signals

MDPI

Technical Note

Evaluating the Performance of Lightweight Ciphers in Constrained Environments—The Case of Saturnin

Panagiotis Podimatas [1] and Konstantinos Limniotis [1,2,*]

[1] School of Pure and Applied Sciences, Open University of Cyprus, Latsia, Nicosia 2220, Cyprus; panagiotis.podimatas@st.ouc.ac.cy
[2] Department of Informatics and Computer Engineering, University of West Attica, Egaleo, 12243 Athens, Greece
* Correspondence: konstantinos.limniotis@ouc.ac.cy

Abstract: The use of lightweight cryptographic algorithms is essential for addressing security in highly constrained environments such as the Internet of Things. In this paper, the performance of lightweight block ciphers in such highly constrained environments is studied. More precisely, focusing—as a case study—on an important family of lightweight ciphers called "Saturnin", which has been evaluated as a candidate for standardization in the relative ongoing NIST's competition, we analyze its efficiency in case that it is implemented in a specific resource-constrained environment. To evaluate the results, a comparative study with the Advanced Encryption Standard (AES) is performed, through an appropriate experimental environment. Our results illustrate that significant gain in performance can be achieved, since Saturnin—whose design is inspired by the design of AES—can be almost two times faster than AES in such restricted environments.

Keywords: lightweight cryptography; performance; Saturnin

Citation: Podimatas, P.; Limniotis, K. Evaluating the Performance of Lightweight Ciphers in Constrained Environments—The Case of Saturnin. *Signals* **2022**, *3*, 86–94. https://doi.org/10.3390/signals3010007

Academic Editor: Chin-Ling Chen

Received: 8 December 2021
Accepted: 18 January 2022
Published: 16 February 2022

1. Introduction

The vision of Internet of Things (IoT) becomes highly prevalent into a wide range of modern applications. It has been stated more than 18 billion of IoT devices (e.g., wireless sensors and embedded devices) were deployed to the market and connected through cloud platform by the end of 2020, whereas more than half of them were of industrial usage [1]; moreover, it is expected that more than 75 billion IoT-connected devices will be in use by 2025, thus having an increase of more than 300% within five years [2]. However, such environments are often characterized by very limited computing power, as well as by low power and memory capacity. In such restricted environments, the classical cryptographic algorithms cannot be efficiently implemented and, thus, are not candidates for providing cryptographic solutions; even the Advanced Encryption Standard (AES) is being considered as too expensive, despite the fact that there are various hardware and software implementations of the algorithm that strive to reduce its requirements in computation power and layout area (see, e.g., [3]). More precisely, the challenges while implementing conventional cryptography in IoT devices are as follows [4]: (i) Limited memory (registers, RAM, ROM), (ii) reduced computing power, (iii) small physical area to implement the assembly, (iv) low battery power (or no battery), and (v) real-time response.

To address the above, lightweight cryptography is being used for providing security services, such as data confidentiality and integrity, in cases which necessitate the efficient implementation of cryptographic algorithms in constrained devices—including not only IoT applications, but also any other application area with the above characteristics (e.g., RFID communications). To this end, NIST has initiated a process to solicit, evaluate, and standardize lightweight cryptographic algorithms that are suitable for use in constrained environments where the performance of current NIST cryptographic standards is not acceptable [5]. More precisely, NIST has initiated a call of algorithms to be considered

as lightweight cryptographic standards. NIST received 57 submissions to be considered for standardization. After the initial review of the submissions, 56 were selected as Round 1 Candidates (announced in April 2019). The second round consisted of 32 out of the initial 56 candidates (announced in August 2019). Currently, this competition is in its third round, where the finalists are 10 algorithms (announced in March 2021).

Due to the high importance of the aforementioned NIST competition, there is a strong research interest in analyzing several aspects of these lightweight ciphers, including their efficiency (see, e.g., [1] and the references therein, as well as [6]). In this paper, we focus, as a case study, on a specific candidate of the NIST competition, called Saturnin [7], in order to estimate its performance (based on measurements of the time needed to perform encryption and decryption) in a realistic restricted environment. Saturnin has been chosen as an indicative example due to the fact that its design is inspired by the design of AES, whereas it is also being considered as a post-quantum secure cipher. Saturnin was a candidate cipher at the 2nd round of the NIST competition, but it is not one of the finalists announced in March 2021. However, as it is analyzed in the status report on the second round of the NIST Lightweight Cryptography Standardization Process [8], the Saturnin team has provided a significant analysis on its security, based on the fact that the security arguments for the AES are also applicable for Saturnin since they share a common structure (although this does not mean that Saturnin is bound to achieve exactly the same security level as AES). According to [8], the main reason that Saturnin has not been chosen as a finalist for the third round was the fact that the original submission is an Encrypt-then-MAC implementation and, thus, *it does not rank as favorably as single-pass candidates in terms of throughput*. However, taking into account the security advantages of the Saturnin, and especially the fact that it is explicitly designed to resist quantum attacks with a 256-bit key and block size (despite the fact that such a resistance was not a formal requirement by the standardization process), it is of interest to further focus on its performance compared to AES. To this end, encryption and decryption have been implemented into an appropriate restricted environment, whereas a comparative study with AES, within the same environment and the same key size, is also conducted.

It should be pointed out that, as it has been shown in a recent comprehensive work [6] in which all the 32 candidates in the Round 2 algorithms were examined in several platforms, the execution time of the algorithms varies significantly. Hence, any new information on the performance of lightweight ciphers on specific restricted environments is of importance, towards assisting the relevant stakeholders in selecting an appropriate algorithm for a given platform. In this technical note, our preliminary analysis illustrates that Saturnin is much more efficient than AES in the environment we check, being able to reduce the Round Trip Time up to the half value (compared to AES). These results are compatible with the relative information given in the original submission of the Saturnin and, thus, they further reveal the performance benefits that can be achieved by the lightweight cryptography.

The structure of the report is as follows. In Section 2, a short description of the lightweight cryptography and, especially, of the Saturnin is given. Section 3, being the main part of the paper, presents the experimental environment as well as the experimental results in terms of comparing the performance between Saturnin and AES. Finally, concluding remarks are given in Section 4.

2. Background

2.1. Lightweight Cryptography

Lightweight cryptography is a subfield of cryptography that aims to provide solutions tailored for resource-constrained devices [9]. Although an obvious approach is to develop efficient implementations of conventional cryptographic standards, the design and analysis of new lightweight primitives seems to be the proper way to address this challenge. To this goal, NIST published in 2018 a document containing the requirements and the evaluation criteria of submissions to the relevant NIST competition for a lightweight cryptography standard [9].

The NIST competition is ongoing, and it is currently in its third round, in which 10 algorithms (from the initial list of 56 algorithms in the first round) have been selected to continue. The selection of the schemes highly relies on the efforts that the cryptographic community puts in order to explore and evaluate their security arguments. It should be pointed out that, as an explicit requirement that NIST has set, each submission to the NIST competition should contain a family of AEAD (Authenticated Encryption with Associated Data) schemes, so as to ensure that not only confidentiality but also authenticity of the data are being provided.

It should be stressed that the design of a lightweight cryptographic algorithm achieving all the design goals—i.e., high security, low cost and high performance—is not an easy task. It is generally assumed that, although any two of these three design goals can be (easily) optimized, it is very difficult to simultaneously achieve all these three goals. Therefore, it is of high interest to examine the performance of a lightweight (i.e., of low cost) cipher with high security standards.

2.2. The Saturnin Cryptographic Suite

Saturnin constitutes a family of cryptographic functions, being able to provide both authenticated encryption as well as hashing operations [7]. More precisely, Saturnin is a 256-bit block cipher with a 256-bit key and an additional 9-bit parameter for domain separation, which can provide the following cryptographic services:

1. Authenticated encryption, through either the so-called mode *Saturnin-CTR-Cascade*, which is an authenticated cipher using the counter mode and a separate MAC, or the so-called *Saturnin-Short* mode which is intended for messages with length strictly smaller than 128 bits.
2. A 256-bit hash function, via the so-called *Saturnin-Hash* mode.

The Saturnin cipher is inspired by the AES, operating on 256-bit blocks and using a 256-bit internal state. Similarly to AES, in which the internal 128-bit state can be seen as a 4×4 square of bytes, the state value in the Saturnin can be viewed as a $4 \times 4 \times 4$ cube of 4-bit nibbles [7]. Therefore, it can be somehow seen as an extension of AES into 3 dimensions. Saturnin is being considered—in a broad sense—as a SPN cipher. A detailed description of Saturnin, as well as its security analysis, are provided in [7]. As it is stated therein, its security analysis follows naturally from that of the AES, providing a high security level even in the presence of quantum computers.

It should be pointed out that *Saturnin-CTR-Cascade*, being actually an AEAD, requires two passes over the data (and, as stated in Section 1, this was the reason that Saturnin is not one of the 10 finalists in the NIST's competition). As the authors state in [10], this is the case because it is hard to achieve security against superposition queries with one pass—and security against such queries seems to be the right way to achieve post-quantum security in symmetric primitives. However, as it is stated in [10], the authors were in the phase of finalizing an one-pass mode, being called QCB.

The original submission of Saturnin included three constant-time implementations: in portable C, in assembly for the ARM Cortex M3 and in assembly for the ARM Cortex M4. Some performance results, in terms of cycles per bytes (cpb), are given in [7]—e.g., for the encryption operation, the costs of the above implementations are 250 cpb, 144 cpb and 147 cpb, respectively. An estimation of the performance of a constant-time AES/GCM implementation on an ARM Cortex M4 is about 161 cpb, as it is described in [10]. Such a comparative study is further elaborated in our present work.

3. Evaluating the Performance

3.1. Methodology—The Testing Environment

Towards studying the performance of Saturnin in a restricted environment for IoT applications, we utilized an ESP8266 System on a Chip (SoC), the Node MCU 0.9 [11]. The ESP8266's CPU is a Tensilica L106 32-bit micro controller (MCU) which also has an embedded Wi-Fi transceiver. It has eleven GPIO (General Purpose Input/output) pins,

and one analog input. Finally, it has one micro-USB port, which is used for power and for communicating with a PC (through serial terminal).

The Node MCU is compatible with any Arduino board. Hence, it can be expanded with various modules connected to its GPIO pins, like Bluetooth, cameras, etc. It also has full network capabilities through Wi-Fi and it can operate either as a server or a client. The Tensilica L106 32-bit RISC processor is able to reach clock speeds up to 160 MHz, whereas the maximum RAM for applications is 80 Kbytes. Because of its full network capabilities through WIFI and its limited RAM and CPU speed, it is ideal for studying the performance | of Saturnin in lightweight environments.

To program the board, the Arduino framework has been used, which is a standard framework for microcontrollers. Indeed, Arduino has been widely used for testing and evaluating standard cryptographic algorithms, as well as lightweight cryptographic algorithms (see, e.g., [12–14]), whilst it is also used in projects employing blockchain technology (see, e.g., [15]). It has thousands of libraries, most of them being cross-platform. In our case, we used, for the AES encryption, the so-called "crypto library". Its programming language is called "wiring" and it is a dialect of C++. For compilation, the framework uses the native C compiler of the board to produce machine code and make the final firmware.

Our testing environment consists of two applications:

1. The desktop application, called "cipher network client". It runs on a PC and sends requests to the ESP8266 board.
2. The "nodemcu" application called "esp cryptoserver". It operates as a server, replying to client requests after performing the necessary encryption.

For the desktop ("client") application, we used C++ as programming language and the Qt framework [16] for the GUI (graphical user interface). Qt is a cross platform framework, under GPL license, able to be deployed across all major operating systems.

For the AES part, we choose an implementation being called "Tiny AES" [17], which is a free public portable implementation of the AES in portable C. It supports various modes of operation like ECB, CTR and CBC, whereas this implementation has been verified against the relevant test vectors provided by the NIST. By utilizing this implementation, we avoided using the implementation that the operating system offers, which may implement certain hardware acceleration for AES and, thus, could highly affect the accuracy of our comparative study. Similarly, for the Saturnin algorithm, we used the original implementation in portable C, i.e., the one submitted to the NIST by the Saturnin development team, and not any implementation using hardware-specific commands.

A screenshot of the desktop client is shown in Figure 1.

When the «connect to server» button is pressed, the client sends requests with the original message to the server (ESP8266) through Wi-Fi and receives three answers: one with the message in an unencrypted form, one with the message being encrypted with AES and one with the message being encrypted with Saturnin. It measures the total time for each answer and prints it on the screen.

The "nodemcu" (ESP8266) application operates as a server and it is written in C++ using the Arduino framework. We used Visual Studio Code [18] with the PlatformIO extension [19]. The compilation to machine code, the preparation of the firmware and its uploading to the board have been automatically performed by the IDE.

For the AES encryption, we used the "crypto" library for the Arduino framework, in AES CTR (CounTeR) mode. For the Saturnin encryption we used the same portable C implementation that we used for the client application. Note that, for our experiments, we utilized the Saturnin-Short version, performing only encryption and not data authentication (since the AES-CTR also performs only encryption). Although the Saturnin-Short should not be used in practice for messages with sizes larger than 128 bits, this restriction, which exists for security reasons, does not affect our analysis since we focus only on its performance. Moreover, Saturnin-short does not require two passes of data (which has been considered as the main drawback of the full version of Saturnin, with respect to the 10 finalists of

the NIST's competition), but uses only one call to Saturnin to provide confidentiality and integrity.

Every Arduino application has two main functions:

- void setup(): This function performs the initial setup. It is being used for the WiFi connection, the server initialization and a global variable setup. Then, it checks the correctness of the algorithms results and prints all the required messages to the terminal (see Figure 2).
- void loop(): This is the main function, which is being executed continuously while a condition holds; otherwise, the program stops running (the board needs reset to run the program again). Here, the server "listens" to the port 2080 and for every request received from the client, it replies sending back the message, first unencrypted, then encrypted with AES and finally encrypted with Saturnin. It subsequently prints to the terminal the time needed for AES encryption and for Saturnin encryption. All times are shown in milliseconds.

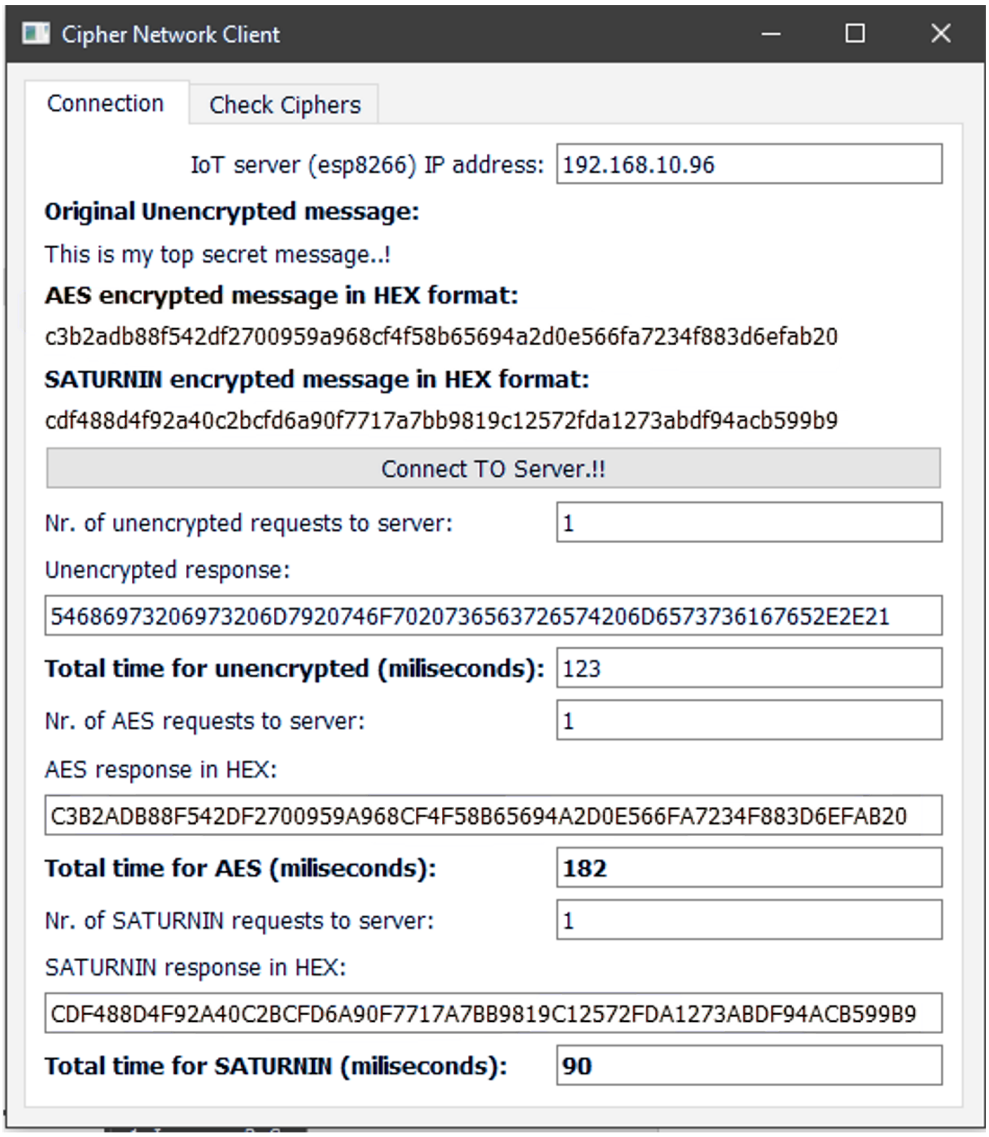

Figure 1. A screenshot from the client application, illustrating the responses that the client receives and the messages that it prints regarding the time measurements.

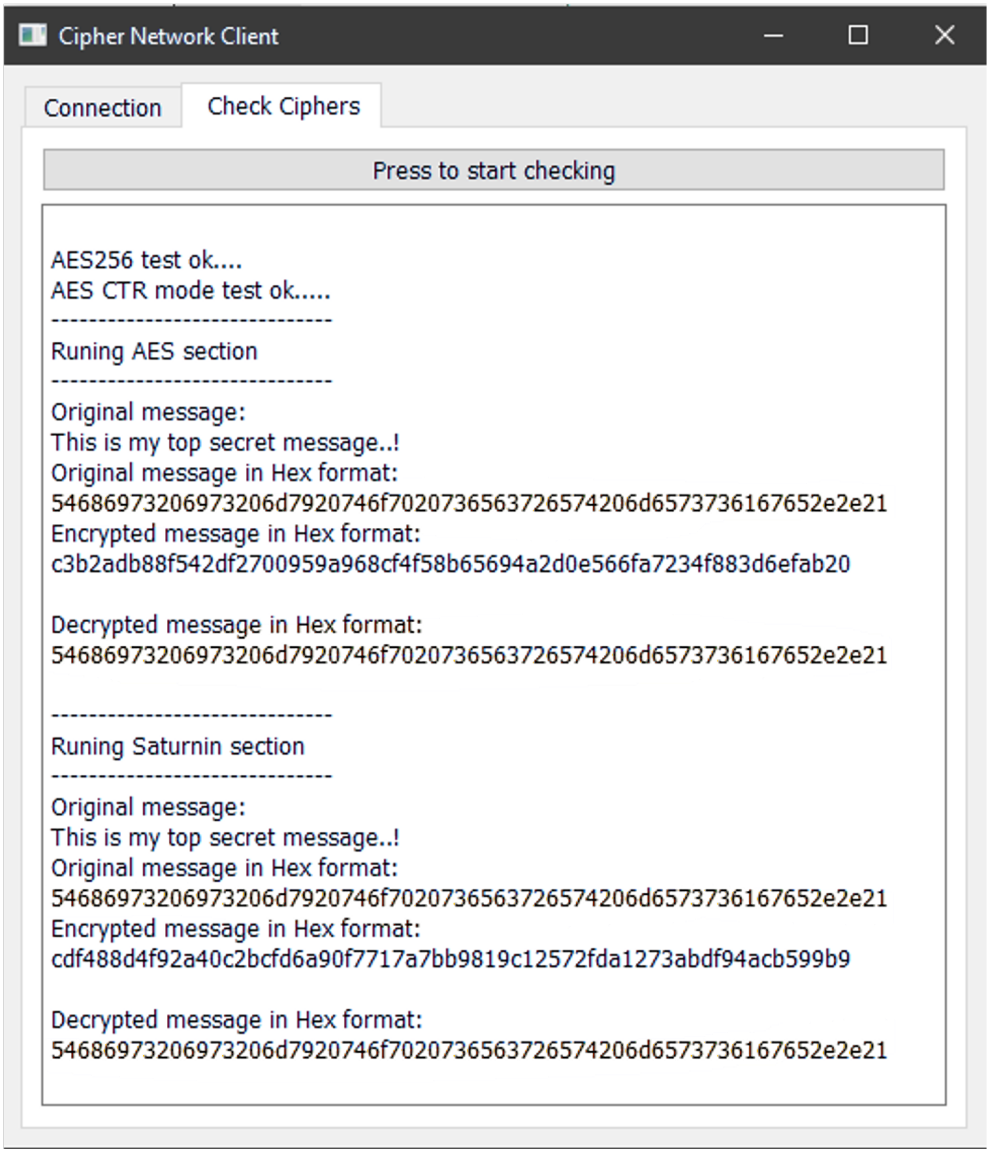

Figure 2. A screenshot from the execution of a setup, illustrating that, for both ciphers, the decrypted messages coincide with the original plaintext.

A serial terminal view after a successful answer to a client's request is shown in Figure 3.

3.2. Experimental Results

For both algorithms, the key size that we utilized is 256 bits—i.e., we chose the maximum possible size for AES key, in order to perform comparisons with the same key sizes. Our original message, before the crypto procedure, is the ANSI string "This is my top secret message…!", having length equal to 32 bytes (256 bits).

To estimate the performance, the client makes 100 requests to the server and measures the total time needed for the following operations:

- Sending the request to server;
- Encrypting the message;
- Transmitting the (encrypted) message back to the client;
- Decryption of the message received;
- Writing the result on the screen.

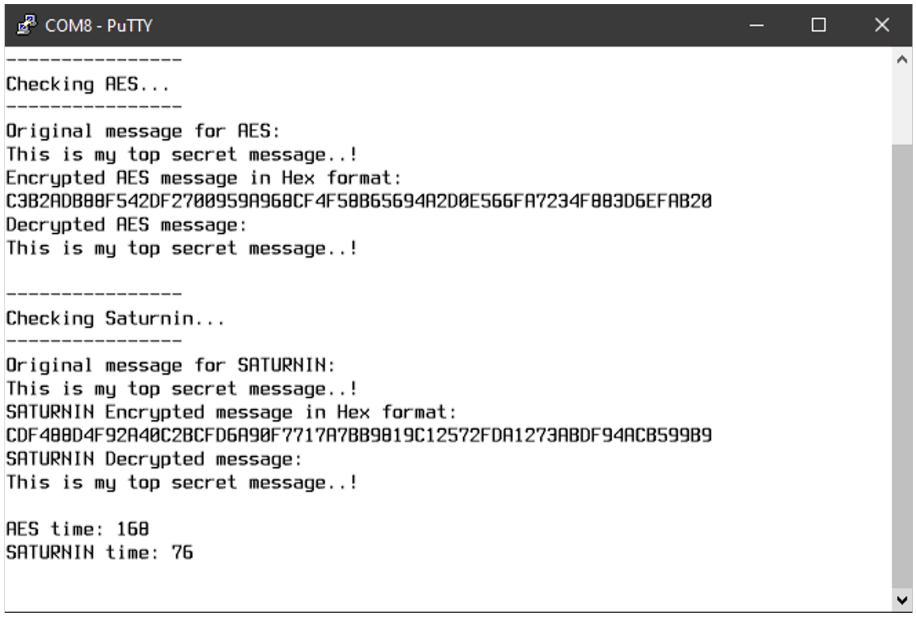

Figure 3. A screenshot after a response from the server to the client, illustrating the time needed for encryption for both AES and Saturnin (which is independent from the network and, thus, constant).

As stated above, the server responds initially by sending back the message unencrypted, then it sends it back with AES encryption and finally, with Saturnin encryption (and, thus, the above operations are being adjusted accordingly). Hence, we actually estimate the time duration for three different scenarios—one for each of the above cases (for the first case, no encryption operation is performed).

In order to illustrate more clearly the comparison between AES and Saturnin in terms of performance, the server repeats, for both cases, the encryption algorithm 500 times and, at the subsequent iteration, it sends the encrypted message to the "client". In other words, we deliberately introduced the same encryption delays for both cases, since otherwise the time needed for encryption was negligible to be measured by the software. By these means, the server encrypts 16,032 bytes (501×32 bytes) in total.

The results of the above are shown in Figure 4, which illustrates that, in almost all cases, Saturnin is nearly two times faster than AES (i.e., the estimated RTT is being reduced almost to the half value). More precisely, the minimum measured RTT for AES was 173 ms and the maximum was 220 ms. For Saturnin, the corresponding minimum value was 83 ms and the maximum 196 ms.

For a more concrete comparative study, we also estimated the mean value of all RTTs measured, for each case (AES and Saturnin). By these means, we can verify that the mean value of the RTT for the Saturnin was 93.71 ms, whereas the corresponding value for AES was 185.27 ms.

It should be stressed that, as we noticed, there exist variations in time measurements on the client side per request, while the time measurements on the server side are always the same. This happens because the client measures the total time of the transaction, whereas the server measures only the time of the encryption. That is why the server almost always prints (measures) 168 ms for AES encryption and 76 ms for Saturnin encryption—and this is actually the pure comparison on the performance of the encryption operations between Saturnin and AES. Therefore, we actually verify that the network load has also a crucial role for the overall performance of an encrypted network communication. In any case though, even if we focus explicitly on these values, we again come to the conclusion that Saturnin is about 2 times faster than AES.

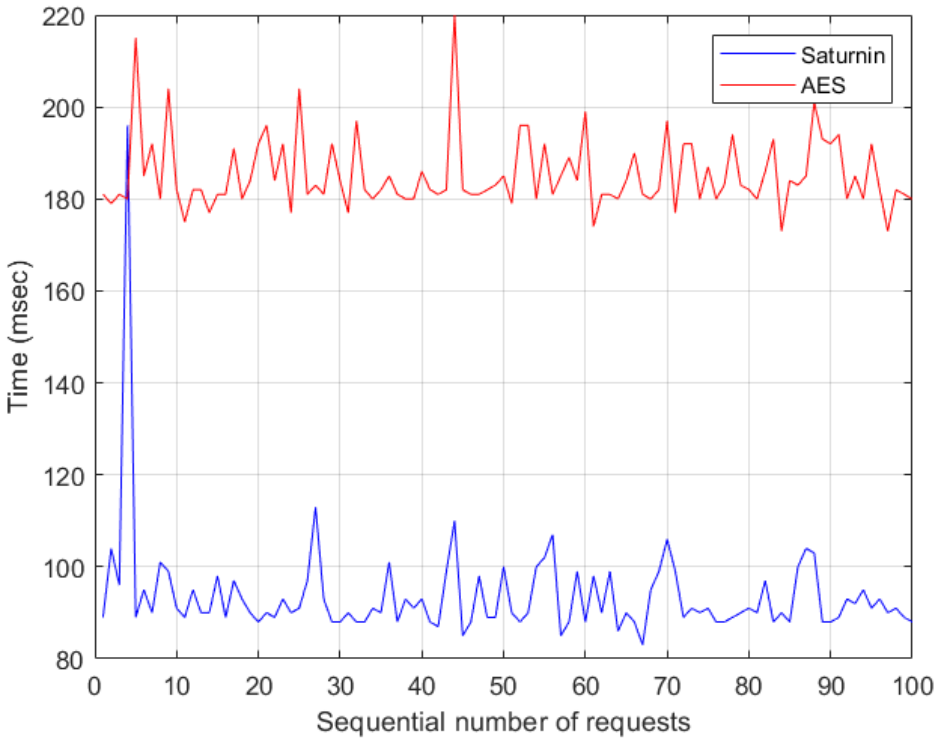

Figure 4. RTTs for 100 transactions, initiated by client's requests, for 32-byte messages.

4. Conclusions

In this note, a preliminary comparative study between Saturnin and AES, with respect to their performance in constrained environments, is carried out. The main outcome from the experiments performed is that Saturnin is significantly faster than AES as block cipher. This performance gain is prevalent in the case of the Saturnin-Short version of Saturnin. However, since Saturnin-Short cannot be used, for security purposes, in encrypting big messages, it is essential to establish a similar comparative study for the case of Saturnin CTR Cascade, which clearly necessitates more CPU resources than Saturnin-Short. It is expected though that, even in this case, Saturnin will still remain faster than AES. Most importantly, it would be of great interest to examine the new version of Saturnin that requires only one pass over the data.

The above result is of importance, taking into account that Saturnin, despite its nice security features (including post-quantum security), will not be standardized by NIST due to its Encrypt-then-MAC nature, which is not as efficient as single-pass AEAD encryption schemes. By this study, it becomes evident that lightweight cryptography suffices to provide practical security solutions in constrained environments. Especially in scenarios which do not necessitate the usage of a standardized algorithm, there are many options for choosing the proper cipher on an ad hoc basis depending on the specific needs (i.e., low battery/energy, need for high speed, etc.). Therefore, the NIST competition, independently from what its final outcome will be, provides the opportunity to have a large pool of important lightweight ciphers.

Author Contributions: P.P. and K.L. have contributed in conceptualization, methodology and writing. All the experiments have been set up and performed by P.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The author declares no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
cpb	cycles per byte
CTR	CounTeR (mode of operation)
IoT	Internet of Things
MAC	Message Authentication Code
NIST	National Institute of Standards and Technology
RAM	Random Access Memory
ROM	Read Only Memory
RTT	Round Trip Time

References

1. Meng, T.X.; Buchanan, W. Lightweight Cryptographic Algorithms on Resource-Constrained Devices. Preprints 2020, 2020090302. Available online: https://www.preprints.org/manuscript/202009.0302/v1 (accessed on 8 December 2021).
2. Abed, S.; Jaffal, R.; Mohd, B.J.; Al-Shayeji, M. An analysis and evaluation of lightweight hash functions for blockchain-based IoT devices. *Clust. Comput.* **2021**, *24*, 3065–3084. [CrossRef]
3. Moradi, A.; Poschmann, A.; Ling, S.; Paar, C.; Wang, H. Pushing the limits: A very compact and a threshold implementation of AES. In *Advances in Cryptology—Eurocrypt 2011, Tallinn, Estonia, May 2011*; Paterson, K.G., Ed.; Lecture Notes in Computer Science; Springer: Heidelberg, Germany, 2011 ; Volume 6632, pp. 69–88.
4. Thakor, V.A.; Razzaque, M.A.; Khandaker, M.R.A. Lightweight Cryptography Algorithms for resource constrained IoT devices: A Review, Comparison and Research Opportunities. *IEEE Access* **2021**, *9*, 28177–28193. [CrossRef]
5. National Institute for Standards and Technology, Lightweight Cryptography Project. Available online: https://csrc.nist.gov/Projects/lightweight-cryptography (accessed on 8 December 2021).
6. Fotovvat, A.; Rahman, G.M.E.; Vedaei, S.S.; Wahid, K.A. Comparative performance analysis of lightweight cryptography algorithms for IoT sensor nodes. *IEEE Internet Things J.* **2021**, *8*, 8279–8290. [CrossRef]
7. Canteaut, A.; Duval, S.; Leurent, G.; Naya-Plasencia, M.; Perrin, L.; Pornin, T.; Schrottenloher, A. Saturnin: A suite of lightweigth symmetric ciphers for post-quantum security. *IACR Trans. Symmetric Cryptol.* **2020**, *2020*, 160–207. [CrossRef]
8. National Institute of Standards and Technology. Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process. 2021. Available online: https://nvlpubs.nist.gov/nistpubs/ir/2021/NIST.IR.8369.pdf (accessed on 8 December 2021).
9. National Institute of Standards and Technology. Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. 2018. Available online: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf (accessed on 8 December 2021).
10. Canteaut, A.; Duval, S.; Leurent, G.; Naya-Plasencia, M.; Perrin, L.; Pornin, T.; Schrottenloher, A. An update on Saturnin. NIST Lightweight Crypto Standardization process (Round 2). 2020. Available online: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/Saturnin_update.pdf (accessed on 8 December 2021).
11. NodeMcu, Connect Things EASY. Available online: http://www.nodemcu.com/index_en.html (accessed on 8 December 2021).
12. El-Haii, M.; Chamoun, M.; Fadlallah, A.; Serhrouchni, A. Analysis of cryptographic algorithms on IoT hardware platforms. In Proceedings of the 2nd Cyber Security in Networking Conference (CSNet), Paris, France, 24–26 October 2018; pp. 1–5.
13. Engineer, M.; Shah, A. Performance analysis of lightweight cryptographic algorithms simulated on Arduino UNO and MATLAB using the voice recognition application. In Proceedings of the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 21–22 December 2018; pp. 1–7.
14. Mathew, B.K. Protecting embedded systems against Class I & Class II cloning attacks using Arduino boards. In Proceedings of the 2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET), Kottayam, India, 21–22 December 2018; pp. 1–4.
15. Okada, T. Handle Smart Contract on Ethereum with Arduino or ESP32. Available online: https://medium.com/@takahirookada/handle-smart-contract-on-ethereum-with-arduino-or-esp32-1bb5cbaddbf4 (accessed on 8 December 2021).
16. Qt Framework. Available online: https://www.qt.io/ (accessed on 8 December 2021).
17. Github, Tiny AES Implementation. Available online: https://github.com/kokke/tiny-AES-c (accessed on 8 December 2021).
18. Visual Studio Code. Available online: https://code.visualstudio.com/ (accessed on 8 December 2021).
19. Platformio. Available online: https://platformio.org/ (accessed on 8 December 2021).