International Journal on Emerging Technologies 11(1): 414-420(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

An Efficient and Lightweight Security Scheme for Big Data

N. Sirisha [1,2] and K.V.D. Kiran[1]
[1]Department of Computer Science and Engineering,
Koneru Lakshmaiah Education Foundation, Vaddeswaram (Andhra Pradesh), India.
[2]Department of Computer Science and Engineering,
MLR Institute of Technology, Dundigal, Hyderabad, India.

(Corresponding author: N. Sirisha)
(Received 07 October 2019, Revised 02 December 2019, Accepted 09 December 2019)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: In the contemporary era, most of the applications are operating online for exploiting benefits of Internet. The comprehensive connectivity among devices in the real world has enabled technologies like Internet of Things (IoT) where devices produce huge amount of data. It is one of the sources of big data where there are massive amounts of data is produced. Such data needs to be protected from malicious attacks. In other words, protecting critical digital infrastructure and its communications is to be given paramount importance. In the proliferation of tools used for data analytics, there is probability of data leakage and theft of data without the knowledge of data owner. Therefore, it is crucial to provide security to big data. Traditional cryptographic primitives are not viable for protecting large volumes of data. Therefore, there is need for lightweight approaches that can improve performance of networks besides safeguarding massive data transfer activities. Both data confidentiality and integrity are essential to protect big data. In this paper, we proposed a Lightweight Security Scheme (LSS) to achieve this. Elliptic Curve Diffie Hellman (ECDH) based lightweight security algorithm is defined. It has mechanism for secure key exchange and support smaller key sizes. A simplified approach is defined to compute modulus of a fractional number. Experiments are made to evaluate the proposed scheme. Empirical results revealed that the LSS scheme shows comparable performance over existing methods like RSA, DH and ECDH. For instance, when key size is 80, DH and RSA needed 1024 bits to have identical security while ECDH and LSS needed only 160 bits. Similarly, when data size is 500 MB, RSA needed 32.0845 seconds of time for data download while DH, ECDH and LSS needed 18.9721, 17.4262 and 16.9856 seconds respectively.

Keywords: Big data, big data security, light weight security protocol, DH, ECDH, RSA.

Abbreviations: DH, Deffie Hellman; ECDH, Elliptic Curve Deffie Hellman, RSA, Rivest-Shamir-Adleman.

I. INTRODUCTION

In the era of big data, it is essential to have security mechanisms that are feasible to the scale of data. With the Internet based innovations like Internet of Things (IoT) there is large volumes of data being produced. It is therefore imperative to cope with such data besides securing communications [9]. Cyber space now includes a broad spectrum of possibilities. It encompasses the connected devices of IoT as well besides networks of banks, telecommunications and digital infrastructure of any organization connected through Internet. This is the critical digital infrastructure that needs to be protected from cyber-attacks.

The traditional security mechanisms like RSA are not suitable for big data security. The rationale behind this is that they are not light weight. In the networks where there is origin of big data, security mechanisms need to be lightweight.

To overcome the aforementioned problem, many lightweight schemes came into existence as in [2, 4, 11, 12]. Usage of RSA in [19, 20] revealed that it as a widely used asymmetric encryption scheme. It is used for big data security as well. However, it has drawbacks in terms of overhead. In order to overcome this, lightweight key sharing methods such as Diffie-Hellman (DH) and Elliptic Curve Diffie Hellman (ECDH) came into existence. From the literature, it is understood that ECDH is lightweight and can cater to the needs of big data. It can be used as a lightweight scheme for secure key sharing along with mechanisms for encryption and decryption.

The drawbacks of RSA are overcome with ECDH. However, in this paper, we enhanced the EC part of the ECDH for improving the mechanism involved in finding modulus of a fractional number. This is important in devices with low power and involved in exchanging large volumes of data as witnessed in IoT use cases. Thus we proposed a Lightweight Security Scheme (LSS) based on ECDH and its proposed extension made in this paper. Empirical study is carried out with different big data workloads in terms of encryption, decryption, upload time and download time. Security analysis is made among RSA, DH, AES and ECDH. The results revealed that both ECDH and LSS outperformed the state of the art. LSS showed better performance over ECDH in some aspects. Our contributions in this paper are as follows.

– A security scheme known as Lightweight Security Scheme (LSS) is proposed based on ECDH key sharing mechanism.

– An extension to ECDH is made to be part of elliptic curve to improve performance of the proposed system.
– A prototype is built to have an empirical study in terms of security analysis, encryption, decryption, upload and download of large volumes of data.

The remainder of the paper is structured as follows. Section II reviews related literature on security to big data. Section III presents traditional cryptographi methods like RSA algorithm and its drawbacks when used with big data. Section IV presents the functionality of Diffie-Hellman and its shortcomings. Section V presents the proposed LSS which is the combination of ECDH and an extension to it. Section VI presents experimental results. Section VII concludes the paper and provides directions for future scope of the research.

II. RELATED WORK

This section reviews relevant literature on security primitives used for big data. Aljawarneh *et al.*, proposed an algorithm for multimedia content associated with big data [1]. It is an AES based scheme known as Feistel Encryption Scheme. They intended to run it in IoT applications in future. Liang *et al.*, proposed a security scheme by combing AES and RSA [2]. Lu *et al.*, focused on privacy to big data and provided useful insights [3]. Tang *et al.*, proposed a security scheme known as Privacy-preserving Fog-assisted Information Sharing (PFIS) scheme which is used to protect big data in healthcare system. They intended to improve it in future to reduce decryption cost [4].

Puthal *et al.*, proposed a selective encryption scheme for sensor data. It incorporated different strategies for confidentiality and data integrity with selective encryption. They intend to improve it to enhance symmetric key encryption [5]. Bai *et al.*, proposed a lightweight encryption scheme for Body Area Network (BAN) in healthcare domain. It has provision for dynamic key updating besides being energy efficient [6]. Bakhtiari *et al.*, proposed a lightweight encryption standard for big data [7]. Kadhim *et al.*, proposed CAST-256 block cipher for securing big data [8]. Usman *et al.*, proposed a secure lightweight mechanism for IoT environments [9]. Talbi and Bouhlel tried lightweight encryption for IoT communications [10]. Hong *et al.*, (2006) proposed Secure IoT (SIT) for lightweight encryption and decryption. They intended to improve algorithm to use with FPGA designs [11]. Al-Souly *et al.*, (2013) enhanced Transposition-Substitution-Folding-Shifting (TSFS) algorithm to improve decryption process by reducing errors [12].

Rajesh *et al.*, [13] proposed a secure and lightweight protocol of symmetric encryption to transfer content among IoT devices. It was known as Novel Tiny Symmetric Encryption Algorithm (NTSA). Big data security issues are explored along with methods to protect data while many encryption techniques used in big data are studied [14-15]. A light weight scheme for secure data sharing is made in [16] for Mobile Cloud Computing (MCC). As explored in [19], RSA a widely used algorithm for encrypting data. Other contributions related big data security in distributed environment include security with Apache Sentry [21], using KNOX [22], protection of big data from encroachments [23], protection from selective forwarding attacks [24], and rise of big data which advocates security aspect [25]. From the literature, it is known that light weight cryptography is essential for handling large volumes of data. There is need for equivalent security to RSA kind of algorithms with less number of bits. We find ECDH as a suitable candidate and enhanced it to define an algorithm named Lightweight Security Scheme (LSS).

III. ISSUES WITH TRADITIONAL CRYPTOGRAPHY

RSA (Rivest-Shamir-Adleman) is one of the cryptographic algorithms widely used. It is an example for asymmetric cryptography where different keys are used for encryption and decryption. It has mechanisms that are complex in nature. Its key size is more for adequate security. It is therefore considered heavy weight and not suitable for the parties or devices to exchange massive amounts of data [20]. The algorithm is as follows.

Step-1: Alice and Bob choose two secret prime numbers such as p=13 and q=19 where both are <n and $p \neq q$.
Step-2: A public parameter such as n=p$*$ q is computed followed by a primitive parameter like $\emptyset(n) = (p-1)(q-1) = 216$.
Step-3: An integer *e* is chosen by Alice such that $1 < e < \emptyset(n)$ and gcd($\emptyset(n), e$)=1
Step-4: A secret key computed by Bob using (e, $\emptyset(n)$) i.e. d $\equiv e^{-1} (\mathrm{mod}\ \emptyset(n))$. For instances if e=31, it results in d = 7
Step-5: The generated public key is P_U = e, n=31, 247 and the private key is P_R= {d, n} = 7, 216.

Algorithm 1: Rivest-Shamir-Adleman (RSA)

As presented in Algorithm 1, RSA enables two parties to have secure communications. Both Alice and Bob select secret numbers. Whereas public keys are provided to other parties. Encryption is carried out with the public key of receiver and then the encrypted message can only be decrypted by the private key of the receiver. It is made possible due to some distinct mathematical relationship between public key and private key pair associated with each party. There are many issues with RSA when it comes to securing big data. First, it was not designed to work with large volumes of data.

Second, the big key size of RSA increases complexity, computation time and communication time. Third, it is heavy weight and not feasible for big data security.

IV. DIFFIE HELLMAN KEY EXCHANGE

Public key cryptography causes overhead on systems due to its complex Public Key Infrastructure (PKI). However, key exchange is made simple with Diffie-Hellman scheme. Instead of sharing secret key, both parties involved in a crypto system compute secret key. The scheme is as follows.

Step-1: Both parties, Alice and Bob, agree to use two big prime numbers such as n & g and there is no need to keep them secret. They can be made public.
Step-2: Alice takes another big random number denoted as X which is kept secret. Then Alice computes A as $A = g^X \mathrm{mod}\ n$
Step-3: Then it is sent to Bob.
Step-4: Similarly, Bob takes another big random number denoted as Y which is kept secret. Then Bob computes B as $B = g^Y \mathrm{mod}\ n$
Step-5: Then it is sent to Alice
Step-6: Alice computes secret key as $K1 = B^X \mathrm{mod}\ n$
Step-7: Bob computes secret key as $K2 = A^Y \mathrm{mod}\ n$
Step-8: When K1 = K2, it is evident that key exchange is completed successfully.

Algorithm 2: Diffie Hellman key exchange scheme

As provided in Algorithm 2, the DH key exchange scheme helps in secure key exchange.

Two parties without having prior knowledge on each other also can exchange keys. The illustration of key exchange is as follows.

1. Alice & Bob agree two prime numbers n, g
   If n=11 and g=7 then
2. X=3 y=6
   $A = g^X \bmod n$ $B = g^Y \bmod n$
3. $A = 7^3 \bmod 11$ $B = 7^6 \bmod 11$
   A=343 mod 11 B=117649 mod 11
   A=343-341 B=117649-117645
   A=2 B=4
4. B=4 A=2
5. $K1 = B^X \bmod n$ $K2 = A^Y \bmod n$
   K1= 4^3 mod 11 $K2 = 2^6$ mod 11
   K1=64 mod 11 K2=64 mod 11
   K1=64-55 K2=64-55
   K1=9 K2=9

It has many advantages. It can be used in encryption methods. Key sharing is done safely and after key exchange, data transfer can be made in insecure channel as well. Finally, it resulted in K1=K2 reflecting successful key exchange.

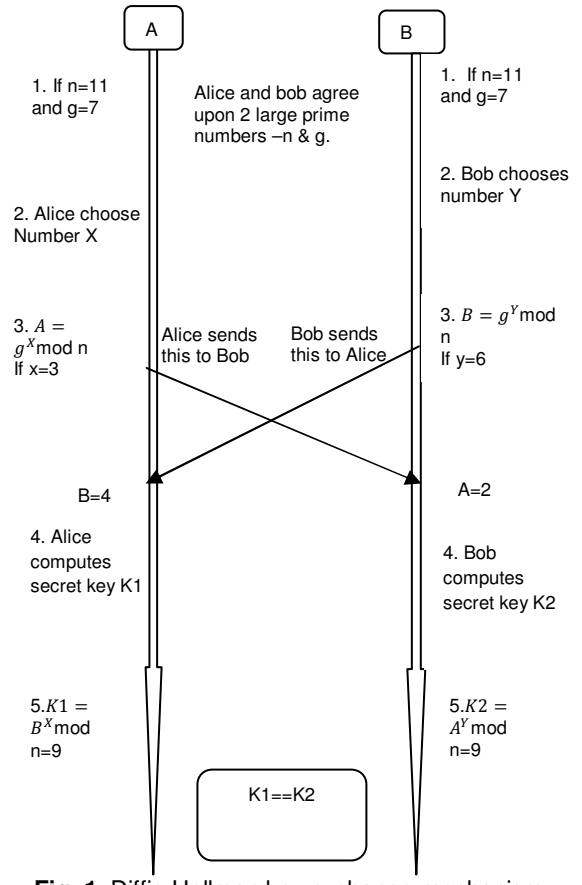

Fig. 1. Diffie Hellman key exchange mechanism.

As presented in Fig. 1, the DH key exchange protocol can be used to compute secret keys by both sender and receiver parties to overcome key exchange problem associated with symmetric key encryption techniques. However, it has certain limitations. It cannot be used for asymmetric key exchange effectively. It is not suitable for signing digital signatures. As it does not authenticate users, it is vulnerable to man-in-the-middle attacks. Fig. 3 shows the graphical flow of the DH algorithm. That is the reason, the proposed scheme in this paper is based on ECDH.

V. PROPOSED LIGHTWEIGHT SCHEME

A security scheme is proposed for big data. It is known as Lightweight Security Scheme (LSS). It is based on ECDH which is well known for key light cryptography. Since it is the combination of EC and DH, it becomes lightweight and robust and viable for massive data transfer among devices. Thus it is suitable for low power IoT devices as well. Before presenting the enhancement to the ECDH, the steps involved in ECDH is provided as in Algorithm 3. It depicts secure key exchange between two parties known as Alice and Bob.

Step-1: Alice chooses an integer denoted as $n_A$as her private key and $n_A < n$. Alice produces her public key with Eqns. 1-4. The public key is denoted as $P_A = n_A * G$ and base point is denoted as E_q(a, b).

Step-2: Similarly Bob chooses a secret integer denoted as $n_B$as private key and $n_B < n$. Then, Bob produces his privacy key $P_B = n_B * G$ and base point is E_q(a, b).

Step-3: Alice computes the secret key K=$n_A * P_B$while Bob computes secret key K=$n_B * P_A$.

Step-4: The generated key for Alice and Bob should be same because $K = nA{*}PB = nA{*}(nB{*}G) = nB{*}(nA{*}G)$.

Algorithm 3: ECDH scheme for key exchange

Usage of elliptic curves in key exchanges makes it lightweight. As explored in [18], elliptic curves are used as follows. First of all, a whole number denoted as *q* is considered. It may be either a number of the structure *2*m or a prime number. Parameters of elliptic curve such as a and b are used to define the value of (a, b). Then a base point denoted as $G = (x1, y1)$ is chosen in (a, b) with order *n* as a large value. The smallest integer like $nG = 0$ can define order n where the parameters are denoted as n and G and these are known to all participants of a crypto system. Then key exchange as per ECDH is carried out as illustrated in Fig. 3.

As can be seen in Fig. 2, it is evident that the key exchange is taken place between two parties without the need for any third party authority. Moreover, it is simple and lightweight. The shared secret key at the end for both parties is K= (4, 2).

Since both parties obtained same key known as shared secret key, the key exchange mechanism is successfully completed. The illustration of ECDH is as follows. Both parties such as Alice and Bob select E as curve denoted as $y^2 = x^3 + x + 6$ over z_7. A public base point denoted as B= (2,4) is chosen by Alice and Bob. Afterwards, $\alpha = 4$ is chosen by Alice and perform computation such as P= $\alpha B = 4(2,4) = (6,2)$. Then the P is sent by Alice to Bob. Alice does not disclose α and keeps it as a secret value. Then β=5 is selected by Bob and computation such as Q= $\beta B = 5(2,4) = (1,6)$ is made. Then Q is sent to Alice while the value of β is kept secret. Afterwards, Alice performs computation like K_A=$\alpha Q = 4(1,6) = (4,2)$ while Bob performs computation such as K_B=$\beta P = 5(6,2) = (4,2)$. Thus the shared key is established as K= (4, 2) which is same at both the parties.

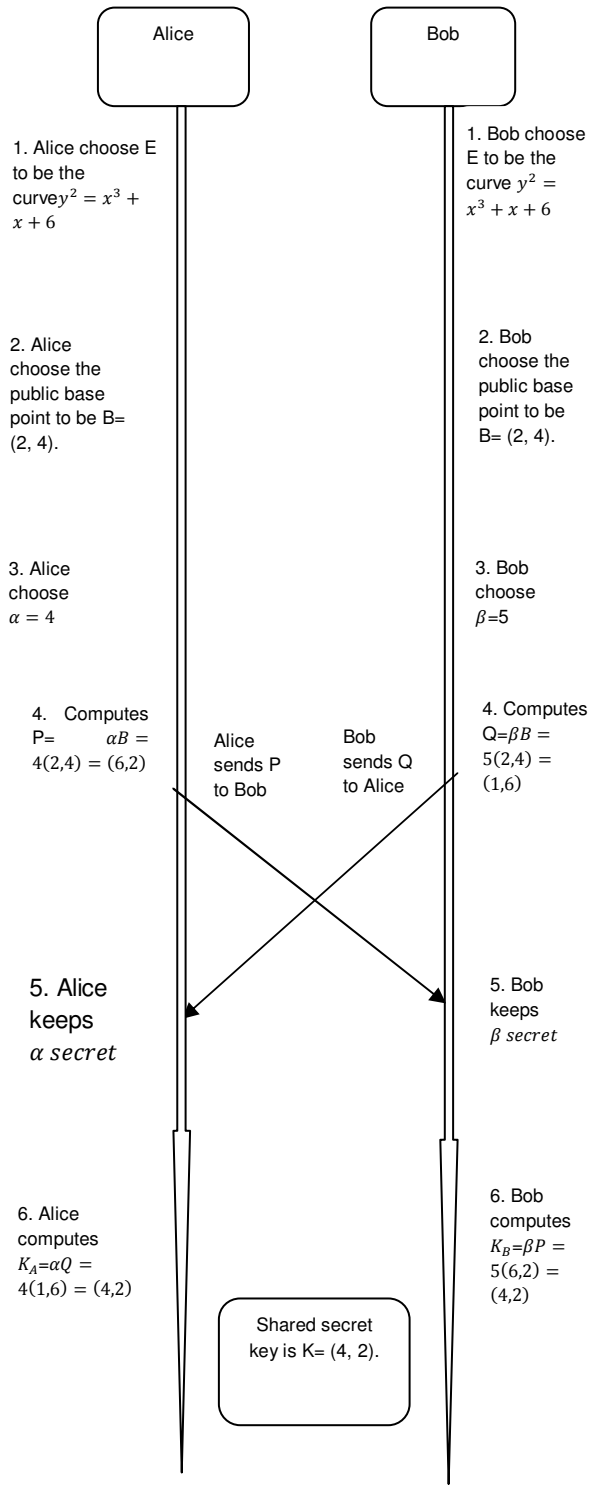

Fig. 2. ECDH key exchange mechanism.

Fig. 2 explains about ECDH exchange technique, in this Alice computes, keeps a key secret key with better encryption.

A. Why Elliptic Curve

Many crypto systems in the real world are based on RSA. However, for adequate security, RSA NEEDS lengthy keys to be used. The key length is being increased to have higher level of protection. Thus it causes much overhead on applications in which RSA is being used. Therefore, RSA is found to be not ideal for applications where massive amounts of data are exchanged. The heavy weight of RSA has its impact on different real world applications such as e-Commerce and banking and IoT where large amount of data is being transferred. To overcome this drawback, Elliptic Curve Cryptography (ECC) provides lightweight mechanisms. It provides similar level of security of RSA with very smaller key size. Thus it reduces overhead drastically. That is the reason, EC based DH approach is preferred in the proposed system. Algebraic addition is used with EC to empower it for computation of increments. As explored in [17], provided two different points denoted as $P = (x_P , y_P)$ and $Q = (x_Q , y_Q)$ respectively and slope denoted as $S = (y_Q - y_P)/(x_Q - x_P)$, the sum of two points denoted as *R=P+Q* is computed as in Eqns. 1, 2.

$$x_R = S^2 - x_P - x_Q \quad (1)$$

$$y_R = -y_P * x_P - x_R \quad (2)$$

When there is doubling operation such as $P + P = 2P = R$ and when $y_P \neq 0$, Eqn. 1 and 2 can be redefined as in Eqns. 3 and 4.

$$x_R = \left(\frac{(3*x_P^2+a)}{2*y_p}\right)^2 - 2 * x_P \quad (3)$$

$$y_R = \left(\frac{(3*x_P^2+a)}{2*y_p}\right) * x_p - x_R - y_P \quad (4)$$

Based on these four equations, the EC related integration is made in ECDH based LSS scheme proposed in this paper.

B. Extension to ECDH

The main focus of this paper is to have a lightweight, low power and robust algorithm for key exchange, encryption and decryption mechanisms suitable for big data. The proposed method is based on ECDH. An improvement is proposed in the area of calculation of modulus of fractional numbers. These numbers are used as EC portion of EC-DH for improved performance. The optimization of EC is carried out with the following algorithm.

Step-1 : Consider natural numbers up to P-1
Step-2 : For each number i in P
Step-3 : Get smallest sun that D$* i > P > D * (i - 1)$
Step-4 : End For
Step-5 : Update N as $N = N * i\%P$
Step-6 : Update D as D=D$* i\%P = D * i - P$
Step-7 : If D$\neq 1$ Then
Step-8 : Repeat steps 2 – 4
Step-9 : End If
Step-10 : Return N

Algorithm 4: Computing modulus of a fractional number

As presented in Algorithm 4, the modulus of a fractional number is computed and returned. It will be used as part of EC which is used in the proposed Lightweight Security Scheme (LSS) which is based on ECDH. With the optimization to ECDH, its power consumption and other performance capabilities are increased.

V. EXPERIMENTAL RESULTS

Experiments are carried out to observe mechanisms like key exchange, encryption and decryption. Observations are made in terms of performance metrics such as execution time, key exchange, power consumption and security comparison. Elliptic curve denoted by $Y^2 = (X^3+aX +b) \bmod P$ is taken where elements of (*P*) are

denoted as *X* and *Y* while *a*, *b* are integers modulo *P* satisfying $4a^3 + 27b^2 \neq 0(mod\ P)$. Values for elliptic curve are generated for which *P*=37, *a*=2 and *b*=0**.**

A. Security Analysis

Key size of any cryptography algorithm has its influence on the security. Thus difficulty in AES is found to be exponential 2^n. When it comes to ECDH, the difficulty is sub exponential $\sqrt{2^n}$. The difficulty is computed in RSA by n as shown in Eqn. 5. Whereas L denotes number of bits in key.

$$\eta = \frac{1.923 * \sqrt[3]{L * \ln(2)} * \sqrt[3]{\ln(L * \ln(2))^2} - 4.69}{\ln(2)} \quad (5)$$

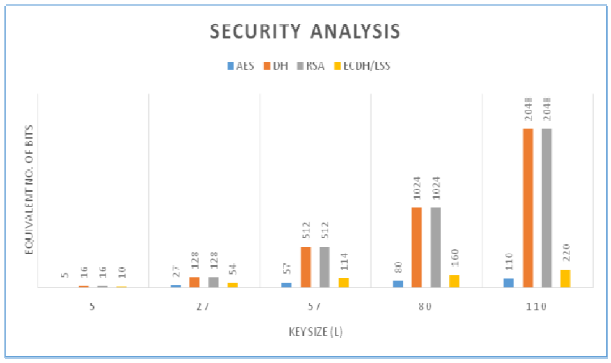

Fig. 3. Results of security analysis.

As presented in Fig. 3, for each algorithm equivalent number of bits in order to provide identical security is provided in vertical axis against different key sizes (L) provided in horizontal axis. The base algorithm for analysis is AES. When key size is 5, DH and RSA needed 16 bits to have identical security while ECDH and LSS need only 10 bits. When key size is 27, DH and RSA needed 128 bits to have identical security while ECDH and LSS needed only 54 bits. When key size is 57, DH and RSA needed 512 bits to have identical security while ECDH and LSS needed only 114 bits. When key size is 80, DH and RSA needed 1024 bits to have identical security while ECDH and LSS needed only 160 bits. When key size is 110, DH and RSA needed 2048 bits to have identical security while ECDH and LSS needed only 220 bits. The results revealed that proposed LSS and ECDH need less key size to provide equivalent security of RSA and DH. Except AES, LSS/ECDH has significant improvement in terms of making the encryption and decryption schemes lightweight.

B. Encryption Time Comparison

Encryption is made with different workloads such as 10 MB, 50 MB, 100 MB and 500 MB. Observations are made on encryption time performance. It is measured in seconds. The algorithms whose execution time is compared are RSA, DH, ECDH and the proposed (LSS).

As presented in Fig. 4, the workload details are provided in X axis and execution time for encryption is provided in Y axis. The workload is found to have influence on the execution time. And different algorithms showed different performance. When data size is 10 MB, RSA needed 2.9938 seconds of time for encryption while DH, ECDH and LSS needed 0.9539, 0.8057 and 0.7989 seconds respectively. LSS revealed improved performance over the existing methods.

This trend is maintained for all workloads. For instance, when data size is 500 MB, RSA needed 25.1956 seconds of time for encryption while DH, ECDH and LSS needed 14.1906, 13.6537 and 12.9896 seconds respectively. Therefore, from the results it is found that LSS shows highest performance while the RSA shows the least performance.

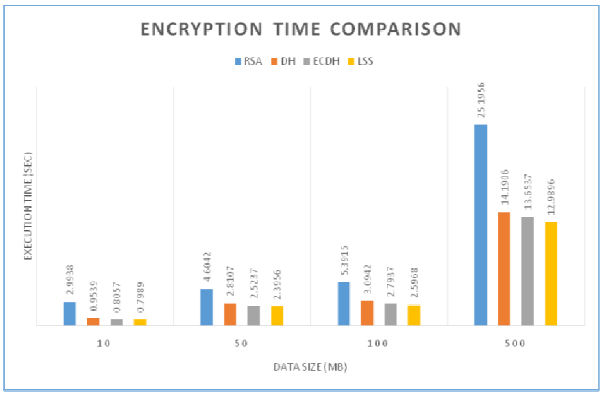

Fig. 4. Execution time comparison for encryption.

C. Decryption Time Comparison

Decryption is made with different workloads such as 10 MB, 50 MB, 100 MB and 500 MB. Observations are made on the time taken by algorithms to decrypt data. It is measured in seconds. The algorithms whose execution time is compared are RSA, DH, ECDH and the proposed (LSS).

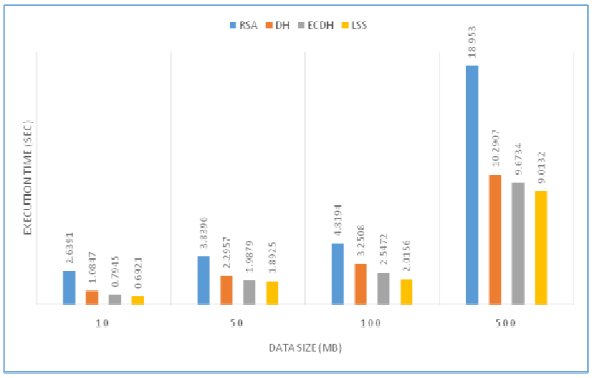

Fig. 5. Time taken for decryption.

As presented in Fig. 5, the workload details are provided in X axis and Y axis shows execution time measured in seconds. The workload has its impact on the execution time. And different algorithms showed different performance. When 10 MB data is used, RSA needed 2.6391 seconds of time for decryption while DH, ECDH and LSS needed 1.0847, 0.7945 and 0.6921 seconds respectively. LSS exhibited improved performance over existing methods.

This trend is maintained for all workloads. For instance, when data size is 500 MB, RSA needed 18.953 seconds of time for decryption while DH, ECDH and LSS needed 10.2907, 9.6734 and 9.0132 seconds respectively. Therefore, from the results it is found that LSS shows highest performance while the RSA shows the least performance.

D. Upload Time Comparison

Data upload time comparison is made with different workloads such as 10 MB, 50 MB, 100 MB and 500 MB. Observations are made on the time taken by different schemes for uploading data. It is measured in seconds. The algorithms whose execution time is compared are RSA, DH, ECDH and the proposed (LSS).

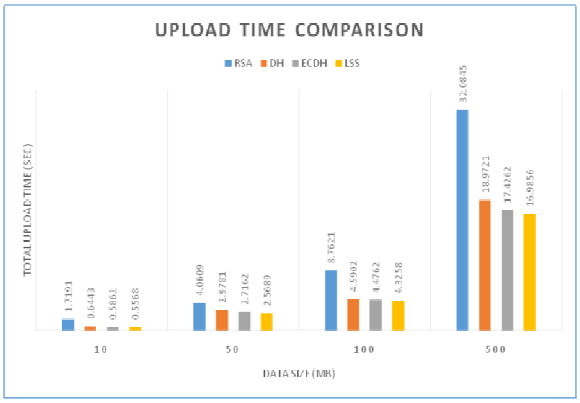

Fig. 6. Total upload time performance.

Fig. 6 showed that, the workload details are provided in horizontal axis while the vertical axis shows execution time taken for data upload. The workload has its impact on the execution time. And different algorithms showed different performance. When 10 MB of data is considered, RSA needed 1.7191 seconds of time for data upload while DH, ECDH and LSS needed 0.6443, 0.5862 and 0.5568 seconds respectively. LSS showed better performance over the state of the art. This trend is maintained for all workloads. For instance, when data size is 500 MB, RSA needed 32.0845 seconds of time for data upload while DH, ECDH and LSS needed 18.9721, 17.4262 and 16.9856 seconds respectively. Therefore, from the results it is found that LSS shows highest performance while the RSA shows the least performance.

E. Download Time Comparison

Data download time comparison is made with different workloads such as 10 MB, 50 MB, 100 MB and 500 MB. Observations are made on the time taken for downloading data. It is measured in seconds. The algorithms whose execution time is compared are RSA, DH, ECDH and the proposed (LSS).

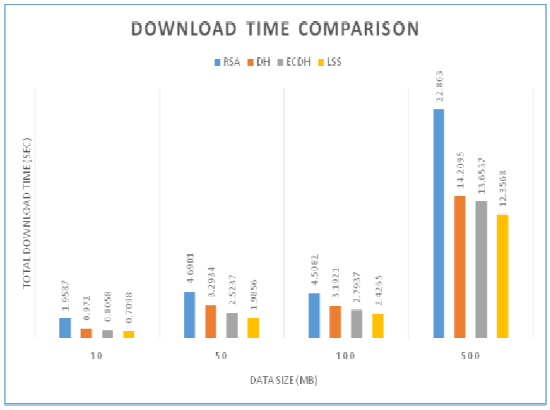

Fig. 7. Comparison of download time.

As presented in Fig. 7, the workload details are provided X axis and Y axis shows download time. The workload has its impact on the execution time. And different algorithms showed different performance. When data size is 10 MB, RSA needed 1.7191 seconds of time for data download while DH, ECDH and LSS needed 0.6443, 0.5862 and 0.5568 seconds respectively. LSS showed performance enhancement when compared with the existing schemes. This trend is maintained for all workloads. For instance, when data size is 500 MB, RSA needed 32.0845 seconds of time for data download while DH, ECDH and LSS needed 18.9721, 17.4262 and 16.9856 seconds respectively. Therefore, from the results it is found that LSS shows highest performance while the RSA shows the least performance.

F. Discussion

The empirical study and results presented in the preceding sub sections are based on the proposed LSS scheme. LSS is based on the EC-DH with an extension to make it more secure and lightweight. The experimental results showed the security advantages of the LSS over other methods. Since it is lightweight, it showed better performance in terms of time taken as well.

VI. CONCLUSION

In this paper, a cryptography scheme is developed for big data. It is termed as Lightweight Security Scheme (LSS) is an improved form of ECDH. ECDH is well known for its lightweight means of key exchange. With the use of Elliptic Curve (EC) and Diffie Hellman (DH) combination along with an improved form of computing modulus of a fractional number used with EC. In the era of big data produced by IoT devices and other sources, it is essential to have lightweight cryptography.

The proposed LSS consumes less power, reduces computational overhead and provides equivalent security when compared with other public key cryptography techniques like RSA. Empirical study is made with a prototype application with different algorithms like RSA, DH, ECDH and LSS. Security analysis and data encryption and decryption revealed that the LSS scheme outperforms the state of the art. Its performance in execution time taken for encryption, decryption, uploading data and downloading data of different size.

VII. FUTURE SCOPE

In future, we intend to improve lightweight security with support for operations such as search and data manipulations directly on the encrypted data.

ACKNOWLEDGEMENTS

I thank to my guide and MLRIT for guiding me to produce this manuscript.

Conflicts of Interest. Authors have no conflict of interest.

REFERENCES

[1]. Aljawarneh, S., Talafha, W. A., & Yassein, M. B. (2017). A resource-efficient encryption algorithm for multimedia big data. *Multimedia Tools and Applications*, *76*(21), 22703-22724.

[2]. Liang, C., Ye, N., Malekian, R., & Wang, R. (2016). The hybrid encryption algorithm of lightweight data in cloud storage. In *2016 2nd International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)* (pp. 160-166). IEEE.
[3]. Lu, R., Zhu, H., Liu, X., Liu, J. K., & Shao, J. (2014). Toward efficient and privacy-preserving computing in big data era. *IEEE Network*, *28*(4), 46-50.
[4]. Tang, W., Zhang, K., Ren, J., Zhang, Y., & Shen, X. (2017). Lightweight and privacy-preserving fog-assisted information sharing scheme for health big data. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, 1-6. IEEE.
[5]. Puthal, D., Wu, X., Nepal, S., Ranjan, R., & Chen, J. (2017). SEEN: A selective encryption method to ensure confidentiality for big sensing data streams. *IEEE Transactions on Big Data*, 1-14.
[6]. Bai, T., Lin, J., Li, G., Wang, H., Ran, P., Li, Z., ... & Jeon, G. (2018). A lightweight method of data encryption in BANs using electrocardiogram signal. *Future Generation Computer Systems*, *92*, 800-811.
[7]. Bakhtiari, M., Zainal, A., Bakhtiari, S., & Mammi, H. K. (2015). Lightweight Symmetric Encryption Algorithm In Big Data. *Int. J. Advance Soft Compu. Appl.*, *7*(3), 45-55.
[8]. Kadhim, F. A., Abdul-Majeed, G. H., & Ali, R. S. (2017). Enhancement CAST block algorithm to encrypt big data. In *2017 Annual Conference on New Trends in Information & Communications Technology Applications (NTICT)* (pp. 80-85). IEEE.
[9]. Usman, M., Ahmed, I., Aslam, M. I., Khan, S., & Shah, U. A. (2017). SIT: a lightweight encryption algorithm for secure internet of things. *International Journal of Advanced Computer Science and Applications*, *8*(1), 1-10.
[10]. Talbi, M., & Bouhalel, M. S. (2018). Application of a Lightweight Encryption Algorithm to a Quantized Speech Image for Secure IoT. 1-16.
[11]. Hong, D., Sung, J., Hong, S., Lim, J., Lee, S., Koo, B. S., & Kim, H. (2006). HIGHT: A new block cipher suitable for low-resource device. In *International Workshop on Cryptographic Hardware and Embedded Systems* (pp. 46-59). Springer, Berlin, Heidelberg.
[12]. Al-Souly, H. A., Al-Sheddi, A. S., & Kurdi, H. A. (2013). Lightweight symmetric encryption algorithm for secure database. *International Journal of Advanced Computer Science and Applications*, *3*(4), 53-62.
[13]. Rajesh, S., Paul, V., Menon, V. G., & Khosravi, M. R. (2019). A secure and efficient lightweight symmetric encryption scheme for transfer of text files between embedded IoT devices. *Symmetry*, *11*(2), 1-21.
[14]. Thayananthan, V., & Albeshri, A. (2015). Big data security issues based on quantum cryptography and privacy with authentication for mobile data center. *Procedia Computer Science*, *50*, 149-156.
[15]. Harinath, D., Babu, K. R.,Chithra, B.,& Murthy, M. V. R., (2015). Encryption Techniques for Big Data in a Cloud. *International Journal of Modern Trends in Engineering and Research*, *2*(8), 1-18.
[16]. Sushmitha, S, Singh, M., & Naaz, F. (2018). A lightweight data sharing scheme using mobile cloud computing. *International Journal of Computer Engineering and Applications*, *12*, 1-6.
[17]. Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE transactions on Information Theory*, *22*(6), 644-654.
[18]. Stallings, W. (2010). Cryptography and Network Security: Principles and Practice, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall Press,
[19]. Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120-126.
[20]. Jonsson, F., & Tornkvist, M. (2017). RSA Authentication in Internet of Things. Available at http://www.divaportal.org/smash/get/diva2:1112039/FULLTEXT01.pdf
[21]. Sirisha, N., & Kiran, K. V. D. (2018). Authorization of Data In Hadoop Using Apache Sentry. *International Journal of Engineering and Technology*, *7*(3), 234-236.
[22]. Sirisha, N., Kiran, K. V. D., & Karthik, R. (2018).Hadoop security challenges and its solution using KNOX. *Indonesian Journal of Electrical Engineering and Computer Science*,*12*(1), 107-116.
[23]. Sirisha, N., & Kiran, K. V. D. (2017). Protection of encroachment on bigdata aspects *International Journal of Mechanical Engineering and Technology*, *8*(7), 550-558.
[24]. Suriya, U., Kumar, R., Dhamodharan, Nagamani, M., & Krishnamoorthy, V. (2019). An Efficient Node Ranking Mechanism for Identifying Selective Forwarding Attacks in WSN. *International Journal on Emerging Technologies, 10*(4), 50-56.
[25]. Myla, S., Marella, S. T., Karthikeya, K., Preetham B., & Ahammad, S. K. H. (2019). The Rise of "Big Data" in the Field of Cloud Analytics. *International Journal on Emerging Technologies, 10*(4), 125-130.

How to cite this article: Sirisha, N. and Kiran, K.V.D. (2020). An Efficient and Lightweight Security Scheme for Big Data. *International Journal on Emerging Technologies*, *11*(1): 414–420.