Experimental Stress Test of Encryption Algorithms on Forex Payment Gateways: Analyzing Performance during Rupiah Exchange Rate Volatility

Audy Alicia Renatha Tirayoh - 18223097
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: aliciatryh@gmail.com , 18223097@std.stei.itb.ac.id

***Abstract*—This research presents an experimental stress test of symmetric encryption layers within foreign exchange (forex) payment gateways, contextualized against the recent macroeconomic shock where the Indonesian Rupiah (IDR) breached the psychological barrier of IDR 18,000 per US Dollar. This extreme volatility triggers exponential spikes in transaction traffic due to panic-hedging, placing cryptographic sub-systems under severe stress. A controlled benchmark was implemented to compare five cryptographic configurations: AES-128-ECB, AES-256-CBC, AES-256-CTR, AES-256-GCM, and ChaCha20-Poly1305, using synthetic payloads modeled after BI-FAST and SWIFT across loads from 100 to 10,000 transactions. Authenticated Encryption with Associated Data (AEAD) modes, specifically AES-256-GCM, achieved the highest throughput, peaking at approximately 385,000 transactions per second, while ChaCha20-Poly1305 exhibited robust stability without hardware acceleration. Conversely, legacy CBC and CTR modes suffered substantial performance degradation, trailing AEAD variants by up to 2.5 times under extreme loads. Furthermore, a block-manipulation exploit on AES-128-ECB is demonstrated to show how transaction data can be silently corrupted without ever knowing the encryption key. These results offer actionable insights for engineering resilient financial architectures capable of preserving data integrity during ongoing macroeconomic crises.**

Keywords—symmetric encryption; forex payment gateways; cryptographic benchmarking; AES-256-GCM; ChaCha20-Poly1305; stress testing; Rupiah exchange rate volatility

I. Introduction

On June 4, 2026, the Indonesian Rupiah broke the IDR 18,000 per US Dollar mark for the first time in history. On June 8, 2026, it touched a low of Rp 18,190.30, making it the weakest the Rupiah has ever been [1]. This wasn't a sudden shock either, it was expected. Since the beginning of 2026, the Rupiah has been changing and already lost around 7.44% of its value against the US Dollar, and the pressure had been building up for months [2].

Destry Damayanti, Bank Indonesia's Senior Deputy Governor, confirmed the main causes in an official statement on June 4, 2026. The main causes were the escalating geopolitical tensions in the Middle East, specifically the US-Israel-Iran conflict. This phenomenon pushed oil prices higher and triggered global risk-off sentiment, causing capital to flow out of emerging markets into safer assets like the US Dollar [2]. On the domestic side, Indonesia's high demand for dollars to pay for energy imports (around 1.5 million barrels per day, 85% of which is subsidized) made things worse. The country's trade surplus had also shrunk drastically, from USD 3.32 billion in March to just USD 0.09 billion in April 2026 [1].

This kind of significant currency depreciation usually sets off a spike in forex activity. Individuals are in a panic to convert their savings. Corporations hedge their bets. Transaction volumes increase for banks and remittance services. Before passing via any payment gateway, whether a bank's SWIFT connection, a fintech application utilizing BI-FAST, or a remittance platform, all of these transactions must be encrypted. In practice, here is where encryption performance truly counts.

The course material regarding block ciphers gives a concrete example of what is at stake: if a forex transaction is encrypted with the wrong mode, an attacker can flip specific ciphertext blocks and modify the transaction amount, without ever knowing the key [3], [4]. That is the kind of vulnerability that becomes especially dangerous when transaction volumes are at peak and monitoring might be overburdened.

Therefore the question this paper is trying to answer is: Can a forex payment gateway's encryption layer handle an increase in transaction volume during a currency crisis? Does performance under that kind of stress depend on the encryption algorithm selected? The ECB block-manipulation attack is demonstrated using a real forex transaction payload, and five encryption schemes are tested across realistic load levels and four exchange-rate scenarios (including one modeled on the actual June 2026 crisis) to show why algorithm choice is not merely a performance consideration.

II. Backgrounds and Related Works

A. Block Ciphers and Modes of Operation

The most widely used symmetric cipher today is AES (Advanced Encryption Standard). It operates on 128-bit blocks

with key sizes of 128, 192, or 256 bits [5]. Given that most real messages are longer than one block, a mode of operation is needed to handle multiple blocks. ECB (Electronic Code Book) encrypts each block independently, which is fast but dangerous. Because identical plaintext blocks result in identical ciphertext blocks, allowing the message's structure to be revealed and allowing blocks to be switched or replayed without detection. CBC (Cipher Block Chaining) chains each block to the previous ciphertext to avoid this, but it still requires padding and does not verify if the data has been tampered with. CTR (Counter) mode addresses the padding issue by treating AES as a stream cipher. It encrypts an incrementing counter value and XOR-s the result with each plaintext block, so no padding is needed and blocks can be processed in parallel. However, like CBC, it still provides no integrity guarantee [6]

B. Authenticated Encryption (AEAD)

It's not enough to simply keep financial data confidential, you also need to know if it has been altered. Both are handled simultaneously by AEAD (Authenticated Encryption with Associated Data). Using a Galois field hash, GCM (Galois/Counter Mode) adds an authentication tag to CTR-mode AES [7]. ChaCha20-Poly1305, which is standardized in TLS 1.3 [8], [9], combines Poly1305 authentication with the ChaCha20 stream cipher. Both provide the same security guarantees, the CPU's hardware support for AES (AES-NI) determines which is actually quicker. GCM is typically faster with AES-NI hardware support, while ChaCha20-Poly1305 tends to outperform GCM on devices without AES-NI [10]].

C. Why This Matters for Payment Gateways

A forex transaction typically has a predictable structure: transaction ID, timestamp, currency pair, amount, exchange rate, account numbers, routing code. If an attacker knows where the amount of field sits, they can overwrite the corresponding ciphertext block with a different one. Which will corrupt the amount of receiver decrypts, without a way for them to detect it. This occurs because ECB encrypts each block independently. [3], [4]. AEAD modes prevent this from happening because the authentication tag will fail to verify if any byte has been changed. This is why using the correct mode isn't just about security theory, but it has a direct impact on financial integrity.

D. The June 2026 Rupiah Crisis as a Gateway Stress Scenario

The historical breach of Rp 18,000 on June 4, 2026 is the real-world context for this paper. In this kind of crisis, forex transaction volumes tend to spike as individuals panic-buy dollars, corporations hedge, and banks intervene. BI even had to restrict each foreign currency purchase to USD 25,000 per person per month starting June 2, 2026 to manage the pressure [1]. In order to replicate the type of surge a gateway would actually see, this research models that transaction spike. Each volatility scenario uses a different exchange-rate distribution, and higher-volatility scenarios are paired with higher transaction volumes.

E. Related Work

Prior benchmarking studies have examined AES performance primarily in the context of disk encryption or TLS, rather than financial transaction workloads. Langley (2015) showed that ChaCha20-Poly1305 consistently outperforms AES-GCM on hardware without AES-NI acceleration, and proposed it as the preferred cipher for mobile TLS connections [10]. The NIST publications defining GCM [7] and the CBC and CTR modes [6] establish formal security properties but do not address throughput under high-volume transactional loads. This study addresses that gap by applying these five configurations to a domain-specific workload, forex payment transactions. And measuring their behavior across realistic load levels and volatility scenarios motivated by an actual macroeconomic event.

III. METHODOLOGY

A. System Overview

Fig. 1 represents the overall system architecture. It is made up of four main components: a transaction simulator that generates payloads (`simulator.py`), a cipher wrapper that presents all five algorithms through the same interface (`algorithms.py`) a benchmark engine that measures performance (`benchmark.py`), and separate outputs for the benchmark results (`benchmark_results.csv`, Fig. 2 - 7) and the ECB attack demonstration (`ecb_attack_result.json`).

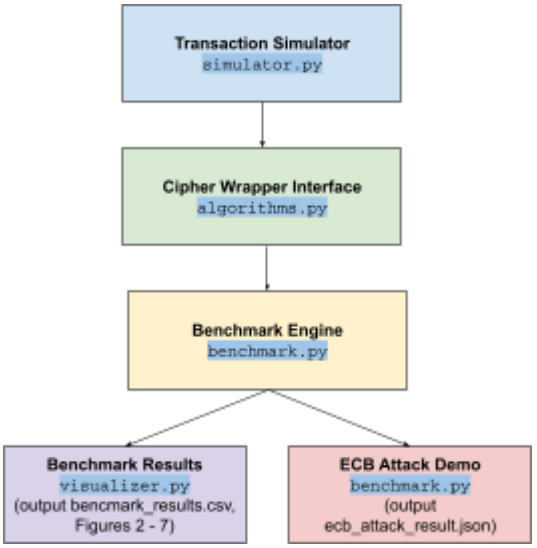

Fig. 1. System architecture.

B. Transaction Payload Simulation

A synthetic transaction generator was built, it produces realistic-looking JSON records instead of utilizing actual financial data. Each data contains UUID, timestamp, transaction type (buy/sell/remittance/wire/swap), currency pair, IDR and foreign amounts, exchange rate, source/destination, bank names and account numbers, a SWIFT-style routing code, a session token, and a checksum field. This is essentially how a real BI-FAST or SWIFT message structure looks. Before encryption, the average size

of each record serialized to compact JSON is roughly 558 bytes. BI-FAST messages follow a structured ISO 20022 format, while SWIFT uses MT103 for cross-border payments. Both share a predictable field layout, which is precisely what makes block-level attacks like the ECB manipulation feasible when an attacker has prior knowledge of the schema. The synthetic payload was designed to mirror this predictability.

C. Volatility Scenarios

There are four exchange-rate scenarios. The NORMAL and MODERATE rates come from Rupiah's range in early 2026. HIGH is based on the all-time low hit in June 2026. EXTREME represents a hypothetical further deterioration of Indonesian Rupiah.

TABLE I. EXCHANGE-RATE SCENARIOS

Scenario	Base rate (IDR/USD)	Std. dev.
Normal	16,500	50
Moderate	17,133	200
High (ATL, 4 June 2026)	17,713	500
Extreme (stress test)	18,200	1,200

D. Load Levels

Five batch sizes were tested: Low (100 tx), Medium (500 tx), High (2,000 tx), Peak (5,000 tx), and Extreme (10,000 tx). These cover the range from a quiet period to a surge comparable to what a gateway might see during a panic-buying period.

E. Cipher Implementation

All five algorithms were implemented in Python using cryptography library, which wraps Open SSL under the hood. Each cipher gets the same encrypt() / decrypt() interface. The benchmark engine just calls these methods and times them.

TABLE II. FIVE ALGORITHMS

Algorithm	Type	Key size	AEAD	Padding
AES-128-ECB	Block	128-bit	No	Yes
AES-256-CBC	Block	256-bit	No	Yes
AES-256-CTR	Stream	256-bit	No	No
AES-256-GCM	AEAD	256-bit	Yes	No
ChaCha20-Poly1305	AEAD	256-bit	Yes	No

F. Benchmark Procedure

For each of the 100 configurations, containing 4 scenarios x 5 loads x 5 ciphers. The transaction batch was pre-generated once and reused across all timing runs. Before timing, warmup passes were run to let the CPU cache settle. Then 5 full encrypt-then-decrypt passes were ran over the batch and averaged the results. Python's perf_counter() function, which provides sub-microsecond resolution, was used for timing. Latency is measured in average milliseconds per transaction, while throughput is measured in transactions per second.

G. ECB Attack Demonstration

To show the vulnerability concretely, a single forex record was encrypted. The record is a USD 1,000 sale at rate 17,713 (IDR 17,713,000), under AES-128-ECB using a fixed key. And then the ciphertext was split into 16-byte blocks and overwrote block 5 of 10 without touching the key. The modified ciphertext was then decrypted. Expecting the receiver to get corrupted data with no error or warning from the decryption routine.

IV. RESULTS AND ANALYSIS

A. Throughput: AEAD Wins by a Large Margin

Fig 2. shows throughput across all five load levels for the NORMAL scenario. AES-256-GCM and ChaCha20-Poly1305 are at a different level from the other three. GCM peaks at around 385,000 transactions per second, and ChaCha20 at around 326,000. The range of the non-AEAD ciphers (ECB, CBC, and CTR) is around 138,000 to 209,000 tx/s. GCM is almost 2.45 times quicker than CBC on average across all tests, whereas ChaCha20 is roughly 2.13 times faster. This is primarily due to the fact that the test environment supports hardware AES acceleration (AES-NI), which advantages GCM.

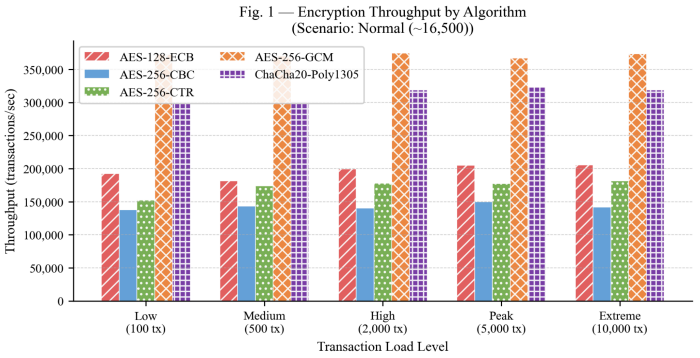

Fig. 2. Encryption throughput by algorithm, NORMAL scenario. Shown AEAD ciphers are clearly faster.

B. Latency: Mostly Flat, But CBC Creeps Up

For the HIGH scenario, Fig. 3 shows the per-transaction delay as the batch size increases from 100 to 10,000 transactions (log scale). Most algorithms stay flat, latency doesn't go up as more transactions pile in. . AES-256-CBC is the lone exception, exhibiting a little increasing trend (from roughly 0.0064 ms at 100 tx to 0.0068 ms at 10,000 tx). It's not really noticeable, but it's the only algorithm that consistently increases, which makes sense for a mode that isn't able to parallelize decryption.

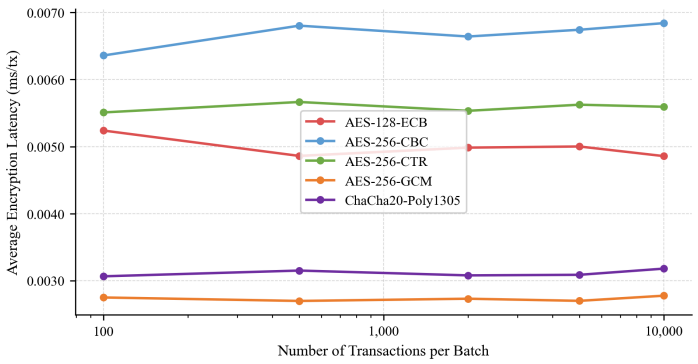

Fig. 3. Per-transaction latency vs load, HIGH Scenario.

C. Round-Trip Across Scenarios: Volatility Does Not Matter

Fig. 4 is probably the most important result for the main question of this paper. It shows round-trip latency, which includes encrypt and decrypt at the Extreme load level. The two AEAD ciphers are shown to be in the green band (0.0045-0.0055 ms), and the legacy ciphers are shown to be in the orange-red band (0.0097-0.0132 ms). The key takeaway is within each row, the numbers barely change across all four scenarios (Normal, Moderate, High, Extreme). Currency volatility does not affect encryption performance.

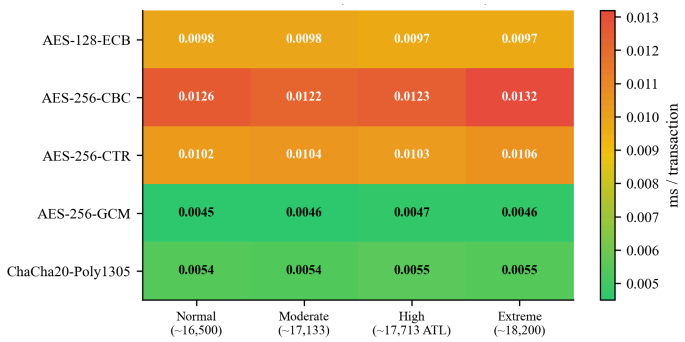

Fig. 4. Round-trip latency heatmap at Extreme load (10,000 tx). Rows are stable across scenarios.

D. Throughput Degradation: Within Noise

Throughput between the NORMAL and EXTREME scenarios are compared in Table III. GCM has changed the most at -3.55%, all other changes are minor. To put things in perspective, these variations are less than what would result from simply performing the identical test again on a computer with typical background processes. There is virtually little impact from the volatility situation. The bars for each method seem the same in all four scenarios at the Peak load level, as Fig. 5 illustrates.

TABLE III. THROUGHPUT CHANGE FROM NORMAL TO EXTREME SCENARIO AT EXTREME LOAD

Algorithm	Normal (tx/s)	Extreme (tx/s)	Change
AES-128-ECB	205,471	204,547	-0.45%
AES-256-CBC	142,001	146,825	+3.40%
AES-256-CTR	181,393	176,501	-2.70%
AES-256-GCM	373,321	360,070	-3.55%
ChaCha20-Poly1305	319,075	313,614	-1.71%

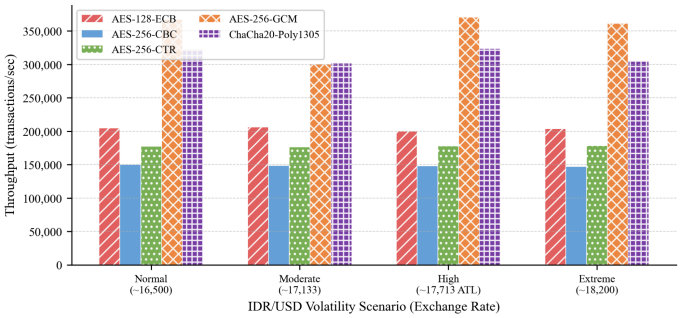

Fig. 5. Throughput across volatility scenarios, Peak load (5,000 tx). Bars per algorithm are nearly identical.

E. Overhead: Small for Everyone

Fig. 6 shows ciphertext size overhead vs plaintext, averaged across all runs. The highest overhead is 5.0% for both GCM and ChaCha20, which includes the nonce and the 16-byte authentication tag. CTR adds 2.9% for the nonce, ECB adds only 1.4% (padding only), and CBC adds 4.3% from its IV and PKCS7 padding. All of these are insignificant in terms of bandwidth for payloads of about 558 bytes.

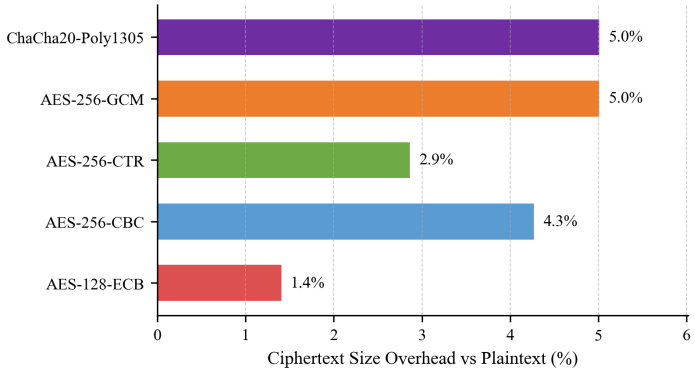

Fig. 6. Ciphertext overhead relative to plaintext size.

F. Encrypt vs Decrypt: Decryption Is Always Faster

For the HIGH scenario at peak demand, Fig. 7 divides the per-transaction time into encryption and decryption. Decryption is faster than encryption for all algorithms. The difference varies from 3% (hardly perceptible) for ECB to 31% for GCM (encryption 0.0027 ms, decryption 0.0019 ms). As anticipated, the AEAD ciphers perform more work during encryption (computing the authentication tag), whereas decryption only verifies it.

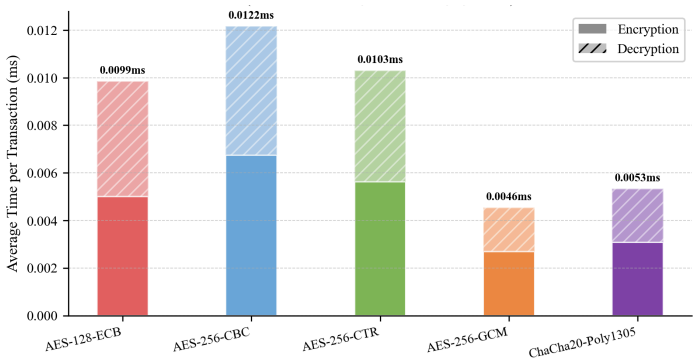

Fig. 7. Encryption vs decryption time per transaction, HIGH scenario. Peak load.

G. The ECB Attack: It Works

The ECB attack worked exactly as expected. A FOREX_SELL transaction of USD 1,000 at rate 17,713 and IDR 17,713,000 was displayed in the original decrypted record. The remainder of the record (tx_id, type, rate, and account numbers) remained intact when block 5 of the ciphertext was intentionally overwritten, but the receiver's decoding resulted in an entirely jumbled amount_idr field that was filled with garbage bytes. Notably, the amount_usd field (USD 1,000) survived intact while the corresponding IDR value (Rp 17,713,000) was completely destroyed. This asymmetric corruption is particularly dangerous in a real gateway where the USD and IDR sides of a transaction are validated separately. The USD leg could pass validation while the IDR leg silently carries corrupted data, potentially going undetected until reconciliation. What matters is that there was no error returned by the decryption procedure. The damaged data was unknown to the recipient. This supports what the course material says about ECB, if an attacker can intercept and alter ciphertext, there is nothing prohibiting them from doing this.

H. What This Means in Practice

There are three things that stand out. First, the AEAD ciphers, containing GCM and ChaCha20, are both more secure and faster than the legacy modes. There is no trade-off to make here. Second, the exchange-rate volatility scenario has essentially no effect on encryption performance.

Therefore if a payment gateway is struggling during a Rupiah crisis, then the network I/O, database load, or application-level logic are more likely to be the cause of a payment gateway's difficulties. For GCM deployments specifically, since a fresh random 96-bit nonce is generated for every transaction, a very high-volume gateway running under a single long-lived key would eventually face a small risk of nonce collision. This does not affect the benchmark results, but it is something a production deployment needs to handle through periodic key rotation or a counter-based nonce scheme rather than relying purely on randomness. Third, the ECB demonstration shows that algorithm selection has actual financial ramifications that go beyond performance: using ECB on forex transaction data poses a genuine security risk, not just a theoretical one.

I. Limitations

This study has a few limitations worth being upfront about. First, all benchmarks were run single-threaded on the author's personal machine. Observations show that running the same benchmark on different days produced different absolute throughput numbers (GCM ranged from roughly 360,000 to 670,000 tx/s across two runs), most likely due to varying background CPU load. The relative rankings stayed the same both times, but anyone attempting to duplicate precise numbers should be aware of this. A proper benchmark setup would pin CPU frequency and run on an isolated server.

Second, only the cryptographic layer is measured in this study. In addition, a real payment gateway must handle network input/output, TLS handshake overhead, database writing, fraud detection, and regulatory logging, all of which are likely to be the true bottleneck during a currency crisis. Our findings indicate that encryption won't be the bottleneck, but they don't specify what.

Thirdly, artificial transaction payloads were used. Although they were designed to resemble actual BI-FAST and SWIFT records, real transactions may differ in size or structure, potentially affecting performance in ways not captured by this study.

V. Conclusion

A stress test of five symmetric encryption schemes was performed across 100 configurations covering four volatility scenarios (including one modeled on the actual Rupiah crisis of June 4, 2026) and five load levels ranging from 100 to 10,000 transactions per batch. AES-256-GCM and ChaCha20-Poly1305 are both faster and more secure than the legacy AES modes. GCM averages 2.45x the throughput of CBC, and ChaCha20 averages 2.13x. Throughput degradation across volatility scenarios never exceeded 3.55%, meaning the encryption layer is not where a gateway bottlenecks during a currency crisis. Additionally, the ECB block-manipulation attack was successful without any errors, demonstrating that transaction data including monetary values should never employ AES-128-ECB. This author’s recommendation for Rupiah-denominated forex gateways: use AES-256-GCM (with AES-NI hardware) or ChaCha20-Poly1305 (where hardware AES isn't available), and avoid ECB entirely. To fully understand where the actual bottlenecks are, future work should expand this test to a full gateway scenario that includes network I/O, HSM overhead, and database demand.

Given that Bank Indonesia's BI-FAST system now processes millions of domestic transactions daily, and that the June 2026 Rupiah crisis demonstrated how quickly transaction volumes can surge in response to macroeconomic shocks, these findings have direct relevance for the design of Indonesia's financial infrastructure. Migrating from legacy non-AEAD modes to AES-256-GCM or ChaCha20-Poly1305 at the gateway layer is a low-cost, high-impact change that simultaneously improves both security and throughput, without requiring hardware upgrades.

Source Code at Github

https://github.com/renathalicia/II4021-Kriptografi-Forex-Payment-Gateway-Encryption-Stress-Test

Video Link at Youtube

https://youtu.be/mpINWo2F-4o

Acknowledgment

The author would like to express sincere gratitude to Dr. Ir. Rinaldi Munir, M.T. for his guidance, course materials, and insights throughout the II4021 Cryptography course, which provided the theoretical foundation for this study.

The author also extends appreciation to the Bidang Kajian, Badan Kesenatoran HMIF ITB 2026/2027, whose review of the Rupiah depreciation and the broader economic challenges facing Indonesia inspired the real-world context of this research. Their work served as a reminder that technology and engineering should not exist in isolation, but should ultimately contribute to understanding and addressing issues that affect society. Finally, the author acknowledges Indonesians who continue striving through rising costs and economic uncertainty. Their perseverance serves as a reminder that technological progress is most meaningful when it improves the lives of the people it is intended to serve.

References

[1] I. R. S. Rahayu, "Rupiah Tembus Rp 18.000 Per Dollar AS, Terlemah Sepanjang Sejarah," *KOMPAS.com*, Jun. 04, 2026. [Online]. Available: https://money.kompas.com/read/2026/06/04/085746426/rupiah-tembus-rp-18000-per-dollar-as-terlemah-sepanjang-sejarah [Accessed: Jun. 10, 2026].

[2] G. Amanda, "BI Ungkap Penyebab Rupiah Tembus Rp 18.000 per Dolar AS," *Republika Online*, Jun. 04, 2026. [Online]. Available: https://ekonomi.republika.co.id/berita/tg3gdp423/bi-ungkap-penyebab-rupiah-tembus-rp-18000-per-dolar-as [Accessed: Jun. 10, 2026].

[3] R. Munir, "Block Cipher (Bagian 1)," Bahan Kuliah II4021 Kriptografi dan Koding, Institut Teknologi Bandung, Bandung, Indonesia, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2025-2026/14-Block-Cipher-Bagian1-STI-2026.pdf. [Accessed: Jun. 19, 2026].

[4] R. Munir, "Block Cipher (Bagian 2)," Bahan Kuliah II4021 Kriptografi dan Koding, Institut Teknologi Bandung, Bandung, Indonesia, 2026. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2025-2026/15-Block-Cipher-Bagian2-STI-2026.pdf. [Accessed: Jun. 19, 2026].

[5] NIST, "Federal Information Processing Standards Publication 197: Announcing the Advanced Encryption Standard (AES)," NIST, Gaithersburg, MD, Nov. 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf.

[6] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: Methods and Techniques," NIST SP 800-38A, Dec. 2001. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a.pdf.

[7] M. J. Dworkin, "Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC," NIST SP 800-38D, Nov. 2007. [Online]. Available: https://doi.org/10.6028/NIST.SP.800-38D.

[8] Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," IETF, RFC 8439, Jun. 2018. [Online]. Available: https://datatracker.ietf.org/doc/rfc8439/.

[9] D. Bernstein, "ChaCha, a variant of Salsa20," Jan. 2008. [Online]. Available: https://cr.yp.to/chacha/chacha-20080120.pdf.

[10] N. Sullivan, "Do the ChaCha: Better Mobile Performance with Cryptography," *The Cloudflare Blog*, Feb. 23, 2015. [Online]. Available: https://blog.cloudflare.com/do-the-chacha-better-mobile-performance-with-cryptography/.

[11] Bidang Kajian, Badan Kesenatoran HMIF ITB 2026/2027, "Kajian Mengenai Melemahnya Nilai Tukar Rupiah dan Ekonomi Indonesia Hari Ini," May 2026. [Online]. Available: https://docs.google.com/document/d/1I4F_3FTQBL5ytsLASY1M6FCKnHCF8qLEIm38y1tuJp0/edit. [Accessed: Jun. 10, 2026].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 19 Juni 2026

Audy Alicia Renatha Tirayoh (18223097)