Comparative Analysis of Cryptographic Algorithms in Context of Communication: A Systematic Review

Shahzad Ahmed[*], Tauseef Ahmed[**]

[*] Faculty of Secure Information Technologies, ITMO University, Russia
[**] School of Information Technologies, Tallinn University of Technology, Estonia

DOI: 10.29322/IJSRP.12.07.2022.p12720
http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

Abstract- Currently, there are billions of IoT devices connected to each other through the internet, manufacturers and IT engineers uses modern cryptographic tools to protect the networks and the devices from attacks. Communication is an important domain in this modern digital era. Nowadays, fast and reliable communication without breaking continuity between the IoT devices is a big challenge, because there are many cryptographic algorithms have been invented. So, it is difficult to determine which one is better and compatible for the concerned device and useful for communication purposes. We tried in this research paper to fill this gap by making a comparative analysis of frequently used cryptographic algorithms. In this research, we compared 23 algorithms with parameters, such as; key size, message size and execution time.

Index Terms- Cryptographic algorithms, Communication, Information Security, Symmetric, Asymmetric

I. INTRODUCTION

By 2023, almost 66 percent of the global population comprises of the internet access [1]. The internet users will be 5.3 billion by 2023 (two-thirds of the global population), up from 3.9 billion in 2018 (51 percent of the world population). By 2023, the number of IP-connected devices will be more than three times the global population, 3.6 networked devices per capita, compared to 2.4 devices per capita in 2018, and total number of 29.3 billion associated devices, compared to 18.4 billion in 2018.

Cryptography is used in all parts of digital technology through security, confidentiality, integrity and authenticity. The standard search strategy was applied in this paper. The seven popular digital libraries, such as; Web of Science, Google Scholar, ACM Digital Library, SCOPUS, IEEE Xplore, Science Direct and Springer were used to search the related paper. Mendeley was used as a reference manager in this research. The attacks and the mitigations of the attacks in the cryptographic algorithms is out of the scope of this paper. This result and comparison analysis is useful for the organizations, manufacturers, IT Engineers and research community to apply the best suitable cryptographic algorithms for communication purposes. The authors considered 23 well-known algorithms in one paper with the comparison of key size, message size, and execution time.

A. Related Work

Darpan et al. have discussed in a review article about identity-based cryptography techniques and applications. The drawbacks and benefits of ID based cryptography have been shown in the research [2].

Advantages:

- It has a limited number of public/private key sets, which means when all users provided the secret keys, the central party can destroy the secret, and the obligation of the central party is to deal with the keys, and that will no done in the identification based encryption schemes.
- Public key authenticity is guaranteed and also the private key is secure in the secret (confidentiality, integrity, and authenticity).

Disadvantages:

The disadvantage is that if the Private Key Generator (PKG) is compromised, all the messages will be compromised, and the messages which are protected by the public-private key pair and used by the server are also compromised.

The researchers have shown the communication cost in bits and forwarded messages of the proposed protocol and the comparison of other protocols is shown in figure 1 [3].

communication cost comparison.

	Forwarded messages	Communication costs in bits
Fouda et al. (2011)	3	3744
Chim et al. (2011)	3	4448
Sule et al. (2012)	3	4416
Mahmood et al. (2016)	2	4768
Li et al. (2017)	2	2752
Proposed	2	1152

- T_{PM}: time for point multiplication $\cong$ 2.226 ms
- T_{HO}: time for hash operation $\cong$ 0.0023 ms

Figure 1 Communication cost comparison [3]

Alese et al. have discussed the comparative analysis of public-key encryption schemes [4]. The researchers tested the RSA Encryption Scheme and the Elliptic Curve ElGamal Encryption Scheme. The authors compared the size of encrypted data, encryption, decryption, and key generation duration. The research work described when the RSA and the ECC ciphers are compared; the ECC has involved much fewer costs as compared to the RSA. Additionally, the ECC has a lot of benefits because it has the ability to provide the same security level as compared to the RSA. However, it has some drawbacks that may hide its attractiveness as a lack of maturity.

Nilesh et al. have discussed the RSA and the Diffie-Hellman [5]. The researchers discussed in detail both the protocols, their steps of the process, the limitations of both protocols, supportive software, and hardware. It is discussed that both algorithms behave differently. The RSA uses the public and private two keys for authentication, and it is more secure as compared to the Diffie-Hellman. The Diffie Hellman cannot authenticate two parties with the man-in-the-middle attack. According to the authors, both algorithms could be edited for their improvement in performance for future prospects.

Shaina et al. have discussed the comparing results of various asymmetric cryptography algorithms using MATLAB [6]. The authors compared the RSA and the ElGamal algorithms and introduced the modification of the ElGamal and the RSA cryptosystems based on the Integer Factorization Problem (IFP) and Discrete Logarithm Problem (DLP).

The authors in [6] compared the time execution between the RSA and the ElGamal algorithms, and after that they implemented the result by using MATLAB. They have proposed a method that is a combination of both the DLP and IFP algorithms.

Antonio et al. have discussed transaction oriented text messaging with Trusted-SMS [7]. The authors have introduced the Trusted-SMS system. The mobile devices will change to the Personal Trust Devices (PTD), which will allow them to send and receive private information from and to the PTD. In [7], it is proposed that this system will be reliable for commercial transactions, bureaucratic delegation, etc.

The existing research has not focused to compare all the frequently used algorithms in one paper in context of communication, and our research filled this gap to guide the suitable communication algorithm.

B. Contributions

The contributions of the research are as follows:

- Research and study the existing literature on comparative cryptographic algorithms
- Compare and analyze the frequently used cryptographic algorithms
- To find the best and most suitable communication algorithm
- To ensure the continuity and scalability of the IoT devices

The rest of this paper is organized as follows; the authors have compared the algorithms on the basis of key size, execution time, and message size in Section 2. In Section 3, the analysis of the experimented results is discussed. Conclusion and future research direction are presented in Section 4.

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

II. COMPARISON OF CRYPTOGRAPHIC ALGORITHMS

1) Diffie-Hellman

This algorithm was invented by Whitfield Diffie and Martin Hellman, so this algorithm's name was derived from their inventor's last names [8]-[9]. The Diffie Hellman (DH) is based on key exchange technique of securely transferring keys on a public network. Cryptographic keys are not essentially transferred, but derived in the same way.

If Alice and Bob want to connect, first they need to consent on (base g) where ($0 < g < p$).

Alice picks a secret integer a which is her private key and then calculates g^a mod p and that is Alice's public key. Bob picks his private/secret key b and calculates his public key by using the same process.

After that, Alice and Bob send their public keys to each other. Now, Alice has a and Bob's public key which is g^b mod p. Alice is unable to calculate the value of b from Bob's public key, because it is a hard mathematical problem which is known as the discrete logarithm problem. But, Alice can calculate $(g^b)^a$ mod p = g^{ab} mod p.

Bob knows b and g^a, therefore he can compute $(g^a)^b$ mod p = g^{ab} mod p. So, both Alice and Bob know a shared secret g^{ab} mod p. An eavesdropper Eve who was listening in on the communication knows p, g, Alice's public key (g^a mod p), and Bob's public key (g^b mod p). Alice is unable to calculate the shared secret key from these values.

In static-static mode, both Alice and Bob keep their private/public keys over several communications. Therefore, the resulting technique shared secret will be the same every time. In the ephemeral-static method, one person will generate a new private or public key every time, hence a new shared private key will be created.

2) RSA Algorithm (Encryption and Signature)

A. RSA Signature

The idea of the digital signatures was to use public-key cryptography, so that anyone can verify the signature, but no one can generate it [8], [10]. Rivest, Shamir, and Adleman described the method of digital signatures, and the method is now widely used which is called RSA. This model is most popular and most proven and gained the broadest adoption by practice and standards bodies. The discrete logarithm and elliptic curve cryptography have also available with several standards, but they are not familiar and not as commonly used as RSA is in practice.

B. RSA Encryption Algorithm

Rivest, Shamir, and Adleman developed the RSA algorithm used by modern computers to encrypt and decrypt messages [11]. It is an asymmetric cryptographic algorithm. The algorithm is based on the fact that finding the factors of a large composite number is difficult; when the factors are prime numbers, the problem is called prime factorization. It is similarly a key pair of private and public keys initiator or we can say generator.

3) ElGamal Algorithm (Encryption and Signature)

A. ElGamal Encryption Algorithm

ElGamal encryption algorithm is an asymmetric cryptography encryption approach, and it uses for sending encrypted messages between two entities [12]-[13]. This approach is difficult to compute g^{ak}, while we are familiar with g^a and g^k.

B. ElGamal Signature Scheme

This scheme is based on the difficulty of computing the discrete logarithms [12]. The public key for the ElGamal signature scheme is k = (p,q,y), where p is a prime, q is a primitive element of Z^*_p and $y=q^x$ mod p, where $1 < x < p$ is the secret key.

To create the signature s of a message m we need to choose a random integer $r \in Z^*_{p-1}$ and calculate s = sig(m,r)=(a,b), where a = q^r mod p and b = $(m-ax)r^{-1}$ mod p-1. The signature s = (a,b) for the message m is valid if $y^a a^b \equiv q^m$ mod p.

4) Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is used for a public key, and it has a shorter key length as compared to other public-key algorithms like RSA [14] - [16]. ECC is an asymmetric public key cryptosystem and it provides equal security with a smaller key size as compared to RSA and other non-ECC algorithms. It makes use of elliptic curves. Elliptic curves are defined by some mathematical function- cubic function, e.g. $y^2 = x^3 + ax + b$ (this is the equation for degree 3).

5) Digital Signature Algorithm

The key for the Digital Signature Algorithm (DSA) is k = (p,q,r,x,y), where p is a large prime, q is a prime dividing p-1, r >1 is a q^{th} root of 1 in Z_p, ($r=h^{p-1/q}$ mod p, where h is a primitive element in Z_p), x is a random integer, such that; $0 < x < q$ and $y = r^x$ mod p. The values p, q, y, and r are made public, x is kept secret [8], [11].

To sign a message m we need to pick a random number k to such as extent that $0 < k < q$ and therefore gcd(k,q)=1. The signature of message m is s = sig(m,k)=(a,b), where a = (r^k mod p) mod q and b = $k^{-1}(m+xa)$ mod q, where $kk^{-1}=1$ mod q. The signature s = (a,b) is valid if ($r^{u1}y^{u2}$ mod p) mod q = a, where u_1 = mz mod q, u_2 = az mod q and z = b^{-1} mod q.

6) Fiat Shamir Signature Scheme

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

Fiat Shamir is used for creating digital signatures. Fiat Shamir heuristic is defined as non-interactive random oracle access and it's widely used for zero-knowledge proofs [12]. This scheme uses discrete logarithms.

7) Schnorr Signature Scheme

In the setup phase of the Schnorr signature scheme a Trusted Authority (TA) chooses a large prime p, a large prime q dividing p-1, an a $\in Z^*_p$ of order q, and a security parameter t such that $2^t<q$. These parameters p,q,a,t are made public [12]. TA creates a secure digital signature approach with a secret signing algorithm sig_{TA} and a public verification algorithm ver_{TA}.

Then in the certificate issuing phase, TA verifies Alice's identity by conventional means and forms a string ID(Alice) which continues her identification information. Alice chooses a secret random $1 \leq a \leq q-1$ and computes $v = a^{-a} \pmod p$ and sends v to the TA. TA generates signature $s = sig_{TA}(ID(Alice),V)$ and sends c (Alice)=(ID(Alice),V,S) back to Alice as her certificate.

A. Identification Protocol steps [12]

i. Alice chooses a random commitment $0 \leq k < q$ and computes $Y = a^k \pmod p$.
ii. Alice sends to BY and her certificate C(Alice)=(ID(Alice),V,S).
iii. Bob verifies the signature of TA.
iv. Bob chooses a random challenge $1 \leq r \leq 2^t$ and sends it to Alice.
v. Alice computes the response $y = (k+ar) \pmod q$ and sends it to Bob
vi. Bob verifies that $Y \equiv a^y v^r \pmod p$.

8) Ciphertext-policy attribute-based encryption (CP-ABE)

The ciphertext communicates with the access structure and the key communicates with the set of attributes [16]. By using public parameters the user encrypts the data and decrypts the ciphertext with the attribute-based private key. The CP-ABE algorithm has four steps, setup, encryption, generating a key, and decryption, as follows:

Step 1: In the setup process, it results the public key PK and a master key MK.

Step 2: In the encryption process, it inputs a message m, as well as the access structure A and PK, and produces the ciphertext C.

Step 3: In the third step of generating a key, it inputs a set of characteristics S, MK, and PK, then produces a decryption key D.

Step 4: In the fourth step of decryption process, it inputs the key D, PK, C, and A. The user can decrypt and obtain the message m, if the numbers of characteristics that satisfy A in all the attributes relate to user D surpasses a specific threshold.

9) Lightweight Cryptography Algorithms

The purpose of a lightweight cryptography (LWC) algorithm is to compromise on a number of aspects, including resource consumption, performance, and cryptographic strength [9], [17]. The LWC is used smaller block sizes than traditional ciphers, and it also uses smaller key sizes. There are no particular requirements to fit in the LWC algorithms, however, block size, key size, code measures, clock cycles, etc. have paramount importance.

Lightweight algorithms use smaller circuitry, Read Only Memory, and Random Access Memory sizes, processing speed, power used by power harvesting devices, and power consumption for battery-powered devices, such as; cameras or sensors to adjust encryption algorithms for resource constraint environments like; SPECK, SIMON, PRESENT, and TWINE [18].

10) Blowfish Algorithm

The blowfish algorithm has two parts (i) the key expansion part which is used for transforming the variable key length to an array of several subkeys. The variable key length has advantages to this algorithm as brute force attack is not possible on it. (ii) data encryption which has 16 rounds of encryption steps and each consists of a permutation-based on key and another permutation, based on key and data [19]. The blowfish algorithm is also used for public-key and it is much faster than other algorithms like DES.

11) Advanced Encryption Standard (AES)

AES has such variations as AES128, AES192, and AES256 that are widespread symmetric block cipher algorithms [10], [19], [20]. AES uses 128, 160, 192, 224, and 256 bits symmetric keys. It combines plaintext with a provided key to calculate the ciphertext using a previous result. AES algorithms are also using block cipher modes of operation, such as; Cipher Block Chaining, Cipher Feedback, Electronic Code Block, etc. Those modes have different parameters and are meant to improve efficiency and provide stronger security [21] - [22].

12) Secure Hash Algorithm (SHA-256, SHA3-256, and SHAKE-256)

SHA-256 is a hash algorithm used in the latest SSL TLS v1.2 protocol [15], [23]. This algorithm is derived from a simpler cipher called Message Digest 4. The message digest formed by this algorithm can be used in software, firmware, and hardware to determine the message integrity [13]. SHA3-256 is an SHA-3 family cryptographic hash function. It is considered to be more secure as it has improved security features, such as; resistance to collision, pre-image and second pre-image attacks. SHA3-256 uses 1088 bits block size compared to 512 bits blocks of SHA-256 [24]. SHAKE-256 is an extendable output function of the SHA-3 family. Its main difference from SHA3 function is the output message length can vary depending on the requirements. The index 256 indicates the supported security level, not the digest length as for other hash functions [25].

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

AES-256, ChaCha20, and ChaCha20-Poly1305 have the largest key size equal to 32 bytes, while SHA3-256 and SHAKE-256 have the largest block size equal to 136 bytes.

13) Comparison between the RSA and the Diffie-Hellman

The RSA and the Diffie-Hellman both of them work differently but they used for the encryption and decryption of data [26] - [28]. The Diffie-Hellman can be integrated with the digital and public key certificates to prevent attacks. The Table 1 describes the comparison between the RSA and the Diffie-Hellman algorithms.

Table 1 Comparison of RSA and Diffie-Hellman

RSA	Diffie-Hellman
Very slow in key generation process	Cannot be used for asymmetric key exchange.
The RSA is slow in signing and decryption process.	Cannot be used for digital signature
Key is vulnerable to various attacks if poorly implemented.	The DH is vulnerable to man-in-the-middle attack and it could be used in Denial of Service attack.

14) Comparison between the RSA and the DSA

The RSA is an asymmetric algorithm, it has a digital signature, encryption, and key exchange functions, but as compared to the DSA, it is only used for digital signature schemes, it is not useable for key exchange and encryption. Table 2 is showing the difference between both the algorithms.

Table 2 Comparison of RSA – DSA

RSA	DSA
The RSA provides digital signatures, encryption, and key exchange.	The DSA is based only on digital signatures, and it has not the function of key exchanging and encryption.
The RSA provides public key method.	The DSA also provides public key method.
The RSA can be used for encryption and decryption	The DSA can only be used for signing and verification
The RSA, on the other hand, is fast at encryption than the DSA.	In the key generation process the DSA is faster than the RSA.
The RSA is a good choice for verification of the digital signatures.	The DSA is faster in decryption process. The DSA will be a good choice in the digital signing process as an encryption algorithm.

15) Key size comparison between the ECC, RSA, DSA, and DH

Table 3 describes the key size comparison between the ECC, the RSA, the DSA and the DH algorithms. The Elliptic Curve Cryptography is growing as a most trusted solution for providing security on embedded systems [29].

Table 3 ECC-RSA-DSA-DH [30]

Security strength/ Symmetric Key size (Bits)	**Key Size**	
	ECC Key size (Bits)	**RSA/ DSA/ DH key size** (Bits)
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

16) Comparison between the DSA, the ElGamal, and the Schnorr

The ElGamal signature works modulo a prime p and requires one modular exponentiation (for generation) or two (for verification) with exponents as big as p, and the signature is two integers modulo p. The Schnorr and the DSA differ in key size, both works in a subgroup, generally a 160-bit subgroup for a 1024-bit modulus. Both the algorithm produces six times smaller signatures as compared to the ElGamal, and also both the Schnorr and the DSA algorithms are six times faster than the ElGamal.

17) Comparison of the RSA, the DES, and the ECDSA

The RSA algorithm is used for encryption and signature scheme in the component of blockchain data transmission and multi-signature. Digital signatures are public key techniques that accustomed to validate communications sent on a public channel, and whoever has a sender's public key is able to verify it [31]. The Data Encryption Standard (DES) algorithm is a symmetric encryption algorithm [32]. It uses the same key for encryption and decryption operations. Also, it is vulnerable if the key is compromised, the datacan be decrypted. The Elliptic Curve Digital Signature Algorithm (ECDSA) is an Elliptic Curve Cryptography-based simulation of the DSA. In addition, the authors in [33] have compared the time taken by the RSA, DES and the ECDSA for encryption and decryption, and for signature and verification.

However, the DES is faster than RSA during the encryption and decryption process, and the RSA is significantly faster than ECDSA during the signature verification procedure. Overall, the RSA has lower latency than the DES and the ECDSA. Besides, the DES require to share the key, and both parties, the sender and receiver need to cope the key.

Salma et al. have discussed hardware Implementation of biomedical data encryption using FPGA [34]. In their research, the encryption process for the DES takes 400 seconds, and for the RSA it takes 300 seconds to encrypt the same amount of data in MATLAB. In the decryption process, the DES consumed 390 seconds and the RSA consumed 350 seconds to complete the decryption process of the same amount of the data in MATLAB. The calculation of 100 characters of encryption/decryption, and signing/verification is shown in Figure 2.

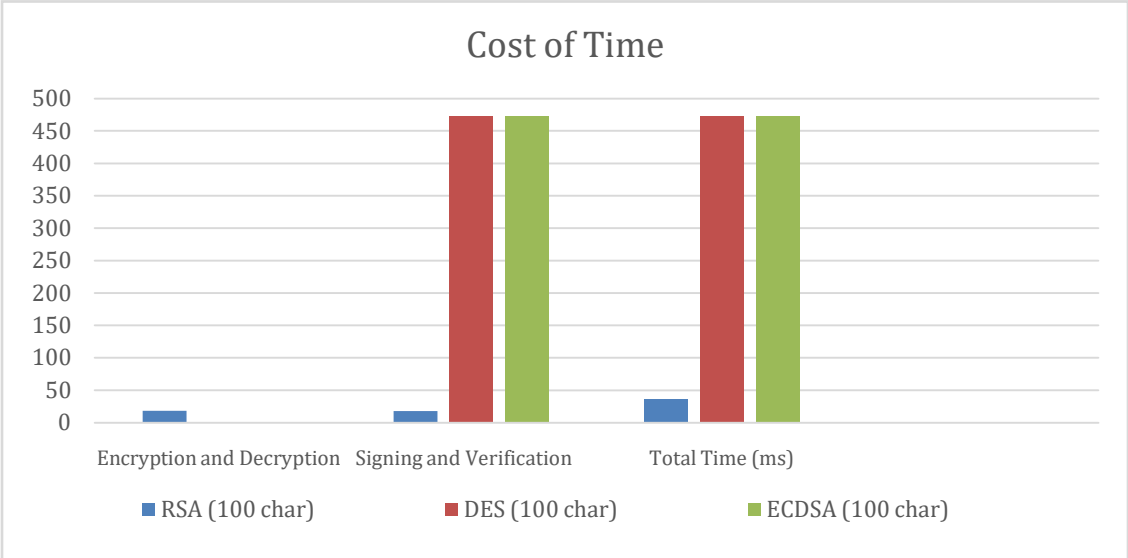

Figure 2 RSA - DES - ECDSA execution duration

18) Comparison of Digital Signatures

The key generation process of different algorithms the ECC, the RSA, and the DSA, with different key sizes, is compared with the execution duration process in Table 4.

Table 4 ECC - RSA – DSA [30]

Algorithms	Size	Sign (s)	Verify (s)	Sign (s)	Verify (s)
ECC	160 bit	0.0001	0.0003	12015.9	3081.8
RSA/ DSA	1024 bit	0.00031	0.000019	3220.6	53656.9
ECC	224 bit	0.0001	0.0002	10658.5	4770.9
RSA/ DSA	2048 bit	0.00204	0.000066	490.3	15257.5

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

ECC	256 bit	0.0002	0.0004	5729.1	2297.3
RSA/ DSA	4096 bit	0.017978	0.000268	55.6	3731.7

Figure 3 shows the signing and verification comparison between the ECC, the RSA and the DSA. The RSA and the DSA are taking less time in the signing process as compared to the ECC, but in the verification process, the ECC is taking very little time as compared to the RSA and the DSA, and also the ECC is using 160-bit size, on the other side, the RSA and the DSA are using 1024 bit size.

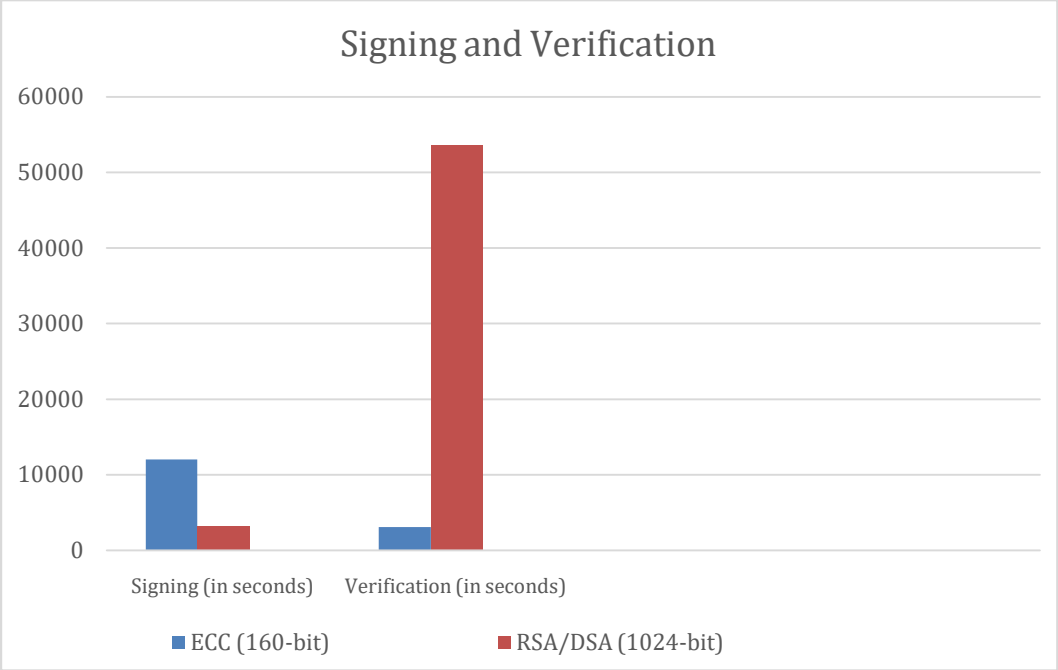

Figure 3 ECC - RSA - DSA signing and verification

19) Comparison of Execution Time

The ElGamal and the RSA algorithms are compared with parameters of execution duration with different message sizes in Table 5.

Table 5 RSA – ElGamal [6]

Message size	Operation	ElGamal	RSA
1 KB	Encryption	0.120 s	0.150 s
	Decryption	0.120 s	0.128 s
2 KB	Encryption	0.250 s	0.292 s
	Decryption	0.245 s	0.262 s
4 KB	Encryption	0.515 s	0.620 s
	Decryption	0.509 s	0.570 s
10 KB	Encryption	1.452 s	1.781 s
	Decryption	1.441 s	1.671 s
20 KB	Encryption	3.522 s	4.355 s
	Decryption	3.5.9 s	4.254 s
40 KB	Encryption	11.805 s	17.322 s
	Decryption	11.711 sec	22 sec

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

100 KB	Encryption	169 sec	195 sec
	Decryption	168 sec	250 sec

The authors have presented a comparison of the RSA, the Diffie-Hellman, the DSA, the ECC, the ElGamal, the DES, the ECDSA, and the Schnorr algorithm based on various criteria. Based on the aforementioned comparison, the RSA could be a good choice as compared to the Diffie-Hellman, because it is a public key cryptography approach, and it produces the digital signatures. ECC works on short key sizes as compared to the RSA, the DSA, and the DH. The Schnorr algorithm could be a suitable choice as compared tothe DSA and the ElGamal, because it provides a multi-signature scheme, it works on the short key size, and it is six times faster than the ElGamal.

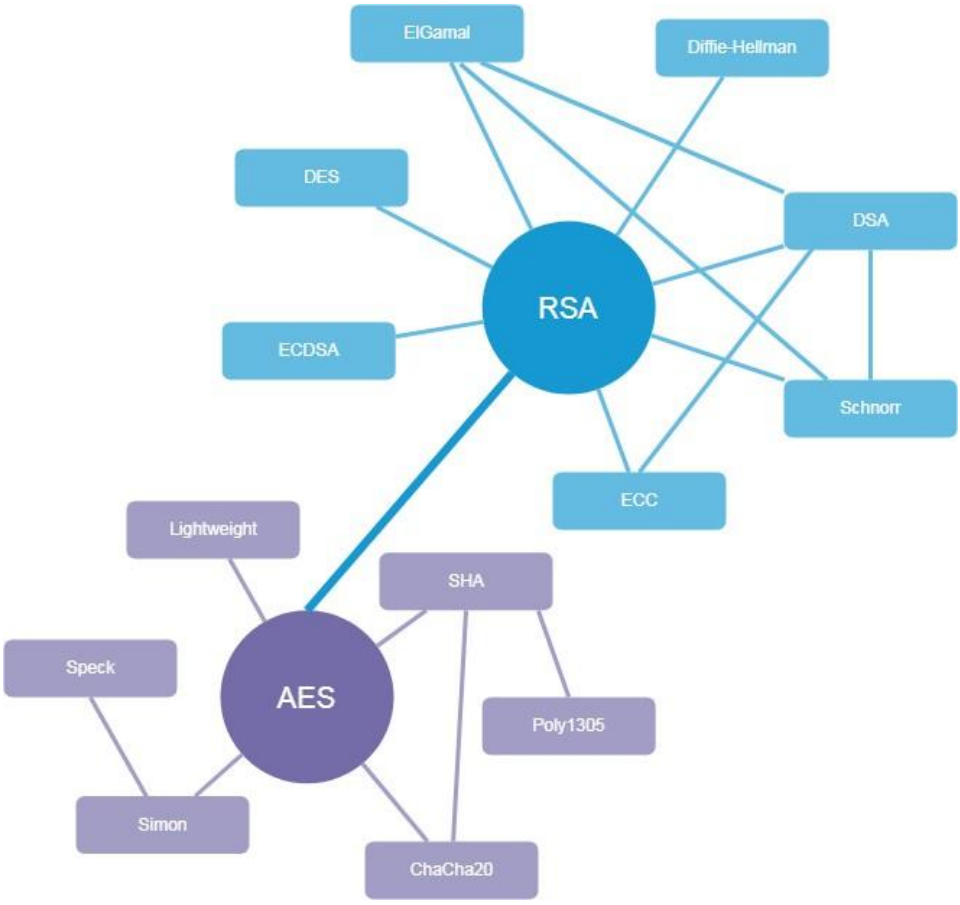

Figure 4 Comparison map

Figure 4 illustrates the comparison map of cryptographic algorithms. The authors focused to compare all the algorithms directly or indirectly with each other to ensure the quality of comparison and research.

III. EXPERIMENTAL ANALYSIS

The algorithms from Tables 6-8 are studied in the python programming language on a Ubuntu 20.04.1 LTS virtual machine with a base memory of 1024 MB, video memory of 16 MB, and a graphic controller VMSVGA. The process is done with 10 iterations of each algorithm, and the result is presented in the average of 10 iterations of each algorithm. In the key generation process shown in Table 6, the ElGamal takes 0.00305 seconds to process, and the DSA and the RSA are taking more time than the ElGamal. In the encryption and decryption process, as shown in Table 7, the ElGamal takes less time than the RSA in the prospect of encryption, and in decryption process the ElGamal takes more time than the RSA. When comparing the total time needed for encryption and decryption, the RSA is faster than the ElGamal. In the process of digital signatures, as shown in Table 8, the signing and verification process, the ElGamal is faster than all other algorithms in the verification process. However, the DSA is overall faster than the RSA, the ElGamal, and the Schnorr multi-signature.

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

Based on the results presented in the Tables 6-8, it is evident that the ElGamal is performing better in the key generation process, the RSA performs better in the encryption and decryption process, and the DSA performs better in the digital signature process.

The result analyzed in Tables 9-11 are experiments in the following environment:

Hardware: Monitor, Wireless Keyboard, Wireless Mouse, Raspberry Pi 3 Model B and Raspberry Pi Zero W, 32 GB Secure Digital Card, Charger 5V 2A, Universal Serial Bus flash drive for simpler algorithm transferring.

Additional Hardware for Raspberry Pi Zero W: High-Definition Multimedia Interface cable, mini HDMI to HDMI adapter and micro USB to USB type A adapter.

Operating System: Raspberry Pi 32-bit OS. The operating system is installed on a 32GB SD card using the Raspberry Pi Imager v1.6.1 application.

File: Created with ".txt" extension.

IDE: Thonny, a built-in python development environment is used for code execution.

As the result shows that the algorithms throughput increased in parallel with the file size, and there are no cardinal changes in connection to the device model as lower performance for the Zero W model is uniform across all cryptographic algorithms. The hash function SHA-256 kept showing the best results in every throughput test. AES-128, AES-192, and AES-256 showed similar and comparatively low results along the way. In the test with 1,048,576 bytes file for Raspberry Pi 3 Model B, AES-128 block cipher reached 8,975,997 bytes per second, while Simon, the lightweight block cipher, gathered 60,506,405 bytes per second.

Table 6 Comparison of Key Generation

Name of Algorithm	**Execution Time (average of 10 iterations)**
RSA	3.75461 s
DSA	0.0065 s
ElGamal	**0.00305 s**

Table 7 Comparison of Encryption and Decryption

Name of Algorithm	**Encryption: Execution Time (average of 10 rounds)**	**Decryption: Execution Time (average of 10 rounds)**	**Total Execution Time**
RSA	0.02269 s	**0.0513 s**	**0.07399 s**
ElGamal	**0.00132 s**	56.52932 s	56.53064 s

Table 8 Comparison of Digital Signatures (Signing and Verification)

Name of Algorithm	**Signing: Execution Time (average of 10 rounds)**	**Verification: Execution Time (average of 10 rounds)**	**Total Execution Time**
RSA	4.90917 s	0.01136 s	4.92053 s
DSA	**0.00143 s**	0.00536 s	**0.00679 s**
ElGamal	0.00347 s	**0.00426 s**	0.00773 s
Schnorr multi-sig	0.33162 s	0.47101 s	0.80721 s

AES-128, AES-192, AES-256: In comparison to the lightweight block algorithms, demonstrated significantly lower throughput. The AES algorithms are on average 5 times slower than lightweight algorithms. Though, the power consumption during encryption of

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

small files is similar to other algorithms, but twice as higher compared to Simon for encryption of large plaintext. Memory usage is not adapted to the needs of the resource constraint environment as it outputs the same size ciphertext.

SHA-256: This algorithm showed the best results of the throughput measurements among all algorithms there are no deviations from the average power consumption compared to others. Among hash functions, SHA-256 required the biggest memory space about twice more than the rest of the hash functions.

SHA3-256 and SHAKE-256: It demonstrated analogous test results. Throughput is noticeably smaller than SHA-256 had, while power consumption results are slightly better for the large size files.

Poly1305-AES: This performed well for the encryption of the large plaintext, had average power consumption and had a fixed output smaller than the SHA family.

ChaCha20: ChaCha20 showed good throughput and average power consumption results for a stream cipher. It produced ciphertext file sizes that are slightly bigger than the input, therefore, not adapted for use in a constraint environment.

ChaCha20-Poly1305: This showed better throughput and power consumption results than ChaCha20 and Poly1305 for the small sized plaintext as ChaCha20 it produces large ciphertext.

Speck and Simon: Speck demonstrated improved performance except in the cases when an extra-large plaintext is provided, in that case, Simon is more effective. The same situation is noticed during the power consumption tests, while memory usage is almost identical.

Table 9 Comparison of AES – SHA – Poly1305 – ChaCha20 – Speck – Simon

Algorithm Name	Key Size (byte)	Initialization vector (byte)	Nonce Size (byte)	Block size (byte)	Rounds
AES128	16	16	-	16	10
AES192	24	16	-	16	12
AES256	32	16	-	16	14
SHA256	-	-	-	64	64
SHA3-256	-	-	-	136	24
SHAKE-256	-	-	-	136	24
Poly1305-AES	16	-	16	16	-
ChaCha20	32	-	12	64	20
ChaCha20-Poly1305	32	-	12	-	-
Speck	16	-	-	16	32
Simon	16	-	-	16	68

Table 10 Analysis on file size 1024 bytes

Algorithm Name	Memory usage (byte)	Execution time (sec)		Throughput (bytes/sec)		Power consumption increase (%)	
		3 B	Zero W	3 B	Zero W	3 B	Zero W
AES-128	1024	0.00349	0.02307	293,410	44,387	87	43
AES-192	1024	0.00343	0.02278	298,542	44,952	84	36
AES-256	1024	0.00347	0.02316	295,101	44,214	77	43
SHA-256	64	0.000054	0.00020	18,962,963	5,120,000	77	36
SHA3-256	32	0.00040	0.00153	2,560,000	669,281	84	29
SHAKE-256	26	0.00041	0.00149	24,975,60	687,248	84	29
Poly1305-AES	16	0.00089	0.00544	1,150,562	188,235	81	36

ChaCha20	1032	0.00063	0.00243	1,625,397	421,399	84	21
ChaCha20-Poly1305	1024	0.00023	0.00089	4,452,174	1,150,562	90	39
Speck	39	0.00072	0.00351	1,422,222	291,738	84	29
Simon	39	0.00129	0.00505	793,798	202,772	81	29

Table 11 Analysis on file size 1048576 bytes

Algorithm Name	**Memory usage (byte)**	**Execution time (sec)**		**Throughput (bytes/sec)**		**Power consumption increase (%)**	
		3 B	Zero W	3 B	Zero W	3 B	Zero W
AES-128	1048576	0.11682	0.29897	8,975,997	3,507,295	87	43
AES-192	1048576	0.13226	0.33979	7,928,141	3,085,953	87	43
AES-256	1048576	0.14863	0.36398	7.054,942	2,880,862	81	36
SHA-256	64	0.01260	0.04819	83,220,317	2,175,920	87	36
SHA3-256	32	0.09432	0.21298	11,117,218	4,923,354	87	36
SHAKE-256	26	0.09465	0.215916	11,078,457	4,856,407	81	36
Poly1305-AES	16	0.02014	0.04972	52,064,349	21,089,622	84	29
ChaCha20	1048584	0.03299	0.08541	31,784,662	12,276,970	84	36
ChaCha20-Poly1305	1048576	0.05555	0.15778	18,876,256	6,645,810	84	36
Speck	39	0.01970	0.06182	53,227,208	16,961,760	81	29
Simon	39	0.01733	0.06212	60,506,405	16,879,845	39	36

According to the result, the authors analyzed that four algorithms are better than others, such as; Schnorr, RSA, Elliptic Curve Cryptography and ElGamal. The authors further try to prioritize them. ElGamal is expensive and the key size of it is larger. ElGamal is slow for signing the signature, and also it needs a random number. ElGamal has a disadvantage in the sense of message expansion factor, it is expanding by a factor of two-place while encryption which means that the ciphertext is two times larger than plaintext. ECC is fast in key generation, and also it has small key sizes, signatures, and cipher text, ECC is also fast for signatures. ECC is faster than RSA in the sense of signature process. But, ECC is difficult to implement securely; importantly it is the standard curve. The standards of the ECC are not the state of the art, especially ECDSA and another disadvantage of ECC is, it is signing a signature with a broken random number generator that has the chance of compromising the key. It may be costly, particularly in a binary curve, and ECC is slow in public key operations. RSA is more suitable to implement as compared to ECC. It is very easy to understand. RSA is similar in encryption and verification, and also it is the same in decryption and signing. It is deployed widely. But RSA is a large key size and the signing and decryption process is slower. RSA is very slow in the key generation process. If it is implemented poorly then it has a chance in the two-part key which may be vulnerable with GCD, and a middleman can attack. If we have a large data size to encrypt, RSA is very slow in this case. Currently, the two most popular approaches to digital signatures are discrete logarithm and elliptic curves. The reason for their popularity is that they have short keys in size in which security is not compromised and computational efficiency is better for both of them. Under the discrete logarithm approach, the ElGamal algorithm based signatures are often deployed. DSA is also a widely used algorithm that was patented but the National Institute of Standards and Technology (NIST) issued a royalty-free license in 1993. Besides DSA, Schnorr is also a great algorithm. It is not commonly used, because it was under patent until 2008. Schnorr provides better digital signatures than others. Its security and security proof is very strong. It is also faster and more efficient than other algorithms. It also has the feature that, it can combine two signatures in a very simple way which will later create a multi-signature. The feature of these multi-signatures is that, this approach gives the benefit of low accountability and high privacy and the key look like a single signature. The advantage of high privacy is that only the participant can view this policy and no third party can view it. There are advantages and disadvantages of low accountability. If there is a signing requirement in it, you cannot see the signers. So, it can be harmful if it is used as a financial multi- signature. And the advantage of it is, helpful for signatures in the ring and group, both of which depend on anonymity.

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

So, finally, the authors prioritized them, first Schnorr when choosing digital signatures, second RSA when choosing encryption, third ECC, and the fourth is ElGamal.

IV. CONCLUSION AND FUTURE RESEARCH

In this research, 23 algorithms have been discussed, compared, and analyzed with key size, message size, and execution time parameters. The authors examined the most familiar algorithms in the python programming language and discussed the result in this paper. The algorithms from Tables 9-11 were analyzed on the Raspberry Pi environment and discussed the result. We recommended the best suitable and fast cryptographic algorithms to fill the communication gap between the IoT devices over the internet and intranet.

The attacks on these cryptographic algorithms and how to mitigate them are out of the scope of this paper. In the future research direction, more cryptographic algorithms have been combined and analyzed, also hash function needs more analysis which is not discussed in detail in this paper.

REFERENCES

[1] "Cisco Annual Internet Report (2018–2023) White Paper." https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] D. Anand, V. Khemchandani, and R. K. Sharma, "Identity-based cryptography techniques and applications (a review)," *Proceedings - 5th International Conference on Computational Intelligence and Communication Networks, CICN 2013*, no. September, pp. 343–348, 2013, doi: 10.1109/CICN.2013.78.

[3] A. A. Khan, V. Kumar, and M. Ahmad, "An elliptic curve cryptography based mutual authentication scheme for smart grid communications using biometric approach," *Journal of King Saud University - Computer and Information Sciences*, no. xxxx, pp. 0–7, 2019, doi: 10.1016/j.jksuci.2019.04.013.

[4] B. K. Alese, E. D. Philemon, and S. O. Falaki, "Comparative analysis of public-key encryption schemes," *International Journal of Engineering and Technology*, vol. 2, no. 9, pp. 1152–1568, 2012.

[5] N. A. Lal, "A Review Of Encryption Algorithms-RSA And Diffie-Hellman," *International Journal of Scientific & Technology Research*, vol. 06, no. 07, pp. 84–87, 2017.

[6] P. Choudhary and C. T. Group, "Comparing Results of Various Asymmetric Cryptography Algorithms Using MATLAB Comparing Results of Various Asymmetric Cryptography Algorithms Using MATLAB," no. August, 2016.

[7] A. Grillo, A. Lentini, G. Me, and G. F. Italiano, "Transaction oriented text messaging with Trusted-SMS," *Proceedings - Annual Computer Security Applications Conference, ACSAC*, no. June 2014, pp. 485–494, 2008, doi: 10.1109/ACSAC.2008.43.

[8] L. J. Dali, "Survey of Cryptography Algorithms for Sub- Saharan Countries," no. January, pp. 1–20, 2021.

[9] P. Shah, M. Arora, and K. Adhvaryu, "Lightweight Cryptography Algorithms in IoT - A Study," *Proceedings of the 4th International Conference on IoT in Social, Mobile, Analytics and Cloud, ISMAC 2020*, no. October, pp. 332–336, 2020, doi: 10.1109/I-SMAC49090.2020.9243437.

[10] K. Ali, F. Akhtar, S. A. Memon, A. Shakeel, A. Ali, and A. Raheem, "Performance of Cryptographic Algorithms based on Time Complexity," *2020 3rd International Conference on Computing, Mathematics and Engineering Technologies: Idea to Innovation for Building the Knowledge Economy, iCoMET 2020*, pp. 2–6, 2020, doi: 10.1109/iCoMET48670.2020.9073930.

[11] M. Ashiqul Islam, A. A. Kobita, M. Sagar Hossen, L. S. Rumi, R. Karim, and T. Tabassum, "Data security system for a bank based on two different asymmetric algorithms cryptography," *Lecture Notes on Data Engineering and Communications Technologies*, vol. 53, no. November 2020, pp. 837–844, 2021, doi: 10.1007/978-981-15-5258-8_77.

[12] L. Notes and C. Protocols, "Lecture Notes Cryptographic Protocols," *Notes*, vol. 2, pp. 1–132, 2008.

[13] Nigel Smart, "Cryptography: An Introduction (3rd Edition) Nigel Smart".

[14] N. Josias *et al.*, "Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptography through RSA Algorithm To cite this version : HAL Id : hal-02926106 Comparative Study on the Performance of Elliptic Curve Cryptography Algorithms with Cryptog," 2020.

[15] S. K. Mousavi, A. Ghaffari, S. Besharat, and H. Afshari, "Improving the security of internet of things using cryptographic algorithms: a case of smart irrigation systems," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 2, pp. 2033–2051, 2021, doi: 10.1007/s12652-020-02303-5.

[16] J. Pan, J. Cui, L. Wei, Y. Xu, and H. Zhong, "Secure data sharing scheme for VANETs based on edge computing," *Eurasip Journal on Wireless Communications and Networking*, vol. 2019, no. 1, pp. 1–11, 2019, doi: 10.1186/s13638-019-1494-1.

[17] A. K. Jadoon, L. Wang, T. Li, and M. A. Zia, "Lightweight Cryptographic Techniques for Automotive Cybersecurity," *Wireless Communications and Mobile Computing*, vol. 2018, 2018, doi: 10.1155/2018/1640167.

[18] O. Toshihiko, "Lightweight cryptography applicable to various IoT devices," *NEC Technical Journal*, vol. 12, no. 1, pp. 67–71, 2017.

[19] R. Fotohi, S. Firoozi Bari, and M. Yusefi, "Securing Wireless Sensor Networks Against Denial-of-Sleep Attacks Using RSA Cryptography Algorithm and Interlock Protocol," *International Journal of Communication Systems*, vol. 33, no. 4, 2020, doi: 10.1002/dac.4234.

http://dx.doi.org/10.29322/IJSRP.12.07.2022.p12720

[20] I. Makarenko, S. Semushin, S. Suhai, S. M. Ahsan Kazmi, A. Oracevic, and R. Hussain, "A Comparative Analysis of Cryptographic Algorithms in the Internet of Things," *3rd International Science and Technology Conference "Modern Network Technologies 2020", MoNeTeC 2020 - Proceedings*, 2020, doi: 10.1109/MoNeTeC49726.2020.9258156.
[21] D. Blazhevski, "Modes of operation of the aes algorithm," *The 10th Conference for Informatics and Information Technology (CIIT 2013)*, no. Ciit, pp. 212–216, 2013.
[22] D. J. Bernstein, "The poly1305-AES message-authentication code," *Lecture Notes in Computer Science*, vol. 3557, pp. 32–49, 2005, doi: 10.1007/11502760_3.
[23] IBM, "Transport Layer Security - TLSv1.2 - IBM Documentation." https://www.ibm.com/docs/en/zvse/6.2?topic=openssl-transport-layer-security-tlsv12 (accessed Jul. 21, 2021).
[24] National Institute of Standards and Technology, "FIPS PUB 202 SHA-3 Standard : Permutation-Based Hash and," *NIST Federal Information Processing Standard*, no. August, 2015.
[25] S. Kelly and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec," May 2007, doi: 10.17487/RFC4868.
[26] StackExchange, "What's the difference between RSA and Diffie-Hellman? [duplicate]." https://crypto.stackexchange.com/questions/42180/whats-the-difference-between-rsa-and-diffie-hellman
[27] C. T.- Quora, "What is the difference between RSA and Diffie Hellman?".
[28] C. B.-C. Threat, "Question 132 : What's the difference between Diffie-Hellman and RSA".
[29] R. Afreen and S. C. Mehrotra, "A Review on Elliptic Curve Cryptography for Embedded Systems," *International Journal of Computer Science and Information Technology*, vol. 3, no. 3, pp. 84–103, 2011, doi: 10.5121/ijcsit.2011.3307.
[30] N. Z. Aitzhan and D. Svetinovic, "Security and Privacy in Decentralized Energy Trading Through Multi-Signatures, Blockchain and Anonymous Messaging Streams," *IEEE Transactions on Dependable and Secure Computing*, vol. 15, no. 5, pp. 840–852, 2018, doi: 10.1109/TDSC.2016.2616861.
[31] A. Faz-Hernández, H. Fujii, D. F. Aranha, and J. López, "A secure and efficient implementation of the quotient digital signature algorithm (qDSA)," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10662 LNCS, no. November, pp. 170–189, 2017, doi: 10.1007/978-3-319-71501-8_10.
[32] E. Edition, O. Systems, S. Edition, and B. D. Communications, *the William Stallings Books on Computer Data and Computer Communications , Eighth Edition*, vol. 139, no. 3. 2011. doi: 10.1007/11935070.
[33] R. Fotohi, S. Firoozi Bari, and M. Yusefi, "Securing Wireless Sensor Networks Against Denial-of-Sleep Attacks Using RSA Cryptography Algorithm and Interlock Protocol," *International Journal of Communication Systems*, vol. 33, no. 4, 2020, doi: 10.1002/dac.4234.
[34] S. M. S. A. E. T. E. M. E. Abo-Elsoud, "Hardware Implementation of Biomedical Data Encryption using FPGA," *International Journal of Science and Research (IJSR)*, vol. 3, no. 5, pp. 834–839, 2014.

AUTHORS

First Author – Shahzad Ahmed, Master of Information Security, mirshahzad2012@gmail.com
Second Author – Tauseef Ahmed, PhD, Lecturer at Tallinn University of Technology, Estonia, tauseef.ahmed@taltech.ee

Correspondence Author – Shahzad Ahmed, Master of Information Security, mirshahzad2012@gmail.com, +923001261007