Traditional Public-key Cryptosystems and Elliptic Curve Cryptography: A Comparative Study

Maria Isaura Lopez, Ayad Barsoum*
St. Mary's University, San Antonio, TX (USA)
Emails: mlopez87@stmarytx.edu; abarsoum@stmarytx.edu

Abstract — The need to establish safer communication channels in a world where technological development is progressing in leaps and bounds is indispensable. Thus, implementing cryptographic algorithms, which are more complex to compromise, improves the possibilities of securing our sensitive data. In this paper, we analyze the algorithmic foundations and perform a comparative analysis of the traditional public-key cryptographic algorithms (e.g., RSA, ElGamal, Schnorr, DSA) and Elliptic Curve Cryptography with NIST recommended curves. In our study, we focus on six different security strengths: 80-, 96-, 112-, 128-, 192-, and 256-bit key sizes. Moreover, this study provides a benchmark among different curves (NIST, SEC2, and IEFT Brainpool) that can be used with various security levels. We study and compare the characteristics and performance of the traditional asymmetric algorithms and the Elliptic Curve algorithms for information security. The results obtained in this study will be graphically visualized through statistical graphs and tables with quantification response times.

Keywords: Cryptography, public key cryptosystems, elliptic curve cryptography

I. Introduction

Due to the rapid growth of digital communications and electronic data exchange, information security issues increase every day. Messages are exchanged worldwide, and publicly accessible computer networks must be protected and have protection mechanisms against manipulation [1].

The ability to access information 24 hours a day from any place in the world is a benefit of the Internet. This fact, however, poses some practical drawbacks, as computer networks and systems have become a new playground for cybercrime. Hackers can intercept electronic communications between two individuals or organizations, access enterprise computer systems, disseminate and sell industrial data, and destroy, modify, or alter mission-critical data, programs, or documents. Communication via the Internet is not inherently secure; encryption needs to be used to guarantee the security of such communications [1]. The science that deals with these aspects and the design procedures for encrypting confidential data is Cryptology (Cryptography and Cryptanalysis).

Two main types of encryption schemes are commonly used: symmetric or secret key cryptography and asymmetric or public-key cryptography [2].

Symmetric, or secret key, algorithms are characterized by being highly efficient and robust. They are so named because the same key is used for encryption and decryption. Symmetric algorithms are based on the use of secret keys that must be previously exchanged in a secure way among authorized parties [2]. That is why key distribution is a concern in symmetric key schemes.

Unlike symmetric algorithms, asymmetric algorithms have different keys for encryption and decryption. Therefore, they are also called public-key algorithms [2]. They allow us to eliminate the inconvenience of how to get the encryption key to the sender. In the case of asymmetric algorithms, a public key and a secret key are used. The public key is published in a location to which the general public has access, while the private one is kept secret. The two keys work together. Therefore, an interception of the public key is useless for deciphering a message since this requires the secret or private key.

The main concern of asymmetric algorithms is the key size. Keys must be large to provide security comparable to symmetric algorithms. They are also slower than symmetric schemes and produce larger encrypted messages [3].

One of the techniques used within public key systems is the elliptic curve cryptosystem/cryptography (ECC). Neal Koblitz and Victor Miller independently proposed ECC in 1985. ECC showed better security conditions, efficient use of computing resources, and reduced memory usage [4].

Elliptic curve-based cryptosystems provide the same security as those based on large number factorization, such as RSA, significantly reducing the number of digits used. Elliptic curves can be implemented with great hardware and software efficiency, and can compete in speed with systems like RSA. They are generally believed to be relatively safe, but studies are still trying to prove their safety features [5].

In this paper, we compare elliptic curve cryptography with the most commonly used public key cryptosystems such as RSA, ElGamal, and Schnorr. We set the advantages and disadvantages of using one over the other encryption system accentuating their main differences. This study also provides a benchmark among the curves (NIST, SEC2, and IEFT Brainpool) [6] [7] [8] that can be used with various security levels in Elliptic Curve Cryptography.

II. LITERATURE REVIEW

In past years, many authors have presented performance analysis studies between Elliptic Curve Cryptography and other methods of encryption.

Mahto *et al.* [9] implemented a security analysis between ECC and RSA. The implementation took place using three separate input size files and four security levels ranging from 80 to 144 bits. Their study found that RSA is very efficient in encryption but slow in decryption, while ECC is slow in encryption but very efficient in decryption. Still, overall, ECC is more efficient than RSA. In their conclusions, they also suggest that ECC may be most favorable for memory constraints devices like Smartphones or IoT devices.

Sann *et al.* [10] implemented a performance comparison between RSA, ElGamal, and ECC on security levels ranging from 80 to 112 bits, given the different algorithms' key sizes. They also used various files ranging from 25 to 500kb in size. In their study, they put much emphasis on the characteristics of the ElGamal algorithm. They claim that performance and speed depend on specific mathematical parameters such as the key length, ranging from 256 bits to an arbitrarily long number. It is also noted that a key length ranging from 1024 to 2018 bits is considered safe for the next 20 years. In their comparison between RSA and ElGamal, they pointed out that, while RSA encryption depends on the difficulty of factoring large integers, ElGamal encryption relies on the complexity of computing discrete logarithms in a large prime modulus. Their conclusions pointed out that RSA and ElGamal are comparable in speeds on encryption and decryption times. Compared to ECC, they demonstrated that ECC is faster than both and also gains increasing advantage on performance while the file size increased. The approximate ratios shown on their results were as follows 4:1 in encryption 8:1 on decryption.

Sinha *et al.* [11] implemented a performance-based comparison between RSA and ECC on security levels that range from 80 – 256 bits. They outlined in their study some of the advantages of ECC over RSA, such as computational overhead. They stated that it is roughly ten times more efficient than RSA. Sinha *et al.* [11] did not tabulate key generation and verification times. They mentioned that ECC key generation and verification are faster than RSA, and ECC offers considerable bandwidth savings over RSA. Their study found that ECC decryption time was slower than RSA but considerably faster at encryption time. In their conclusions, they mentioned that RSA Security on their website states that ECC is the technique in demand in the future, but RSA is well researched and trusted.

Gobi *et al.* [12], in their performance study they compared ECC using Koblitz curves against RSA and AES they found that at 163-bit ECC/1024-bit RSA security level, elliptic curve exponentiation for general curves over arbitrary prime fields is roughly 5 to 15 times as fast as an RSA private key operation. At 256-bit ECC/3072-bit RSA security level, the ratio has already increased to between 20 and 60, depending on optimizations. Based on their results to secure a 256-bit AES key, ECC-521 can be expected to be on average 400 times faster than 15,360-bit RSA. In their conclusions, they observed that ECC was the best compared to the RSA algorithm in terms of Authentication, based on execution time, speed, scalability, flexibility, reliability, security, and limitation essential for secure communication. They state that although the RSA algorithms were competent, RSA takes more time than ECC, and memory usage and encryption performance were better using ECC.

Saho & Ezin [13] did in-depth performance analysis of asymmetric cryptography. They used ECNR, ECDSA, ECIES, and RSA using Java libraries to do their study. The results obtained at key generation time found that the ECIES algorithm's key generation is much faster than the key generation in the RSA algorithm. The time increased exponentially as the security level increased for RSA. In the process of encryption, they noted that the RSA algorithm is faster than ECC. The gap between the two computational times was not showing a big gap. Therefore, from an encryption point of view, they concluded that the two algorithms presented the same performance level, but RSA was slightly better. In the decryption process, they found the opposite results, decryption by ECC algorithm was faster than RSA. In the process of key generation for digital signatures, they discovered no significant performance difference between ECNR and ECDSA. Still, for RSA, time exponentially increased when compared to the ECC as the security level increased, and they concluded that ECC key generation was optimal at this level. At signature generation, the analysis showed that ECDSA performed better than ECNR and RSA, and although all algorithms show the same performance at verification, RSA's performance was better than ECC. They concluded that ECC's computational resources were lower than RSA, thus offering a lower computational cost to benefit resource-constrained devices.

Other research studies have been done by Mahto & Yadav [14, 15], Durge & Hajare [16], Alese *et al.* [17], and Vahdati *et al.* [18].

III. CRYPTOGRAPHY FUNDAMENTALS

Cryptology is the discipline of cryptography and cryptanalysis and their interaction. Its aim is secrecy and confidentiality: the practice of keeping secrets, maintaining privacy, or concealing valuables. Figure 1 shows the main branches of cryptology.

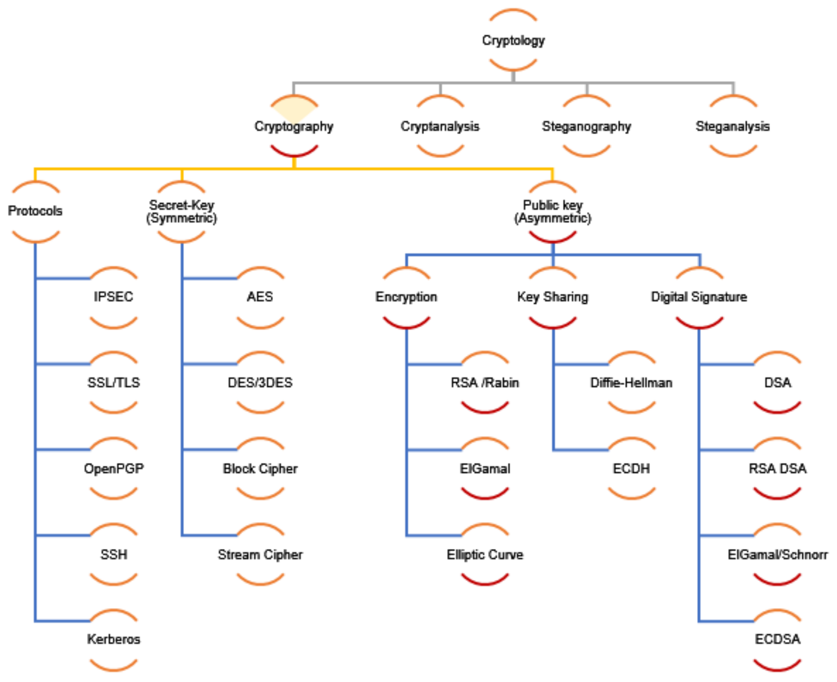


Figure 1: Main Branches of Cryptology

Cryptography is concerned with encryption, which is the process of converting plaintext (the original message) into ciphertext (the disguised message). The reverse process of encryption is called decryption (transforming the ciphertext into plaintext). Cryptanalysis is the discipline of deciphering a ciphertext without having access to the key. Cryptanalysis is mainly used by attackers to break the cryptosystems [19].

Symmetric Cryptography

In symmetric type encryption, the sender and receiver have the same key to encrypt and decrypt [3]. There are two types of symmetric encryption: stream cipher and block cipher. The difference between the two is that stream cipher is designed to encrypt data of arbitrary size that sometimes comes in the way of a stream. The encryption is done bit by bit (or byte by byte). Block cipher converts the plain text into ciphertext by taking a plain text's block of fixed-size data at a time. In both cases, the secret key must be shared among authorized users in a secure way. Figure 2 shows a typical symmetric encryption scheme. Examples of symmetric encryptions include DES (Data Encryption Standard), 3DES, AES (Advanced Encryption Standard), Blowfish, CAST (Carlisle Adams and Stafford Tavares), IDEA (International Data Encryption Algorithm), RC2, and RC4 [3].

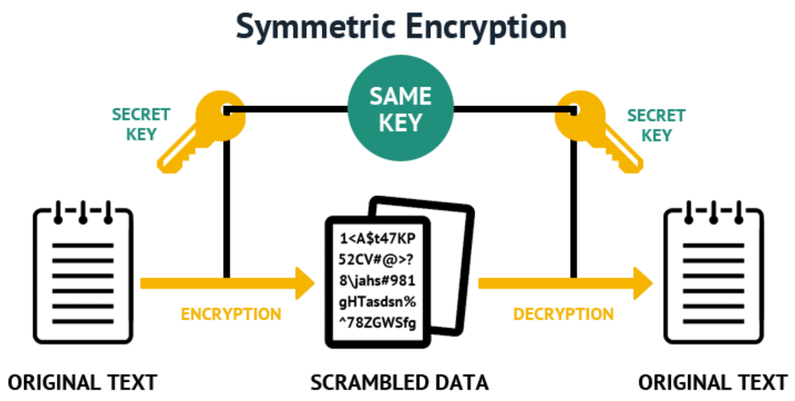


Figure 2: Symmetric Encryption

Asymmetric Cryptography

Asymmetric cryptography uses two complementary keys called the private key and the public key [3]. Figure 3 shows a typical symmetric encryption scheme

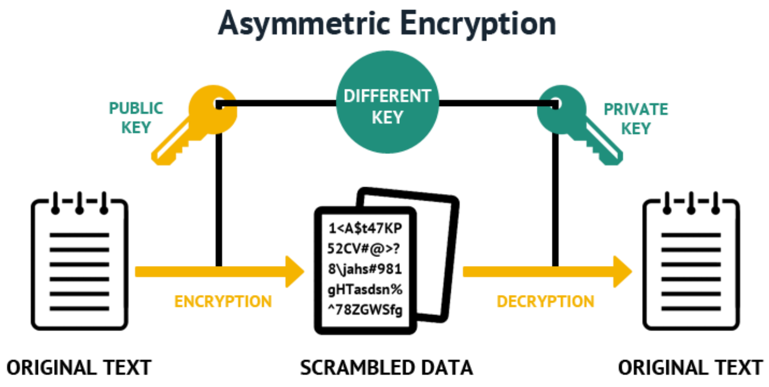


Figure 3: Asymmetric Encryption

For public key or asymmetric type encryption, the main feature is that the decryption key cannot be calculated from the encryption key. Private keys should be known only by their owner. In contrast, the corresponding public key may be released publicly. If a user wants to send a message to another user so that only the receiver can understand it, it will encode it with the receiver’s public key. The receiver uses his private key to decrypt the message. With an asymmetric system, any user can send an encrypted message to another using the receiver's public key. Still, only those who have the corresponding secret key can correctly decrypt the message [3]. Examples of asymmetric encryptions include RSA (Rivest, Shamir, Adleman), Diffie-Hellman, and ECC (Elliptic Curve Cryptography) [3].

Cryptanalysis

Cryptanalysis is the study of methods to obtain the content of encrypted information without being the user authorized. Typically this means getting the secret key. This practice is known as breaking the code [20]. The methods and techniques of cryptanalysis have changed drastically throughout Cryptology's history, adapting to a growing cryptographic complexity, ranging from the pencil and paper methods of the past, through machines like Enigma, to computer-based systems of today. Cryptanalysis can be done using: brute force attack, frequency analysis, differential cryptanalysis, and linear cryptanalysis [3].

In brute force attack, exhaustive test is done with all possible keys to decrypt a cryptogram. Frequency analysis is the most basic tool to break the classic ciphers. Since in all known languages, certain letters of the alphabet appear more frequently than others. For example, in English, there are often used vowels like E, O, A, or a consonant T. They occupy approximately 35% of the text [19]. On the other hand, there are some infrequent letters, such as Z or X, that their sum-frequency does not reach 1% [19]. The frequency analysis will reveal the original content if the encryption used cannot hide these statistics.

Differential cryptanalysis is based on the observation of ciphertext pairs, which have clear texts that have specific differences between themselves. If we study the evolution of these differences, we can realize that they can go through the 16 rounds of DES cryptogram since it is encrypted with the same key.

Linear cryptanalysis uses linear approximations to describe the actions of a block cipher. It is the technique that attempts to find linear approximations based on the transformations that a cryptosystem executes on a text. Differential and linear techniques are known as know-plaintext exploits because they use a known text to perform the attacks [3].

IV. Elliptic Curve Cryptography

Elliptic curves are not ellipses; they are set of points defined through a cubic equation. In general, cubic equations for elliptic curves have the form:

$$y^2 + axy + by = x^3 + cx^2 + dx + e$$

Where a, b, c, d, and e are real numbers, x and y have values over the real numbers. In general, an elliptic curve over the real numbers can be defined by the points (x, y), which satisfy the equation of an elliptic curve of the form: $y^2 = x^3 + ax + b$. Figure 4 shows an elliptic curves for $y^2 = x^3 - 4x + 0.77$ – $a=-4$ $b=0.77$

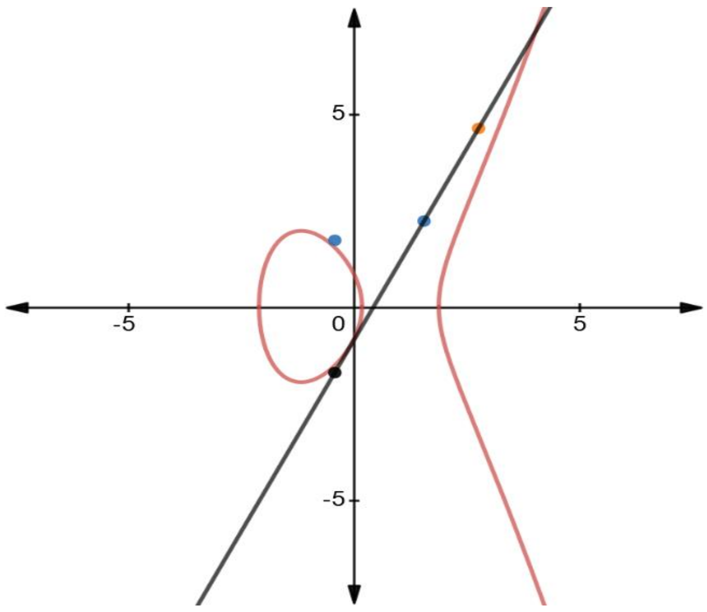
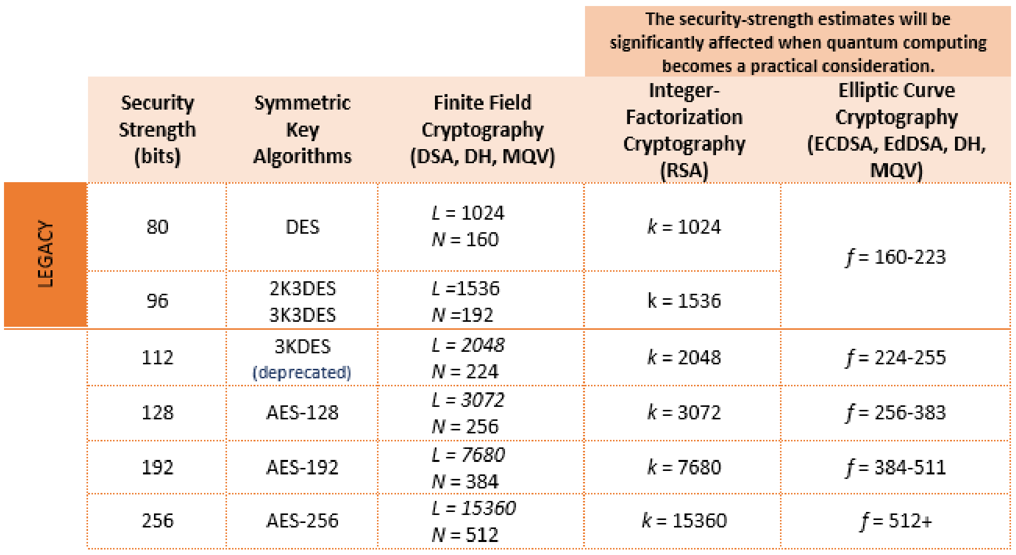


Figure 4: Elliptic Curve $y^2 = x^3 - 4x + 0.77$ – a=-4 b=0.77

Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC allows smaller keys compared to non-EC cryptography to provide equivalent security.

Table 1 provides estimated and comparable maximum-security strengths per key size or public and private key in the case of finite field cryptography [21].



				The security-strength estimates will be significantly affected when quantum computing becomes a practical consideration.	
	Security Strength (bits)	Symmetric Key Algorithms	Finite Field Cryptography (DSA, DH, MQV)	Integer-Factorization Cryptography (RSA)	Elliptic Curve Cryptography (ECDSA, EdDSA, DH, MQV)
LEGACY	80	DES	*L* = 1024 *N* = 160	*k* = 1024	*f* = 160-223
	96	2K3DES 3K3DES	*L* =1536 *N* =192	k = 1536	
	112	3KDES (deprecated)	*L = 2048* *N* = 224	*k* = 2048	*f* = 224-255
	128	AES-128	*L = 3072* *N* = 256	*k* = 3072	*f* = 256-383
	192	AES-192	*L = 7680* *N* = 384	*k* = 7680	*f* = 384-511
	256	AES-256	*L = 15360* *N* = 512	*k* = 15360	*f* = 512+

Table 1: Comparable security strengths of symmetric and asymmetric key algorithms

V.PERFORMANCE COMPARISON AND DATA ANALYSIS

Most public key systems today use 1024-bit keys for RSA. The National Institute for Standards and Technology (NIST) says that this key size was sufficient until 2010. Since then, it was recommended that the key length be increased to maintain the security level. This means that as we have more processing-capable equipment, we will need to increase the key size to provide an adequate level of security.

It should also be noted that some elements such as handheld computers, USB sticks, mobile phones, limited bandwidth networks, and the next ultra-mobile computers (UMPCs) need to protect their data regardless of whether they have the resources necessary to handle keys that are getting bigger every day to maintain reliable security [22].

Table 2 shows a comparison between the key sizes of RSA and ECC for different security levels.

Security Index	RSA Key Size [bits]	ECC Key Size [bits]
80-bit	1024	160
112-bit	2048	224
128-bit	3072	256
192-bit	7680	384
256-bit	15360	512

Table 2: Comparison of key size between RSA and ECC

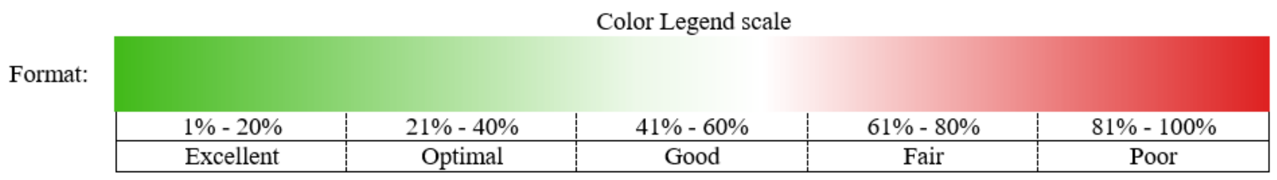
It is observed that the elliptic curve system requires smaller keys than RSA for different security levels. These key sizes greatly affect the computational times RSA and ECC.

In our study, we have implemented traditional public key systems and systems based on elliptic curves. The implementation has been divided into two categories with the algorithms shown in Table 3.

	Traditional	Elliptic Curve	ECC key generation
Key Generation	DSA ElGamal Schnorr RSA	ECDSA Schnorr ElGamal	ECC Individual pair ECC Keygen
Encryption – Decryption	RSA ElGamal	ECC	

Table 3: Categories of this study

The Computational Power of our study: Intel Core i5-3350P CPU @ 3.10ghz, 8GB RAM, and Windows 10 home 64bits. The used platform: PyCharm 2020.1.2 and Python version 3.8.0. Table 4 shows the color scale that we used to measure our results.



Color Legend scale

Format:

1% - 20%	21% - 40%	41% - 60%	61% - 80%	81% - 100%
Excellent	Optimal	Good	Fair	Poor

Table 4: Color scale to measure results on tables

Key Generation

Figure 5 and Table 5 allows us to see the difference between ECC key generation times vs. Traditional cryptography.

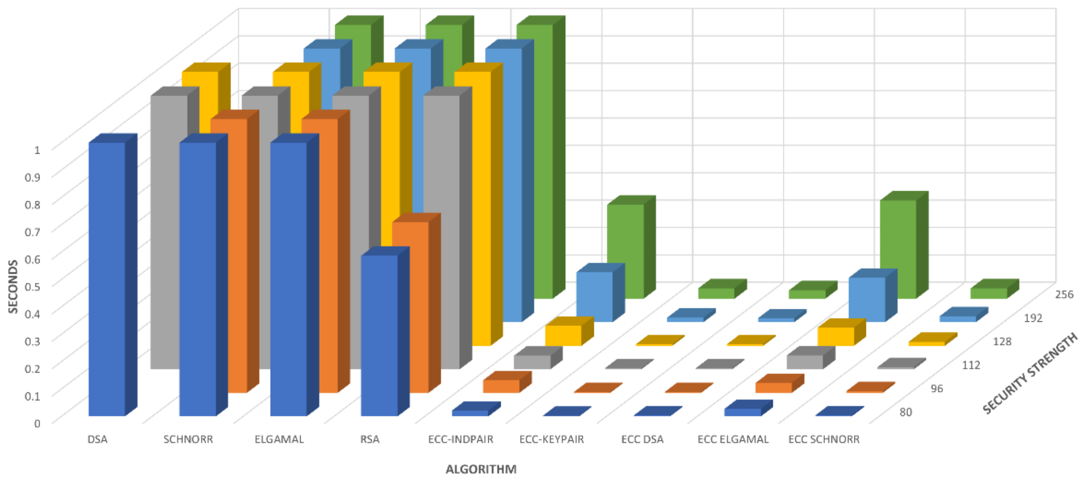


Figure 5: Key Generation - All Algorithms

Average of Seconds	Key Strength					
Algorithm	80	96	112	128	192	256
DSA	4.024756908		37.62644392	228.8692261		
SCHNORR	1.168286085	4.162094682	6.779845655	47.87722653	968.3491119	9873.557607
ELGAMAL	2.128114727	13.0825071	181.8626009	893.4288153	1878.814974	3110.545372
RSA	0.58663559	0.623687983	5.935323	42.60560727	641.8927538	7903.985229
ECC-INDPAIR	0.021320025	0.045971394	0.049969355	0.072621822	0.182219903	0.343889952
ECC-KEYPAIR	0.002332369	0.002997716	0.003997326	0.00599575	0.016655922	0.037643274
ECC DSA	0.003122896	0.003130923	0.004247	0.005663017	0.014615933	0.030814211
ECC ELGAMAL	0.027648767	0.036310991	0.05030179	0.066626549	0.162565788	0.358444611
ECC SCHNORR	0.003789097	0.006495714	0.007953068	0.014156173	0.021903843	0.037809779

Table 5: Key Generation - All Algorithms

In Figure 5, we only use 1 second as a max measure to allow ECC to show their speed graphically. The first four rows of Table 5 correspond to traditional cryptography, and the last five rows to ECC.

If we need to generate a key that will allow for a 128-security strength with ECC, it will take barely 0.005 seconds, while RSA keygen will take 42 seconds. It is not that much, (you would think, is only about half a minute) but while generating 1 RSA key, we can generate roughly 8000 ECC keys. A Ratio of 1:8400.

A 256-security level Digital Signature using ECC Schnorr's algorithm will take 0.037 seconds, while on Schnorr, traditional cryptography will take more than 2 hours only to generate the key. A ratio of 1:267,000.

Traditional Cryptography – RSA and ElGamal – Encryption/Decryption

RSA and ElGamal are two algorithms, where their strength lies in the bit length used. The degree of difficulty in RSA lies in the factorization of large primes, while in ElGamal lies in the calculation of discrete logarithms. After testing, it is proven that RSA performs a faster encryption process than ElGamal. However, ElGamal decryption process is faster than RSA. RSA is a deterministic algorithm, while ElGamal is a probabilistic algorithm with distinct functions on their algorithms. That said, we can also prove that RSA performs better than ElGamal. We can also observe that its performance on encryption-decryption is better than RSA. We can observe from our results that there was a peak at ELGAMAL-1536 which was probably due to a computer process in the background as the performance on ELGAMAL-2048 goes back to a consistent speed. In Figure 6, we set the limit at 1600 seconds to observe with detail the lowest speeds. Table 6 shows the data points for RSA and ElGamal - Encryption /Decryption operations.

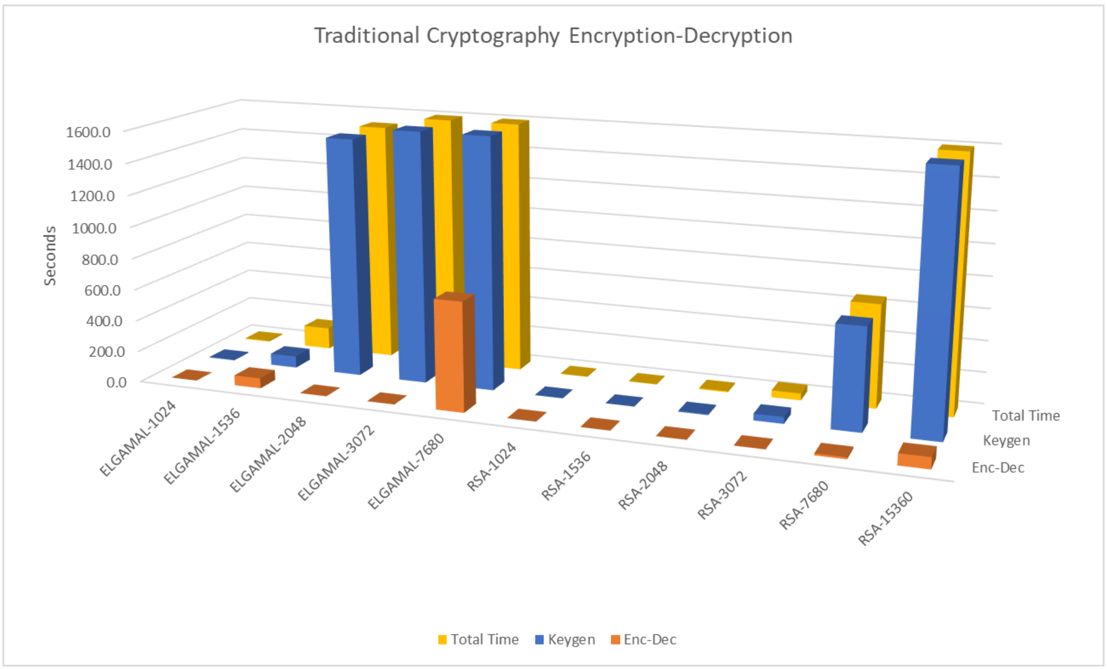
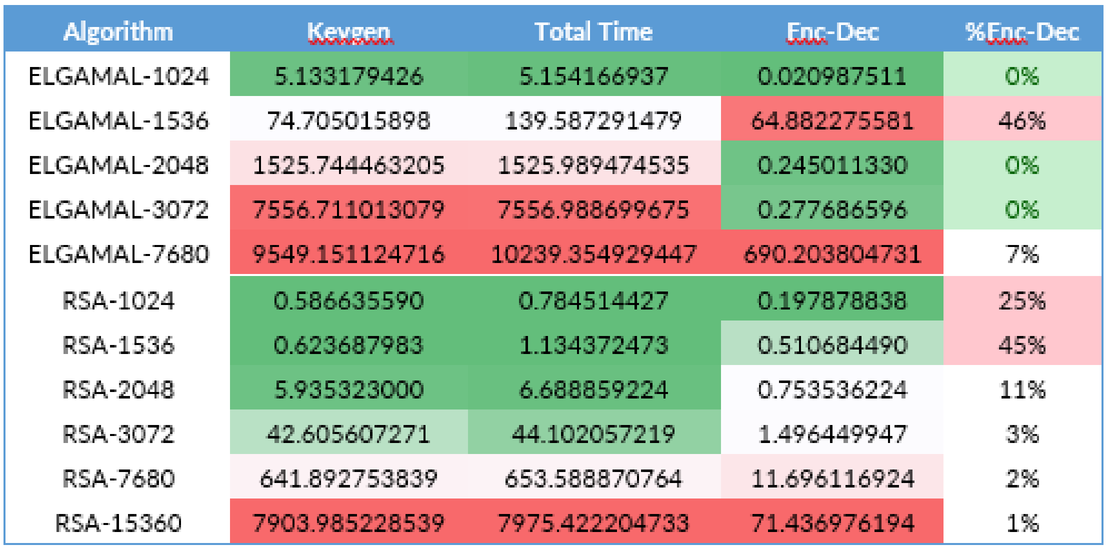


Figure 6: RSA and ElGamal - Encryption/Decryption



Algorithm	Keygen	Total Time	Enc-Dec	%Enc-Dec
ELGAMAL-1024	5.133179426	5.154166937	0.020987511	0%
ELGAMAL-1536	74.705015898	139.587291479	64.882275581	46%
ELGAMAL-2048	1525.744463205	1525.989474535	0.245011330	0%
ELGAMAL-3072	7556.711013079	7556.988699675	0.277686596	0%
ELGAMAL-7680	9549.151124716	10239.354929447	690.203804731	7%
RSA-1024	0.586635590	0.784514427	0.197878838	25%
RSA-1536	0.623687983	1.134372473	0.510684490	45%
RSA-2048	5.935323000	6.688859224	0.753536224	11%
RSA-3072	42.605607271	44.102057219	1.496449947	3%
RSA-7680	641.892753839	653.588870764	11.696116924	2%
RSA-15360	7903.985228539	7975.422204733	71.436976194	1%

Table 6: RSA and ElGamal - Encryption /Decryption - data

Elliptic Curve Cryptography – ECC and EC-ElGamal – Encryption/Decryption

ElGamal encryption using ECC can be described as an analog of the ElGamal cryptosystem that uses Elliptic Curve arithmetic over a finite field [23]. Therefore, there are not too many differences between ECC and ElGamal ECC. We can see that in the performance that both algorithms show. We experienced a slight performance improvement using ElGamal ECC with SEC2 curves, but overall the results are comparable. Once again, we can observe that the process employs about 10% of the time on the key generation and the rest in encrypting and decrypting the data. Figure 7 and Table 7 present the performance results for ECC and ElGamal using elliptic curves (EC- ElGamal).

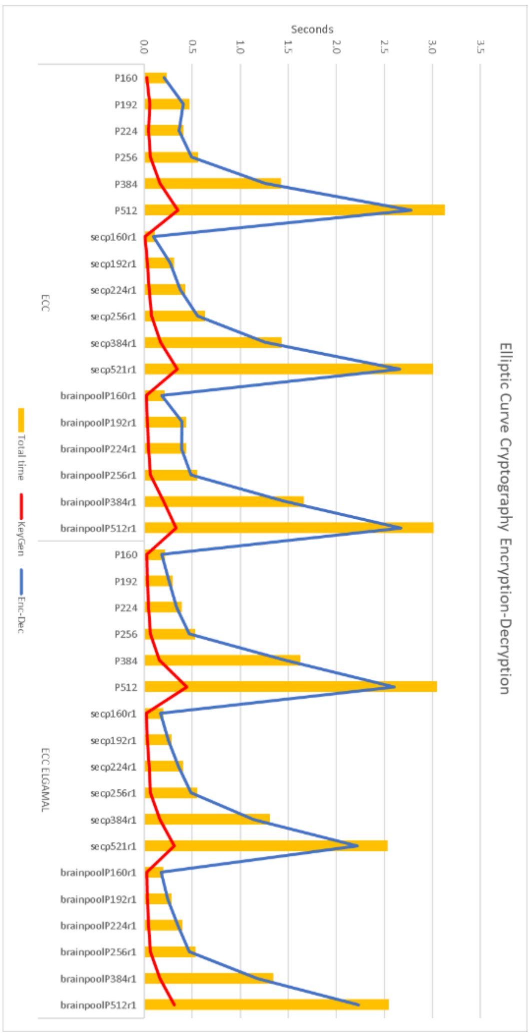


Figure 7: ECC and EC-ElGamal – Encryption/Decryption - graph

Curve	Keygen	Total time	Enc-Dec	%Enc-Dec	% Keygen
Elliptic Curve Encryption - Decryption					
P160	0.025983334	0.237740993	0.211757660	89%	11%
P192	0.061961412	0.471707821	0.409746408	87%	13%
P224	0.047970295	0.412746191	0.364775896	88%	12%
P256	0.068957329	0.563230753	0.494273424	88%	12%
P384	0.166895866	1.426116467	1.259220600	88%	12%
P512	0.350783825	3.131976843	2.781193018	89%	11%
secp160r1	0.012992144	0.112930298	0.099938154	88%	12%
secp192r1	0.037976980	0.311808109	0.273831129	88%	12%
secp224r1	0.051968575	0.427736759	0.375768185	88%	12%
secp256r1	0.079950571	0.635607958	0.555657387	87%	13%
secp384r1	0.174890041	1.435109854	1.260219812	88%	12%
secp521r1	0.346296787	3.007834435	2.661537647	88%	12%
brainpoolP160r1	0.024984598	0.211869240	0.186884642	88%	12%
brainpoolP192r1	0.037975788	0.436730146	0.398754358	91%	9%
brainpoolP224r1	0.049969196	0.437731743	0.387762547	89%	11%
brainpoolP256r1	0.068957567	0.557671309	0.488713741	88%	12%
brainpoolP384r1	0.204873800	1.662974834	1.458101034	88%	12%
brainpoolP512r1	0.334589243	3.010927677	2.676338434	89%	11%
ElGamal Elliptic Curve Encryption - Decryption					
P160	0.031979561	0.216864109	0.184884548	85%	15%
P192	0.036977768	0.296818733	0.259840965	88%	12%
P224	0.049968719	0.392755747	0.342787027	87%	13%
P256	0.064959764	0.532670021	0.467710257	88%	12%
P384	0.160900116	1.625991106	1.465090990	90%	10%
P512	0.443723440	3.051104546	2.607381105	85%	15%
secp160r1	0.024982929	0.197872400	0.172889471	87%	13%
secp192r1	0.036976814	0.287820101	0.250843287	87%	13%
secp224r1	0.050967932	0.406747818	0.355779886	87%	13%
secp256r1	0.066957951	0.553655386	0.486697435	88%	12%
secp384r1	0.163898468	1.307724714	1.143826246	87%	13%
secp521r1	0.317803621	2.534441471	2.216637850	87%	13%
brainpoolP160r1	0.025983810	0.202873707	0.176889896	87%	13%
brainpoolP192r1	0.034978390	0.284607649	0.249629259	88%	12%
brainpoolP224r1	0.049968719	0.401751757	0.351783037	88%	12%
brainpoolP256r1	0.067961931	0.535687447	0.467725515	87%	13%
brainpoolP384r1	0.162898779	1.342167854	1.179269075	88%	12%
brainpoolP512r1	0.313806772	2.546424627	2.232617855	88%	12%

Table 7: ECC and EC-ElGamal – Encryption/Decryption – data

.

Elliptic Curve Cryptography vs. Traditional Public-key Cryptography

Figure 8 shows side by side the average time of ECC vs. Traditional Public-key cryptography on a scale of 1 to 3500 seconds (1:3500 ratio). We can see the efficiency in time that ECC and ECC ElGamal encryption has over RSA and ElGamal.

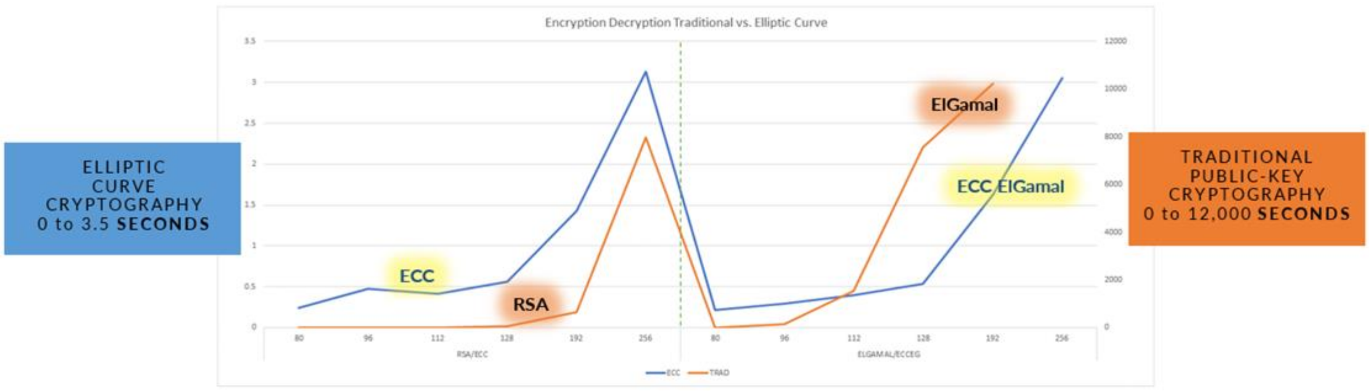


Figure 8: Elliptic Curve vs. Traditional Encryption/Decryption

VI. Conclusion

Cryptography provides us with confidentiality, authenticity, and non-repudiation for sensitive data. Cryptography comes with the cost of computational overhead, which can potentially compromise the viability of many systems.

Encryption algorithms rely on the tremendous difficulty of solving specific mathematical problems. These algorithms that were once effective are falling into the face of increased computational power of current hardware and the advances in cryptanalysis. As a countermeasure, we need to increase the key length of our current encryption schemes to maintain the same security level. This may be an issue for some resource-limited devices. Hence, comes the need for schemes based on Elliptic Curve Cryptography. ECC shows promising results over traditional asymmetric cryptography. ECC is efficient and is being used more and more these days. Elliptic curves can be used for digital signatures, key exchange, encryption-decryption, and end-to-end communications.

References

[1] Loredana., M. M. (2021). Pro Cryptography and Cryptanalysis creating advanced algorithms with c# and .net. S.l.: APRESS.
[2] Knospe, (2019) Knospe, H. (2019). A course in cryptography. Providence, RI: American Mathematical Society.
[3] Stallings, (2017) Stallings, W. (2017). Cryptography and network security: principles and practice (7th ed.). Boston: Pearson.
[4] Koblitz, (2012) Koblitz, N. (2012). A course in number theory and cryptography. New York: Springer.
[5] M. Subhashini, P. Srivaramangai, (2019) “Data Protection Using Elliptic Curve Cryptography,” International Journal of Computer Sciences and Engineering, Vol.07, Special Issue.02, pp.134-138, 2019.
[6] NIST, (2013) Digital Signature Standard (DSS) FIPS PUB 186-4. National Institute of Standards and Technology. DOI:10.6028/nist.fips.186-4
[7] Lochter, M. and J. Merkle, (2010) “Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation.” RFC 5639, DOI 10.17487/RFC5639, https://tools.ietf.org/html/rfc5639
[8] Brown, D. R. (2010) SEC 2: Recommended Elliptic Curve Domain Parameters (Vol. 2.0, Standards for Efficient Cryptography). Certicom. doi:https://www.secg.org/sec2-v2.pdf
[9] Mahto, D., Khan, D. A., & Yadav, D. K. (2016, June). Security analysis of elliptic curve cryptography and RSA. In Proceedings of the world congress on engineering (Vol. 1, pp. 419-422).
[10] Sann, Z., Soe, T. T., & Nwe, K. M. (2019). Comparison of Public Key Cryptography in Different Security Level. International Journal of Recent Development in Engineering and Technology, 8(12), ISSN 2347-6435.
[11] Sinha, R., Srivastava, H. K., & Gupta, S. (2013). Performance based comparison study of RSA and elliptic curve cryptography. International Journal of Scientific & Engineering Research, 4(5), 720-725.
[12] Gobi, M., Sridevi, R., & Rahini priyadharshini, R. (2015). A comparative study on the performance and the security of RSA and ECC algorithm. International journal of advanced network and application.
[13] Saho, N. J. G., & Ezin, E. C. (2020). Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm. https://hal.archives-ouvertes.fr/hal-02926106
[14] Mahto, D., & Yadav, D. K. (2018). Performance Analysis of RSA and Elliptic Curve Cryptography. IJ Network Security, 20(4), 625-635.

[15] Mahto, D., & Yadav, D. K. (2017). RSA and ECC: a comparative analysis. International journal of applied engineering research, 12(19), 9053-9061

[16] Durge, P., & Hajare, H. R. (2020). Comparative Analysis of RSA and ECC Algorithm. International Research Journal of Engineering and Technology Vol 7, Issue 6, ISSN:2395-0056.

[17] Alese, B. K., Philemon, E. D., & Falaki, S. O. (2012). Comparative analysis of public-key encryption schemes. International Journal of Engineering and Technology, 2(9), 1552-1568.

[18] Vahdati, Z., Yasin, S. M., Ghasempour, A., & Salehi, M. O. H. A. M. M. A. D. (2019). Comparison of ECC and RSA algorithms in IoT devices. Journal of Theoretical and Applied Information Technology, 97(16).

[19] Tilborg & Jajodia, (2011) Tilborg, Henk C. A. van, and Sushil Jajodia. Encyclopedia of Cryptography and Security. Springer, 2011.

[20] Katz and Lindell, (2020) Katz, J., & Lindell, Y. (2020). Introduction to modern cryptography. Boca Raton, FL : Chapman &amp; Hall/CRC.

[21] Barker, E. B. (2020) NIST Special Publication 800-57 Part 1 Revision 5: Recommendation for key management (Vol. Rev 5). Gaithersburg, MD: National Institute of Standards and Technology, Technology Administration. doi:https://doi.org/10.6028/NIST.SP.800-57pt1r5

[22] Wirdum, A. (2019, November 07). The Power of Schnorr: The Signature Algorithm to Increase Bitcoin's Scale And Privacy Retrieved September 19, 2020, from https://bitcoinmagazine.com/articles/the-power-of-schnorr-the-signature-algorithm-to-increase-bitcoin-s-scale-and-privacy-1460642496

[23] Sunuwar, R., &amp; Samal, S. K. (2015). Elgamal Encryption using Elliptic Curve Cryptography. Retrieved December 9, 2015, from https://cse.unl.edu/~ssamal/crypto/EEECC.pdf

Maria Isaura Lopez a graduate student at St. Mary's University, San Antonio, Texas enrolled in the MS Cybersecurity degree program. She is the information system technology administrator at St. Mary's University.
She received her MS in computer science from Texas A&M international university. She received her B.A. in computer/information technology administration and management from Autonomous University of Tamaulipas in Mexico.

Ayad Barsoum is an Associate Professor in Computer Science Department at St.Mary's University, San Antonio, Texas. He is the Graduate Program Director of MS in Cybersecurity. Dr. Barsoum received his Ph.D. degree from the Department of Electrical and Computer Engineering at the University of Waterloo (UW), Ontario, Canada in 2013. He is a member of the Centre for Applied Cryptographic Research at UW.

He received his B.Sc. and M.Sc. degrees in Computer Science from Ain Shams University, Cairo, Egypt, in 2000 and 2004, respectively.

At the University of Waterloo, Barsoum has received the Graduate Research Studentship, the International Doctoral Award, and the University of Waterloo Graduate Scholarship. Dr. Barsoum has received "Amazon Web Services in Education Faculty Grant" for funding his research and teaching through using Amazon cloud infrastructure