On The Impact of Cryptographic Block Sizes on Energy Constrained Networking Embedded Systems

Rodrigo Lopes Ferreira
Master of Telecommunications and Informatics Engineering
ISCTE
Lisbon, Portugal
https://github.com/rodfer0x80/meti-srsi

***Abstract*—Evaluating the impact of energy consumption for hardware accelerated cryptographic algorithms on a common device used on energy constrained networked systems, the ESP32.Using 51B, 242B, 1460B and 4096B total block size minus internal and cryptographic headers. Utilising cryptographic algorithms such as AES128-CTR+DH+HMAC (AES128 in CTR mode with DH key exchange and HMAC integrity checking) and AES256-GCM+RSA (AES256 in GCM mode with GCM integrity checking and RSA for key exchange) following the PRISEC III framework recommendations on efficient and secure cryptographic algorithms for IoT devices, along with a cleartext version as control group. Two methods are used for testing, unidirectional communication from client to server and bidirectional. Charge in mAh and data transferred in bytes is measured locally and leveraged to calculate Energy Cost per Byte in µJ/B for each block size, method and algorithm used. Charge measured using a power meter between the device and the power source and data transferred is calculated on the server. The proposed method should produce a quick and rough, low-cost reproducible set of steps to evaluate the trade-offs between energy consumption and security in IoT devices.**

***Index Terms*—cryptography, networking, IoT security, ESP32, energy consumption**

I. Introduction

A. Cryptography

In the modernisation of infrastructure using small and energy efficient devices in the world of IoT, security and privacy is a human right that cannot be forgotten and is ever more challenged. Hardware acceleration is built into many embedded devices to improve their efficiency and should be leveraged whenever available. Also frameworks as the PRISEC III [1] exist to guide engineers and developers of these services and application to ensure their users' data and devices, data and services remain confidential, integral and available from bad actors. The best recommendations from the PRISEC III framework are AES128-CTR and AES256-GCM+RSA for a good security vs. energy consumption compromise.

To establish a secure communication channel, the system employs a Hybrid Cryptosystem that combines the efficiency of symmetric encryption with the security of asymmetric key exchange. This process begins with a handshake phase utilizing the Diffie-Hellman key exchange protocol for the AES-128-CTR implementation. In practice, this necessitates that both parties first agree on a common generator g and a large prime modulus p. Each party independently generates a private integer (a and b respectively). Subsequently, they compute their public keys, $A = g^a \pmod p$ and $B = g^b \pmod p$, and exchange them over the insecure channel. Upon receipt, each party computes the shared secret S using the peer's public key. Due to the properties of modular exponentiation, both parties arrive at the identical symmetric key:

$$S \equiv (g^b)^a \equiv (g^a)^b \equiv g^{ab} \pmod p$$

This resulting shared secret is then utilized to seed the AES-128-CTR algorithm. It is then passed through a Key Derivation Function using SHA-256, from the raw Diffie-Hellman output a fixed-length 256 bit string is generated, which is then truncated to form the 128-bit session key required for AES128-CTR. The initial 256 bits are expanded to 768 bits by summing the first hash of the session key with the hash of the first hash and then extracted the 128 bit AES128 key, the 256 bit HMAC key, 64 bit TX and 64 bit RX nonces to check the integrity and perform the rotation of the data according to the CTR mode respectively

The AES256-GCM implementation relies on RSA Key Encapsulation. The server generates a private/public RSA key pair for the session and shares the public key with the client in raw over the wild. The client generates the pre-master key and encrypts it with the server's RSA public key and sends it to the server. Only the server can decrypt this session key with their private key. It is then passed through a HKDF-SHA256 to expand the key material where the 256 bit AES encryption key, 96 bit Initial IV, the 64 bit TX and 64bit RX nonces are derived. The Authenticated Encryption with Associated Data scheme is inherent in AES256-GCM, so integrity is verified using the internal GHASH function without requiring the external HMAC-SHA256 used in the AES128-CTR implementation.

From there the 256 bit session key is established to be used for AES256-GCM since the client generates a high entropy 256-bit random bitstring and transmits it securely via RSA, this value serves directly as the AES encryption key. The GCM mode implementation uses a 12 bit IV before the data and a 16 bit tag after the data so there is no need to run HKDF with HMAC-SHA256 as done for AES128-CTR.

The AES256-GCM+RSA implementation is vulnerable to Active Man-in-the-Middle attacks unless the server holds an Identity Private Key that never changes and the public key is hardcoded in the client, then the private key is used to sign the RSA public key such that the client can verify it's confidentiality and integrity. Otherwise it is wiser to use the Elliptic Curve Diffie-Hellman Key Exchange algorithm.

B. Networking

Common packet sizes for IoT energy constrained devices are 51 bytes to 242 bytes for those following the LoRaWAN protocol. In LoRaWAN architecture, the maximum payload size is dynamic and inversely proportional to the Spreading Factor used. To maintain an acceptable Time-on-Air and comply with regional duty cycle regulations (e.g. EU868 ISM band), the protocol enforces smaller payloads such as 51 bytes when using higher spreading factors for long-range communication. Conversely, lower spreading factors, which offer higher data rates but shorter range, allow for larger payloads up to approximately 242 bytes. This Adaptive Data Rate mechanism is used to minimize energy consumption, thus these sizes were chosen for evaluation.

In contrast to the constrained LoRaWAN payloads, broadband IP networks use the standard Ethernet II Maximum Transmission Unit of 1500 bytes. To prevent IP fragmentation, the Maximum Segment Size is set to 1460 bytes. This value is derived by subtracting the standard overheads from the MTU: 20 bytes for the IPv4 header and 20 bytes for the TCP header. This configuration ensures that the maximum amount of application data is transmitted while strictly adhering to physical layer constraints, guaranteeing that each packet fits within a single Ethernet frame.

Additionally, a block size of 4096 bytes was selected as it is a common standard for bulk data transfer operations to and from the embedded system. This size matches the standard virtual memory page size used by most modern operating systems and processor architectures. Aligning data chunks to this boundary minimizes processing overhead by ensuring that buffer operations coincide with the system's native Memory Management Unit.

C. Hardware

For the evaluation was used an ESP32-D0WD-V3 (revision v3.1) with WiFi (802.11 b/g/n), Dual-Core 32-bit Xtensa LX6 microprocessor @240MHz

allowing to perform cryptographic operations in on core and networking operations on another.

Average current draw range from 120–260 mA during transmission, 95–100 mA during reception and 80 mA when idle with WiFi mode on.

It also has hardware acceleration for cryptographic algorithms such as AES128 and AES256 with support for CTR and GCM modes, as well as SHA256 and RSA with support for DH and HMAC. Due to hardware acceleration these operations are executed significantly faster and with lower power consumption compared to a pure software implementation, directly influencing the power consumption results.

D. Energy

Reading is done using a USB-USB meter in between the device and the power source in repeated transmission runs to try and average incorrect accuracy due to the measurement device's imprecision. The device gives us an estimation of charge drawn in mAh which we use to calculate the Energy Cost in µJ/B and Throughput in KB in each scenario. This is such that we have two metrics to compare the results, the amount of data it can output per second and the cost of each byte.

The experimental constants are defined as follows:

$$V = 3.3\,\mathrm{V} \qquad \text{(Voltage constant)}$$
$$t = 60\,\mathrm{s} \qquad \text{(Time constant)}$$
$$N_r = 3 \qquad \text{(Number of Runs)}$$

The average current I, in Amperes, is calculated by converting the measured charge Q from mAh to Coulombs and dividing by the duration of the transmission t, in seconds. This provides the instantaneous rate of charge flow:

$$I = \frac{Q}{t} \times \frac{3600}{1000}$$

Power consumption P, in Watts, is defined as the product of the constant voltage and the calculated current:

$$P = V \times I$$

The total Energy E, in Joules, consumed by the system is derived from the power consumption over the duration of the test:

$$E = P \times t$$

Throughput T, in KB/s, is the quotient of the average data transmitted D, in bytes, and the time t, normalized to KB:

$$T = \frac{D}{t \times 1024}$$

Energy Cost E_c represents the efficiency of the transmission. It is calculated as the energy consumed per unit of data, in microjoules per byte (μJ/B):

$$E_c = \frac{E}{D}$$

II. Related Work

Recent research in building frameworks for cryptographic schemes in IoT, balancing energy consumption and security such as PRISEC III [1] highlight the usage of AES128-CTR and AES256GMC+RSA. Such algorithms have native hardware acceleration support. The framework aforementioned proposes implementing security at 4 distinct levels (Guest, Basic, Advanced, Admin) using a multi-layered approach.

In [8] and [9], the authors proposed a method for estimating the energy consumption of cryptographic algorithms. The method involved counting the number of executed instructions and multiplying them by their theoretical energy cost. Although this technique is very interesting and might need more exploration, it is more narrowly applicable to the cryptographic algorithms themselves, and loses consistency when measured programs that perform transmission of packets and thus it is not a well applicable for this experiment.

In [5] a multi-layer approach was used for the ESP32 that explores rotating the cryptography algorithm depending on battery availability in order to reduce energy consumption using AES128 and RC4. Although the concept is interesting, using RC4 is not recommended by the PRISEC III framework due to poor security. The limitation of this

approach is further exacerbated by the fact that these devices have AES128 hardware accelerators.

Given that joulescope measurements fall outside the practical scope of this project, a hybrid estimation approach is adopted. This combines theoretical energy estimation derived from [4] and [7], with empirical readings obtained from a low-cost USB power meter placed between the device and its power source, averaged across multiple measurement runs.

While these approaches have been validated in prior work, the influence of data block size on energy consumption and performance has not been systematically explored. This work addresses this by investigating how varying block sizes energy consumption in different real-world scenarios and reinforces the validity of the PRISEC III framework.

—- DONE TILL HERE —-

III. Methodology

A. Server

The server logic is distributed in three files.

The protocol defines the algorithms to perform encryption and decryption of data, sending and receiving raw frames with the length header size, DH and RSA handshake and key derivation.

The server handles the logging and networking side abstracting from the specifics of the implementation of data exchange and cryptographic algorithms.

The frontend interface that runs and logs for each method and cryptography algorithm performing the data exchange loops and summing the data from each packet sent or received excluding the length header and cryptographic headers such as HMAC, IV and TAG. This is used to calculate the Throughput and Energy Cost.

B. Client

The client is implemented in C code using the ESP-IDF library for the bootloader, flashing, multithreading, timing functions, data types, logging, networking and native hardware accelerated cryptography (mbedtls).

The code for the client consists of a two functions to connect to a WiFi network, two functions to send and receive raw frames of data including a length header at the beginning, a function to perform establish the DH or RSA handshake, and a function to run the main logic of the program according to their method (unidirectional, bidirectional) and cryptographic needs, along with some more global variables for WiFi, cryptography and payload calculations for packet encapsulation.

Each run of the algorithm is timed for 60 seconds using the above mentioned library inside the main loop of the core function, only accounting for the time spent during the encrypt-send or encrypt-send-receive-decrypt loops.

A USB power meter is connected in series between the device and it's power source to measure the Charge while running the algorithm used to calculate the Energy Cost.

C. Results

Method (Size in Bytes)	**I** (mA)	**P** (mW)	**E** (J)	**Data** (B)	**Throughput** (KB/s)	**EnergyCost** (μJ/B)
Cleartext Unidirectional						
51B	120	396	23.76	9,372,897	152.55	2.53
224B	140	462	27.72	22,163,512	360.73	1.25
1460B	60	198	11.88	33,479,264	544.91	0.35
4096B	100	330	19.80	34,655,148	564.05	0.57
Cleartext Bidirectional						
51B	500	1650	99.00	89,770	1.46	1102.82
224B	500	1650	99.00	433,319	7.05	228.47
1460B	240	792	47.52	9,530,005	155.11	4.99
4096B	120	396	23.76	13,238,984	215.48	1.79
AES128-GCM+DH+HMAC-SHA256 Unidirectional						
51B	100	330	19.80	1,235,515	20.11	16.03
224B	120	396	23.76	11,177,148	181.92	2.13
1460B	120	396	23.76	22,907,888	372.85	1.04
4096B	120	396	23.76	27,229,067	443.18	0.87
AES128-GCM+DH+HMAC-SHA256 Bidirectional						
51B	560	1848	110.88	27,010	0.44	4105.15
224B	580	1914	114.84	390,988	6.36	293.72
1460B	360	1188	71.28	9,411,691	153.19	7.57
4096B	500	1650	99.00	13,138,160	213.84	7.54
AES256-GCM+RSA Unidirectional						
51B	160	528	31.68	1,787,533	29.09	17.72
224B	120	396	23.76	11,314,660	184.16	2.10
1460B	100	330	19.80	17,568,684	285.95	1.13
4096B	80	264	15.84	23,559,008	383.45	0.67
AES256-GCM+RSA Bidirectional						
51B	480	1584	95.04	32,515	0.53	2922.93
224B	500	1650	99.00	361,340	5.88	273.98
1460B	300	990	59.40	8,023,456	130.59	7.40
4096B	360	1188	71.28	8,835,136	143.80	8.07

TABLE I: Energy Consumption Results (3.3V, 60s, averaged out of 3 runs each)

D. Analysis

A consistent trend is observed across all three methods, energy cost per byte shares an inverse relationship with block size.

For the Cleartext Unidirectional baseline, increasing the block size from 51B to 1460B results in an efficiency improvement of approximately 86% in energy cost, demonstrating that small packets suffer from a high ratio of protocol overhead as expected. A large amount of the data sent is lost in IP and TCP headers (20B + 20B) and cryptography suite headers such as IV and TAG for AES256 and HMAC for AES128 as per the implementation used in this evaluation. As the block size approaches the Maximum Segment Size of 1460 bytes and the page size of 4096 bytes, this overhead becomes much more negligible.

Bidirectional communication introduces a massive energy penalty compared to unidirectional stream. This varying order of magnitude can be attributed to the half duplex nature of the WiFi radio and the ESP32's processing logic. In bidirectional modes, the CPU must context switch between encryption and decryption, while the radio constantly switches states between RX and TX. The wait for response latency also prevents the CPU from entering light-sleep modes or batch processing data efficiently, and is likely the cause of higher current consumption with lower throughput resulting in a worse energy cost. The duality of transmission remaining mostly linear between cleartext and cryptographic data only highlights the performance decrease and energy consumption increase in these mode of communication. Except for larger block sizes where an encryption tunnel mostly stabilises while cleartext energy consumption per byte drops, possibly highlighting that encrypted communications aiming for smaller packet sizes can aim for the Maximum Segment Size for the smallest packet with the best performance.

An unexpected finding is the comparative performance of the cryptographic suites. Intuitively, AES-128 is expected to be lighter than AES-256. However, the AES256-GCM+RSA implementation frequently outperforms AES128-CTR+DH+HMAC-SHA256 in terms of energy cost, particularly in the unidirectional 4096B. It would make sense for smaller packets has the AES128 implementation has an extra 6 bytes of control headers than AES256 but not for larger packets. The possible cause seems to be cache access, a common constraint and maybe obvious as performing repeated read/write operations with a large volume of data proportional to the available cache and memory can constraint performance and increase energy usage. Further tests should be performed with more accurate measuring, such as using a joulescope, stabilising WiFi connection between tests and reviewing the algorithmic implementation for possible errors causing this discrepancy. In theory this could be attributed to the mode of operation. The AES128 implementation utilizes CTR mode with HMAC-SHA256 for integrity. HMAC is a two-pass algorithm and, while SHA256 has native hardware acceleration on the ESP32, the combined operation is still computationally expensive, meanwhile the AES256 implementation GCM which can be computed in parallel with the encryption while still being hardware accelerated. If further evaluation remains true, the efficiency gained by using GCM outweighs the computational penalty of the larger 256-bit key size and can validate the usage of AES256-GCM with RSA or even ECDH lighter encryption algorithms such as AES128-CTR, by providing better securing IoT devices even ones with energy constrains for networking operations.

IV. Conclusion

After an evaluation of the energy impact of cryptographic block sizes on the ESP32 as an energy constained embedded system, using PRISEC III framework recommendations—specifically AES128-CTR and AES256-GCM+RSA a few insights were revealed.

The results demonstrate that packet fragmentation and small payload sizes are energy inefficient for networking communications. The energy cost per byte is minimised the payload approaches the Maximum Segment Size for the network protocol, 1460 bytes for TCP/IP and larger packet sizes optimised for system processing such as 4096 bytes, the memory page boundary provide better results by

reducing the usage of the transmission hardware, which is the main energy consumption component in embedded devices.

The impact of hardware acceleration is also highlighted by reducing the impact of running cryptographic algorithms to secure the data before transmission, These can result in the trade offs between algorithms being thrown out of proportion and thus requiring a re-evaluation per implementation or device. The tests show that contrary to the intuitive assumption that a 128 bit key would inherently consume less energy than a 256 bit key. The results suggest that the computational overhead of the two pass HMAC scheme outweighs the cost of the larger key size in AES-256. Concluding that as cryptographic algorithms become hardware accelerated, cache access becomes a relevant constraint for optimisation. This also highlighting that cryptographic selection might not be able to be generic but rather, algorithms and suites must be evaluated against the specific hardware accelerators of the device to optimize the tradeoff between performance and energy consumption.

V. Further Work

- Expand evaluation for more modern and relevant protocols than LoRaWAN such as DECT NR+
- Further tests should be performed with more accurate measuring, such as using a joulescope, stabilising WiFi connection between tests and reviewing the algorithmic implementation for possible errors causing this discrepancy between AES128-CTR+DH+HMAC-SHA256 and AES256-GCM+RSA on the ESP32
- Extend testing to other IoT devices.

VI. Acknowledgments

Acknowledgments are in order to the professors Valderi Leithardt and Humza Sohail for leading the SRSI module, and for developing the PRISEC III framework.

References

[1] Sohail, H., Leithardt, V., & Trigo, A. (2025). *PRISEC III: Cryptographic Techniques for Enhanced Security*.

[2] EXPRESSIF SYSTEMS. (2016). *ESP-WROOM32 Manual (FCC Reviewed)*. Available: https://fcc.report/FCC-ID/2AC7Z-ESPWROOM32/3212970

[3] Silva, C., Cunha, V. A., Barraca, J. P., et al. (2024). *Analysis of the Cryptographic Algorithms in IoT Communications*.

[4] Maitra, S., Richards, D., Abdelgawad, A., & Yelamarthi, K. (2019). *Performance Evaluation of IoT Encryption Algorithms: Memory, Timing, and Energy*.

[5] Rafat, S. H., et al. (2025). *Lightweight Cryptographic Algorithm Analysis for Secure IoT Communication on ESP-32 Platforms*.

[6] Suárez-Albela, M., Fernández-Caramés, T. M., Fraga-Lamas, P., & Castedo, L. (2018). *Clock frequency impact on the performance of high-security cryptographic cipher suites for energy-efficient resource-constrained IoT devices*.

[7] Patterson, J. C., Buchanan, W. J., & Turino, C. (2025). *Energy Consumption Framework and Analysis of Post-Quantum Key-Generation on Embedded Devices*.

[8] Guo, C., Yang, Y., Zhou, Y., Zhang, K., & Ci, S. (2021). *A Quantitative Study of Energy Consumption for Embedded Security*.

[9] Guo, C., Ci, S., Zhou, Y., & Yang, Y. (2021). *A Survey of Energy Consumption Measurement in Embedded Systems*.

[10] Vaz, Y. S., Mattos, J. C. B., & Soares, R. I. (n.d.). *Lightweight AES Algorithm for Internet of Things: An Energy Consumption Analysis*.

[11] *ESP32 and Cryptography Courses: https://www.luisllamas.es/en/*.

[12] *python3 cryptography library documentation: https://cryptography.io/en/latest/*.

[13] *EspressIDF ESP-IDF documentation: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/index.html*.

[14] *archlinux wiki: https://wiki.archlinux.org/*.