ESKİŞEHİR TECHNICAL UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY
A- APPLIED SCIENCES AND ENGINEERING

Estuscience – Se, 2024, 25 [2] pp. 278-288, DOI: 10.18038/estubtda.1427488

RESEARCH ARTICLE

PERFORMANCE COMPARISON OF ECC LIBRARIES FOR IOT DEVICES

İsmet Kaan ÇEKİŞ [1], Armağan TOROS [2], Nimet APAYDIN [3], İlker ÖZÇELİK [4,*]

[1] Department of Computer Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Türkiye
cekiskaan@gmail.com - 0009-0001-9709-0465

[2] Department of Computer Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Türkiye
armagantoros7@gmail.com - 0009-0008-8950-7559

[3] Department of Computer Engineering, Graduate School of Natural and Applied Sciences, Eskişehir Osmangazi University, Eskişehir, Türkiye
nimetapaydin02@gmail.com - 0009-0002-2110-3255

[4] Department of Software Engineering, Faculty of Engineering and Architecture, Eskişehir Osmangazi University, Eskişehir, Türkiye
ilker.ozcelik@ogu.edu.tr - 0000-0002-2032-1640

Abstract

As the prevalence of IoT devices increases, the need for strengthening security becomes inevitable. Lightweight encryption solutions play a pivotal role, particularly in addressing security concerns associated with IoT authentication and privacy. This study concentrates on the performance evaluation of open-source and lightweight encryption libraries. Various open-source encryption libraries underwent testing on a Raspberry Pi, revealing noteworthy variations in key generation, signing, verification times, and memory usage. This research provides comprehensive assistance for the selection of encryption libraries for IoT applications. Consideration extends beyond library performance, encompassing factors such as user base and documentation quality, to deliver optimal security solutions.

Keywords

ECC,
IoT,
Resource-Constrained,
Performance Testing

Time Scale of Article

Received :31 January 2024
Accepted : 08 May 2024
Online date :28 June 2024

1. INTRODUCTION

The Internet of Things (IoT) is experiencing an explosive rise in popularity. Now with IoT devices all over the world, it is evident that this technology is rapidly transforming our lifestyle, work, and interactions with the world around us. From smart homes to industrial systems, the IoT is playing an increasingly critical role in shaping our future. As the use of IoT devices grows significantly, the importance of ensuring their security has increased. Currently, IoT devices are susceptible to vulnerabilities due to limited hardware capabilities, such as processing power and memory. These limitations prevent the use of complex security functions on these devices. Therefore, insufficient authentication and confidentiality are prominent security problems for the IoT domain. Cryptography can be used to address these issues.

*Corresponding Author: ilker.ozcelik@ogu.edu.tr

Cryptosystems can be put into two broad categories: symmetric and asymmetric. Symmetric encryption uses the same key to encrypt and decrypt data. In this approach encryption and decryption can be done quickly. However, the key must be shared with the relevant parties that need to encrypt/decrypt data. Key distribution is usually a challenging task that can lead to critical security issues like intercepted or stolen keys especially on untrusted networks. Asymmetric encryption uses two different keys called public and private keypairs. The public key is made available to anyone who wants to send encrypted data to the key owner, but only the recipient has the private key to decrypt messages. This feature addresses the key distribution problem of symmetric cryptosystems, but it also works slower than symmetric encryption Moreover, the use of currently reliable and widely used asymmetric encryption methods such as River-Shamir-Adleman (RSA) on IoT devices is not always feasible because of the hardware limitations. Elliptic Curve Cryptography (ECC) has become a promising alternative because it requires less resources [1]. It uses a smaller key size compared to other asymmetric cryptographic methods like RSA and can be used on devices with limited resources and is considered more efficient. ECC is used in a variety of applications, including secure communications [2], data encryption [3], digital signatures [4], and blockchain technology [5].

It is both safer and recommended for application developers to use well-established cryptographic libraries rather than implementing cryptographic functions by themselves. There are many open-source cryptographic libraries available online that are reviewed and used in many projects by researchers. Most of the well-known crypto algorithms were implemented in these libraries. However, it is not always an easy task to decide which one of these libraries is best for the intended application. In this paper we address this problem and present results of a heuristic study that can help choose the right crypto library for resource constrained systems.

In ECC, both different key sizes and their implementation can affect the performance of key generation signing and signature verification. Therefore, choosing an ECC library that provides a variety of key sizes, and the best performance is crucial for IoT based implementations. We found several ECC libraries that are open source and suitable for IoT implementations, including BearSSL, libecc, Mbed-TLS, OpenSSL, MIRACL and RELIC. BearSSL, a C-based SSL/TLS implementation, targets small embedded systems and specialized contexts like bootstrap code; it prioritizes efficiency by avoiding dynamic allocation. It supports elliptic curves such as SECP256R1, SECP384R1, SECP521R1, and Curve25519 for ECDHE key exchange but lacks TLS 1.3 compatibility. OpenSSL is a cryptographic toolkit designed primarily for flexible applications in common computing environments, offering compatibility with a range of platforms such as web, Windows, Solaris, Linux, Android and OSX. The toolkit includes three core components: the libcrypto library, the libssl library and a command-line utility for performing cryptographic tasks. It enables TLS 1.3 support, key parameter generation, CSR and CRL creation as well as protocol support including SSL/TLS/DTLS/QUIC, and the handling of S/MIME signed or encrypted mail. MbedTLS is crafted to seamlessly integrate with diverse environments, spanning from embedded systems such as ARM-based platforms to personal computers, smartphones, and even gaming consoles. The range of features it offers includes handling X.509 certificates, supporting SSL/TLS and DTLS protocols. MIRACL (Multiprecision Integer and Rational Arithmetic Cryptographic Library) stands out by extending capabilities beyond traditional environments, catering to highly constrained platforms like embedded systems, mobile applications, and SCADA. It supports ECC, RSA and symmetric encryption. MIRACL's multifunctionality extends to programming languages including C/C++, Go, Rust, Python, Java, JavaScript and Swift. Libecc is a C++ elliptic curve cryptography library created for maximum speed and efficiency without heap allocation and relies solely on constant global variables. It supports various signature algorithms outlined in the ISO 14888-3:2018 standard, along with other signature algorithms and ECDH primitives. RELIC is a modern research-oriented cryptographic meta-toolkit with emphasis on efficiency and flexibility. RELIC can be used to build efficient and usable cryptographic toolkits tailored for resource-constrained environments like embedded systems. Like MIRACL, RELIC also provides various symmetric and asymmetric encryption, digital signature, and hashing techniques.

These libraries allow developers to customize them to meet the specific needs of their applications. Also, they are suitable for use in resource constrained IoT devices that need to process data in real time. In this paper, we performed a comparative analysis of five ECC libraries written in C/C++. We evaluated their performance under different key lengths. Our analysis includes benchmarking each library based on its execution time for key generation signing and verification.

The rest of the paper is organized as follows. Section II examines the fundamental aspects of ECC and gives necessary background information. Section III reviews previously published articles on the subject matter. Section IV overviews the performance metrics used in this study. The results are presented in Section V and finally the paper concludes with a discussion of the findings in Section VI.

2. ECC OVERVIEW

Public Key cryptographic systems are designed using public key schemes which contain a pair of public and private keys for each user. In contrast to the public key, which is accessible to all users, the private key is unique to each user and is kept confidential. The private key remains unpredictable, even with knowledge of the public key. The security of Public Key cryptographic systems relies on the difficulty of the mathematical approach used. In the literature, there are three well-known algorithms which use three different difficult problems to implement three different approaches. These algorithms are based on the factorization of integers, the discrete logarithm problem, and the elliptic curve discrete logarithm problem. Cryptosystems designed using these algorithms include examples such as RSA, ElGamal, and ECC. RSA uses the integer factorization algorithm based on the problem of finding the prime factors of a specially selected positive integer. ElGamal public-key encryption relies on the discrete logarithm problem in a finite field, and ECC is based on the difficulty of the elliptic curve discrete logarithm problem (ECDLP) [6].

Elliptic curve cryptography was introduced by Neal Koblitz and Victor Miller in 1985. It comprises a set of points defined on an elliptic curve generated over a finite field, with a prime field size [7]. Users have the flexibility to select different elliptic curves, even when they operate within the same finite field. The field parameters employed in the design of the elliptic curve play a pivotal role in determining the key size. In comparison to alternative algorithms, the chosen parameters enable a more compact key size [8].

In general, the equation form of the elliptic curve is as follows:

$$y^2 = x^3 + ax + b$$

The set of points (x, y) in the equation represents the elliptic curve. 'a' and 'b' are constant numbers. The given equation is the simplified form of the Weierstrass equation.

ECC provides low resource consumption, small parameters, and small key size compared to other algorithms for the same security levels. Therefore, it is preferred in various areas such as authentication systems in resource-limited devices, data sharing systems, and blockchain.

3. RELATED WORK

The popularity of elliptic curve cryptography has increased in secure IoT systems because it requires less power and memory. Many researchers have studied the implementation performance of this cryptosystem.

Hannes and Manuel [9] compared the performance of Symmetric, DH/DSA/RSA and ECC. They also explored the time required for sign and verify operations on various ARM-based devices using different ECC curves. They underscored the computational requirements of ECC, emphasizing that acceptable delays rely on the specific requirements of the application. The performance results provided in the paper are contingent on factors such as enabled optimizations, key sizes, curve type and CPU speed, showcasing the importance of selecting an appropriate microprocessor based on the expected usage environment. In a study conducted by Mahto and Yadav ECC was compared [10] with three RSA variants (Basic, Chinese Remainder Theorem (CRT), Multi-prime). The authors used NIST recommendations as key sizes and compared average encryption and decryption times. The authors concluded that ECC outperforms RSA and its variants in terms of operational efficiency and security with lesser parameters. The performance of RSA and ECC cryptosystems on wireless sensor networks (WSN) was compared by Zagrouba et. al. [11]. This study evaluated energy consumption and observed the time to encrypt and decrypt messages in different key sizes (8, 64 ,256 bits). The results showed that RSA encryption times are better than ECC. However, ECC gives better performance in key size, energy consumption and decryption time.

In a study done by Gupta et. al. [12] elliptic curve cryptography was utilized to improve SSL protocol performance. They tested RSA-based and ECC-based handshakes with and without client authentication. Additionally, they compared encryption, decryption, digital signature, and verification times. These authors observed performance improvements for both workstations and hand-held devices when using ECC-based cryptosystems. They concluded that ECC offers significant performance benefits to SSL clients and servers. Koppl et al. [13] measured and compared the time intervals for the sign, verify, and key generation operations of ECDH and ECDSA algorithms for different ECC curves (NIST and BRAINPOOL). They used the OpenSSL tool on the Ubuntu operating system to measure the time it took for each process. At first, they assumed that NIST curves provided more security than BRAINPOOL curves, but then they concluded that it is not necessary to prioritize the NIST curves over other curves by considering their test results.

Pigatto et. al. [14] compared the time performance of MIRACL and RELIC libraries using two curves (having key size of 160 and 256 bits) and different message sizes (50 and 100 kb). They performed their tests on a system with a Pentium Dual-Core CPU and Ubuntu Linux installed. The experiment results showed that using a 256-bit key size to encrypt and decrypt the same message took an average of 20.9 and 10.6 seconds for the MIRACL and RELIC libraries, respectively. The authors concluded that RELIC outperformed MIRACL on average response time. In another study, Popa et. al. [15] compared modern ECC libraries (MIRACL, RELIC) and WolfCrypt. The authors performed their tests on an ARM based 32-bit microprocessor (Infineon TC297) used in the automotive industry. Also, they compared the implementation performance of ECDH, BLS, ECDSA protocols on three libraries using different curves and hash functions. The results showed that TC297 could handle ECC-based cryptosystems and the RELIC library was the fastest and provided the most configuration possibilities.

The Di Matteo et al. [16] study introduces a noteworthy advancement in the field of elliptic curve cryptography (ECC) for real-time Internet of Things (IoT) applications, focusing on the development of a secure ECC crypto processor. This work emphasized the importance of hardware acceleration in enhancing the performance and energy efficiency of ECC in IoT devices. The study has been performed both on 45 nm Silvaco and 7 nm Artisan TSMC technologies and verified on a Xilinx ZCU106 board. By supporting key ECC schemes on NIST P-256/-521 (SECP256R1/SECP521R1) elliptic curves, the research highlights the processor's capability to provide high-speed cryptographic operations with robust security through smaller key sizes. This contribution is pivotal in demonstrating the vital role of hardware accelerators for secure and scalable IoT systems, offering a complementary perspective to the software-centric discussions in the existing literature. The study conducted by Aikins-Bekoe and Hayfron-Acquah [17] delves into the efficacy of Elliptic Curve Diffie-Hellman (ECDH) for securing

Wireless Sensor Networks (WSNs), comparing it against traditional algorithms such RSA. The tests were performed by using PyCryptodome and eciespy python libraries. Their findings highlight ECC's advantage of requiring smaller key sizes for equivalent security, which is crucial for the resource-constrained environments typical of WSNs. This research underscores ECC's superiority in computational efficiency and operational speed, making it an ideal cryptographic solution for WSNs. This comparison demonstrates the practical benefits of ECC in a specific application area.

These articles concentrated on various curves within the same cryptographic library or compared two ECC libraries for a specific purpose. Our study involved evaluating the performance of six different ECC libraries, BearSSL, libecc, MbedTLS, OpenSSL, RELIC and MIRACL. We conducted our tests on Raspberry Pi. We provide commentary on the strengths and weaknesses of each library, discussing the best-suited libraries for various use case scenarios.

4. PERFORMANCE EVALUATION

In this research, our emphasis was on exploring open-source libraries for Elliptic Curve Cryptography, with a detailed examination of three key metrics: key-pair generation time, sign time, and verification time. For our analysis, we chose three curves across these libraries, namely SECP256R1(NIST P-256), SECP384R1(NIST P-384), and SECP521R1(NIST P-521). Every curve starts with "SEC" to represent "Standards for Efficient Cryptography," followed by a "P" indicating parameters over a prime field (F_p), and then a number "x" signifying the bit length of the field size P, which also determines our key sizes. Following this, there is an "R" followed by a sequence of numbers "n" indicating that this curve is the nth recommended curve with an x-bit field size in the Standards for Efficient Cryptography Group (SECG) standards.

Our tests were performed on the Raspberry Pi 4 model B with 8GB RAM. The key generation, sign, and verify operations were repeated for 200 000 iterations to assess their efficiency. The obtained results revealed noteworthy variations in the speed of these operations across different libraries.

Key-Pair Generation:

The key-pair generation time, which represents the duration to create both a private key and its corresponding public key, is a critical factor influencing the efficiency of cryptographic systems. The efficiency of this key generation impacts the performance of cryptographic systems.

Signing and Signature Verification:

The sign and verify times provide insights into the duration required for both signing a message and subsequently verifying its signature. Digital signatures play a crucial role in guaranteeing the authenticity and integrity of data. In our assessments, these execution times serve as fundamental metrics, gauging the efficiency of ECC libraries in generating digital signatures (sign) or verifying signatures (verify). A reduced execution time signifies quicker cryptographic operations, a highly desirable trait for applications requiring real-time or low-latency performance.

5. RESULTS

In our evaluations, we measured the key generation, signing, and verification times of selected libraries across various key lengths. The average processing time for each function across different key lengths for each library is presented in Table 1. Furthermore, the performance of individual libraries for key generation, signing, and verification across different key lengths is illustrated in Figures 1, 2, and 3, respectively.

Table 1. Mean Time Performance (in milliseconds) Across Three Key Length.

Operation	MIRACL	BearSSL	Libecc	MbedTLS	OpenSSL	RELIC
Key Gen.	7.51	5.69	41.76	1.39	4.01	1.37
Sign	2.76	20.54	22.76	1.48	4.21	1.54
Verify	3.25	37.43	43.53	5.75	3.30	5.16

Key generation is a crucial aspect of cryptographic operations. While MbedTLS and RELIC stands out for its remarkably fast key generation, even with shorter key lengths, BearSSL, OpenSSL, and MIRACL exhibit comparable average performance. In contrast, Libecc demonstrates a relatively higher key generation time (see Figure 1).

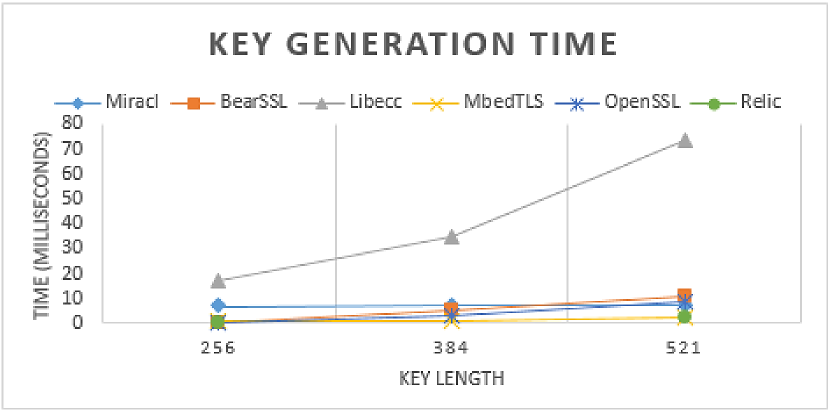

Figure 1. Key generation times (in milliseconds) of different libraries

When it comes to signing, MbedTLS and RELIC distinguish themselves yet again by delivering the fastest execution times among the listed libraries, especially with short key lengths. OpenSSL and MIRACL consistently demonstrate superior performance on average, as illustrated in Figure 2. Conversely, BearSSL and Libecc show slower signing speeds compared to their counterparts. Notably, as the key length increases, BearSSL and Libecc experience a significant decrease in performance.

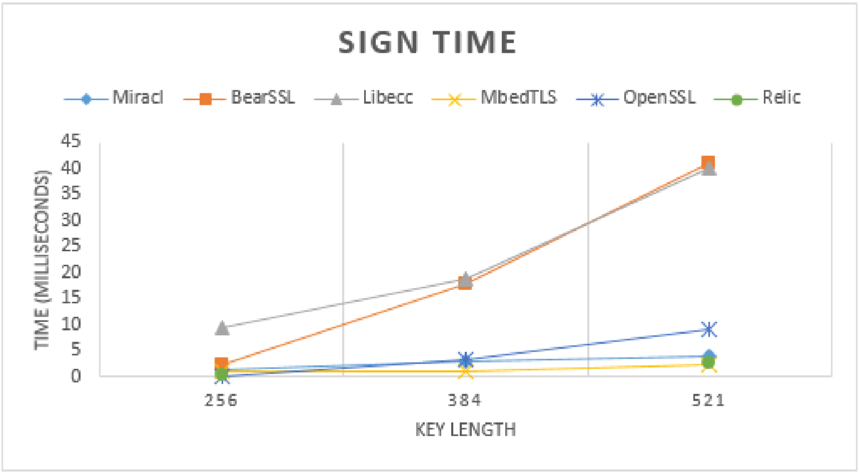

Figure 2. Signing times (in milliseconds) of different libraries

In the process of verification, MIRACL and OpenSSL libraries take the lead. While OpenSSL delivers the quickest results for short key lengths, it lags behind MIRACL on the average. MbedTLS and RELIC follow behind, as illustrated in Figure 3. Conversely, BearSSL and Libecc highlight slower verification times across the range of libraries.

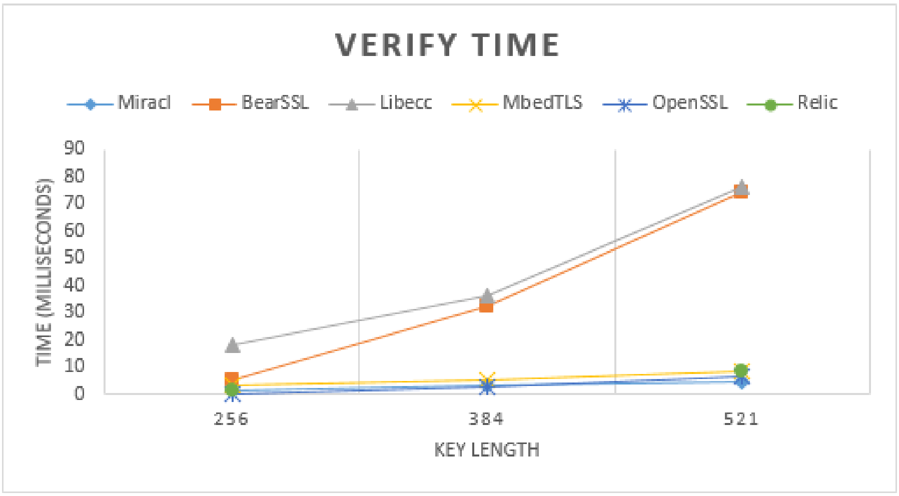

Figure 3. Verification times (in milliseconds) of different libraries.

Efficient memory utilization is a critical factor in cryptographic operations, particularly within resource-constrained environments. Table 2 presents the average memory consumption of libraries for various operations, while Figures 4, 5, and 6 illustrate the variations in memory consumption for key generation, signing, and verification across different key lengths.

Table 2. Memory Utilization Averages (in Bytes) for Three Different Key Lengths

Operation	MIRACL	BearSSL	Libecc	MbedTLS	OpenSSL	RELIC
Key Gen.	74837	74457	79889	109432	624404	75036
Sign	76885	76505	79889	131001	717221	75036
Verify	76885	76505	79889	198556	724491	75036

Upon a comprehensive analysis of memory usage across different curves, it became evident that MIRACL, RELIC, Libecc and BearSSL exhibit consistent behaviour regardless of the operation and key length, displaying efficient memory utilization within the range of 74837 to 79889 bytes. MbedTLS exhibits a significant increase in memory use, reaching up to 241,956 bytes depending on the operation. Additionally, the memory requirements of MbedTLS increase with key size, unlike other libraries. OpenSSL, while excelling in performance metrics, tends to consume notably more memory, with memory results for key generation ranging from 620,952 to 627,206 bytes and for sign-verify from 710,229 to 731,630 bytes. Due to this memory consumption, the details of Libecc and MbedTLS are difficult to distinguish in the table below; thus, a magnification of their results is provided in the speech bubble. It is also crucial to note that OpenSSL is primarily designed for general-purpose applications on traditional computing platforms, demanding more resources and potentially being unsuitable for certain IoT devices.

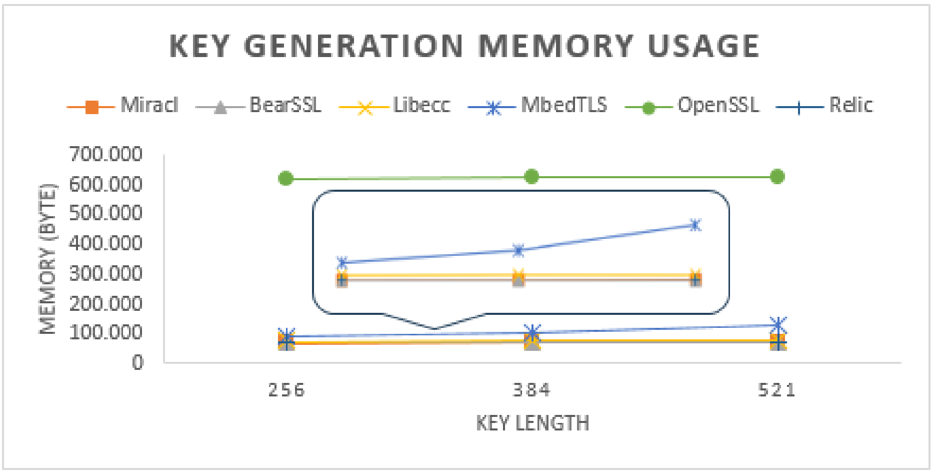

Figure 4: Key generation function memory usage (in bytes) of different libraries.

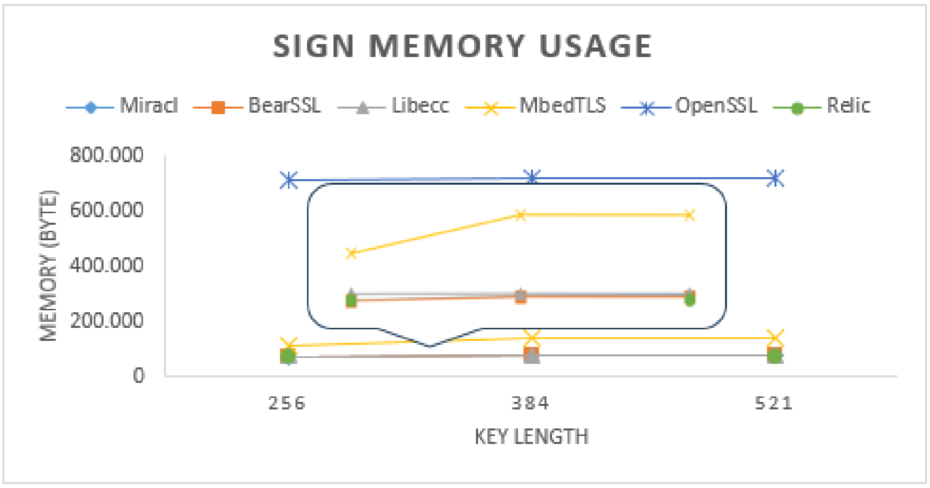

Figure 5: Sign function memory usage (in bytes) of different libraries.

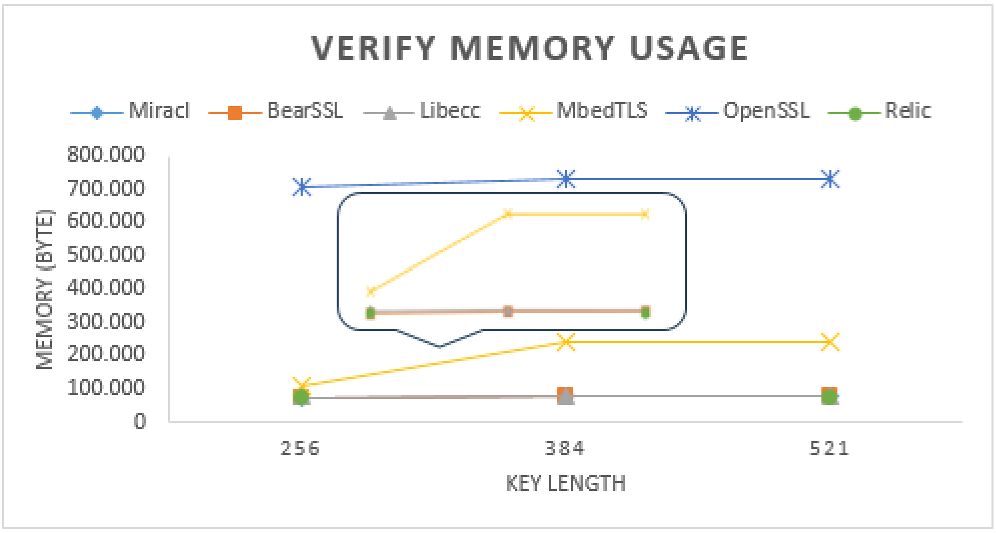

Figure 6. Verify function memory usage (in bytes) of different libraries.

6. DISCUSSION AND CONCLUSION

The rising prevalence of IoT devices necessitates improved security measures. Lightweight cryptographic solutions become crucial in addressing security concerns, particularly those related to authentication and confidentiality in the IoT domain. Given the challenges with key distribution in symmetric encryption, asymmetric methods, such as ECC, become more practical due to their lighter and additive operations when compared to resource-intensive multiplicative operations in methods like RSA. Consequently, our study focuses on performance tests of ECC-based open-source cryptographic libraries to help decide the right library for a target application.

In this study, we conducted tests on various open-source cryptographic libraries, including MIRACL, RELIC, BearSSL, Libecc, MbedTLS, and OpenSSL. The evaluations were conducted on a Raspberry Pi. Our findings underscore the importance of selecting cryptographic libraries based on platform constraints and performance requirements. For instance, OpenSSL demonstrates a salient performance for all operations, however it requires more resources (code size, memory usage) since it was designed for legacy computing environments and may not be feasible to run on certain IoT devices. The rest of the libraries evaluated in this study are optimized for resource constraint platforms. Our experiments revealed distinct performance differences in key generation, signing, verification durations of these libraries.

In our tests, RELIC and MbedTLS exhibited remarkable efficiency in key generation and signing but had an average verification performance. MIRACL showed comparatively slow key generation but balanced performance in signing and verification functions. BearSSL demonstrated efficiency in key generation but experienced longer signing and verification durations. Libecc was the slowest among all the libraries.

Memory requirements are another critical factor when deciding on a library. When we consider both performance and memory usage at the same time RELIC, MIRACL, MbedTLS and BearSSL stand out. When prioritizing key generation, notable options include RELIC, BearSSL and MIRACL. On the other hand, if signing and verification performance is paramount, MIRACL and RELIC are strong contenders. Alternatively for devices with less memory constraints MbedTLS presents a decent performance for all operations.

RELIC and MbedTLS demonstrate fast signing times and moderate verification times, rendering it a favourable choice for applications prioritizing quick signing operations. Also, RELIC, MbedTLS and BearSSL excel in key generation, and they are suitable for applications demanding rapid cryptographic key establishment. MIRACL demonstrates commendable performance for both signing and verification, making it suitable for applications where a balanced performance is critical. Therefore, MIRACL is recommended for applications requiring authentication or data integrity checks.

In this study we used a heuristic approach to compare six open-source crypto libraries. While our results provide valuable insights into performance variations between different open-source ECC libraries, aiding in the selection of an appropriate library for a specific application, we did not delve deeper into the reasons behind these variations through methods like clock cycle analysis or a meticulous code review. Future enhanced performance analysis and code review studies would help increase the efficiency of existing crypto-libraries and become a guide for future cryptosystem implementation projects.

Consequently, cryptographic libraries based on Elliptic Curve Cryptography (ECC) present viable choices for enhancing the security of IoT systems. Our findings empower informed decision-making when implementing security features on resource-constrained devices. Nevertheless, it is essential to adopt a well-rounded approach to library selection, considering factors beyond performance. Alongside performance, considerations such as user base, development status, and documentation quality should be factored in when making a library choice.

ACKNOWLEDGEMENT

This material is based upon work supported by the Scientific and Technological Research Council of Türkiye (TÜBİTAK) contract/grant number 122E517. The authors gratefully acknowledge this support and take responsibility for the contents of this report. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Scientific and Technological Research Council of Türkiye (TÜBİTAK).

CONFLICT OF INTEREST

The authors stated that there are no conflicts of interest regarding the publication of this article.

CRediT AUTHOR STATEMENT

İsmet Kaan Çekiş: Software, Validation, Investigation, Writing-Original Draft, Visualization.
Armağan Toros: Software, Validation, Investigation, Writing-Original Draft
Nimet Apaydın: Writing-Original Draft
İlker Özçelik: Funding acquisition, Project administration, Supervision, Conceptualization, Writing-Original Draft, Review & Editing

REFERENCES

[1] Yassein MB, Aljawarneh S, Qawasmeh E, Mardini W, Khamayseh Y. Comprehensive study of symmetric key and asymmetric key encryption algorithms. International conference on Engineering and Technology (ICET) 2017; 1-7.

[2] Islam T, Youki RA, Chowdhury BR, Hasan AT. An ECC based secure communication protocol for resource constraints IoT devices in smart home. In Proceedings of the International Conference on Big Data, IoT, and Machine Learning 2021; 431-444.

[3] Tawalbeh LA, Mowafi M, Aljoby W. Use of elliptic curve cryptography for multimedia encryption. IET Information Security 2013; 7(2): 67-74.

[4] Caelli WJ, Dawson EP, Rea SA. PKI, Elliptic curve cryptography, and digital signatures. Computers & Security 1999; 18(1): 47-66.

[5] Alshahrani MY. Implementation of a blockchain system using improved elliptic curve cryptography algorithm for the performance assessment of the students in the e-learning platform. Applied Sciences 2021; 12(1): 74.

[6] Cheng R, Wu K, Su Y, Li W, Cui W, Tong J. An efficient ECC-based CP-ABE scheme for power IoT. Processes 2021; 9(7): 1176.

[7] Brychta J. Benchmarks with points on elliptic curves. InProc. 25th Conf. Student Eeict 2019; 520-524.

[8] Hijawi U, Unal D, Hamila R, Gastli A, Ellabban O. Performance evaluation of no-pairing ECC-based KPABE on IoT platforms. IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT) 2020; 225-230.

[9] Tschofenig H, Pégourié-Gonnard M. Performance investigations. IETF Proceeding 2015; 92.

[10] Mahto D, Yadav DK. Performance Analysis of RSA and Elliptic Curve Cryptography. Int. J. Netw. Secur. 2018; 20(4): 625-35.

[11] Kardi A, Zagrouba R, Alqahtani M. Performance evaluation of RSA and elliptic curve cryptography in wireless sensor networks. 21st Saudi Computer Society National Computer Conference (NCC) 2018; 1-6.

[12] Gupta V, Gupta S, Chang S, Stebila D. Performance analysis of elliptic curve cryptography for SSL. In Proceedings of the 1st ACM workshop on Wireless security 2002; 87-94.

[13] Koppl M, Siroshtan D, Orgon M, Pocarovsky S, Bohacik A, Kuchar K, Holasova E. Performance Comparison of ECDH and ECDSA. 2nd International Conference on Electronics, Communications and Information Technology (CECIT) 2021; 825-829.

[14] Pigatto DF, da Silva NB, Branco KR. Performance evaluation and comparison of algorithms for elliptic curve cryptography with El-Gamal based on MIRACL and RELIC libraries. Journal of Applied Computing Research. 2011; 1(2): 95-103.

[15] Popa L, Groza B, Murvay PS. Performance evaluation of elliptic curve libraries on automotive-grade microcontrollers. In Proceedings of the 14th International Conference on Availability, Reliability and Security 2019; 1-7.

[16] Di Matteo S, Baldanzi L, Crocetti L, Nannipieri P, Fanucci L, Saponara S. Secure elliptic curve crypto processor for real-time IoT applications. Energies. 2021; 14(15): 4676.

[17] Aikins-Bekoe S, Hayfron-Acquah JB. Elliptic curve Diffie Hellman (ECDH) analogy for secured wireless sensor networks. International Journal of Computer Applications. 2020; 176(10): 1-8.