2019 IEEE International Conference on Fog Computing (ICFC)

A Performance Evaluation of Data Protection Mechanisms for Resource Constrained IoT Devices

Clemens Lachner
Distributed Systems Group
TU Wien
Vienna, Austria
c.lachner@dsg.tuwien.ac.at

Schahram Dustdar
Distributed Systems Group
TU Wien
Vienna, Austria
dustdar@dsg.tuwien.ac.at

***Abstract*—Data Protection is a major research topic concerning the Internet of Things (IoT). IoT systems continue to permeate deeper into our personal lives, where devices sense, process, and store all kinds of data. This poses various challenges to security and privacy aspects, especially to applications running on resource constrained devices. In this paper we evaluate selected, well established data protection mechanisms that enable confidentiality and integrity of data. Specifically, we look into the performance of different cryptographic block and stream ciphers, hashing algorithms, message authentication codes, signature mechanisms, and key exchange protocols executed on state-of-the-art resource constrained devices. By providing limitations and data throughput values, our obtained results ease the calculation of performance/data protection thresholds and facilitate the design and development of secure IoT systems.**

***Index Terms*—IoT, data protection, resource constrained devices, performance evaluation, integrity, confidentiality, authenticity, encryption, microcontroller**

I. Introduction

Different interconnected devices with minimal to average computing power make up the majority of the Internet of Things (IoT). They sense, process, and store various data from different domains and have become an integral part of our daily lives [1]. The specific nature of data related to critical domains, such as healthcare, public surveillance or home automation, requires tailored data protection mechanisms concerning security and privacy aspects. Data Integrity, confidentiality, authenticity, anonymization and source location privacy pose different, partly overlapping challenges for the design and implementation of an IoT system. Those challenges become even more demanding when dealing with resource constrained devices. In this paper we refer to devices that lack computing power, i.e., microcontrollers (MCU) with low CPU power and limited memory. Furthermore, we distinguish between devices typically found in Wireless Sensor Networks (WSN), which could be seen as an infrastructural subset on the IoT, and typical resource constrained IoT devices. Most of the devices found in WSN are sensors and actuators mostly adhering to the IEEE 802.15.4 protocol, using communication frameworks like ZigBee. We define a resource constrained IoT device as a device with limited computational power but integrated Ethernet or WiFi capabilities and USB connectivity for facilitated programming.

The design of an IoT system, dealing with sensitive data, is often accompanied with the definition or followed upon a set of distinct data protection rules, e.g., in the form of policies. The overall risk that such a system is exposed to could be assessed by looking on its various attack vectors of an adversary. Concerning privacy, protecting the system from attacks on confidentiality of the data is crucial. Other characteristic threats to those system are spoofing attacks, message altering, replay, and flooding attacks [2]. Well established countermeasures reducing or eliminating the attack surface on a system are often not designed to be executed on resource constrained devices or may perform poorly on those [3]. However, as technology advances, generalized statements on performance of data protection mechanisms in IoT, such as:*"Asymmetric Cryptography is infeasible"*, should not be made. Therefore, it is necessary to continuously evaluate those various mechanisms and to show whether such statements still hold true or not. In this paper we evaluate the performance of different algorithms concerning data integrity, authenticity, and confidentiality. Our testset of algorithms comprises well established standard implementations of encryption, key exchange, signing, and hashing methods, as well as lightweight implementations, respectively. Many of widely used, commercially available, resource constrained IoT devices come with built in *crypto chips*, i.e., specialized hardware to accelerate specific cryptographic applications. These chips usually feature methods used by the SSL/TLS standard, but they do not support the many other different algorithms available used for data confidentiality and data integrity that could be needed to facilitate the development of custom protocols, e.g., onion routing, relying on secure and privacy preserving data transfer. Hence, we formulate the following two research questions:

RQ1: What are the limitations of representative IoT devices performing selected data protection mechanisms?
RQ2: What are the individual throughput rates that are needed for performance/data protection trade-off calculations?

The results gathered by our evaluation can aid developers and system engineers in designing a system that adheres to specific security guidelines, yet maintains adequate performance. Additionally, systems may encounter changes of several parameters during runtime, such as sensor reading frequency, data

978-1-7281-3236-5/19/$31.00 ©2019 IEEE
DOI 10.1109/ICFC.2019.00015

TABLE I
TESTBED OVERVIEW OF REPRESENTATIVE RESOURCE CONSTRAINED IOT DEVICES

Device Name	Processor	CPU Speed	SRAM	Flash Memory
Arduino MKR1000 WiFi	Cortex-M0+ 32-Bit	48 MHz	32 KB	256 KB
Wemos ESP8266 D1 mini	Xtensa LX106 32-Bit	80-160 MHz	160 KB	4 MB
Espressif ESP32-WROOM-32	Xtensa LX6 32-Bit DualCore	160-240 MHz	512 KB	4 MB

size, or a specific *level* of data protection. The level of data protection could manifest itself via changing security policies enforcing a minimal key size of an encryption algorithm or ensuring data integrity using a message authentication code. In particular this becomes relevant for adaptive systems, e.g., fog nodes running an risk-assessment engine which needs the end devices (i.e., our resource constrained IoT devices) to be flexible concerning data protection methods.
The rest of the paper is structured as follows: Section II provides an overview of related work. Relevant information on cryptographic background, methodology, and experimental setup is given in Section III. In Section IV we present the corresponding results. Finally, in Section V, we conclude the paper and give an outlook on future research.

II. RELATED WORK

As IoT devices continue to permeate deeper into our personal lives, security and privacy aspects have become a major research topic in this field. Suo et al. [3] divide an IoT system into four layers. Bottom-up these are: Perception Layer, Network Layer, Middleware Layer, and Application Layer. An overview of possible threats and adverse scenarios on each of those layers is provided by Farooq et al. [2], concluding that the majority of threats are located in the Network Layer. To mitigate or at least lower the risk of these threats, trade-offs have to be made either at design time or runtime of a system. Those trade-offs involve parameters like energy consumption, strength of encryption or data throughput. Altering one of these parameters will most likely affect one or many of the others. In the area of WSN, where energy consumption and management is particularly of interest, security evaluations have been done by several researchers. The majority of related work in this fields deal with lightweight implementations of cryptographic algorithms. In [4], various security solutions were analyzed. It provides an overview of the individual security characteristics, requirements, attacks, and encryption algorithms, considered to be useful for designers of a secure WSN. Alharby et al. [5] studied the security costs in terms of energy consumption focusing on the IEEE 802.15.4 transmission protocol definition. In their simulation they demonstrate the impact of security message overhead on data latency and throughput, followed by an evaluation of the effects those overheads have on energy consumption and their memory footprint. In this paper we consider such network overheads as negligible, because of the many others factors that affect network performance, like signal strength or available bandwidth. Also evaluating the costs of security in WSN, but focusing on energy consumption, Lee et al. [6] measured and compared four lightweight encryption algorithms on MicaZ and TelosB sensor motes. Those devices reside at the lowest end of the WSN spectrum, regarding computing power. In typical IoT environments, such as Smart Buildings, energy consumption may not be of primary concern, while data protection and throughput is considered more important. Therefore, in this paper we solely focus on the evaluation of data protection mechanism and their performance in terms of computing power, based on measurements gathered from, what we classify as, typical resource constrained IoT devices. While our classification is based on a devices computing capabilities, other classifications can be found in literature, e.g., Bormann et al. [7] suggest a classification scheme based on the memory capacity of an IoT device. Rani et al. [8] studied literature on the performance of various lightweight encryption algorithms, focusing on the medical domain. In their work they provide an overview of addressed problems, methodology and outcome of different research articles. However, the addressed papers deal with evaluating the performance of lightweight cryptographic algorithms only, mostly implemented on Field Programmable Gate Arrays (FPGA). An overview of various security mechanisms in the IP-based IoT is provided by Cirani et al. [9]. In their work they discuss different algorithms located at the network, transport, and application layer. They focus on a detailed description of each algorithm and its properties like key size, code size, block size, etc., but do not evaluate their performance in terms of processing speed or computational effort. Closely related to our approach is the work of Ertaul et al. [10]. In their paper the performance of lightweight stream ciphers is evaluated. Implementation of three different algorithm were deployed on a NodeMCU development kit and their performance evaluated. Their measurements include i) stream cipher throughput, ii) power consumption, iii) memory utilization and iv) WiFi Round Trip Throughput. However, they solely focused on lightweight stream ciphers and their most powerful device corresponds to the least powerful one of our testbed. Another publication, closely related to our work is provided by Sethi et al. [11] in RFC8387. They evaluate various symmetric and asymmetric encryption algorithms with different key sizes on a 8-bit microcontroller. In their work they state that: *"It is important to state that 32-bit microcontrollers are much more easily available, at lower costs, and are more power efficient. Therefore, real deployments are better off using 32-bit microcontrollers that allow developers to include the necessary cryptographic libraries. There is no good reason to choose platforms that do not provide sufficient computing*

TABLE II
OVERVIEW ON EVALUATED DATA PROTECTION MECHANISMS

Algorithm	Type	Purpose
AES	Block Cipher	Confidentiality
ChaCha	Stream Cipher	Confidentiality
Ed25519	Digital Signature Scheme	Signatures
SHA3	Hashing Algorithm Family	Integrity
Poly1305	Message Authentication Code	Message Authentication
ECDH	Key Agreement Protocol	Key Exchange

power to run the necessary cryptographic operations." In our work the performance of algorithms related to both, integrity and confidentiality, is evaluated using a testbed comprising only 32-bit microcontrollers.

III. METHODOLOGY

In this section we discuss our evaluation process. First, we present the constellation and an architectural overview of the used testbed. Second, we briefly introduce selected, well established data protection mechanisms which we chose for our test scenarios, and how we conduct our experiments.
We remark that entropy generation for random number generation or specific system watchdog timer restart implementations are out of scope of this evaluation.

A. Testbed

Our testbed comprises three representative resource constrained IoT devices. We chose only commercially available and well established microcontrollers mounted on development boards which integrate our desired capabilities. This decision is based on our goal to facilitate the development of secure IoT systems, in a way that our obtained results can be easily adjusted to real world use cases. Our main criteria for selecting appropriate devices, is i) limited CPU power and memory, ii) an integrated WiFi chip and iii) USB connectivity. We see such devices residing at the lower end of the IoT spectrum, close to WSN devices which we position at the lowest end. Out of this device set, we deliberately pick only MCUs that differ from each other in terms of computing power and memory, and which are prevalent on IoT development boards. Though there are a lot more devices commercially available, the majority does not differ in terms of computing power, because most of them come in the form of development boards, designed for various purposes, but featuring the same MCU mounted on the board. Table I provides an overview of the selected devices. All of those devices are programmed using either *C* or *C++*, while the sourcecode is compiled and deployed using the Arduino IDE.

B. Test Scenarios

This subsection provides an overview on the various data protection mechanisms evaluated. We are particularly interested in algorithms supporting confidentiality, authenticity, and integrity. Confidentiality refers to mechanisms enabling the protection of data from disclosure to parties not authorized to read or act on this data. Integrity mechanisms, on the other hand, deal with the prevention of altering data, either stored or during transfer, by unauthorized parties [12]. Authenticity refers to authentication mechanisms used to verify a data source. Table II gives an overview on the evaluated algorithms, their type, and which purpose they fulfill.

1) Encryption: Encryption is a fundamental technique to establish confidentiality, which can be divided into two main categories, namely i) symmetric encryption and ii) asymmetric encryption. Substitution and transposition are the key features of encryption algorithms. Those algorithms are commonly referred to as *ciphers*. Symmetric ciphers use a (pre-)shared key for both encryption and decryption, and are further classified into i) stream ciphers and ii) block ciphers. A stream cipher algorithm encrypts one bit or byte of plaintext at a time and uses an infinite stream of pseudorandom bits as the key. Block cipher algorithms, on the other hand, encrypt plaintext with fixed blocks of n-bits size at one time [13]. While block ciphers seem to be applicable to a broader range of applications and are therefore much more researched and implemented, it has been shown that stream ciphers in general perform better than block ciphers regarding CPU time [14] [13]. In this paper, we are focusing on the Advanced Encryption Standard (AES) block cipher, and the ChaCha stream cipher.
Asymmetric cryptography uses a pair of keys that are generated based on one-way functions. A one-way function easily computes any input, e.g., calculate $f(x)$, but it is hard to compute its inverse function, i.e., given the value of $f(x)$ it is hard to calculate x. Such a key pair consists of a private key which should be known to its owner only, and a public key which can be distributed and made freely available to anybody [15]. Asymmetric cryptography can be used for both, encryption/decryption and digital signatures. Encrypting a message can be done by anybody using the receiver's public key, but decryption of that message is only possible with the corresponding receiver's private key. Signatures work the opposite way. First, a message is hashed and the resulting hash then encrypted with the senders private key which will be send to one or more recipients alongside with the plaintext message. Authenticity and integrity of the message can then be verified by everyone in possession of the senders public key [16]. In recent years, a special form of asymmetric cryptography, namely elliptic curve cryptography (ECC), has gained increased attention. It is based on the algebraic structure of elliptic curves over finite fields The main advantage of ECC is that it provides equivalent security, compared to non-ECC techniques, but uses smaller keys [17]. Therefore, ECC in general becomes particularly interesting in IoT scenarios, as corresponding applications often operate on resource constrained devices. In this paper we focus on signing and verifying a message using ECC, specifically Ed25519. This technique uses SHA-512 and the elliptic curve 25519.

2) Hashing and Message Authentication Codes: The concept of hashing functions is to transform a message of arbitrary length into an output sequence of fixed length, usually much shorter than the input message. The ultimate goal for cryptographic hash functions is to create a unique value

characteristic for a specific message, commonly referred to as hash of the message. Hashing algorithms are also based on previously described one-way functions, commonly used by data protection mechanisms concerning integrity of data, or for efficient data persistence mechanisms, i.e., usage of hash tables [18]. In our experiments we look into the performance of the latest Secure Hashing Algorithm (SHA) family, i.e., SHA3. To preserve message integrity and authenticity during transfer, simply hashing a message and appending the corresponding hash is not sufficient. An adversary with knowledge about the used hashing algorithm could perform an active man-in-the-middle attack, by altering the message and calculating a new hash for that message, being then forwarded to the intended recipient. Message authentication codes (MAC) use specific techniques that offer protection against such attacks, by incorporating a (pre-)shared secret key into their generation process. The values generated from a MAC are then verified by applying the same secret key used to calculate the MAC, which poses the main difference to digital signatures [19]. MAC algorithms can be implemented using either cryptographic hash functions or cipher algorithms. In this paper we are focusing on the latter, by evaluating the Poly1305 one-time authenticator algorithm [20].

3) Key Exchange: For at least two parties to exchange confidential messages, a secret key needed for encryption and decryption must be shared among all participants. This key exchange can be seen as the weakest point, in terms of security, of symmetric cryptography. To tackle this issue, Whitfield Diffie and Martin Hellman proposed a cryptographic protocol, namely the Diffie-Hellman key exchange (DH). It leverages techniques based on asymmetric cryptography, enabling communication parties to securely exchange a common key, even if an adversary is eavesdropping the communication channel. A modern ECC variant of the DH is called Elliptic-curve Diffie-Hellman (ECDH). The ECDH protocol consists of three basic steps for two parties that want to establish a shared secret key via an insecure channel. First, each party must generate a private and public key pair based on a common base point on the same elliptic curve on the same finite field. Second, the public keys are exchanged between the two parties. Third, a common secret key is derived from calculations that take the secret key of a communication partner and the previously exchanged public key of the other partner as input [21]. In this paper we do not generate ephemeral key pairs, i.e., temporarily used key pairs, and evaluate the key generation as a separate step, because a static key generation is usually done only once when setting up a device. Furthermore, we do not evaluate step two to avoid incorrect measurements which take network latency into account that could be affected by environmental specific factors like signal interferences.

C. Experimental Setup

All measurements are based on open source implementations of each algorithm, separately compiled and deployed for each device, by using the Arduino IDE v.1.8.7 running on 64bit Linux Mint 19.1. We determine the performance of each algorithm by taking the arithmetic mean of results obtained after 100 runs, respectively. Those performance results are expressed then as *limitation* and *throughput*. Limitation corresponds to the time it takes for an algorithm to process one byte of data given in μs per byte ($\mu s/B$), while throughput describes how many bytes can be processed in one second (B/s). With provided throughput values, a trade-off value for data protection/performance can be then calculated.
Key generation for ECC asymmetric algorithms were measured separately. For AES we also measure the performance of the algorithm with different keysizes (128bit and 256bit) and different block modes (ECB and CTR), respectively. Furthermore, two specific versions (based on encryption rounds) of the ChaCha algorithm are evaluated, namely ChaCha20 and ChaCha12, both using a keysize of 256bit. From the SHA3 family, we evaluate two different versions. First, the SHA3-256, which produces a hash with 256bit in size, and second, the SHA3-512 algorithm, which produces a hash with 512bit in size. ED25519 and ECDH were evaluated by performing corresponding operations on a base64 encoded JSON file, 840B in size, which acts as typical information structure an IoT device would send to the Fog or Cloud. The performance of signing and verifying large messages mostly depends on the internally used hashing algorithm, which for ED25519 is SHA-512.

IV. Results

This section provides our experimental results on evaluated data protection mechanisms. It is followed by a discussion on the obtained values and how they can be interpreted. The results are put into corresponding categories, namely i) confidentiality, ii) integrity, iii) authenticity, and iv) key exchange. To increase readability we converted throughput rates to kB/s.

A. Confidentiality

The following tables display the results of evaluated AES and ChaCha variants. Top values in each row correspond to encryption speed, while the bottom values correspond to decryption speed. Values in brackets correspond to limitation, while values outside the brackets correspond to throughput. Table III shows our obtained results for the AES block cipher with different modes of operation and different key length, respectively. Table IV shows our obtained results for the ChaCha stream cipher with different configuration, each using a 256bit key. As expected, performance of encryption and decryption scales linearly with CPU frequency. For the ESP8266 and ESP32, CPU frequency can be adjusted easily via the Arduino IDE. This becomes particularly interesting if energy consumption is of major concern in system design. The immense increase in performance of AES ciphers on the ESP32 is most likely due to the built-in hardware acceleration for AES.

B. Integrity

Table V and VI show the results of integrity related data protection mechanisms. Top values of each cell correspond to

TABLE III
PERFORMANCE OF AES-ECB AND AES-CTR EACH WITH 128BIT AND 256BIT KEYS

	AES128-ECB	AES128-CTR	AES256-ECB	AES256-CTR
Arduino MKR1000	100.70kB/s (9.70μs/B) 52.34kB/s (18.66μs/B)	91.66kB/s (10.65μs/B) 91.64kB/s (10.66μs/B)	71.40kB/s (13.68μs/B) 36.76kB/s (26.56μs/B)	68.04kB/s (14.35μs/B) 68.04kB/s (14.35μs/B)
ESP8266@160MHz	304.58kB/s (3,21μs/B) 213.19kB/s (4.58μs/B)	288.30kB/s (3,39μs/B) 288.24kB/s (3,39μs/B)	217.53kB/s (4.49μs/B) 150.94kB/s (6.47μs/B)	211.56kB/s (4.62μs/B) 211.56kB/s (4.62μs/B)
ESP8266@80MHz	152.35kB/s (6.41μs/B) 106.63kB/s (9.16μs/B)	144.16kB/s (6.77μs/B) 144.13kB/s (6.78μs/B)	108.76kB/s (8.98μs/B) 75.47kB/s (12.94μs/B)	105.78kB/s (9.23μs/B) 105.78kB/s (9.23μs/B)
ESP32@240MHz	2,591.21kB/s (0.38μs/B) 2,595.51kB/s (0.38μs/B)	1,812.12kB/s (0,54μs/B) 1,803.75kB/s (0.54μs/B)	2,371.02kB/s (0.41μs/B) 2,363.84kB/s (0.41μs/B)	1,859.84kB/s (0,53μs/B) 1,854.87kB/s (0.53μs/B)
ESP32@160MHz	1,736.11kB/s (0.56μs/B) 1,692.85kB/s (0.58μs/B)	1,218.32kB/s (0.80μs/B) 1,212.06kB/s (0.81μs/B)	1,667.56kB/s (0.59μs/B) 1,627.60kB/s (0.60μs/B)	1,249.63kB/s (0.78μs/B) 1,246.63kB/s (0.78μs/B)

TABLE IV
PERFORMANCE OF CHACHA12-256 AND CHACHA20-256

	ChaCha12	ChaCha20
Arduino MKR1000	711.33kB/s (1.37μs/B) 709.90kB/s (1.38μs/B)	543.94kB/s (1.80μs/B) 543.07kB/s (1.80μs/B)
ESP8266@160MHz	4,045.83kB/s (0.24μs/B) 4,032.00kB/s (0.24μs/B)	3,149.09kB/s (0.31μs/B) 3,140.70kB/s (0.31μs/B)
ESP8266@80MHz	2,023.05kB/s (0.48μs/B) 2,016.13kB/s (0.48μs/B)	1,574.51kB/s (0.62μs/B) 1,570.31kB/s (0.62μs/B)
ESP32@240MHz	5,130.10kB/s (0.19μs/B) 5,113.73kB/s (0.19μs/B)	4,078.84kB/s (0,24μs/B) 4,066.63kB/s (0.24μs/B)
ESP32@160MHz	3,407.48kB/s (0.29μs/B) 3,394.71kB/s (0.29μs/B)	2,708.68kB/s (0.36μs/B) 2,701.77kB/s (0.36μs/B)

TABLE V
PERFORMANCE OF SHA3-256 AND SHA3-512

	SHA3-256	SHA3-512
Arduino MKR1000	102.24kB/s (9.55μs/B) 1,260.29μs	55.13kB/s (17.71μs/B) 1,266.31μs
ESP8266@160MHz	420.99kB/s (2.32μs/B) 307.20μs	226.17kB/s (4.32μs/B) 307.37μs
ESP8266@80MHz	210.50kB/s (4.64μs/B) 1,627.64μs	113.08kB/s (8.64μs/B) 1,626.73μs
ESP32@240MHz	712.28kB/s (1.37μs/B) 179.64μs	384.51kB/s (2.54μs/B) 179.73μs
ESP32@160MHz	473.03kB/s (2.06μs/B) 270.50μs	255.34kB/s (3.82μs/B) 270.64μs

throughput and limitation rates regarding hashing operations of each algorithm. The bottom value corresponds to the *finalize* operation of the algorithm, where internal states are resetted. As for encryption and decryption speed of previously discussed algorithms regarding confidentiality, hashing performance also scales linearly with CPU frequency. However, the ESP32 and ESP8266 are remarkably faster than the MKR1000 WiFi, which is also most likely due to hardware acceleration.

TABLE VI
PERFORMANCE OF POLY1305 MESSAGE AUTHENTICATION CODE

	Poly1305
Arduino MKR1000	282.24kB/s (3.46μs/B) 70.76μs
ESP8266@160MHz	1,811.75kB/s (0.54μs/B) 11.68μs
ESP8266@80MHz	905.93kB/s (1.08μs/B) 32.00μs
ESP32@240MHz	4,675.87kB/s (0.21μs/B) 4.83μs
ESP32@160MHz	3,105.13kB/s (0.31μs/B) 7.26μs

TABLE VII
PERFORMANCE OF ED25519 SIGNATURE AND ECDH

	ED25519	ECDH
Arduino MKR1000	0.916s (1.467s) 0.907s	0.492s
ESP8266@160MHz	0.157s (0.248s) 0.154s	0.082s
ESP8266@80MHz	0.313s (0.495s) 0.309s	0.164s
ESP32@240MHz	0.050s (0.078s) 0.049s	0.026s
ESP32@160MHz	0.076s (0.118s) 0.073s	0.039s

C. Authenticity

For the purpose of better readability, results of ED25519 and ECDH are combined into a single table. Table VII provides results regarding the performance of the signing operation (top value without brackets), verify operation (top value in brackets), and key generation (bottom value). To increase readability, we converted obtained results from microseconds to seconds. As for confidentiality and integrity related results, Ed25519 operations also scale linearly with CPU frequency. Also, hardware acceleration for ECC on the ESP32 and ESP8266 are the most likely cause for the immense perfor-

mance increase.

D. Key Exchange

Table VII shows the results of the third step of the ECDH. The performance of the first step of ECDH, i.e., key generation, is the same as for Ed25519. As for Ed25519, the same statements regarding performance scaling with CPU frequency and hardware acceleration apply for ECDH.

V. Conclusion

In this paper we present an evaluation of selected, well established data protection mechanisms, including cryptographic block and stream ciphers, hashing algorithms, message authentication codes, signature algorithms, and key exchange protocols. Specifically, we measured how those algorithms performed, in terms of computational effort, on representative state-of-the-art resource constrained IoT devices. Our measurements provide results that can aid the design and development of secure IoT systems incorporating resource constrained devices. We provide values for different limitations regarding the configuration and execution of several algorithms on certain devices. Furthermore, we provide results on data throughput rates for each device. Limitations and throughput rates can be used to calculate protection/performance trade-offs for a specific hardware configuration. Our evaluation shows that an IoT system running applications relying on certain data protection mechanisms will largely benefit from incorporating microcontrollers that come with built-in hardware acceleration for several cryptographic tasks. Especially for ECC related tasks, like signatures and key exchanges, hardware acceleration has the most significant impact. Future work will include measurements on power consumption, evaluations of asymmetric encryption algorithms, like RSA or El-Gamal, as well as network routing strategies and anonymization techniques.

References

[1] O. Vermesan and P. Friess, *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers, 2013.

[8] D. J. Rani and S. E. Roslin, "Light weight cryptographic algorithms for medical internet of things (iot)-a review," in *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*. IEEE, 2016, pp. 1–6.

[2] M. U. Farooq, M. Waseem, A. Khairi, and S. Mazhar, "A critical analysis on the security concerns of internet of things (iot)," *International Journal of Computer Applications*, vol. 111, no. 7, 2015.

[3] H. Suo, J. Wan, C. Zou, and J. Liu, "Security in the internet of things: a review," in *2012 international conference on computer science and electronics engineering*, vol. 3. IEEE, 2012, pp. 648–651.

[4] M. Dener, "Security analysis in wireless sensor networks," *International Journal of Distributed Sensor Networks*, vol. 10, no. 10, p. 303501, 2014.

[5] S. Alharby, N. Harris, A. Weddell, and J. Reeve, "The security trade-offs in resource constrained nodes for iot application," *International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering*, vol. 12, no. 1, pp. 52–59, 2018.

[6] J. Lee, K. Kapitanova, and S. H. Son, "The price of security in wireless sensor networks," *Computer Networks*, vol. 54, no. 17, pp. 2967–2978, 2010.

[7] C. Bormann, "Guidance for light-weight implementations of the internet protocol suite," 2013.

[9] S. Cirani, G. Ferrari, and L. Veltri, "Enforcing security mechanisms in the ip-based internet of things: An algorithmic overview," *Algorithms*, vol. 6, no. 2, pp. 197–226, 2013.

[10] L. Ertaul and A. Woodall, "Iot security: Performance evaluation of grain, mickey, and trivium-lightweight stream ciphers," in *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer , 2017, pp. 32–38.

[11] M. Sethi, J. Arkko, A. Keranen, and H. Back, "Practical considerations and implementation experiences in securing smart object networks," Tech. Rep., 2018.

[12] M. Schumacher, *Security engineering with patterns: origins, theoretical models, and new applications*. Springer Science & Business Media, 2003, vol. 2754.

[13] W. Stallings, *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, 2017.

[14] S. O. Sharif and S. Mansoor, "Performance analysis of stream and block cipher algorithms," in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, vol. 1. IEEE, 2010, pp. V1–522.

[15] J. Katz, A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, *Handbook of applied cryptography*. CRC press, 1996.

[16] L. Lamport, "Constructing digital signatures from a one-way function," Technical Report CSL-98, SRI International Palo Alto, Tech. Rep., 1979.

[17] M. Amara and A. Siad, "Elliptic curve cryptography and its applications," in *International Workshop on Systems, Signal Processing and their Applications, WOSSPA*. IEEE, 2011, pp. 247–250.

[18] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," in *Proceedings of the twenty-first annual ACM symposium on Theory of computing*. ACM, 1989, pp. 33–43.

[19] D. Russell, D. Russell, G. Gangemi, S. Gangemi, and G. Gangemi Sr, *Computer security basics*. " O'Reilly Media, Inc.", 1991.

[20] D. J. Bernstein, "The poly1305-aes message-authentication code," in *International Workshop on Fast Software Encryption*. Springer, 2005, pp. 32–49.

[21] D. Hankerson and A. Menezes, *Elliptic curve cryptography*. Springer, 2011.