A comparative analysis of the computation cost and energy consumption of relevant curves of ECC presented in literature

PIM MULDER, University of Twente, The Netherlands

With the Internet of Things (IoT) becoming exponentially more prevalent, the need for lightweight cryptography functions increases simultaneously. Namely, IoT end devices are mostly limited by their resource-constrained capabilities and therefore cannot rely on heavyweight cryptographic algorithms such as Rivest-Shamir-Adleman (RSA) or Diffie-Hellman (DH) for security. Elliptic Curve Cryptography (ECC) offers a more lightweight alternative by being based on a mathematical problem named the Elliptic Curve Discrete Logarithm Problem (ECDLP) which is not known to be solvable in sub-exponential time. Within the field of ECC, many different curve types exist following various standards for this protocol. In this paper, the energy consumption and time consumption for key generation, encryption, and decryption are measured within the ElGamal protocol using ECC for the various curves. To measure this, a Raspberry Pi 4B and a Personal Computer are used to conclude the disproval of the hypothesis that the Twisted Edwards curve performs most efficient to achieve its security strength. Namely, Brainpool curves function most efficient within this benchmark, after which Short Weierstrass curves follow. Moreover, it is concluded the performance pattern for both data processors and data providers is equal to each other for all message sizes.

Additional Key Words and Phrases: Elliptic Curve Cryptography, ECC, Internet of Things, IoT, Benchmark, Cryptography, Montgomery Curve, Short Weierstrass Curve, Twisted Edwards Curve, Brainpool Curve

1 INTRODUCTION

With the recent start of the era of Industry 4.0, the Internet of Things (IoT) has become incredibly more significant within society [14]. The IoT is used in many applications including smart environments, healthcare, logistics, personal use, et cetera [1, 12]. In these settings, the end devices' prime function often is to collect or process data in some manner [1, 11]. Consequently, many data are transferred from, and to these devices. Following this, to ensure the safety of the data, the need for cryptography becomes apparent [13].

Traditionally, for such purposes an asymmetric algorithm such as the Rivest-Shamir-Adleman (RSA) [31] or the Diffie-Hellman (DH) [10] algorithm would be used; Both operate by maintaining public and private keys, where the public key is used to ensure safety in sending the data and the private key is needed to guarantee the safe receiving of the message [10, 31].

However, both 'traditional' algorithms are built upon mathematical problems which all have proven to be solvable sub-exponentially [6, 24]. Namely, the RSA algorithm is built upon the Integer Factorisation Problem (IFP) [24], and the DH algorithm is built upon the discrete logarithm problem (DLP) [25].

Consequently, the key size for both RSA and DH is recommended to be at least 2048 bits by several organizations [3, 20]. Therefore, a system utilizing these algorithms is required to have access to a considerable amount of resources [22]. Combined with the observation that IoT devices are resourcefully limited, there is now a concluded need for a lightweight cryptography algorithm.

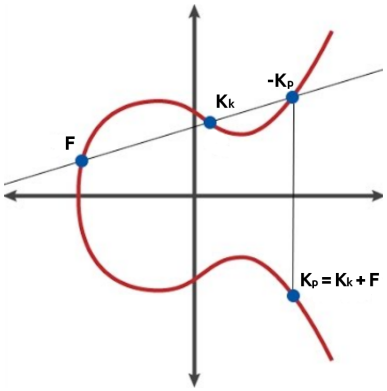


Fig. 1. Elliptic curve displaying the operation $P + Q = R$ [15]

Elliptic curve cryptography (ECC) provides a lightweight trapdoor function by being based on the elliptic curve discrete logarithm problem (ECDLP). This problem is not yet known to be solvable in sub-exponential time. As a consequence, the keys used for ECC become much smaller in comparison to those used by alternatives [3, 24].

The algorithm works by the entire network agreeing on both a curve EC, an order of the base point, resulting in base point F, and a field size n. Then, each user selects a private key K_k and a public key K_p such that:

$$K_p = K_k + F \pmod{n} \tag{1}$$

as visualised in Figure 1. Moreover, the arithmetic operation to add points varies per protocol and is determined per network. To encrypt a message M_d, the sender can apply the following function to generate an encrypted message M_e:

$$M_e = M_d + K_k \pmod{n} \tag{2}$$

This process can be reiterated a predetermined amount of times to generate a fully secure message. The decryption of the message is done by finding the third intersection of the line crossing M_e and K_p.

However, despite being considered a promising lightweight cryptography algorithm, limited research has been done in benchmarking ECC in the context of end devices for the IoT. Additionally, no benchmark has been made comparing the different parameters that can be used within ECC.

Hence, the goal of this paper is to answer the following research question:

TScIT 37, July 8, 2022, Enschede, The Netherlands

RQ 1. What combination of curve and field size, creates the most efficient ECC algorithm in the context of the IoT?

Then, to help answer Research Question 1, two additional research questions are answered:

RQ 2. What combination of curve and field size creates the most efficient ECC algorithm on a Personal Computer?

RQ 3. What combination of curve and field size creates the most efficient ECC algorithm on a Raspberry Pi?

Herein, the choice is made for a Personal Computer (PC) and Raspberry Pi (RPi) to simulate receivers and senders for data in the IoT end devices respectively. So, using these three questions, a comparative analysis will be made of the computation cost and energy consumption of relevant curves of ECC presented in literature.

Gaining an answer to these questions will first be done by drawing out the background in Section 2. Therewith, an overview of the recommended standards is created as well as an analysis of how ECC is used in practice and the related research within the field. Following, an insight is gained into the state of the art and the missing literature. Consequently, the research questions will be hypothesised upon.

Then, measurements can be made to create a complete benchmark on the different parameters used in ECC in Section 4. In order to do so, the methodology is first explained within Section 3. Namely, in this section, the hardware and software will be walked through: Explaining the platforms, libraries, and algorithms used.

Following, the measurements are made and presented in Section 4. Moreover, they will be explained and analysed to seek out any outliers or otherwise non-expected results. Note, that any results that follow the research questions will be explained in Section 5. Namely, within the conclusion section, the outcome of relevant research will be put up against the measurements to answer the research questions.

Thereafter, recommendations for future work are made. Additionally, a reflection on the research is conducted in Section 6.

2 BACKGROUND

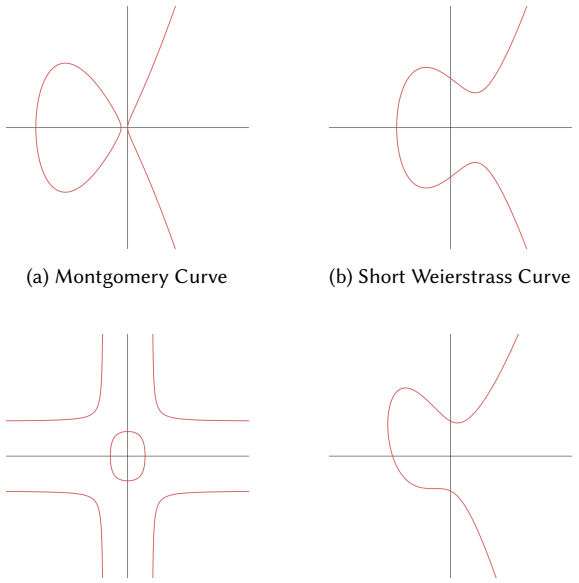
With the purpose of creating a taxonomy based on the relevant standards, a definition of the relevant standards is apt. Therefore, this section seeks to outlay any standard used within the industry.

As there exist many different standardisation institutions, it must be added that the scope of this paper will be put at any standard mentioned within the latest draft document of the National Institute of Standards and Technology (NIST) [8].

Following the disquisition of standards, an explanation will be provided of how ECC is utilised in practice, and in what way those algorithms function. Then, related research will be reviewed, in the hope that any missing literature is identified. Achieving this is done by discussing several papers which sought to create a taxonomy for the same goal as this research.

Within the latest version of the NIST document [8], it has been mentioned that the standards advised were inspired by several organisations' standards. These institutions include, but are not limited to: The Internet Engineering Task Force (IETF) [29], the Standards for Efficient Cryptography (SEC) [5], and the previous version of the NIST recommendation [19].

In regards to the curve type used for ECC, the NIST has identified several different curve types. To visualise, these curves are displayed in Figure 2:



(a) Montgomery Curve (b) Short Weierstrass Curve

(c) Twisted Edwards Curve (d) Short Weierstrass (Binary)

Fig. 2. ECC Curve types recommended by the NIST

Firstly, the 'traditional' curve type is the Short Weierstrass form displayed in Figure 2b. This form follows:

$$y^2 = x^3 + ax + b \tag{3}$$

Wherein a and b are configurable parameters with:

$$4a^3 + 27b^2 \neq 0 \tag{4}$$

These a and b are both generated by a specific seed - on which the curve is based. The standards for this curve are: P-192, P-224, P-256, P-384, P-521, W-25519, and W-448. Herein, P-192 is only for legacy use - and thus retracted from the latest version - and W-25519 and W-448 have been newly added. It is phrased as being 'traditional' as that was the earliest form of EC introduced.

In this notation, the first letter is either a P or W: P indicates it is a curve over a prime field and W indicates it is both a curve over a prime field, and that it is isomorphic to the curve with the same number sequence for the Montgomery curves. For example, W-25519 is isomorphic to Curve25519 - for the numeric values of each curve, the Reader can refer to the draft publication of the NIST [8].

Additionally, the Montgomery curve type provides Curve448. Both Curve448 and Curve25519 follow the form:

$$Bv^2 = u(u^2 + Au + 1) \tag{5}$$

Wherein:

$$A \in GF(q),\ B \in GF(q),\ A \neq \pm 2,\ B \neq 0 \tag{6}$$

Security Strength	Bit Length of n	Prime Field	Binary Field
112	224 - 255	len(p) = 224	m = 233
128	256 - 383	len(p) = 256	m = 283
192	384 - 511	len(p) = 384	m = 409
256	≥ 512	len(p) = 521	m = 571

Table 1. Recommended bit length

Moreover, these curves have been added to the most recent version following the earlier recommendation of the IETF [29]. Next to that, the Edwards Curve type was introduced in the latest draft. These curves follow:

$$ax^2 + y^2 = 1 + dx^2y^2 \tag{7}$$

With:

$$a, d \neq 0, a \neq d \tag{8}$$

Within the NIST recommendation [8], it is stated that curves over this form provide the highest performance using the protocol Edwards Curve Digital Signature Algorithm (EdDSA). This form introduces the curves: Edwards25519, Edwards448, and E448.

Then, the NIST does mention the existence of Brainpool curves introduced by the IETF [29]. However, they do not recommend them as they only put them in their last appendix. Regardless, this type introduces curves following the form of a Short-Weierstrass curve on a prime field, with different parameters. The standards existing are: brainpoolP192r1, brainpoolP224r1, brainpool256r1, brainpoolP320r1, brainpoolP384r1, and brainpoolP512r1. Alternatively, when using EdDSA twisted curves could be used, for this protocol exists the standards: brainpoolP192t1, brainpoolP224t1, brainpoolP256t1, brainpoolP320t1, brainpoolP384t1, and brainpoolP512t1.

Finally, the NIST suggests curves Short-Weierstrass curves over a field with size GF(2^m) where m is any positive integer.

In this curve-form exist two distinctions, the curve is either a Koblitz curve or a pseudo-random curve. For either curve they are marked as the identifier followed by the exponent m used for the binary field GF(2^m); Koblitz curves are identified with a K and pseudo-random curves with a P. Koblitz curves follow the form:

$$y^2 + xy = x^3 + ax^2 + 1 \tag{9}$$

And pseudo-random curves follow the form:

$$y^2 + xy = x^3 + x^2 + b \tag{10}$$

The standards existing are: K-163, K-283, K-409, K-571, B-163, B-233, B-283, B-409, B-571. Similar to the Short Weierstrass form on a prime field, the smallest field size has been removed from the recommendation. Namely, with the increase in computation power the security of this variation came at risk. However, following the discovery of the GHS Weil Descent Attack [17], this curve is only mentioned and not recommended for federal use.

This is confirmed by combining the security strength tested in SP 800-57 [2] and the field size of each curve. Namely, in Table 1, it is noticed that for the binary field, an exponential increase in the difference between len(p) and m forms when increasing security strength.

Furthermore, transferring information using ECC is done by using the ElGamal scheme [16], as proposed independently by Miller [26] and Koblitz [21] [23]. The algorithm functions using the concept explained in Section 1:

Suppose that *Alice* and *Bob* want to share information over an insecure channel. In this situation, *Alice* has secret x_a and *Bob* has secret x_b, n is the size of the module, and α any primitive element (mod n) - the latter two being both known. Then *Alice* computes $y_a = \alpha^{x_a} \pmod n$ and *Bob* computes $y_b = \alpha^{x_b} \pmod n$. This value is communicated cross-party, and consequently, a common secret K_{ab} can be established by exponentiating y_a with x_b or vice versa.

Following, the encrypted message m comes in the form (c_1, c_2), where:

$$c_1 = \alpha^{K_{ab}} \pmod n \quad c_2 = K_{ab}m \pmod n \tag{11}$$

Decryption to the message m is done by dividing K_{ab} with c_2. Note that, for this, multiplication on an EC is needed, this is achieved by getting the tangent to the point P after which point Q, the intersection with EC is the result of one iteration of multiplication.

Notwithstanding the precise algorithmic implementation, Section 1 claimed the absence of relevant literature in benchmarking ECC in the setting of the IoT. However, the literature is abundant in theoretical comparisons between possible combinations. Therefore, here an overview is created of their methodologies to hypothesise on the research questions, and to find which flaws need to be covered up. Moreover, the choice of parameters is argued for.

In the literature, already some overviews of the parameters in use have been made. For example, Julio López and Ricardo Dahab [23] created an overview in 2000 of the then-dominant ECC parameters splitting the variables out into: the finite field arithmetic, the type of curve, the algorithms for group operation, and elliptic curve protocols. However, as this overview stems from 2000 it is heavily out of date. This is proven by comparing this study with the current, aforementioned, standards.

Contrasting, this research has only two variables in the form of the type of curve, and the key size used. For the reason that the NIST [8] has put finite field arithmetic, group algorithms and elliptic curve protocol inherent to the type. Continuing, several other institutions or studies standardise a fifth variable, the group arithmetic. This is also disregarded, as that is made inherent by the NIST similarly. Ergo, the standards of the NIST [8], SEC [5], and IETF [29] have primarily contributed to the selected division.

Following the selection of parameters, it can be found how the combinations of each parameter interact with each other. However, as many permutations exist, it is more practical to apply a benchmark instead of theoretical research. This will be done by measuring the process of the entire transaction, similar to how it has been done previously to standalone ECC [4, 18]. Consequently, measurements of key generation, encryption, and decryption will follow.

Seeing the relevance of the work of Gura [18] and Branovic [4] it is reasonable to refresh their work by adding the newly found combinations to their results. Additionally, as the Internet of Things was not as popular then as it was now [1], the efficiency of ECC was not yet placed in the context of the IoT.

Dhanda [9] did, however, have the knowledge of IoT and addressed in their research the need to reduce keys even more, and to

reduce resource needs. Observing that they have only made measurements for different algorithms, it is believed to find a more secure alternative with the same efficiency.

However, seeing that none of the mentioned research conducts precisely similar measurements. Hypothesising the quality of each combination is challenging. Be that as it may, the NIST recommendation [8] does mention that Twisted Edward curves are the quickest curve type of all recommendations. Additionally, as aforementioned, binary fields require a greater key size to match the security strength of curves over prime fields. Therefore, it is hypothesised that to reach the same security strength, Twisted Edwards curves will be the most efficient curves, followed by other curves on prime fields to then end the rank with Koblitz curves, and other curves over a binary field. It is expected that this hypothesis will uphold for both PC and RPi.

3 MEASUREMENTS

To answer the Research Questions set in Section 1, a systematic approach will be taken to make the measurements. In this section, the algorithm used to make these measurements is elaborated upon as well as the media this software is executed upon. In doing so, the choice of PC and RPi is explained. Moreover, a thorough understanding of the software used is created, i.e. the imported, and self-made programs.

To start, within the IoT two device types exists, data-collectors and data-processors. Herein, the data-collectors are frequently resource-constrained, whereas the data-processors allow for more computational power. Moreover, this balance is simulated using an RPi and PC respectively.

Firstly, the RPi used within this research is a Raspberry Pi 4B with 4GB of RAM (Random Access Memory). As the goal of the research is to have the RPi simulate an IoT device, a 32-bit OS without a graphical interface was chosen. Consequently, the minimal version of Raspian 32-bit was chosen as an OS.

Furthermore, to simulate a realistic setting for the data processor, the PC operates on a 64-bit Ubuntu Linux distribution. However, due to the only PC being available that of the Researcher, it does provide a graphical interface, and thus reduces the performance of the software. This PC operates on an Intel i7-8750H CPU clocked at 2.20GHz, with 16GB of RAM clocked at 2400 MT/s.

Then, the software [27] used is designed to output graphs measuring the performance of the encryption, decryption and key generation efficiency as described in Section 2. Herein, efficiency is defined as the time it takes to run these processes and the energy that is consumed by the CPU during this time. Furthermore, the software measures this performance for the Montgomery curves, Short Weierstrass curves over a prime field, Twisted Edwards curves, and Brainpool curves in either Short Weierstrass or Twisted Edwards form.

Continuing, within the curve types, the key size is the changed value after which the efficiency values should follow. Additionally, to outtake any bias towards certain message sizes, the test is repeated for message sizes of byte size 2^m with m being an integer ranging from 0 to 5. If a conclusion can be set that this bias is nonexistent, the experiment can focus on one specific message size to more easily draw conclusions - and thus answer the research questions.

Furthermore, creating the benchmark is done by importing an ECC python library [7] which allows for ElGamal encryption using the specified curve types. To this library, the old and new standards were added, as previously described, as that was not offered by default. Moreover, the measurements are made using either PyRAPL [30] or vcgencmd [28]. Namely, PyRAPL provides a measurement interface for the Intel chipset, whereas vcgencmd is used for the alternatives. Consequently, by running this script concurrently, the results can be made reasonably efficient resulting in those presented in Section 4.

4 RESULTS

In this section, the results of the measurement script as described in Section 3 are discussed. To create the clearest overview, the analysis is done by going through the separate variables in an orderly manner. Thus, first, the difference between message sizes on both the PC and RPi will be analysed, to then compare the difference among the curve types.

Within Figure 3, the results of data encryption on an RPi are presented for all measured message sizes. Notably, no significant difference can be found between any of those measurements. Similarly, the time for key generation and decryption followed the same pattern of no significant difference amongst message sizes - note that, for the reason of conciseness, these graphs are not displayed. Additionally, the energy usage on an RPi displayed no significant change when altering the message size. Therefore, further analysis of the measurements on an RPi is done on only one message size for the rest of this section.

Likewise, Figure 4 displays no significant differences between the decryption of data for several message sizes on a PC. Therefore, the remainder of this section will focus on a message size of 1 byte, as those data are more easily gathered. Additionally, it is analysed that the pattern for RPi and PC equal. Accordingly, the focus is shifted towards the results of a PC, as those data are easier to be gathered following the more plentiful resource pool.

As visualised in Figure 5, again, the patterns amongst different processes follow equal trends. For that reason, the tabular data for only one combination of process, measurement type, and message size is given in Table 2. Within this table, each row displays the energy it takes per - if available - field size to encrypt a message of 1 byte.

From these results, no real surprises are found that do not relate to the research questions. Therefore, those findings will be discussed in Section 5. Moreover, if it is desired to inspect the results with greater repetition, the Reader could refer to the source code [27], and repeat the experiment to gain and confirm the same benchmark.

5 CONCLUSION

As mentioned in Section 1, this section provides an answer to the set Research Questions by first answering Research Question 2 and Research Question 3, to then answer Research Question 1. Furthermore, within Section 4 it was found that the used platform or message size made no significant difference within the efficiency of

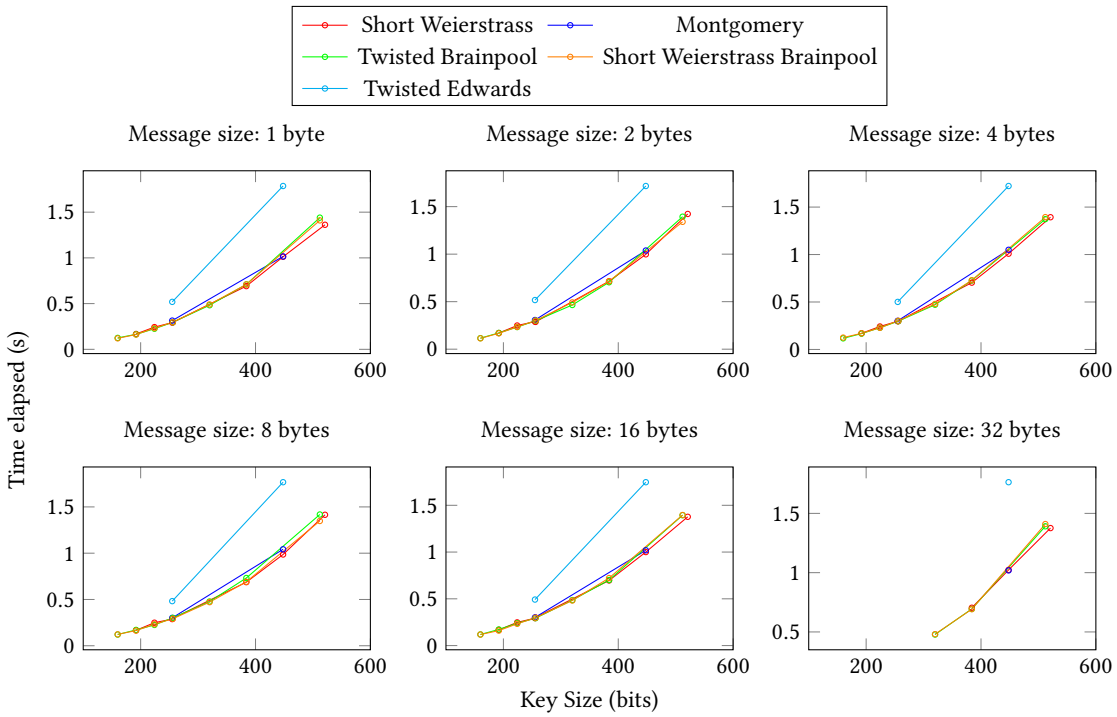


Fig. 3. Time (s) taken to encrypt data in ElGamal with ECC on RPi for several message sizes, n=100

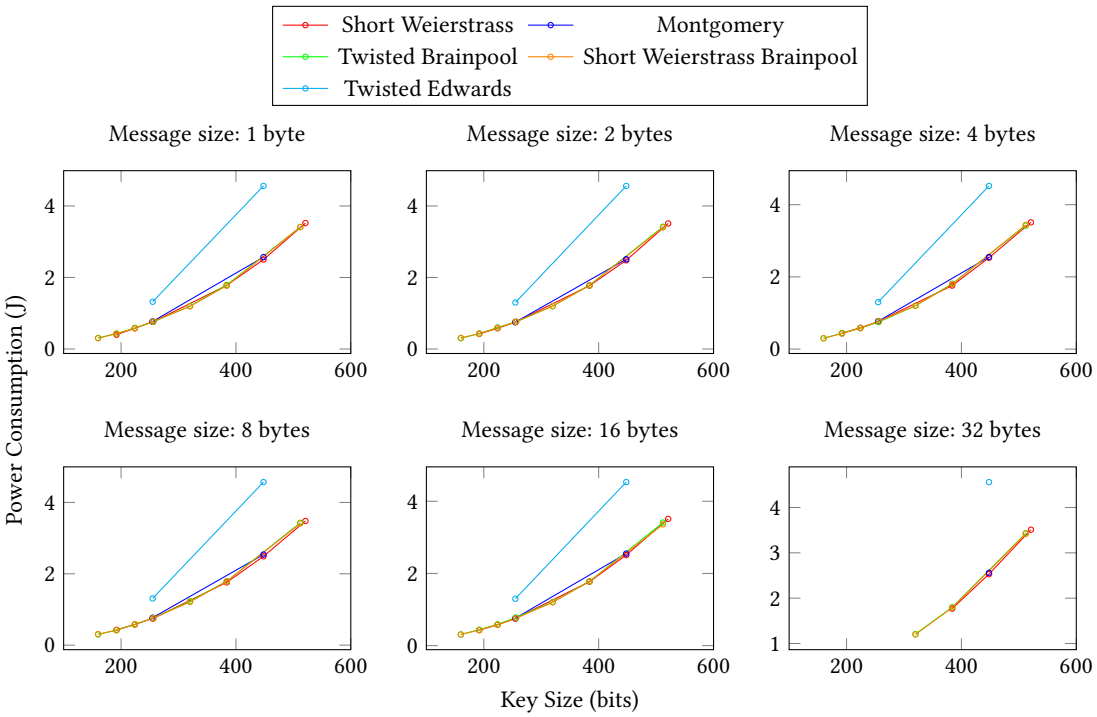


Fig. 4. Energy Consumed (J) whilst decrypting data in ElGamal with ECC on PC for several message sizes, n=1500

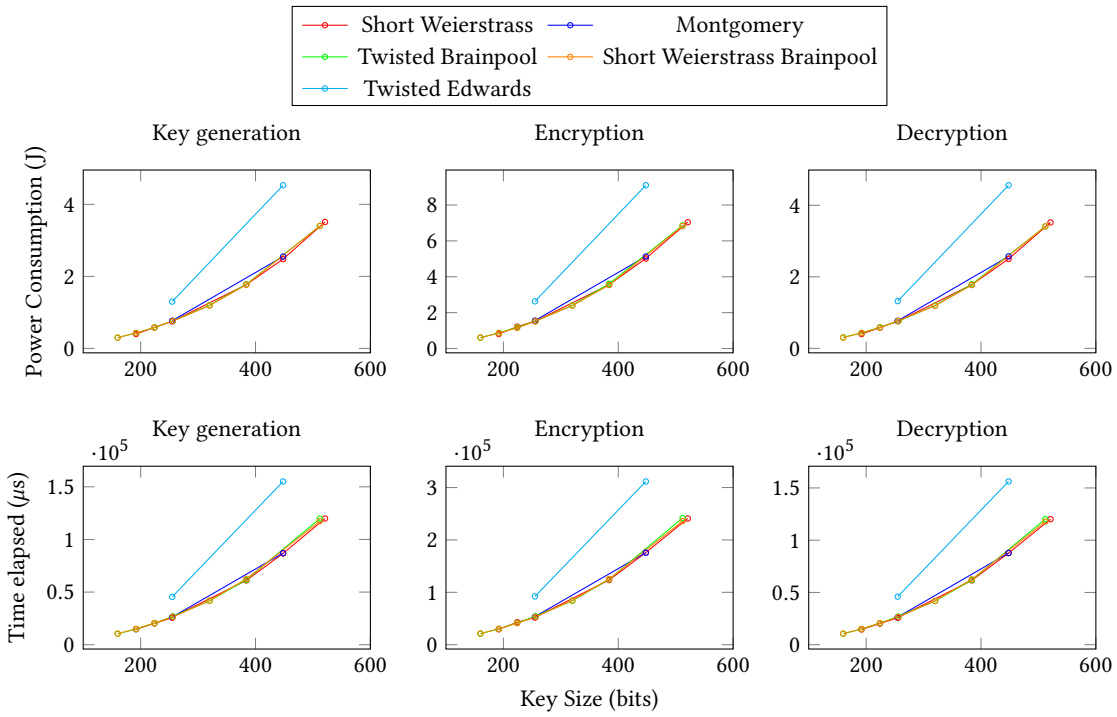


Fig. 5. Measurements for key generation, encryption, and decryption process for message size of 1 byte, n=1500

Curve Type	160	192	224	255	256	320	384	448	512	521
Short Weierstrass		0.81	1.22	1.52	1.53		3.56	5.02		7.04
Montgomery				1.56				5.13		
Twisted Brainpool	0.61	0.87	1.18		1.53	2.41	3.61		6.86	
Short Weierstrass Brainpool	0.61	0.86	1.15		1.53	2.39	3.54		6.83	
Twisted Edwards				2.63				9.1		

Table 2. Power (J) consumed during encryption of message of size 1 byte, n = 1500

	224	255	256	320	384	448	512	521	Average
Short Weierstrass	0.0109	0.0119	0.0120		0.0185	0.0261		0.0275	0.0178
Montgomery		0.0122	0.0137			0.0267			0.0175
Twisted Brainpool	0.0105		0.0120	0.0188	0.0188		0.0268		0.0174
Short Weierstrass Brainpool	0.0103		0.0120	0.0187	0.0184		0.0267		0.0172
Twisted Edwards		0.0205				0.0474			0.0340
Security Strength	112	128	128	128	192	192	256	256	

Table 3. Power (J) consumed per 1 bit security strength, n = 1500

ECC. Therefore, this section will provide an answer to the Research Questions following the data provided in Table 2. However, prior to that, the expectation of Section 2 is restated. Namely, within that section, it was predicted for the Twisted Edwards Curve to perform as the most efficient curve type.

Since Section 4 concluded no difference in results between an RPi and a PC regarding the pattern of the results, the answer of Research Question 2 and Research Question 3 can be found in an equal manner: Following the found security strength of Table 1 in Section 2, the supposed security strength of each curve can be combined per curve. Moreover, using the power consumption of Table 2, the amount of power consumed per one bit strength can be found. This leads to Table 3. Consequently, the most efficient algorithm will then be found by identifying the lowest number of this table. In this case, that result belongs to the Brainpool curve in Short Weierstrass curve with a field size of 224. Additionally, overall

the Brainpool curves perform the most efficient on average as well. Furthermore, notice that in Table 3, the field sizes smaller than 192 have been removed following the recommendation of the NIST [8], as discussed in Section 2.

Following this result, the hypothesis is disproven for Research Question 2 and Research Question 3 since Twisted Edwards curves proved not to be the most efficient curve type. Additionally, it can be concluded that the answer of Research Question 2 and Research Question 3 can be copied to Research Question 1 as Section 4 concluded the same efficiency pattern to uphold for either medium.

Concluding, it has been found that the most efficient curve type is one which is not, currently, recommended by the NIST in the form of Brainpool curves. Closely following are the other Short Weierstrass curves and the Montgomery curves. Though, the hypothesised most efficient curve resulted to be two times less efficient in comparison with the most efficient curve type.

6 FUTURE WORK

Following the disproven hypothesis, this section will provide recommended work to extend the quality of this research. This is done by initially reflecting on the quality of this research, after which extra work is proposed.

Firstly, following the literature research of Section 2, a security strength for each curve has been assumed as is. However, as this score was extrapolated from Table 1, it may be that the results in Table 3 may be mistaken as a consequence. Therefore, what could be done is to put a fourth variable in place to measure security strength. By measuring security strength, it is meant that the energy and time required for decryption by brute force is measured. Following, a more realistic energy per bit strength value could be produced, disproving this entire research.

Moreover, having followed the suggestion of the NIST to disregard curves over binary fields [8] it has never been formally proven per benchmark that these curves indeed perform worse. Again, this would require the Researcher to check for security as otherwise only an ideal scenario would be benchmarked.

Then, both these suggestions and the rest of this research could be repeated while also taking into account more resource-constrained devices such as an Arduino. However, due to resourceful limitations during this research, this was not to be benchmarked in this paper. Furthermore, when re-implementing the source code [27], the quality of the code could be reevaluated to rewrite it in a more efficient programming language than Python. Possibly, this could find the reason behind Twisted Edwards curves' performing significantly better on a PC for a message size of 1 byte as well.

Finally, as discussed, communication protocols using ECC rely on different algorithms compared to RSA or DH. Therefore, it is advised to benchmark communication costs per complete message transaction to truly test if ECC is a more lightweight protocol to RSA or DH.

To summarise, it is concluded the research in benchmarking ECC in the context of the IoT is far from complete. Future work can add a security value, not disregard binary curves and use other mediums to add more common instances to this work. Moreover, the claim of being a more lightweight protocol can be backed by also measuring communication costs.

REFERENCES

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. 2010. The Internet of Things: A survey. *Computer Networks* 54, 15 (2010), 2787–2805. https://doi.org/10.1016/j.comnet.2010.05.010

[2] Elaine Barker, Elaine Barker, William Burr, William Polk, Miles Smid, et al. 2006. *Recommendation for key management: Part 1: General*. National Institute of Standards and Technology, Technology Administration ....

[3] Elaine Barker and Allen Roginsky. 2018. *Transitioning the use of cryptographic algorithms and key lengths*. Technical Report. National Institute of Standards and Technology.

[4] Irina Branovic, Roberto Giorgi, and Enrico Martinelli. 2003. A workload characterization of elliptic curve cryptography methods in embedded environments. *ACM SIGARCH Computer Architecture News* 32, 3 (2003), 27–34.

[5] Daniel RL Brown. 2010. Standards for Efficient Cryptography 1 (SEC-1). *Standards for Efficient Cryptography, 2009* (2010).

[6] Daniel RL Brown and Robert P Gallant. 2004. The Static Diffie-Hellman Problem. *IACR Cryptol. ePrint Arch.* 2004 (2004), 306.

[7] Liu Chang. 2022. ecc-benchmarker. https://github.com/lc6chang/ecc-pycrypto.

[8] Lily Chen, Dustin Moody, Andrew Regenscheid, and Karen Randall. 2019. *Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters*. Technical Report. National Institute of Standards and Technology.

[9] Sumit Singh Dhanda, Brahmjit Singh, and Poonam Jindal. 2020. Lightweight cryptography: a solution to secure IoT. *Wireless Personal Communications* 112, 3 (2020), 1947–1980.

[10] Whitfield Diffie and Martin E Hellman. 2019. New directions in cryptography. In *Secure communications and asymmetric cryptosystems*. Routledge, 143–180.

[11] Mohammed El-Haii, Maroun Chamoun, Ahmad Fadlallah, and Ahmed Serhrouchni. 2018. Analysis of Cryptographic Algorithms on IoT Hardware platforms. In *2018 2nd Cyber Security in Networking Conference (CSNet)*. IEEE, 1–5.

[12] Mohammed El-hajj, Maroun Chamoun, Ahmad Fadlallah, and Ahmed Serhrouchni. 2017. Analysis of authentication techniques in Internet of Things (IoT). In *Cyber Security in Networking Conference (CSNet), 2017 1st*. IEEE, 1–3.

[13] M. El-Hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni. 2019. Ethereum for Secure Authentication of IoT using Pre-Shared Keys (PSKs). In *2019 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. 1–7. https://doi.org/10.1109/WINCOM47513.2019.8942487

[14] Mohammed El-Hajj, Ahmad Fadlallah, Maroun Chamoun, and Ahmed Serhrouchni. 2019. A survey of internet of things (IoT) Authentication schemes. *Sensors* 19, 5 (2019), 1141.

[15] Youssef El Housni. 2018. Introduction to the Mathematical Foundations of Elliptic Curve Cryptography. (Dec. 2018). https://hal.archives-ouvertes.fr/hal-01914807 Chapter III of the pre-published version of the book: Elliptic Curve Cryptography. The name of the commercial publisher will be added once it is published.

[16] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory* 31, 4 (1985), 469–472.

[17] Steven D Galbraith, Florian Hess, and Nigel P Smart. 2002. Extending the GHS Weil descent attack. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 29–44.

[18] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz. 2004. Comparing elliptic curve cryptography and RSA on 8-bit CPUs. In *International workshop on cryptographic hardware and embedded systems*. Springer, 119–132.

[19] Cameron F Kerry and Patrick D Gallagher. 2013. Digital signature standard (DSS). *FIPS PUB* (2013), 186–4.

[20] Mikko Kiviharju. 2017. On the fog of RSA key lengths: Verifying public key cryptography strength recommendations. In *2017 International Conference on Military Communications and Information Systems (ICMCIS)*. 1–8. https://doi.org/10.1109/ICMCIS.2017.7956481

[21] Neal Koblitz. 1987. Elliptic curve cryptosystems. *Mathematics of computation* 48, 177 (1987), 203–209.

[22] Arjen K Lenstra, Thorsten Kleinjung, and Emmanuel Thomé. 2013. Universal security. In *Number theory and cryptography*. Springer, 121–124.

[23] Julio Lopez and Ricardo Dahab. 2000. An overview of elliptic curve cryptography. (2000).

[24] Dindayal Mahto and Dilip Kumar Yadav. 2017. RSA and ECC: a comparative analysis. *International journal of applied engineering research* 12, 19 (2017), 9053–9061.

[25] Ueli M Maurer and Stefan Wolf. 2000. The diffie–hellman protocol. *Designs, Codes and Cryptography* 19, 2 (2000), 147–171.

[26] Victor S Miller. 1985. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*. Springer, 417–426.

[27] Pim Mulder. 2022. ecc-benchmarker. https://github.com/P1mguin/ecc-benchmarker.
[28] Sushant Nadkar. 2020. vcgencmd. https://github.com/sushantnadkar/vcgencmd.
[29] Yoav Nir, Simon Josefsson, and Manuel Pégourié-Gonnard. 2018. Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier. RFC 8422. https://doi.org/10.17487/RFC8422
[30] PowerAPI. 2019. pyRAPL. https://github.com/powerapi-ng/powerapi.
[31] Ronald L Rivest, Adi Shamir, and Leonard Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM* 21, 2 (1978), 120–126.