Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 230 (2023) 493–502

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Evolutionary Computing and Mobile Sustainable Networks (ICECMSN 2023)

Security Implementation on IoT using CoAP and Elliptical Curve Cryptography

Rathnakar Achary[1], Chetan J Shelke[1], Kavin Marx[1], Aishwarya Rajesh[1]

[1]*Dept. Computer Science and Engineering, Alliance College of Engineering and Design*
Alliance University, Bangalore, India
rathnakar.achary@alliance.edu.in, chetan.shelke@alliance.edu.in, mkavinbtech20@ced.alliance.edu.in, raishwaryabtech20@ced.alliance.edu.in

Abstract

IoT devices typically encompass objects or devices equipped with software and internet connectivity, allowing them to collect data. They have limited processing power, memory, and storage. These limitations can make it difficult to secure these devices, as traditional security measures may not be practical. Implementing security for IoT devices can be challenging due to these characteristics, but several steps can be taken to mitigate these challenges. Elliptic Curve Cryptography (ECC) can secure resource constrained IoT devices by providing a more efficient method of encryption and authentication than traditional methods such as RSA. In this paper, we analyzed the lightweight cryptographic algorithm ECC for securing resource-constrained devices such as IoT and evaluated its performance compared to the RSA cryptosystem. The analysis result indicates that ECC is considered to provide stronger security than RSA for the same key length. The CoAP protocol is applied to devices with limited resources, and an examination of its impact is conducted, considering three key performance metrics: CPU usage, bandwidth efficiency, and message communication latency.

Keywords: IoT, OSCORE, CoAP, Elliptical Curve Cryptography, RSA, Resource-constrained devices

1. Introduction

IoT technology has become important in industry because it allows for collecting and analyzing data from connected devices and equipment, which can be used to improve efficiency, reduce costs, and increase revenue. Some of its areas of implementation are; predictive maintenance, smart building, smart supply chain operation, and smart cities. However, the implementation of IoT technology in the industry is not without its challenges.

10.1016/j.procs.2023.12.105

Corresponding author:
E-mail address: rathnakar.achary@alliance.edu.in

Which include lack of standardization, limited resources like computing power and memory, complexity due to the heterogeneous nature of the devices, Data management and analysis, lack of software updates, scalability, and security. Security implementation is a paramount requirement in IoT for several reasons. First, IoT devices collect and transmit enormous amounts of confidential data, such as personal information and location data, which can be exploited by malicious actors if not properly secured. Second, these devices are also connected to other systems and networks, such as home automation systems and industrial control systems, which can be compromised if the IoT devices are not secure. This can result in serious consequences, such as loss of data, damage to equipment, or even physical harm. Third, IoT devices are susceptible to a wide-ranging range of cyber-attacks, including hacking, malware, and denial of service attacks, which can cause outages and service disruptions. Lastly, the increasing popularity and use of IoT devices have made them a prime target for cybercriminals. Ensuring the security of IoT devices is therefore crucial for protecting individuals, organizations, and critical infrastructures from these risks. Security implementation is a main challenge for IoT because the devices that make up the IoT are often small and have limited resources, making it difficult to implement robust security measures. Additionally, these devices are connected to the public network and may be remotely monitored, which increases the risk of hacking and other forms of cyber-attacks. Furthermore, many IoT devices are manufactured by different companies, which can make it difficult to ensure that all devices are secure. Lastly, IoT devices are often used in critical infrastructures and personal data that could be targeted by malicious actors, making security of paramount importance.

1.1. Possible security attacks on IoT Devices

IoT devices are vulnerable to various types of cyber-attacks that can be launched remotely. These attacks include; hacking, malware, denial of service (DoS) attacks, man-in-the-middle (MitM)attacks, eavesdropping, physical attacks, data breaches, device spoofing, command injection, and exploiting firmware

These are a few examples of security attacks that can be launched against IoT devices. It is important to note that IoT security threats are constantly evolving, and new attack methods are emerging regularly.

1.2. RSA algorithm and challenges of its implementation on IoT

The RSA system is a public-key cryptography method that employs separate keys for encryption and decryption. The intricacy of the RSA algorithm is rooted in the mathematical characteristics of sizable prime numbers and the challenge of factorizing extensive composite numbers. The mathematical representation of the RSA algorithm involves three main steps:

Key generation: To generate the keys select two large prime numbers, p, and q, and then compute $n = pq$. The private key is then calculated by selecting a number, d, that is relatively prime to $(p-1)(q-1)$ and then finding the modular multiplicative inverse of d modulo $(p-1)(q-1)$.

Both e and $(p-1)(q-1)$ have no common factor other than 1. Select e such that $1<e< \emptyset(n)$, where e is prime to $\emptyset(n)$. $gcd(e(d(n))=1$. The public key is then calculated by selecting a number, e, that is relatively prime to $(p-1)(q-1)$ and then finding the modular multiplicative inverse of e modulo $(p-1)(q-1)$.

Encryption: The sender uses the receiver's public key (n, e) to encrypt the message m. The ciphertext is obtained by encrypting the message m as, $c = me \pmod n$.

Decryption: The original plaintext message is regenerated by using the receiver's private key, d, and obtains the plaintext message as, $m = cd \pmod n$.

Using RSA the security level relies on the fact that factoring large composite numbers is a computationally difficult problem. As long as the prime numbers used to generate the keys are large enough, it would take an attacker an infeasible amount of time to factor n and determine the private key. The RSA cryptosystem is a widely used method for secure data encryption and digital signature, but it may not be the best choice for the security of IoT for several reasons such as, resource constraints nature of the devices, scalability for large scale deployments, latency in communication between the IoT devices, and Quantum computing threats. This could make RSA encryption vulnerable to attacks soon. Other encryption algorithms, such as ECC and AES are more lightweight and efficient, and

therefore more appropriate for these resource-constrained devices. These algorithms are more scalable and can handle many devices with less computational power and memory. The rest of this paper is structured in the following manner. Section II the literature review: This section reviews relevant literature on the topic, including previous studies and theories. In section III we explained the new IoT protocol OSCORE. Section IV analyzed the performance of RSA and ECC: This section describes the research methods used in the study, including the design and analysis procedures and in section V we summarize the main findings and conclusions of the study.

2. Literature Review

The researcher Kothmayr T et.al [1] proposed security implementation using RSA, the most used public key encryption method. The objective was to provide better interoperability and minimized overhead. However, DTLS handshake consumes a substantial amount of resources, which is an implementation challenge of this algorithm. Authors Raza et.al [2] used a combined technique, which integrates the features of DTLS and CoAP together. This allows to access the CoAP automatically. The result analysis shows a significant decrease in processing time and a reduction in packet size by DTLS compression. The authors Branchmann et.al in [3] proposed end-to-end encipherment using secure IP-based IoT for many IoT devices. This mechanism will secure the communication between HTTP and CoAP using DTLS and 6LowPAN using DTLS and Border router which acts as a proxy. A single session key is used to establish a secret connection among a group of devices using DTLS. A top-down systematic method proposed by Alghandi et.al in [4] to analyze the cyber risk in an IoT Network. For the implementation, the author analyzed DTLS and IPSec protocol. This process is not completely an optimized method hence they are not advised for resource constraint devices. In [6] the author proposed a lightweight 128-bit AES algorithm and lightweight security for CoAP. The limitation of this proposed method is that it is still suited for tracking the location of a vehicle in GPS network and other applications, this method may not be suitable for an IoT network. Authors Bhattacharyya A, et.al in [7] proposed a session security mechanism using the approach using CoAP and DTLS known as lightweight establishment of a secure session (LESS). This mechanism provides better results only in unicast security, but not for multicast security.

3. Object Security For Constrained Restful Environments (OSCORE): A Look At The New IoT Security Protocol

OSCORE is a new security mechanism for IoT, developed to address the security challenges faced by constrained devices in RESTful (Representational State Transfer) environments. OSCORE is designed to provide a lightweight, easy-to-use, and secure communication mechanism for these devices, which often have limited resources and capabilities. One of the key features of OSCORE is that it uses artificial intelligence (AI) to help secure communications between devices. Specifically, OSCORE uses machine learning algorithms to dynamically adapt to the changing security environment and to identify and mitigate potential threats. This allows OSCORE to provide strong security protections without requiring significant resources or processing power from constrained devices. OSCORE is a secure communication protocol for IoT devices, and it has the potential to significantly improve the security of these devices in the future. The CoAP protocol is specially designed for resource-constrained devices and networks, such as those found in IoT systems. OSCORE is designed to protect the request/response message communication between endpoints corresponding to the application layer. This is developed not only to secure the data part of a specific resource, but also the request method, the resource identifier, and the content format of the data part. In this way, application-relevant data and endpoint communication semantics can be protected in a way that is separate from message passing and is also light on overhead, because the size of the original CoAP message can only be 11-13 bytes.

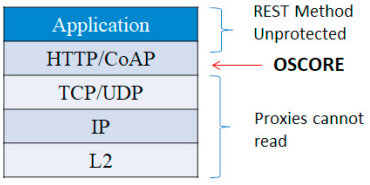


Fig.1. OSCORE Layer-wise protocols

3.1. CoAP and IoT communication security

CoAP is a communication protocol designed for use in IoT networks. It is like HTTP in that it allows for the transfer of resources and is based on the insecure UDP protocol. CoAP provides some basic security features such as message integrity, confidentiality, and endpoint identity protection using DTLS (Datagram Transport Layer Security). However, it is important to note that CoAP alone is not sufficient for securing IoT communications and additional security measures such as authentication and access control should be implemented. However, a secured endpoint communicating with IoT devices should be determined by business logic instead of transport protocols and endpoint availability as in figure (1). This is possible by protecting the message passing through different network layers, even with low-power radio devices without impacting the performance. The other security protocols CoAP support are, OAuth, IPSec, TLS(transport layer security), and JWT(JSON web token). Resource-constrained devices require a dedicated protocol for secure communications, which minimizes performance impact while flexibly supporting different trust models. A gateway in the communication path between the thing and the cloud may perform important functions to support end-to-end communication, but still cannot be trusted to access application layer data.

Design of ECC Algorithm

The design of a secured communication protocol to communicate between the IoT devices and the next higher layers of the IoT network is explained in this section. The implementation steps are presented in the algorithm as in figure (2) which represents the key generation (private and public) and data encryption and decryption to secure CoAP.

Key generation and message encryption using ECC.

In Secure CoAP to communicate the data between IoT networks, we need message encryption by using a public-private key pair. Let `G` be a point on an elliptic curve as in figure (2)A point within the ECC is regarded as a public key, while its corresponding private key is represented as a scalar value. Use the following steps to generate the key pair.

1. Choose a suitable elliptic curve and a finite field to work with. Let the curve be over the field $GF_{(p)}$ can be used. In which p is a prime number.
2. Choose point `G` on the curve as the generation point. This point is publicly known and is used to generate all other points on the curve.
3. Choose a random integer between 1 and the order of `G` and consider it as a private key.
4. Calculate the public key `Q = d*G`, which is a point on the curve. Here both `d` and `Q` form the public-private key pair.
5. ***Encryption and decryption***
6. ECC is an end-to-end public key cryptosystem to secure the information. The public key `Q` and a random number `k` are used to encrypt the message `m`. The following steps can be taken to cipher a message `m`, as explained in the following steps.

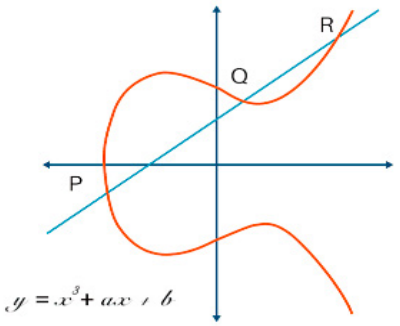


Fig. 2. Elliptical Curve

ECC Algorithm for key generation

1. SA request to RD for public key
   SA → RD: KU_{RD}
2. RD generates Private and Public key $[KP_{RD}, KU_{RD}]$ using ECC
3. RD communicate KU_{RD} to SA
   RD → SA: KU_{RD}

Message Encryption and Decryption

4. SA encrypt the message using a public key and send ciphertext to RD
   SA→ RD: CT
5. RD decrypt the CT with private key KP_{RD}

Fig. 3. ECC Algorithm for key generation and encryption

Encryption.

1. Calculate the point `R = k*G`, where `G` is the generator point.
2. calculate the point `S = k*Q`, where `Q` is the receiver's public key.
3. Calculate the ciphertext `c = (R, S, m)`.

Decryption

1. Calculate the point S_1 `= c[1]-c[0]*d`, where `d` is the receiver's private key and `c[0]` and `c[1]` are the two points in the ciphertext.
2. Calculate the message m_1 `= c[2]*S1^(-1)`, where `c[2]` is the message and S_1`^(-1)` is the inverse of S_1 modulo the order of `G`.

4. Comparative Analysis -ECC vs RSA

RSA and ECC are two popular algorithms used in public-key cryptography. Figure (4) represents some of the main differences that are compared here.

Table.1 Comparison of ECC and RSA in terms of key size

Symmetric key Size (bits)	RSA Size (bits)	ECC key Size (bits)
80	1024	160
112	2048	224
128	3072	256
192	7680	384
256	15360	521

Key size: The smaller key size of ECC will enable it to perform faster than RSA to achieve an identical security level. If the ECC key size is 256 bits, the equivalent key size in RSA is 3072 bits in terms of security. This means that ECC is more efficient in terms of key size and is suitable for applications that need to exchange larger amounts of data.

Performance: The performance of ECC is much better than RSA encipherment, especially for smaller key sizes. However, RSA has better performance for larger key sizes.

Implementation: The cryptanalysis of RSA is due to the large key size and factorization of these large composite numbers, while in ECC it is due to the complexity of finding the discrete logarithm of a point on an elliptic curve. As a result, ECC may be more suitable for implementation on constrained devices, such as smart cards and embedded systems.

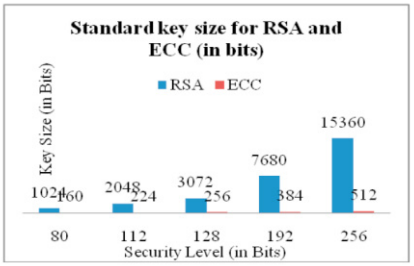


Fig. 4. Security Level (in bits)

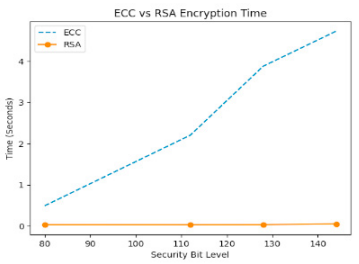
Security: Both RSA and ECC are considered secure if implemented properly. However, ECC has the potential to be more secure due to its complexity of solving the elliptic curve discrete logarithm challenge, as compared to the difficulty of factoring large composite numbers in RSA.

ECC is generally faster and more efficient than RSA, especially for smaller key sizes. However, RSA may be faster for larger key sizes and is widely used in many applications. Both algorithms are secure and suitable for different use

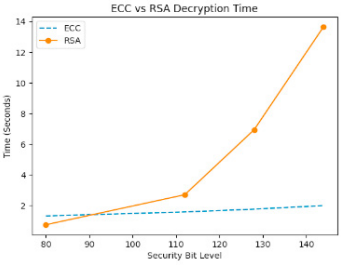
cases.

Table. 2. RSA vs ECC: Encryption and Decryption for 8-bit input

Input = 8 bits						
Security Bit-level	**Encryption**		**Decryption**		**Total Time**	
	ECC Enc. Time	**RSA Enc. Time**	**ECC Dec. Time**	**RSA Dec. Time**	**ECC Total Time**	**RSA Total Time**
80	0.4885	0.0307	1.3267	0.7543	1.8152	0.7850
112	2.2030	0.0299	1.5863	2.7075	3.7893	2.7375
128	3.8763	0.0305	1.7690	6.9409	5.6453	6.9714
144	4.7266	0.0489	2.0022	13.6472	6.7288	13.6962



(a)



(b)

Fig. 5 (a) and (b). RSA vs ECC Encryption and Decryption

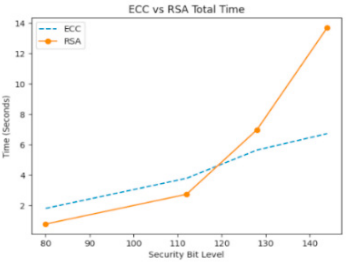


Fig. 6. RSA vs ECC Processing Time

Encryption and Decryption of 8-bit data using ECC and RSA

ECC and RSA are two different algorithms used for encryption and decrypting data. The implementation of ECC is mainly related to the elliptical curve and its algebraic structure it, while RSA is based on the mathematical concept of prime factorization. Both ECC and RSA can be used for symmetric and asymmetric encryption, but they differ in their performance, security level, and key size.

64 bits – Encryption, Decryption, and Total Time (in seconds)

From the analysis result the security level of ECC is better than RSA for a given key size, due to the inherent difficulty of the underlying mathematical complexity. However, ECC requires less computation and has faster encryption and decryption times compared to RSA, especially for smaller key sizes. For 8-bit and 64 bits, ECC may have significantly faster encryption and decryption times compared to RSA. However, as the key size increases, the difference in performance between the two algorithms becomes less significant. The performance of an encryption algorithm depends on various factors, including the hardware and software being used, the size and complexity of the data being encrypted, and the specific implementation of the algorithm. As such the relative performance of ECC and RSA may vary in different scenarios as in Tables 2 and 3 and Figures 5 and 7 respectively by taking 64 bit data for encryption and decryption. The physical network setup for the implementation of CoAP protocol is shown in Figure (8). The network includes components such as a network switch which will isolate the network from the rest of the Internet, and a Raspberry Pi working on a server. This IoT device is connected to multiple sensors and fetches data from these sensors. Network traffic is emulated by a network emulator, which is a virtual machine, and a broker integrates the functionalities of the CoAP client and server.

Table 3. Time for Encryption and Decryption

Input = 64 bits						
Security Bit-level	**Encryption**		**Decryption**		**Total Time**	
	ECC Enc. Time	**RSA Enc. Time**	**ECC Dec. Time**	**RSA Dec. Time**	**ECC Total Time**	**RSA Total Time**

80	2.1685	0.1366	5.9099	5.5372	8.0784	5.6738
112	9.9855	0.1635	6.9333	20.4108	16.9188	20.5743
128	15.0882	0.1672	7.3584	46.4782	22.4466	46.6454
144	20.2308	0.1385	8.4785	77.7642	28.7093	77.9027

Given this, the logical network scenario segregates this network segment, compelling network traffic between servers, clients, and brokers to traverse the network emulator. The security requirement of the message transmission process using CoAP protocol is achieved by using TLS cryptosystem. The analysis of CoAP is performed only after configuring the network scenario. This implementation covers both `client.c` and `server.c` for CoAP client and server respectively. The CoAP server provides access to two resources, 'time' and 'asyns,' enabling the comparison of piggybacked and separate responses. Specifically, 'asyns' serves to investigate the time difference between sending the ACK and sending the separate response. When the request for these two resources is received, the sensors connected to the IoT must read the value. This data is received by the CoAP server using TLS 1.2 cyber suit to encrypt the message.

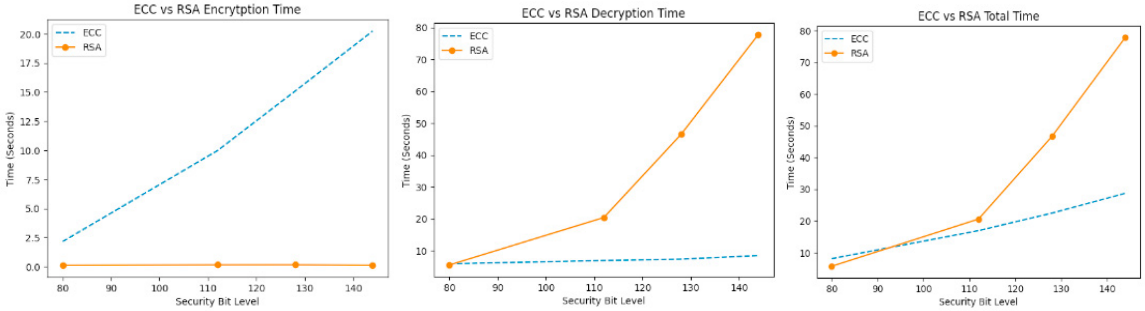


Fig. 7 (a) and (b). RSA vs ECC Encryption, Decryption and Total Time for 64-bit input

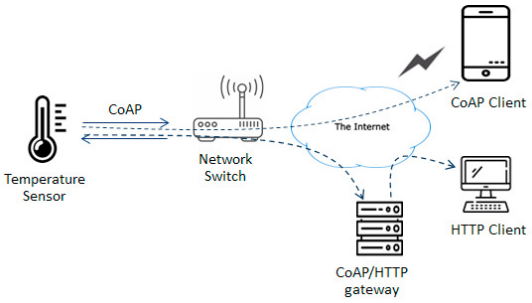


Fig.8 Implementation of CoAP Protocol

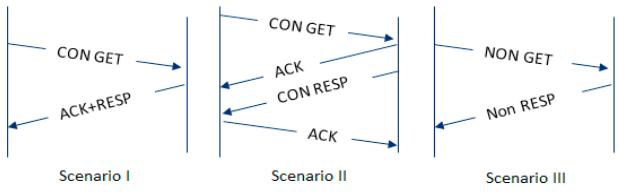


Fig. 9. Scenarios representing CoAP Information retrieval

Analytical Study of CoAP Protocol and Result Analysis

The performance of the CoAP protocol on a network with no packet loss determines the amount of data needed to retrieve sensor information without encryption. Considering this, the table below presents the count of packets and the data shared between the client and server. In all cases, the message communication delay is almost the same. The total delay is due to the delay in capturing sensor data and the time taken to exchange the data in the network. The IoT used in the experimentation functions as a server, it takes about 720 msec to read the data from the sensor. CoAP uses UDP protocol for message communication, a loss of message may be detected at the application layer. This is performed in a confirmed mode and the possible scenarios I, II, and III represent the CoAP information retrieved is shown in Figure (9).

Scenario	I	II	III
Packets	2	4	2
Bytes	134	258	134

In this operational mode, the retransmission of ACK_TIMEOUT occurs after a two-second interval following the transmission of the CON message. If the timer expires without receiving an ACK, it is transmitted again with a level of randomness introduced after ACK_RANDOM_FACTOR to mitigate potential collisions. The likelihood of packet loss resulting from network congestion is expressed as the probability that a message is received successfully, denoted as 'q' and computed as 'k = 1 − p.' Additionally, accounting for the time required for sensing, reading the sensor, and network RTT, the usual delay in receiving a response when employing a CON message is as follows:

$$\text{Delay} = T_{sensing} + k^2 + RTT + (1-k^2)k^2 \times (RTT + ACK_{TIMEOUT}) + (1-k^2)^2 + k^2 + (2RTT + ACK_{TIMEOUT}) + (1-k^2)^3 + k^2(3RTT = +ACK\_TIMEOUT) + \ldots.$$

Neglecting the RTT against *T_sensing* and ACK_TIMEOUT and if this sum has infinite terms we have.

$$\text{Delay} = T_{sesning} + k^2 + ACK_{TIMEOUT} + \sum_{n=1}^{\infty} n(1-k^2)^n$$

$$\text{Considering } \sum_{n=1}^{\infty} nx^n = ACK\_TIMEOUT \left[\frac{1-k^2}{k^2}\right]$$

If *T_sensing* = 720msec ACK_TIMEOUT = 2000msec

The delay for different probability of packet loss p is shown in table (4)

Table 4. Protocol Delay against probability of packet loss

T_Sensing = 720 msec ACK TIMEOUT = 200msec	
P	**Delay msec**
0	720
0.05	936
0.1	1,189
0.15	1,488
0.2	1,845

The performance analysis of CoAP protocol when applied to a resource-constrained device such as IoT, obtained from the experimental results are analysed focusing on the following three performance indicators, such as CPU utilization, bandwidth efficiency and latency of message communication. CPU utilization is an important indicator of performance in two different aspects. CPU utilization is important for finding the average number of packets employed per message communication. It is also an important indicator for energy efficient message communication. Higher the number of CPU cycle implies a large amount of energy consumption; this also requires larger processing capabilities. The energy consumption is a fixed component per packet transmitted but it varies as several bytes packet transmission due to retransmission problems. CPU utilization varies by adopting different types of ciphering modes. Among the three scenarios in scenario III there is a decrease in the number of CPU cycles. This is due to the increased packet loss and retransmission of the packets. So, the CPU usage has reduced in the server side.

The bandwidth utilization depends on the total number of bytes transferred per message communication. Bandwidth utilization is analyzed based on the three scenarios I, II, and III. In scenarios I and III the results obtained are practically indistinguishable for a secured communication both in PSK mode and PKI mode, whereas in scenario II loss rate increases as we change from no securing-to-securing mode. The experimental results obtained are shown in the table.

CoAP Bandwidth Utilization			
Scenario	**No Securing (Bytes)**	**Securing (Bytes)**	
		PSK	**PKI**
I, III	134	1769	2299
II	258	1854	2453

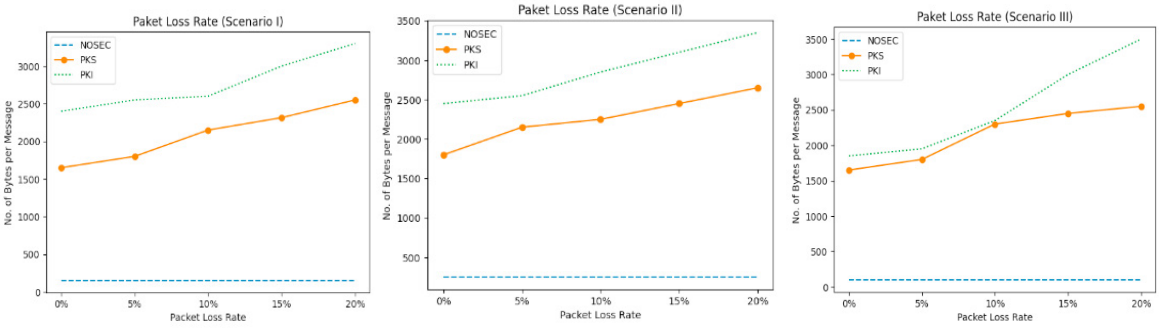


Fig.10 Bandwidth usage by Scenario I, II, III

The energy consumption in scenario I is more compared to scenarios II and III. The distinction between Scenario I and Scenario II lies in Scenario I's need for an immediate response, which entails extra server-side effort. In contrast, the contrast with Scenario III arises from the absence of a relationship, eliminating the need for a retransmission timer and managing large number of requests. The greater number of retransmissions in this context will raise the energy consumption of the network interface, consequently affecting the overall energy consumption.

5.1 CoAP Latency

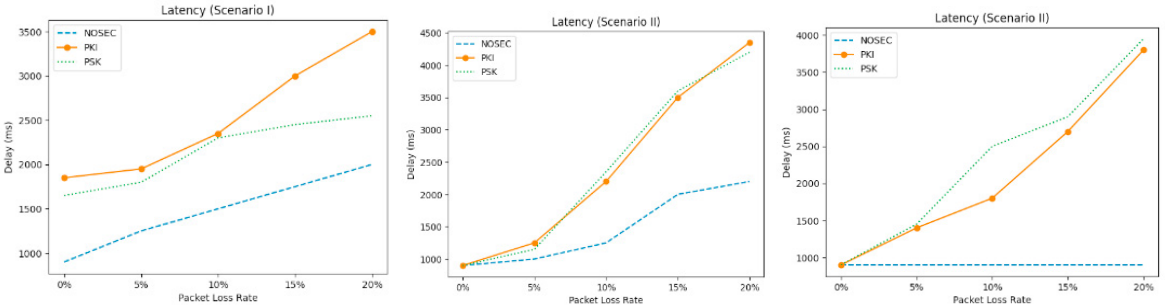


Fig.11 Time delay per Message Transmission Scenario I,II, III

Latency is assessed in the experiment by measuring the time gap between the initial and final packet exchanges. In the case of IoT or resource-limited devices, this duration encompasses the time taken by an IoT device to capture the temperature data. The latency varies across the three scenarios mentioned. In the case of lossless networks, scenario II the average delay is 720.02msec under no securing mode and 723.1432 and 728.2249msec for PSK and PKI modes respectively. In the non-secure scenarios, I and II exhibit similar behavior as the loss rate rises. However, the time delay in Scenario II is slightly higher than that in Scenario I due to a greater number of packets per message. In the case of PKS and PKI encryption modes in Scenarios I and II, they follow an alike pattern as there is a rise in the loss rate, being more affected by losses compared to the non-secured scenario.

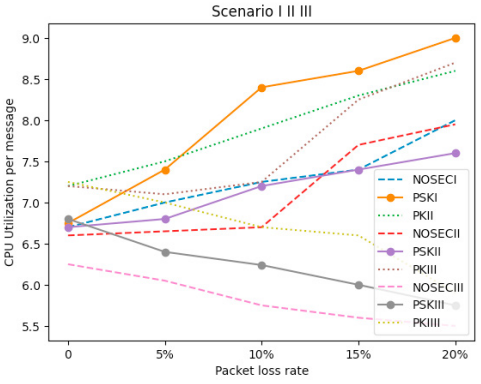


Fig. 12 CPU Usage Scenario I, II, III

5. Conclusion

To ensure the security of CoAP protocol in IoT, two prominent protocols proposed are DTLS and IPSec. This article explores the analysis and the provision for the implementation of CoAP. The analysis highlights the point that CoAP did not fulfil the security requirements of IoT devices and networks of IoT devices. In addition to this application of DTLS and IPSec, may not be possible due to the resource limitations in IoT devices. The article, therefore, claims the need for new trivial, security approaches such as elliptical curve cryptography for the secure version of CoAP. Comparing the ECC and RSA algorithms related to performance, speed of operation, and security level, the experimental result proves that ECC is much suited for low-resource devices such as IoT with smaller key sizes for the same level of security.

References

[1] Kothmayr T., "Security Architecture for Wireless Sensor Networks Based on DTLS," M.S. Thesis, the University of Augsburg, 2011.
[2] Raza S., Shafagh H., Hewage K., Hummen R., and Voigt T., Lithe: "Lightweight Secure CoAP For The Internet Of Things," IEEE Sensors Journal, vol.13, no.10, pp. 3711-3720, 2013.
[3] Brachmann M., Garcia-Morchon O., and Kirsche M., "Security For Practical CoAP Applications: Issues And Solution Approaches," Technical Report, 2011
[4] Alghamdi T., Lasebae A., and Aiash M., "Security Analysis of The Constrained Application Protocol In The Internet Of Things," in Proceedings of Second International Conference on Future Generation Communication Technology, London, pp. 163-168, 2013.
[5] Ukil A., Bandyopadhyay S., Bhattacharyya A., Pal A., and Bose T. "Lightweight security scheme for IoT applications using CoAP," International Journal of Pervasive Computing And Communications, vol. 10, no. 4, pp. 372-392, 2014
[6] Bhattacharyya A., Bose T., Bandyopadhyay S., Ukil A., and Pal A., "LESS: Lightweight Establishment of Secure Session: A Cross-Layer Approach Using CoAP and DTLS-PSK Channel Encryption," in Proceedings of IEEE 29th International Conference on Advanced Information Networking and Applications Workshops, Gwangiu, pp. 682-687, 2015.
[7] Liu Yuxuan Research on FPGA-based High-performance Elliptic Curve Cryptography Acceleration Technology>, 2021-05, Hefei University of Technology.
[8] Gao Wei, Luo Yixuan, Li Jiakun, Wu Haixia High-Performance Hardware Implementation of Elliptic Curve Cryptography Point Multiplication over GF(p), 2021-09, Beijing Institute of Technology, Transaction of Beijing Institute of Technology.
[9] Wei Wei, Chen Jiazhe, Li Dan, Zhang Baofeng Research on the Bit Security of Elliptic Curve Diffie-Hellman, 2020-04-24, Journal of Electronics & Information Technology.
[10] C. A. Lara-Nino, A. Diaz-Perez, and M. Morales-Sandoval, ''Elliptic curve lightweight cryptography: A survey,'' IEEE Access, vol. 6, pp. 72514–72550, 2018, doi: 10.1109/ACCESS.2018.2881444.
[11] P. Gupta, D. K. Verma, and A. K. Singh, ''Improving RSA algorithm using multi-threading model for outsourced data security in cloud storage,'' in Proc. 8th Int. Conf. Cloud Comput., Data Sci. Eng. (Confluence), Jan. 2018, pp. 14–15.
[12] A. Rawat, K. Sehgal, A. Tiwari, A. Sharma, and A. Joshi, ''A novel accelerated implementation of RSA using parallel processing,'' J. Discrete Math. Sci. Cryptogr., vol. 22, no. 2, pp. 309–322, Feb. 2019, doi: 10.1080/09720529.2019.1582864.
[13] X.-L. Huang, Y.-X. Dong, K.-X. Jiao, and G.-D. Ye, ''Asymmetric pixel confusion algorithm for images based on RSA and Arnold transform,'' Frontiers Inf. Technol. Electron. Eng., vol. 21, no. 12, pp. 1783–1794, Dec. 2020, doi: 10.1631/FITEE.2000241.
[14] RSA-232 Number Has Been Factored—HBM PAH [Internet]. Accessed: Oct. 11, 2021. [Online]. Available: https://www.inm.ras.ru/math_center_en/rsa-232-number-has-been-factored-5/
[15] S. Shin, K. Won, and S. Shin, ''Size efficient pre-processed symmetric RSA for wireless body area network,'' ACM SIGAPP Appl. Compute. Rev., vol. 20, no. 1, pp. 15–23, Apr. 2020, doi: 10.1145/3392350.3392352.
[16] A. S. V. Nair and R. Achary, "Social Engineering Defender (SE.Def): Human Emotion Factor Based Classification and Defense against Social Engineering Attacks," *2023 International Conference on Artificial Intelligence and Applications (ICAIA) Alliance Technology Conference (ATCON-1)*, Bangalore, India, 2023, pp. 1-5, doi: 10.1109/ICAIA57370.2023.10169678.
[17] R. Achary, R. R, R. K and P. V, "Effect of Temperature and Relative Humidity on Onion farms and its Monitoring by using IoT Based Smart Farming System," *2022 International Conference on Communication, Computing and Internet of Things (IC3IoT)*, Chennai, India, 2022, pp. 1-6, doi: 10.1109/IC3IOT53935.2022.9767884.
[18] C. J. Shalke and R. Achary, "Social Engineering Attack and Scam Detection using Advanced Natural Langugae Processing Algorithm," *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2022, pp. 1749-1754, doi: 10.1109/ICOEI53556.2022.9776697.
[19] R. Achary and C. J. Shelke, "Fraud Detection in Banking Transactions Using Machine Learning," *2023 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, Bengaluru, India, 2023, pp. 221-226, doi: 10.1109/IITCEE57236.2023.10091067.