MINISTRY OF EDUCATION AND TRAINING

FPT UNIVERSITY

Capstone Project Summary

A study on End-to-End encryption in IoT by using Elliptic Curve Cryptography

GSP22IA05	
Group member	Huynh Phuc An - SE140698 Nguyen Le Bao Truong - SE140940 Nguyen Tuan Khoi – SE140949
Supervisor	Nguyen Tan Cam
Ext Supervisor	
Capstone Project code	SP22IA09

- Ho Chi Minh, 04/2022 -

Table of Contents

Table of Figures

Table of Tables

CHAPTER 1: INTRODUCTION

1.1. Project Information

Project name: A study on End-to-End encryption in IoT by using Elliptic Curve Cryptography

1.2. The Participants

1.2.1. Supervisor

Table 1. Supervisor

Full name	Phone	Email	Title
Nguyen Tan Cam	0909332547	camnt5@fe.edu.vn	Dr.

1.2.2. Team Members

Table 2. Team members

Full name	Student code	Phone	Email	Role in group
Huynh Phuc An	SE140698	0968402802	anhpse140698@fpt.edu.vn	Leader
Nguyen Le Bao Truong	SE140940	0796731459	truongnlbse140940@fpt.edu.vn	Member
Nguyen Tuan Khoi	SE140949	0349938240	khointse140949@fpt.edu.vn	Member

1.3. Actual IA problem

1.3.1. Overview

As the incoming 4.0 industry revolution, the growth of IoT has been accelerated every day. Along with the development, there are certainly a lot of questions about IoT security problems. The security solutions for IoT devices have been concerned and researched for some time.

One of the most alarming security problems of IoT systems is the Lack of encryption techniques. When communicating in plain text, the exchanging information between IoT devices or service servers may be impacted by a Man-in-the-Middle attack. These data can be captured for sensitive information or intentionally altered the communication.

As IoT systems and devices sometimes have high requirements of operating speed and small data storage, choosing an effective encryption algorithm should be concerned.

1.3.2. The statistic:

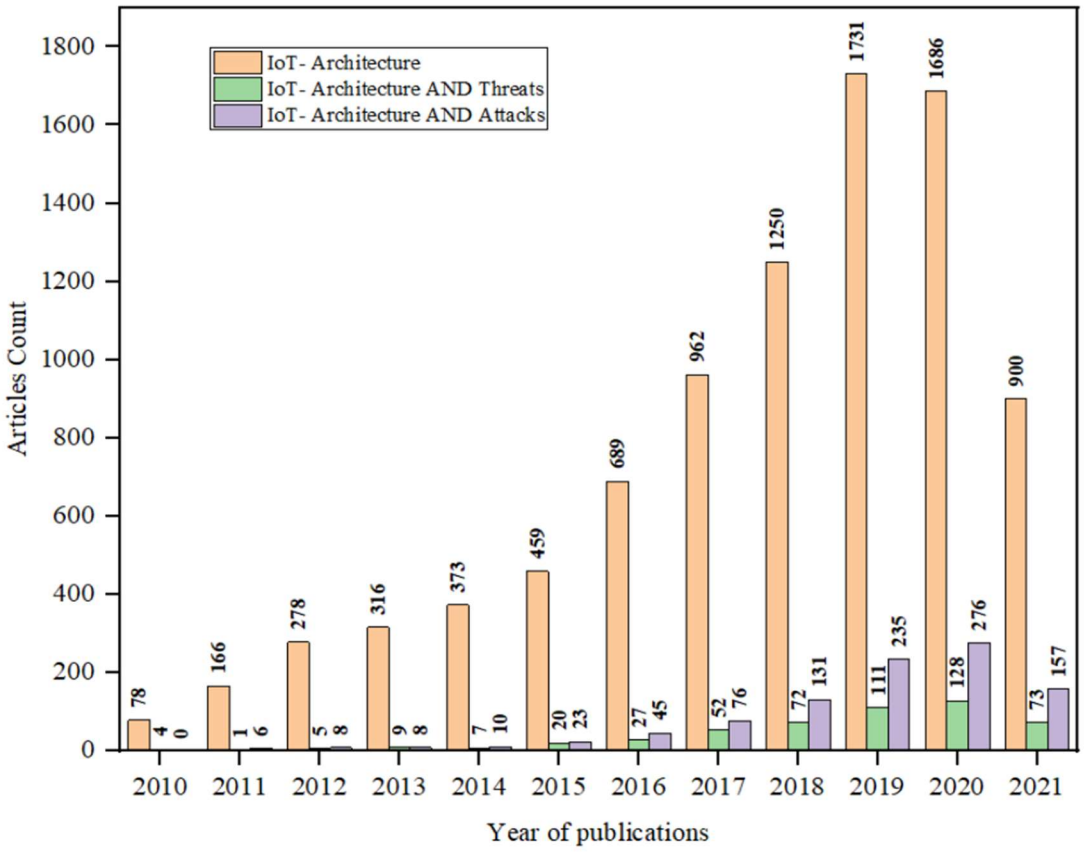

Figure 1. Literature statistics on IoT architecture, IoT architecture and threats, and IoT architecture and attacks[1]

The number of topics about IoT is growing at high speed in the past few years, but only a few of them shows their concern about the vulnerability of being attacked.

Less than 15% of the IoT articles talk about the risks and attacks on IoT systems. The security solutions for IoT devices have been concerned. But as Figure 1 shows, it is not enough to say that we are safe when using IoT.[2]

1.3.3. Solution

Applying encryption for IoT systems is always a real challenge. Most of the popular encryption techniques that have been used do not satisfy the requirement of IoT systems. For example, AES will provide high operating speed but secure strength, while RSA meets the security requirements but it needs large-sized keys and time-consuming calculations.

Meanwhile, Elliptic Curve Cryptography provides an equivalent level of encryption strength as the RSA algorithm with a shorter key length. As a result, the speed and security offered by ECC are faster than RSA does.

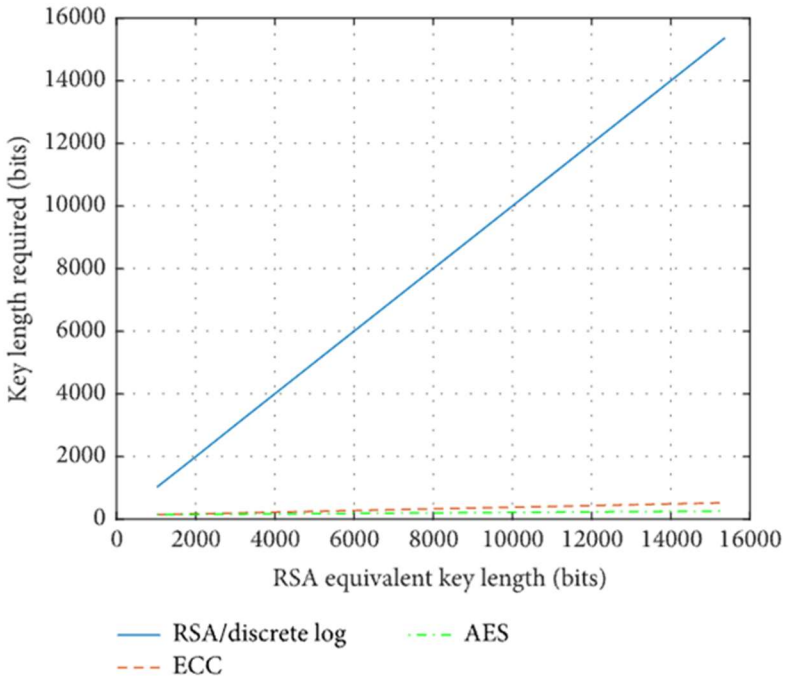

Figure 2. Effectiveness comparison between RSA, ECC and AES [3]

A 384-bits key in ECC is strong enough to protect the US government's top secret. Comparing to RSA, which needs a 7680-bits key for an equivalent encryption strength, ECC will be an effective algorithm to encrypt IoT data.[4]

CHAPTER 2: IA PROJECT MANAGEMENT PLAN

2.1. Problem Setting

2.1.1. The Current Situations

In the development, IoT systems are now aiming to the full-automation operations. Machine-to-Machine communications play an important role to achieve complete automation. If attackers are able to retrieve and exploit these communications, it would be hard for the machines themselves to detect and defend against the threats. An IoT system that lacks encryption will be an easy target for hackers to break through.

2.1.2. The Proposed Solution

Securing IoT products is always an issue for every organization working in this field. Our team has researched and developed a demo IoT system that is secured by Elliptic Curve Cryptography in order to evaluate the effectiveness of the algorithm within IoT services. Then make a comparison with other cryptographic algorithms.

2.1.3. Boundaries of the Solution

At first, we developed virtualized Raspberry Pi machines using VMWare Workstation. After development, the communication within our system will be secured by ECC and others cryptographic algorithms.

After we make sure that the development works well on Raspbian OS, the code will be built on our Raspberry Pi 3 and Raspberry Pi Zero for realistic data. We measure and calculate the effectiveness of all the algorithms on the environment devices, visualize the comparison, and make the decision if ECC is suitable to be used to secure IoT systems.

2.1.4. Development Environment

We are using the VMWare Workstation for the development of the virtual IoT system.

We used 2 IoT devices: Raspberry Pi 3 and Raspberry Pi Zero as measurement environment. Raspberry Pi 3 – have a higher CPU and Memory size – plays a role as a rich-resourced environment. Meanwhile, Raspberry Pi Zero is used as a low-resourced environment.

By using Golang and VS code, we also building CLI for developing test cases, and so does the cryptography functions.

For task management, our team uses Trello to split the tasks and manage deadlines. For code management, we used GitHub.

Table 3. Tools and Techniques

No	Tools	Function
1	Go Language	Used to build CLI and cryptography functions
2	VS Code	Development IDE
3	Raspbian OS	An OS designed for IoT systems control.
4	Power measuring unit	Used to measure the power usage of the devices during their algorithms execution
5	Highcharts	A JavaScript-based tool used to visualize the comparison charts
6	Benchmark	Go Language feature for measuring processing time and resources
7	GitHub	Code management
8	Trello	Task management
9	VMWare Workstation	Development of the virtual IoT system

2.2. Researching

Table 4. Researching

Description	Each member researched for the latest information about MQTT, Raspberry Pi, ECC and Benchmarking
Distribution	Researching MQTT, Raspberry Pi, ECC and Benchmarking
Resources	Knowledge, documents from the Internet.
Dependencies and Constrains	Collect appropriate information involving the project's purpose.

CHAPTER 3: ELLIPTIC CURVE CRYPTOGRAPHY RESEARCH

3.1. Elliptic Curve Arithmetic

3.1.1. Elliptic Curve

In cryptography a shortened and simplified equation is used for more efficiency:

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

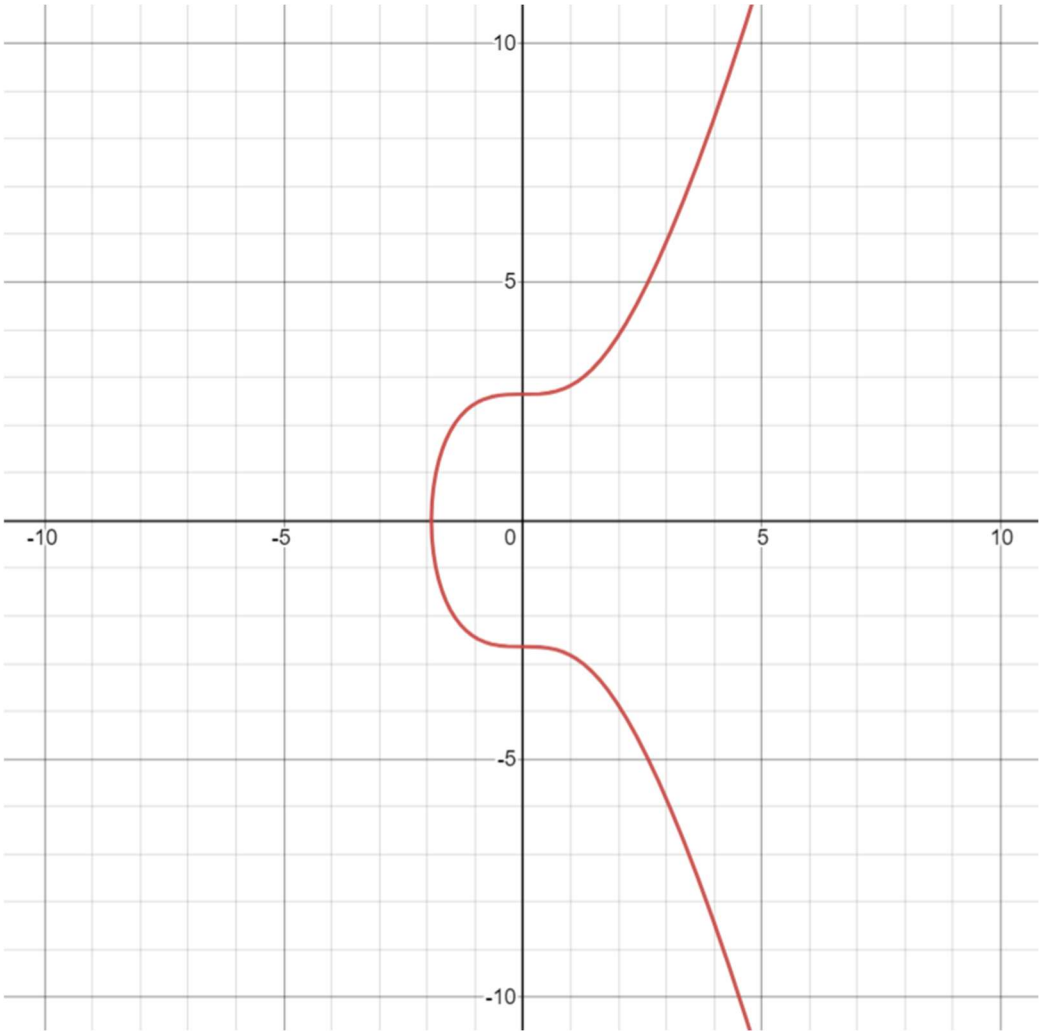

Figure 3. Curve $E : y^2 = x^3 + 7$ (Having a = 0 and b = 7)

Although the equation looks simply, it is more effective than the complex one because it will provide faster generation and calculation while being strong enough to protect the secret messages.

3.1.2. Adding and Multiplying Points

It is possible to add two points on the elliptic curve and the result is a different point. This operation is called EC point addition. If we add a point P to itself, the result is:

$$P + P = 2 * P$$

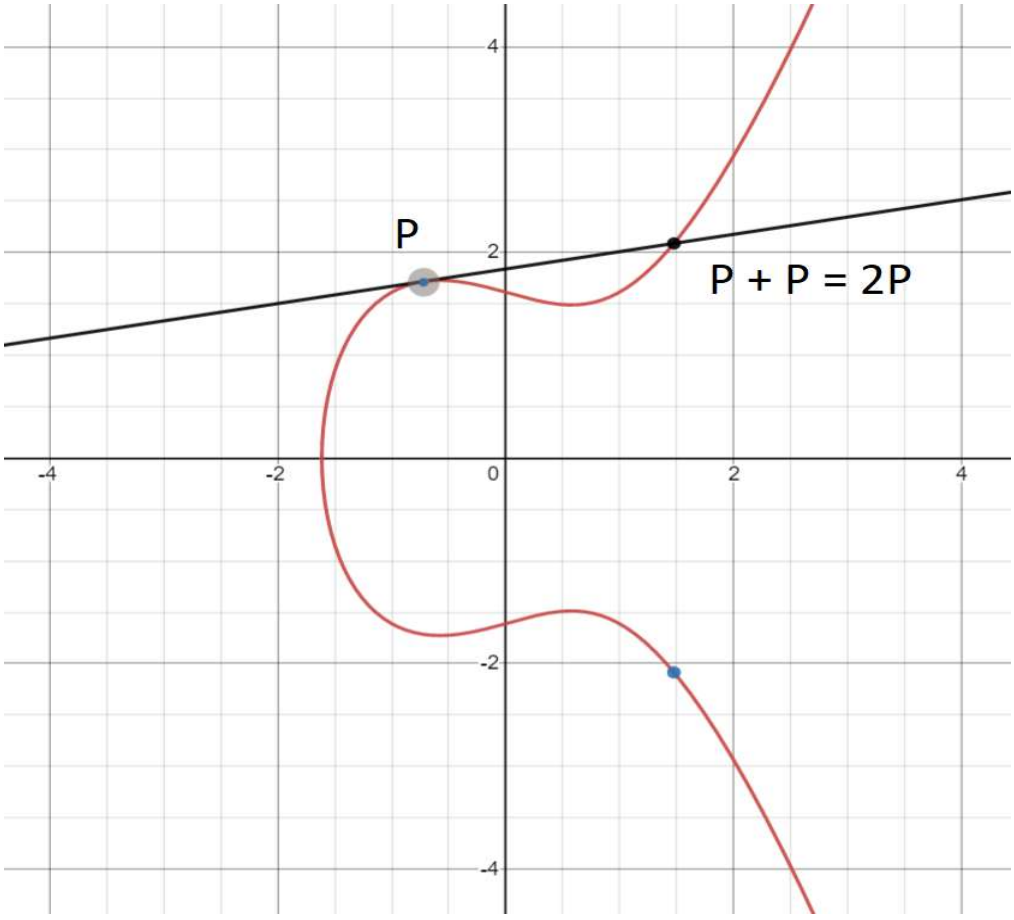

Figure 4. Adding a point on a curve

If we add P again to the result, we get 3 * P and so on. This is how the EC score multiplication is determined.

A point P over an elliptic curve can be multiplied by an integer k and the result is another EC point G on the same curve and this operation can be calculated easily and very fast

$$G = k * P$$

It is important to know is that multiplying EC point by integer returns another EC point on the same curve and this operation is really fast.

3.2. Weaknesses of ECC

Because of the mathematic complexity, only a small number of people can completely understand ECC algorithms. We don't even know whether there are any vulnerabilities related to the algorithms, from which they can create a backdoor to attack the system.

Another weakness of ECC is that we are using recommended pre-defined Elliptic Curves. It is necessary to stay alert that the organizations who provide these curves (such as NIST) can have a backdoor in them.

The fact that using complicated mathematic algorithms does not make encryption stronger. Ideally, we should use some algorithms that anyone can easily understand, not something we have to rely on a small number of people, hoping them to be on our side.

ECC is also somehow vulnerable to bruteforce attacks. Like all other public-key cryptographic algorithms, from a public-key, we can obtain the private-key through bruteforce attacks. Therefore, it is necessary to consider for a bigger-sized key.

CHAPTER 4: COMPARATIVE ALGORITHMS DEVELOPMENT

4.1. ElGamal Encryption Algorithm

Suppose that Alice wants to communicate with Bob：

1. Bob generates public and private keys:

- Bob chooses a very large number q and a cyclic group Fq.

- From the cyclic group Fq, he choose any element g and an element a such that gcd(a, q) = 1.

- Then he computes h = ga.

- Bob publishes F, h = ga, q, and g as his public key and retains as a private key.

2. Alice encrypts data using Bob's public key:

- Alice selects k from cyclic group F such that gcd(k, q) = 1.

- Then she computes p = gk and s = hk = gak.

- She multiples s with M.

- Then she sends (p, M*s) = (gk, M*s).

3. Bob decrypts the message:

- Bob calculates s′ = pa = gak.

- He divides M*s by s′ to obtain M as s = s′.

4.2. RSA Encryption Algorithm

The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is also derived from

the same two prime numbers. So, if somebody can factorize the large number, the private key is compromised. Therefore, encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.

4.3. General comparison of ECC, RSA and ElGamal

Table 5. General comparison of ECC, RSA and ElGamal

Factors	**RSA**	**ElGamal**	**ECC**
Development	1997	1984	1985
Performance	Fast	Slow	Very Fast
Power Consumption	High	Low	Low
Hardware requirement	High and not efficient	Low and efficient	Very low and very efficient
Mathematic based algorithm	Factoring the product of 2 large prime numbers	Computing discrete logarithms in a finite field	Computing elliptic curve discrete logarithm
Security Strength	Strong	Strong	Very strong

CHAPTER 5: IMPLEMENTATION OF PROCESS MEASURING

5.1. Comparison Criteria

5.1.1. Time Performance

The very first criterion we think of when talking about the effectiveness of Cryptographic algorithms is the amount of time they take to calculating. Especially in IoT systems, where the size of processing devices is not very suitable for high-performance CPU and Memory but have high requirements in the processing time.

Because of that, we have decided the consumption of time is the most important criterion in comparing the cryptographic algorithms. The shorter time it takes to calculate an algorithm, the more effective the algorithm is.

5.1.2. CPU Performance

In our development, we want to measure each Cryptography Algorithm by its CPU performance percentage on Raspberry Pi in an amount of time.

We expected that all three algorithms will have the same result of CPU performance percentage as they are running on the same environment of Raspberry Pi 3 or Raspberry Pi Zero.

The outcome of the measurement is an average usage of CPU for calculating the algorithms. Analyzing the CPU average usage with the total processing time will help evaluating the effectiveness of the algorithms.

5.1.3. Memory Performance

Calculating cryptographic algorithms can be determined as a very memory-consuming task. It needs a large amount of memory to compute heavy

calculations. Therefore, in order to ensure that the calculations worked properly, we assume that memory performance is also an important criterion to be measured

5.1.4. Power consumption

In IoT systems, there are numerous microcontrollers and stripped-down SBCs like the Raspberry Pi Zero that are more energy efficient options than a full Raspberry Pi 4. But that efficiency comes with a cost of its own in terms of a reduction in features and functionality. But in the same testing environment, we should be looking at ways to save as much unused power consumption on the Raspberry Pi as possible.

We have used an external measuring tool to get the power consumption amount of Raspberry Pi Zero and Raspberry Pi 3 while they are running on processing difference cryptographic algorithms. From which, we can determine whether there are any differences between those algorithms. Then, determine which algorithms is the best for saving power usage.

5.2. Environment setting up

5.2.1. Devices

- Raspberry Pi 3:

+ Quad Core 1.2GHz Broadcom BCM2837 64bit CPU

+ 1GB RAM

+ Micro USB power source up to 2.5A

- Raspberry Pi Zero:

+ 1GHz, single-core CPU

+ 512MB RAM

+ Micro USB power

5.2.2. Power Source

- Electric potential: 220V
- Power supply unit for Raspberry Pi 3 Adapter: 2.5A – 5V
- Power supply unit for Raspberry Pi Zero: 2A – 5V

5.2.3. Measuring Unit

- Voltage measuring range from 3V ~ 20V
- Electric current range from 0A ~ 3A

5.3. Process Measuring Development

Naturally, we cannot directly measure the CPU performance and Memory performance of the calculating process. Then, we found Gosputil, which is a Go language library package that provide process measuring functions.

5.4. Power Measuring Development

In our process of power measurement, a testing unit is plugged directly between power source and the devices. Because there is no suitable method to collect and transfer the data to a log file. All the power consumption usage is logged and calculated manually.

CHAPTER 6: RESULT ANALYSIS

6.1. Time of processing analysis

Using a wordlist file with 100 random generated 1024-bits messages, we let our 2 machines Raspberry Pi 3 and Raspberry Pi Zero send and receive the messages from and to each other, one by one. With every message sent and received, we wrote a line on our running-time-log files. Each line indicates how much time had been consumed to encrypt or decrypt the message. We have run the code for 10 times, which results in a total of 60 log files.

Each log file has 100 lines indicating encryption time and 100 lines indicating decryption time.

Having the collected data from the log files, we wrote a simple script calculating the average time consumption of the processes. The average numbers are calculated by adding all the lines of the log files with their specific identifiers.

After calculation, we got the data, which is suitable for us to draw the comparison chart, shown as below:

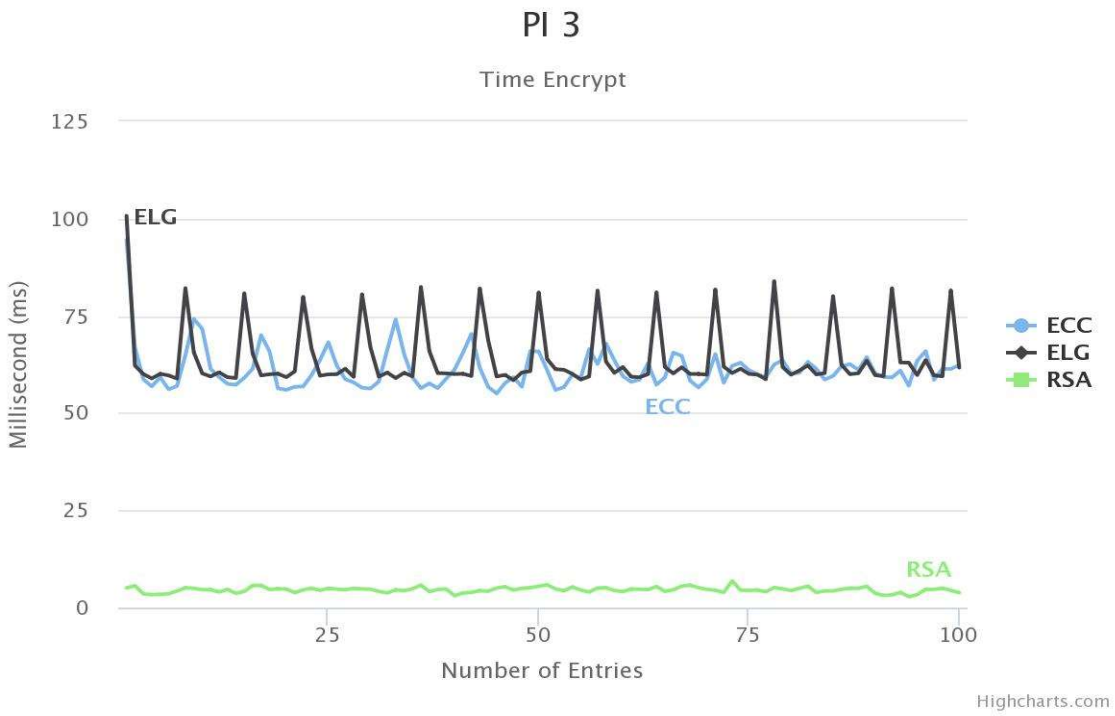

Figure 5. Encrypting time comparison between algorithms on Raspberry Pi 3

Figure 6. Encrypting time comparison between algorithms on Raspberry Pi Zero

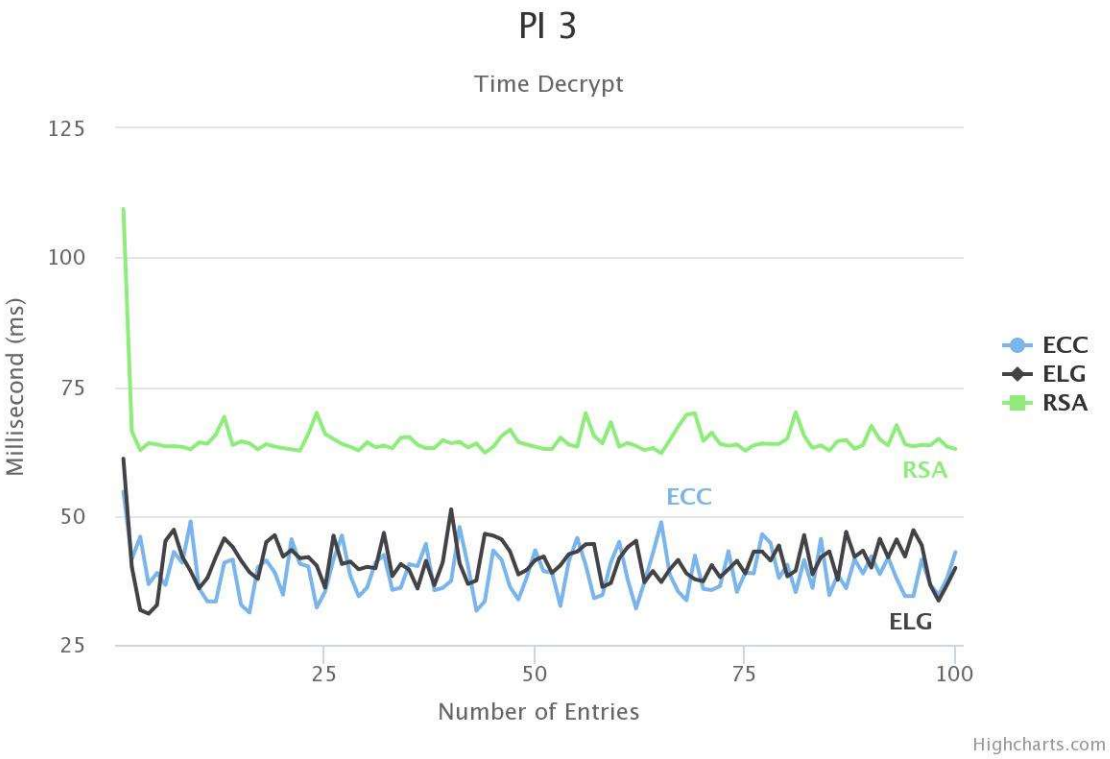

Figure 7. Decrypting time comparison between algorithms on Raspberry Pi 3

Figure 8. Decrypting time comparison between algorithms on Raspberry Pi Zero

With the data, we parsed the numbers into Highcharts – a JavaScript based tool – to visualize the time-consumption comparison between the algorithms.

Compare to other RSA, Elliptic Curve Cryptography unexpectedly shows that it is not faster in encryption phase. But in decryption phase, ECC shows that it is about 2 times faster than RSA. On the other hands, ECC and ElGamal has a quite equivalent of processing time in both encryption and decryption phases.

On the first glance, we may see that ECC is not so effective in the criterion of time consumption. But on the next criterion of CPU and Memory usage, which is further described in section 6.2, it will be explained. Besides, although the processing time is similar, comparing to ElGamal which require much more resource for key generation, ECC is also much more effective.

6.2. CPU and memory performance analysis

Using the same method of collecting time performance data, we have generated the log files showing the records of CPU and Memory performance. The scripts for collecting those data are run at the same time and on the same environment as the time measuring process. Therefore, the data can be used for explain why ECC's time consumptions is not the same as we have expected.

The calculation of average CPU and Memory usage is also similar to previous section. Having the engineered data, we, again, have Highcharts visualize the comparison as below:

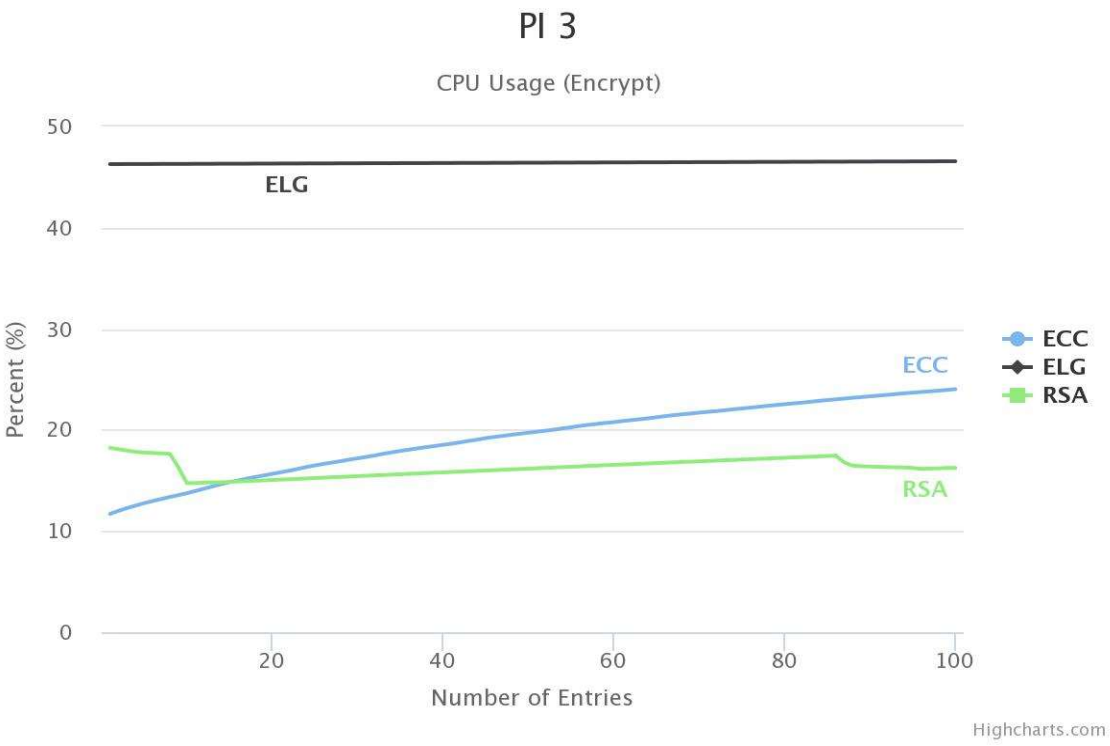

Figure 9. Comparison of CPU consumption during Encryption on Raspberry Pi 3

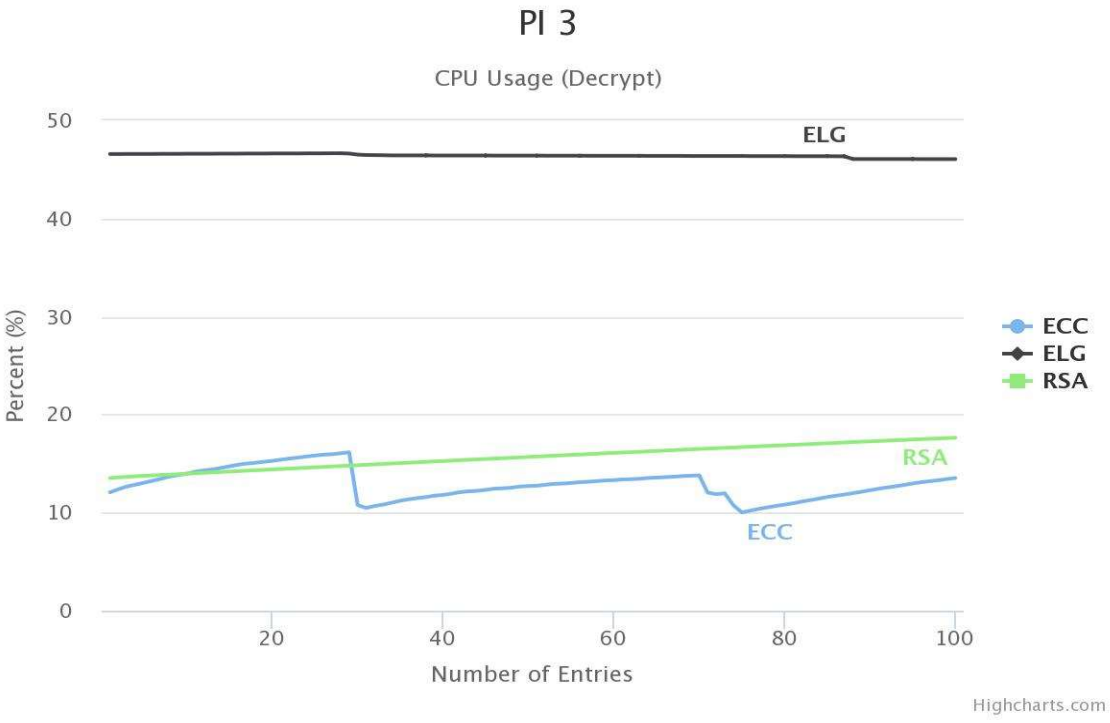

Figure 10. Comparison of CPU consumption during Decryption on Raspberry Pi 3

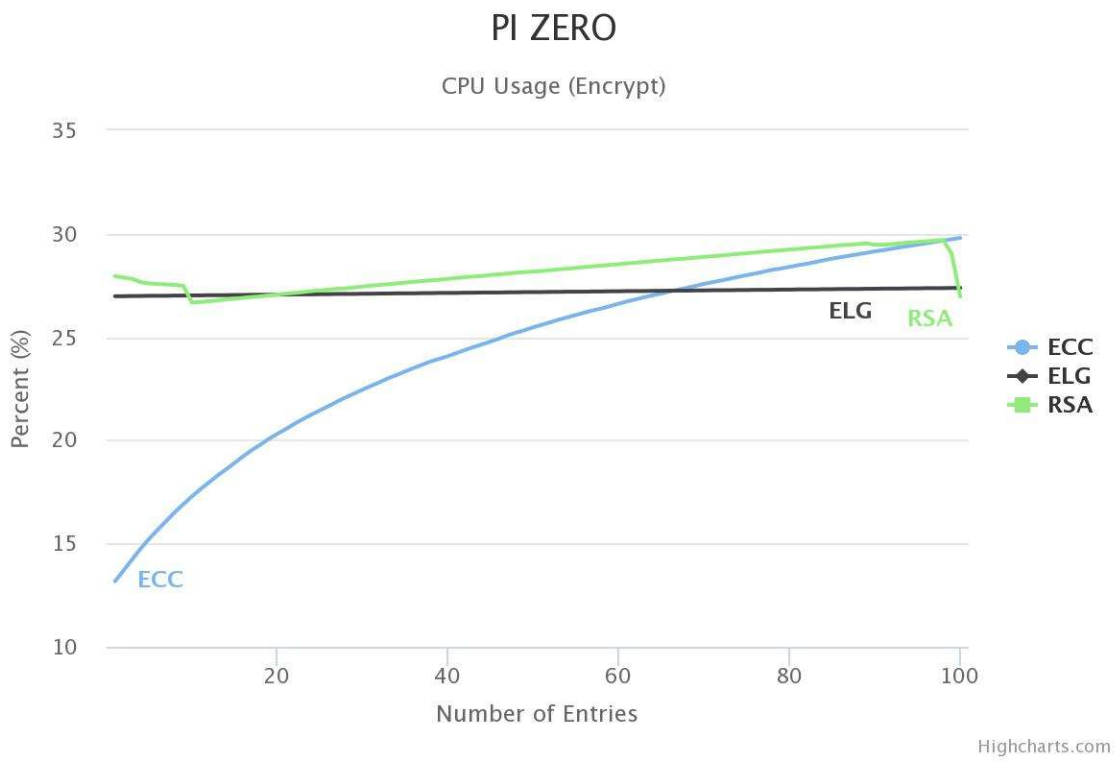

Figure 11. Comparison of CPU consumption during Encryption on Raspberry Pi Zero

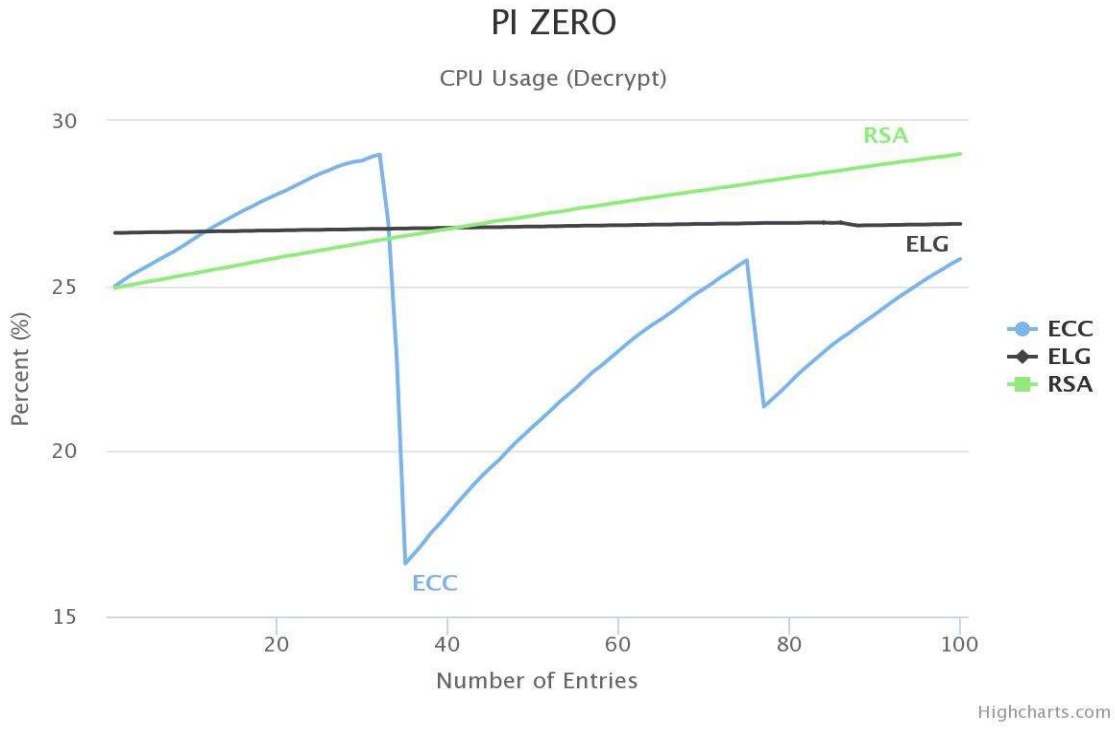

Figure 12. Comparison of CPU consumption during Decryption on Raspberry Pi Zero

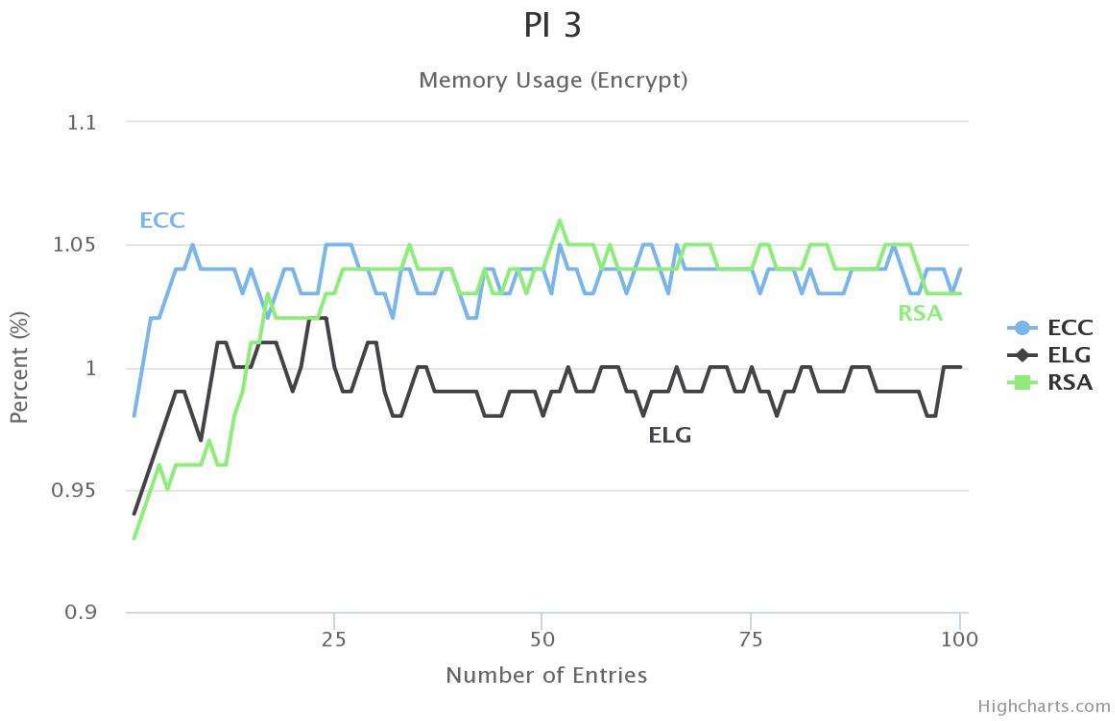

Figure 13. Comparison of Memory usage during Encryption on Raspberry Pi 3

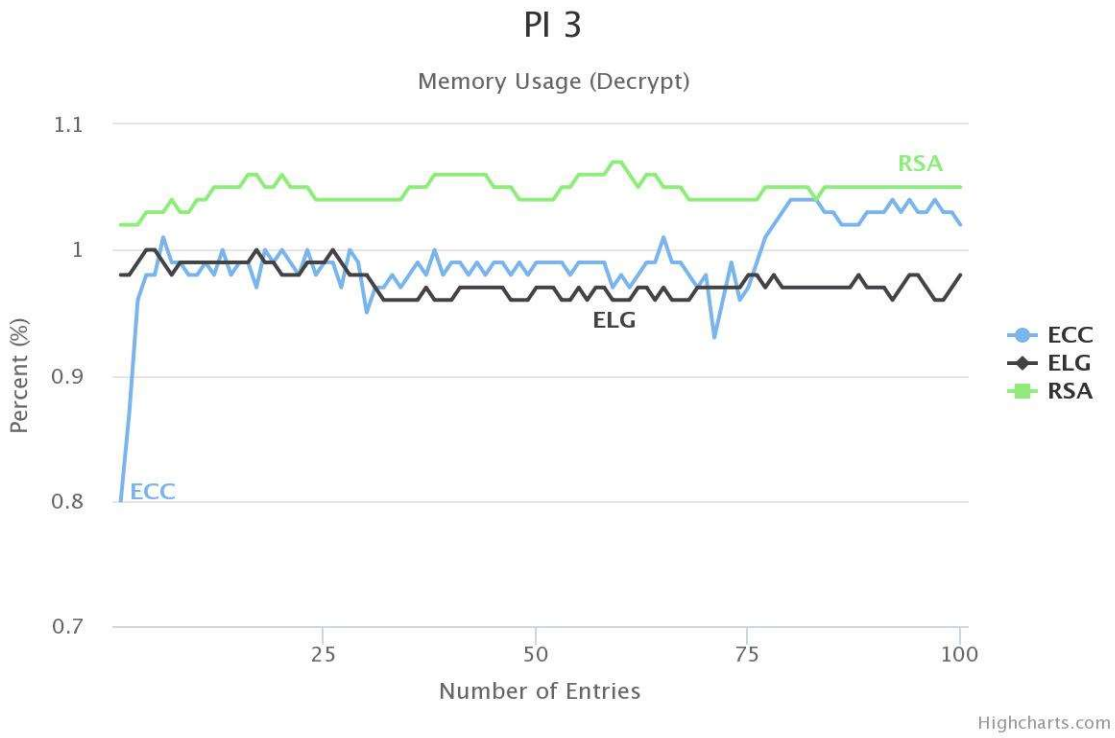

Figure 14. Comparison of Memory usage during Decryption on Raspberry Pi 3

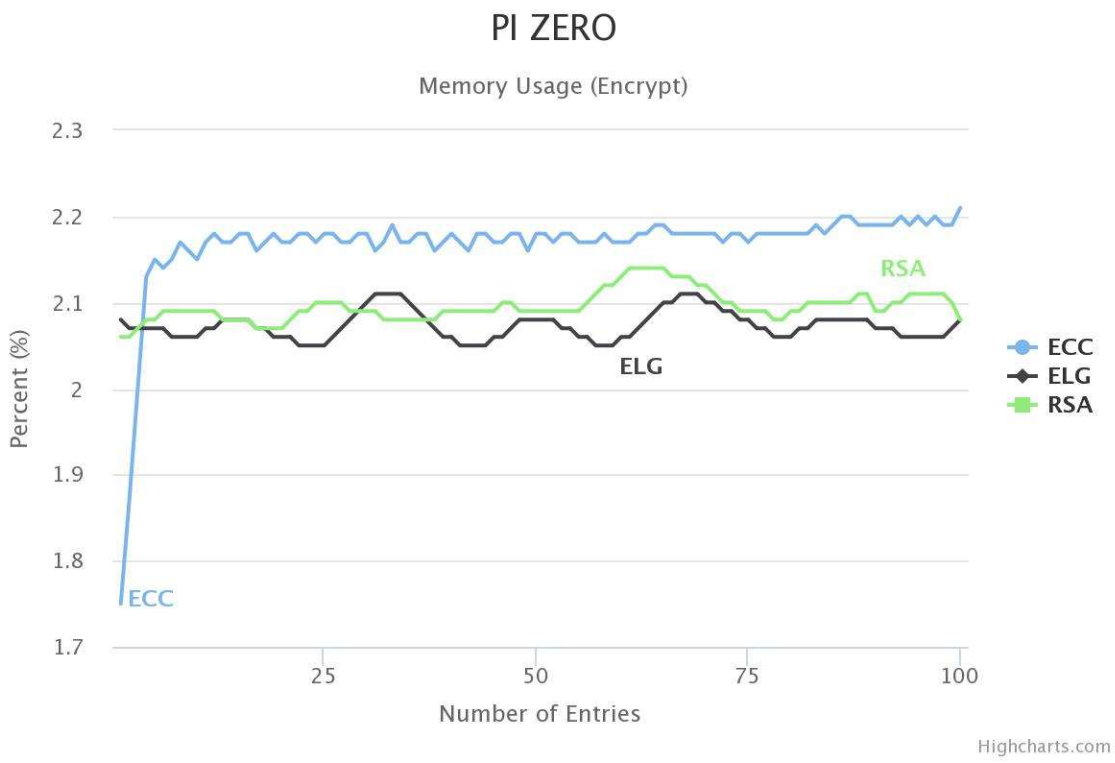

Figure 15. Comparison of Memory usage during Encryption on Raspberry Pi Zero

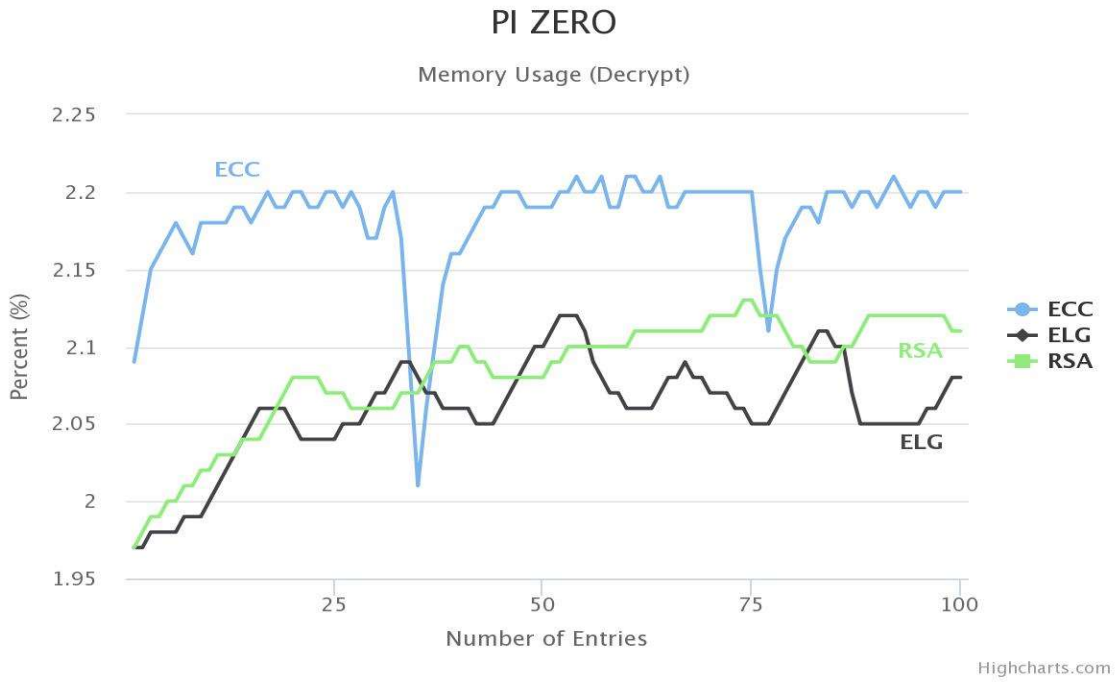

Figure 16. Comparison of Memory usage during Decryption on Raspberry Pi Zero

As we can see, in rich resourced environment like Raspberry Pi 3, the CPU performance of ECC and RSA is quite similar at low demand, while ElGamal consumes a huge amount of CPU resource. As in low resourced environment like Raspberry Pi Zero, ECC at some first operation consumes much lower CPU resource than others. But for the later time, CPU consumption increased and maintain at the same amount as RSA and ElGamal. This problem may due to the overloading on low resource environment.

In the analysis of both comparison criteria: time consumption and CPU/Memory usage, we can evaluate the ECC's effectiveness by comparing the total resources it consumed while processing.

Comparing to RSA, ECC have higher encrypting time but lower in decryption which can be count as an equivalence. At the same time, we can see that ECC uses much lower resource than RSA. So, we can see that the total resource that ECC used is lower than RSA did. In conclusion, ECC is quite more effective than RSA on the criteria of resource consumption.

Besides, when comparing to ElGamal, ECC shows equivalence in both encryption and decryption time consumed but a much lower resource usage. ElGamal also have the high resource and time demand at key generation process as its own drawback. So, we can claim that ECC is also more effective than RSA.

Considering all off the fact, ECC beats the other algorithms on the criteria of resource consumption.

6.3. Power consumption analysis

Using the external measurement unit connected directly between power source and the devices, we measured and take notes for every 2 seconds during the processing time of the devices.

After measuring, we can calculate the average voltage and electric current every encrypt/decrypt phase. Then we can calculate the power usage per second by using Ohm's Law:

$$P\ (W) = U\ (V) \times I\ (A)$$

Table 6. Average power consumption

Algorithms	**Raspberry Pi 3**	**Raspberry Pi Zero**
Idle	0.20A – 5.01V ~ 1.00W	0.17A – 5.01V ~ 0.85W
ECC	0.23A – 5.05V ~ 1.16W	0.18A – 5.05V ~ 0.91W
RSA	0.23A – 5.04V ~ 1.15W	0.19A – 5.05V ~ 0.96W
ElGamal	0.32A – 5.17V ~ 1.65W	0.26A – 5.13V ~ 1.33W

From the table, it is easy to see that the power usage of ECC and RSA is quite similar but RSA's is a little bit higher. Meanwhile, ElGamal's power usage is much higher, shows that ElGamal is not so effective when widely used.

So, in the criteria of power usage, ECC or RSA must be more effective when used in IoT systems, which will be worldwide used in the future.

6.4. Conclusion

As IoT systems and devices sometimes have high requirements of operating speed and small data storage, ECC, at the currently time, is the most effective encryption algorithm should be used in these systems for their best performance.

References

[1] *Figure 1* - Krishna, R.R.; Priyadarshini, A.; Jha, A.V.; Appasani, B.; Srinivasulu, A.; Bizon, N., 23 August 2021. *State-of-the-Art Review on IoT Threats and Attacks: Taxonomy, Challenges and Solutions*. Retrieved from: https://doi.org/10.3390/su13169463

[2] Sjoerd Langkemper. *The Most Important Security Problems with IoT Devices.* Retrieved from: https://www.eurofins-cybersecurity.com/news/security-problems-iot-devices

[3] *Figure 2* - Nishaal J. Parmar, January 12, 2019. *A Comparative Evaluation of Algorithms in the Implementation of an Ultra-Secure Router-to-Router Key Exchange System.* Retrieved from: https://www.hindawi.com/journals/scn/2017/1467614/

[4] Darrel Hankerson, Alfred Menezes, Scott Vanstone, 2004. *Guide to Elliptic Curve Cryptography.*