaptive policy	AES-256 at low/medium load; AES-128 at high load	Tests SDN-controlled selection against its matched baseline.

2.3 Experiment Matrix and Analysis

The experiment combined four types of policies with three load levels of 20%, 50% and 80%, three IoT devices number of 5, 10, and 20, three payload sizes of 256, 1024 and 4096 bytes and repeated 30 times. The design gives on productions of 4 x 3 x 3 x 3 x 30 = 3240. What was the same in fixed policies and adaptive decisions. Analysis reports medians and quartiles for skew-resistant latency and throughput summaries mean values for process metrics and Friedman tests for policy comparisons The outcome is viewed via practical significance, besides p-value.

Many classes of load are viewed as controlled stress settings, rather than a public network observation. The number of emulated IoT participants in your topology is given by device count, and the application message size is referred to as the payload size. With 30 repetitions for each condition, our final results rely on a distributional summary that mitigates one transient scheduling event's impact. The test does not imply a global benchmark score, but rather how the policies respond given these specific operational aspects are altered systematically.

Latency is reported from sender to receiver application therefore it includes the ciphered application path rather than just the encryption time. The reported throughput values in Mbps are interpreted with payload size because larger payloads change the ratio of useful data to fixed-message overhead. CPU and resident memory refer to process level observations. The energy result is a software estimate obtained through the experiment and is only intended for relative comparison.

According to these definitions, a software run time estimate is not a hardware measure.

Table 3. Full-factor experiment matrix

Factor	Levels	Values
Policy	4	AES-128-GCM; AES-256-GCM; ChaCha20-Poly1305; adaptive
Load	3	20%; 50%; 80%
IoT devices	3	5; 10; 20
Payload	3	256; 1,024; 4,096 bytes
Repetitions	30	Per complete condition
Total	3,240	4 x 3 x 3 x 3 x 30

3. Results

3.1 Delivery Integrity

Completion of all 3,240 experiments. A total of 756,000 planned MQTT messages were received with no aggregate losses. The logs recorded no authentication failure as well as policy-selection error. As the results demonstrate, the comparisons that follow refer to delivered authenticated finished rather than to a policy that enjoys a superficial performance by dropping or circumventing protected traffic.

The integrity outcome is important since authenticated encryption only succeeds if the receiver accepts a valid tag and the application is delivered the intended protected message. It will be incomplete without confirmation for latency comparison. The count of selection errors being zero also corroborates that the adaptive configuration applies the deterministic rule which states that low and medium load invokes AES-256-GCM, and high load invokes AES-128-GCM for every scheduled condition. Consequently, the reliability result is a necessary precursor to the performance analysis, not a substitute.

Table 4. Reliability and execution outcomes,

Measure	Result	Interpretation
Completed runs	3,240 / 3,240	Full experiment matrix completed.
Planned/received messages	756,000 / 756,000	Zero aggregate delivery loss.
Authentication failures	0	No failed integrity checks recorded.
Selection errors	0	Adaptive decision executed as configured.

3.2 Overall Policy Comparison

All the policies gave rise to closely matched end-to-end latency and throughput values in all situations. The median latency using adaptive policy was 12.015 ms like the AES-128-GCM case which

was 12.092 ms, AES-256-GCM the 12.180 ms and ChaCha20-Poly1305 the 12.027 ms . The median throughput 3.429 Mbps is in a narrow range of fixed baselines. AES-128-GCM had a lower mean CPU value as compared to other alternatives. Consequently, one should interpret the observed differences as policy-equivalent for the aggregate workload; responsiveness to conditions rather than universal work performance superiority creates selection value.

The timing components display a similar pattern. The average time taken to encrypt a packet was 0.896 milliseconds for AES-128-GCM, 0.923 milliseconds for AES-256-GCM, 0.906 milliseconds for ChaCha20-Poly1305, and 0.900 milliseconds for the adaptive policy. AES-128-GCM and the adaptive policy had a mean decryption time of 0.119 ms, 0.122 ms for AES-256-GCM, and 0.128 ms for ChaCha20-Poly1305. Not only are these sub-millisecond differences useful for implementation characterization, but they are also already much smaller than end-to-end effects arising from load, topology activity and message size.

Friedman tests did not show a significant policy impact on the encryption time ($p = .2488$), mean end-to-end latency ($p = .9319$), p95 latency ($p = .9402$), throughput ($p = .7434$) or CPU use ($p = .7722$). The decryption time, memory, and the energy proxy were different in statistical terms. Nonetheless, proof of statistical significance does not prove operational significance. The resident memory range across policies was just about 0.043 MB and the energy indicator is not an actual power figure. This distinction steers the discussion set out in Section 4.5.

Table 5. Overall policy results across the full matrix

Policy	End-to-end median (ms)	Throughput median (Mbps)	Mean CPU (%)	Mean RSS (MB)
AES-128-GCM	12.092	3.421	6.048	33.069
AES-256-GCM	12.180	3.438	6.145	33.070
ChaCha20-Poly1305	12.027	3.437	6.130	33.111
Adaptive	12.015	3.429	6.151	33.068

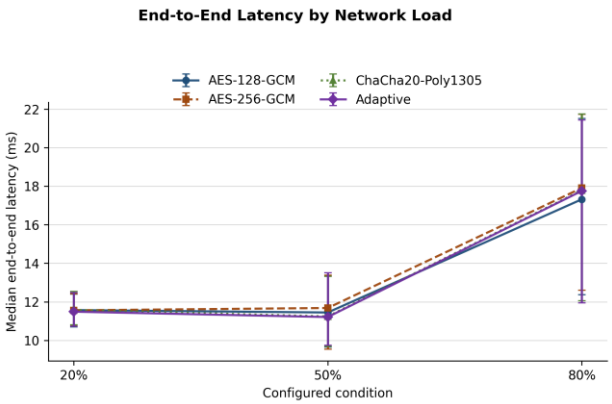


Figure 2. End-to-end latency distribution by offered load.

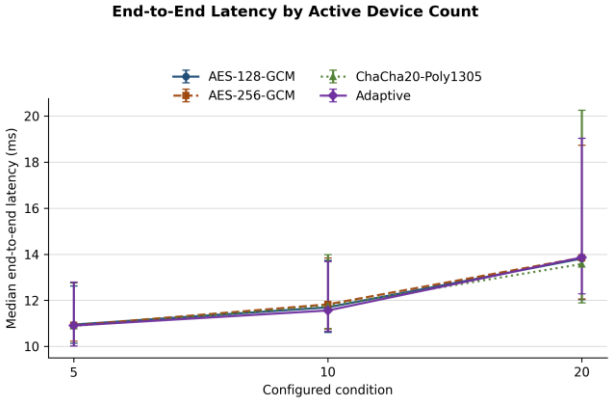


Figure 3. End-to-end latency distribution by device count.

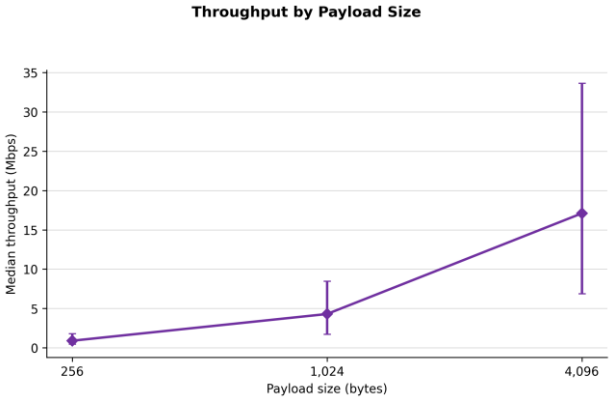


Figure 4. Throughput distribution by payload size.

3.3 Effects of Operating Conditions

The influence of operating conditions was greater than policy label. At low, medium and high load end to end latencies of 11.545 ms, 11.322 ms and 17.689 ms respectively. The latency increased from 10.909 milliseconds with 5 devices to 13.760 milliseconds with 20 devices. According to expectations, throughput scale was influenced by payload size: the median rose by 0.888 Mbps at 256 bytes to 17.097 Mbps at 4096 bytes while latency illustrates the effect due to protocol overhead and serialization. Analysis with the Friedman tests has shown that the load, the number of devices and the payload all have significant impacts on the main performance measures, almost always with $p < .001$.

The fact that the median at 20% and 50% load is non-monotonic should not be overinterpreted in terms of performance benefits related to medium load. The small difference, relative to the high-load increase, reflects the variability of a virtualized testbed with application, broker, forwarding, and

controller processes. Under 80% load, a clear change in latency happens, with CPU median growing from 2,000% to 10,489%. Consequently, the experiment gives rise to a policy rule that places more weight on persistent high load rather than a very small median difference between lightly loaded states.

Table 6. Operating-condition effects, pooled across policies

Factor	Level	Latency median (ms)	Throughput median (Mbps)
Load	20%	11.545	0.861
Load	50%	11.322	4.316
Load	80%	17.689	8.435
Devices	5	10.909	1.728
Devices	10	11.721	3.445
Devices	20	13.760	6.888
Payload	256 B	13.312	0.888
Payload	1,024 B	10.635	4.289
Payload	4,096 B	13.100	17.097

3.4 Adaptive Policy Against Its Matched Baseline

The evaluation of adaptive policy against a fixed policy was carried out the rule of which selected in the same load class. The adaptive median latency at low load was 11.751 ms versus 11.947 ms for AES-256-GCM. Against medium load, AES-256-GCM was 11.767 ms versus 11.742 ms. At high load, the adaptive latency was 29.568 ms, whereas that of AES-128-GCM was 30.452 ms; thus, a reduction of 2.118%. The matched comparisons yielded p-values of .299, .715, and .440. As a result, the adaptive policy closely followed the intended baseline behavior. Further, it did not result in an overall latency improvement that reaches statistical significance. For these reasons, its practical value is controlled selection with no penalty to reliability.

The throughput that matched was still quite similar. The adaptive median and AES-256-GCM were 1.759 and 1.760 Mbps under low load. Under medium load conditions, these speeds were 8.805 and 8.773 Mbps. At high load, adaptive and AES-128-GCM recorded 14.289 and 14.281 Mbps. The difference in latency between high-load and low-load is not due to a lower useful-data rate. It further affirms that the existing rule acts as a policy selector rather than a traffic-throttler.

Table 7. Adaptive policy versus the selected fixed baseline

Load	Matched fixed policy	Adaptive median (ms)	Baseline median (ms)	Difference	p-value
Low	AES-256-GCM	11.751	11.947	-0.676%	.299
Medium	AES-256-GCM	11.742	11.767	-0.158%	.715
High	AES-128-GCM	29.568	30.452	-2.118%	.440

4. Discussion

4.1 Worst-Case Condition and Discussion

When placed under the most challenging tested conditions – 80% load, 20 devices and 4096-byte payloads – mean end-to-end latency was 134.133 ms for AES-128-GCM; 132.427 ms for AES-256-GCM; 128.473 ms for ChaCha20-Poly1305; and 127.714 ms for the adaptive policy. The throughput median at the same conditions was between 40.931 and 43.531 Mbps. The stress of workload, we confirm, dominates the aggregate policy effect. Moreover, a control rule should be assessed by its degree of stability and transparent decision logic, rather than just by the headline averages.

The worst-case scenario also shows why just one overall average isn't enough. A policy that appears virtually identical to another policy over the complete matrix can exhibit a different ordering at high load and high payload. On the other hand, a small difference in one harsh condition cannot produce an overall winner. The adaptive policy is best understood as a device that makes this choice evident and reproducible. When a more sophisticated decision rule is warranted, it can be extended by a new telemetry or a different deployment objective.

The Friedman findings bolster this assertion. There were no significant differences between the policies in terms of encryption time ($p = .2488$), mean end-to-end latency ($p = .9319$), p95 latency ($p = .9402$), throughput ($p = .7434$), or CPU use ($p = .7722$). Although differences were statistically significant for each of the decryption time, resident memory, and the software energy proxy, the absolute memory range was about 0.043 MB and energy measure is not a physical power reading. As such, the evidence does not support a claim that adaptive control significantly exceeds all fixed ciphers. The controller can choose reliably according to conditions and with complete authenticated delivery.

5. Conclusion

The study demonstrated and tested the SDN-controlled authenticated-encryption policy on resource-constrained IoT traffic. The experiment ran 3,240 total runs on four policies, three load levels, three device numbers, three payload sizes, and 30 repetitions. All 756,000 messages were successfully delivered with no failed authentications and no adaptive-selection errors. The adaptive policy had an overall median end-to-end latency of 12.015 ms and remained within 1.65 ms of the fixed policies across the entire matrix.

The experiment has these explicit limitations: the environment is a tested Linux/WSL2 emulated environment (not an extensive real-world field deployment on multiple distinct physical IoT boards). The controller rule relies on manually defined load classes instead of a learned predictor or direct hardware energy sensing. Critical provisioning and control-plane authentication are omitted from the evaluation mechanism. These limitations aim to keep the experiment interpretable while also pinpointing which future validations should precede deployment-specific advice.

The main point is not that one algorithm wins all the time. Load, number of devices, and payload size were the most significant performance variables, whereas the adaptive rule simply maintained the baseline behavior and slightly improved the median latency by 2.118% over that baseline at high load without a significant difference to AES-128-GCM at high load. This is an interesting result for policy management, where one can make the decision to manage security-performance explicitly rather than having an implicit default, untested policy.

Further work will include: more detailed telemetry, more endpoint hardware, physical energy metrics, longer duration traffic, and a policy that responds to measured CPU and memory thresholds rather than a configured load class. The architecture is extensible with formal policy verification, more IoT application protocols, and larger topologies. All extensions should follow the design principle used here: compare adaptive behavior with transparent fixed baselines for the same traffic at the same conditions.

5.1 Data and Code Availability

The full reproducibility package for this work is available at the GitHub repo https://github.com/hadihm8/sdn-iot-adaptive-aead and the archived Zenodo release https://doi.org/10.5281/zenodo.22803282. This package contains the source code, FAUCET config, Mininet topology, MQTT and authenticated-encryption scripts, experimental procedure, raw and processed data from all the 3,240 experiments, data analysis scripts and presentation figures. It can be used for independent validation, reproduction of the experimental results, and for extending the proposed SDN-controlled adaptive authenticated-encryption concept.

REFERENCES

[1] V. A. Thakor, M. A. Razzaque, and M. R. A. Khandaker, "Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities," IEEE Access, vol. 9, pp. 28177-28193, 2021, doi: 10.1109/ACCESS.2021.3052867.

[2] S. R. Mishra, B. Shanmugam, K. C. Yeo, and S. Thennadil, "SDN-enabled IoT security frameworks: A review of existing challenges," Technologies, vol. 13, no. 3, Art. no. 121, 2025, doi: 10.3390/technologies13030121.

[3] E. Gilliard and J. Liu, "CALIS: AI-driven context-aware encryption for SDN-enabled smart-home IoT," Journal of King Saud University Computer and Information Sciences, 2026, doi: 10.1007/s44443-025-00404-9.

[4] S. Koteshwara and A. Das, "Comparative study of authenticated encryption targeting lightweight IoT applications," IEEE Design & Test, vol. 34, no. 4, pp. 26-35, 2017, doi: 10.1109/MDAT.2017.2682234.

[5] Y. Sovyn, V. Khoma, and M. Podpora, "Comparison of three CPU-core families for IoT applications in terms of security and performance of AES-GCM," IEEE Internet of Things Journal, vol. 7, no. 1, pp. 339-348, 2020, doi: 10.1109/JIOT.2019.2953230.

[6] R. Serrano, C. Duran, M. Sarmiento, C.-K. Pham, and T.-T. Hoang, "ChaCha20-Poly1305 authenticated encryption with additional data for transport layer security 1.3," Cryptography, vol. 6, no. 2, Art. no. 30, 2022, doi: 10.3390/cryptography6020030.

[7] M. Gheisari, G. Wang, W. Z. Khan, and C. Fernandez-Campusano, "A context-aware privacy-preserving method for IoT-based smart city using software defined networking," Computers & Security, vol. 87, Art. no. 101633, 2019, doi: 10.1016/j.cose.2019.02.006.

[8] M. Pourvahab and G. Ekbatanifard, "An efficient forensics architecture in software-defined networking-IoT using blockchain technology," IEEE Access, vol. 7, pp. 99573-99588, 2019, doi: 10.1109/ACCESS.2019.2930345.

[9] A. Al-Hayajneh, M. Z. A. Bhuiyan, and I. McAndrew, "Improving Internet of Things security with software-defined networking," Computers, vol. 9, no. 1, Art. no. 8, 2020, doi: 10.3390/computers9010008.

[10] A. Alamer, "Security and privacy-awareness in a software-defined fog computing network for the Internet of Things," Optical Switching and Networking, vol. 41, Art. no. 100616, 2021, doi: 10.1016/j.osn.2021.100616.

[11] M. Priyadarsini and P. Bera, "Software defined networking architecture, traffic management, security, and placement: A survey," Computer Networks, vol. 192, Art. no. 108047, 2021, doi: 10.1016/j.comnet.2021.108047.

[12] N. Dayal and S. Srivastava, "Analyzing effective mitigation of DDoS attack with software defined networking," Computers & Security, vol. 132, Art. no. 103269, 2023, doi: 10.1016/j.cose.2023.103269.

[13] A. A. Reshi and S. Sholla, "Challenges for security in IoT, emerging solutions, and research directions," International Journal of Computing and Digital Systems, vol. 12, pp. 1895-1905, 2022, doi: 10.12785/ijcds/120198.

[14] J. Ohri and S. Neogi, "Software-defined networking security challenges and solutions: A comprehensive survey," International Journal of Computing and Digital Systems, vol. 12, pp. 311-328, 2022, doi: 10.12785/ijcds/120131.

[15] M. Kumhar and J. Bhatia, "Software-defined networks-enabled fog computing for IoT-based healthcare: Security, challenges and opportunities," Security and Privacy, vol. 5, no. 6, Art. no. e291, 2022, doi: 10.1002/spy2.291.
[16] Y. Zhang, N. Wu, F. Zhou, X. Zhang, and J. Zhang, "High performance AES-GCM implementation based on efficient AES and FR-KOA multiplier," IEICE Electronics Express, vol. 15, Art. no. 20180559, 2018, doi: 10.1587/elex.15.20180559.
[17] K. Kim, S. Choi, H. Kwon, H. Kim, Z. Liu, and H. Seo, "PAGE: Practical AES-GCM encryption for low-end microcontrollers," Applied Sciences, vol. 10, no. 9, Art. no. 3131, 2020, doi: 10.3390/app10093131.
[18] Y. Naito, Y. Sasaki, and T. Sugawara, "AES-LBBB: AES mode for lightweight and BBB-secure authenticated encryption," IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2021, no. 3, pp. 298-333, 2021, doi: 10.46586/tches.v2021.i3.298-333.
[19] C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schlaffer, "Ascon v1.2: Lightweight authenticated encryption and hashing," Journal of Cryptology, vol. 34, Art. no. 33, 2021, doi: 10.1007/s00145-021-09398-9.
[20] G. Cagua, V. Gauthier-Umana, and C. Lozano-Garzon, "Implementation and performance of lightweight authentication encryption ASCON on IoT devices," IEEE Access, 2025, doi: 10.1109/ACCESS.2025.3529757.
[21] S. Ullah and R. Zahilah, "Curve25519 based lightweight end-to-end encryption in resource constrained autonomous 8-bit IoT devices," Cybersecurity, vol. 4, Art. no. 11, 2021, doi: 10.1186/s42400-021-00078-6.
[22] I. Sokol, P. Hubinsky, and L. Chovanec, "Lightweight cryptography for the encryption of data communication of IoT devices," Electronics, vol. 10, no. 21, Art. no. 2567, 2021, doi: 10.3390/electronics10212567.
[23] G. Kuldeep and Q. Zhang, "Design prototype and security analysis of a lightweight joint compression and encryption scheme for resource-constrained IoT devices," IEEE Internet of Things Journal, vol. 9, no. 8, pp. 5800-5812, 2022, doi: 10.1109/JIOT.2021.3098859.
[24] D. Xu et al., "Ring-ExpLWE: A high-performance and lightweight post-quantum encryption scheme for resource-constrained IoT devices," IEEE Internet of Things Journal, vol. 10, no. 4, pp. 3551-3564, 2022, doi: 10.1109/JIOT.2022.3189210.
[25] S. Pandey and B. Bhushan, "Recent lightweight cryptography based security advances for resource-constrained IoT networks," Wireless Networks, vol. 30, pp. 3147-3171, 2024, doi: 10.1007/s11276-024-03714-4.
[26] O. Sabri, B. Al-Shargabi, A. Abuarqoub, and T. A. Hakami, "A lightweight encryption method for IoT-based healthcare applications: A review and future prospects," IoT, vol. 6, no. 2, Art. no. 23, 2025, doi: 10.3390/iot6020023.
[27] C. Silva, N. Tenorio, and J. Bernardino, "Lightweight encryption algorithms for IoT," Computers, vol. 14, no. 12, Art. no. 505, 2025, doi: 10.3390/computers14120505.
[28] A. Aljaedi, A. R. Alharbi, A. Aljuhni, M. K. Alghuson, S. Alassmi, and A. Shafique, "A lightweight encryption algorithm for resource-constrained IoT devices using quantum and chaotic techniques with metaheuristic optimization," Scientific Reports, vol. 15, 2025, doi: 10.1038/s41598-025-97822-6.
[29] N. Sivanesan, N. Parthiban, S. Vijay, and S. N. Sheela, "Comparison of mitigating DDoS attacks in software defined networking and IoT platforms," Cyber Security and Applications, vol. 3, Art. no. 100080, 2025, doi: 10.1016/j.csa.2024.100080.
[30] S. Dhanda, B. Singh, and P. Jindal, "AES 32: An FPGA implementation of lightweight-AES for IoT devices," International Journal of Computing and Digital Systems, vol. 15, no. 1, pp. 925-934, 2024, doi: 10.12785/ijcds/160167.
[31] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: Rapid prototyping for software-defined networks," in Proc. 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, pp. 1-6, doi: 10.1145/1868447.1868466.
[32] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," Proceedings of the IEEE, vol. 103, no. 1, pp. 14-76, 2015, doi: 10.1109/JPROC.2014.2371999.

[33] OASIS, MQTT Version 5.0, OASIS Standard, 2019. [Online]. Available: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html
[34] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF protocols," RFC 8439, Jun. 2018, doi: 10.17487/RFC8439.
[35] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST SP 800-38D, Nov. 2007, doi: 10.6028/NIST.SP.800-38D.