2023 33rd International Conference Radioelektronika (RADIOELEKTRONIKA) | 979-8-3503-9834-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/RADIOELEKTRONIKA57919.2023.10109046

Comparison of Hardware Implementations of Cryptographic Algorithms for IoT Applications

Sushree Sila P. Goswami
Dept. of Electronics, and Electrical Engineering
Indian Institute of Technology
Guwahati, India
sushree@iitg.ac.in

Gaurav Trivedi
Dept. of Electronics, and Electrical Engineering
Indian Institute of Technology
Guwahati, India
trivedi@iitg.ac.in

***Abstract*—This paper compares various algorithms for application in the Internet of things (IoT). This technology contains a vast network of systems, sensors, and products, taking advantage of its low computing power and miniaturization of electronic networks and interconnected networks to provide extended capabilities that were not possible with the previously-used technology. Many IoT devices per the customer's use, such as Internet-enabled appliances, automation components, and management devices, are helping to lead the vision of a "smart life". This paper aims to find out and implement cryptographic algorithms that can be used in various IoT applications. For the hardware implementation, a XILINX FPGA ZedBoard xc7z020clg484-1 was used. It was found that Elliptical curve cryptography using the Diffie-Hellman algorithm (ECCDH) uses approximately 52% and 77% less power as compared with AES and ECC, respectively. Also, the operating frequency in the case of ECCDH is 355% and 9% more than the operating frequency of AES and ECC, respectively. Again, our design uses 43% less power and 374.4% more frequency than the existing ECCDH architecture. Therefore our proposed fast and resource efficient ECCDH algorithm is suitable for IoT applications.**

Index Terms*—*Cryptography, Security, IoT, FPGA Implementation, resource, power

I. Introduction

It is the inherent need of human beings to communicate data and share information but selectively, and it gives rise to the art of coding the messages so that only the intended person gets to access the data or information or even if the messages in the scrambled form reach the unauthorized person, he could not be able to extract any information. This gave birth to the concept called Cryptography.

Cryptography is the art that can be used to obscure information to make the data secure. The word Cryptography originates from the combination of two greek words," Krypto"means *hidden* and "graphene" means *writing*, coined together to form the hidden writing. The Cryptography process provides confidentiality, data integrity, authentication, and no repudiation. Cryptography has been used to secure communication for centuries. Due to the worldwide growth of digital networks, many applications are using Cryptography to ensure their data, for example, in WLAN, WSN, and digital cards. Therefore, it is becoming crucial to generate low-power, efficient, and secure Cryptographic algorithms. Moreover, there are many cryptographic algorithms that are widely available and in use to secure the information.

The concept of interconnecting networks and computers has been around for years, but the term "Internet of Things" is relatively new. For example, connecting the electrical grid through the telephonic lines to monitor it remotely was already in use by the late 70s. By the 1990's progress in technology made machine-to-machine solutions and enterprise for operation and watching common. IoT is a vast network of devices and objects connected to the Internet through a router and network devices. IoT has created various opportunities as well as challenges across the world.

Applications of IoT includes smart homes, smart city, connected Health, wearables, industrial Internet, smart farming, smart retails, smart supply chain, etc.

The following section represents the architectures and implementations of various Cryptographic algorithms. Section II gives a brief of the existing implementations. Section III presents the FPGA implementation details and architectures for Cryptographic algorithms. Conclusions are presented in Section IV.

II. Related Work

The introduction of the new design either aims towards low memory consumption or high frequency or the optimization between both parameters. Security algorithms like AES can be implemented on Field Programmable Gate Array (FPGA) [1] - [9]. In [10], DES ECC algorithm is implemented and the FPGA Implementation of Pipelined Blowfish algorithm was done by S. Roy et al. [11]. The implementation of these encryption algorithms is proposed in terms of resource, and power optimization is implemented, while some focus on reducing area occupancy. Fast, resource efficient, and lightweight ECC algorithms are implemented for various applications like web sensors, IoT, wireless sensor networks [12] - [19].

The implementation of advanced encryption standards (AES) on FPGA was efficiently done by Sridevi et al. [6]. The number of modes the AES performs is four, and these are cipher block chaining, output feedback mode, electronic code block, and Cipher feedback block. The variable key size of 128, 192, and 256 bits can be used for the plain text size of 128 bit with the number of rounds 10, 12, and 14, respectively. Sub bytes, shift rows, mix columns, and addroundkey are the steps involved in the algorithm. A Silicon platform was used

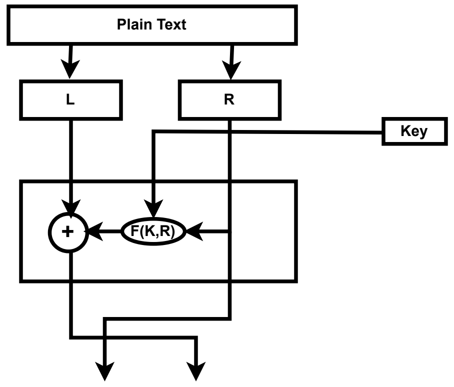
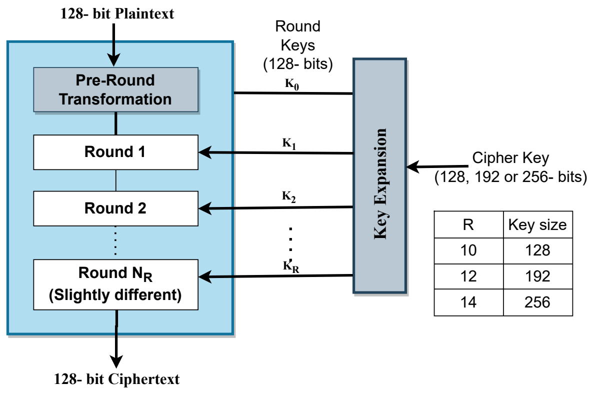


Fig. 1. Fiestel Structure Block Diagram.



R	Key size
10	128
12	192
14	256

Fig. 2. AES Algorithm Block Diagram.

to implement the AES, and it was pipelined to be implemented on the FPGA kit. T box was used to replace the S box, which in turn was also pipelined, showing higher throughput on the FPGA kit.

S. Roy et al. [11] proposed a pipelined structure for blowfish. Key size of 32 bits is used for the plain text of 64 bit of size. It also uses the Feistel network structure with the number of rounds to be 16 for improved Security, it provided that the throughput of the pipelined blowfish is less than that of non-pipelined blowfish. Blowfish shows better performance than the already existing AES and DES algorithm.

Reza et al. [9] Proposed a 2-slow retiming technique for the AES algorithm and implemented it on an FPGA kit. They proved that throughput could be increased by Pipelining concept, Rolling concept, and Sub stage pipelining concept. They also proved that the 2-slow retiming technique is better than the c-slow retiming technique. The 2-slow retiming technique breaks the critical path into the number of pieces. Due to that, registers are increased. To overcome the more number of registers, the author approached the multiplication process in the third stage of the AES algorithm.

Miguel et al. [16] presented a FPGA prototyped low power hardware accelerator scalar multiplier in elliptic curve cryptography (ECC), which provides security services in the IoT, such as confidentiality and authentication. Their design is area efficient than other FPGA architectures and targeted elliptic curves defined over binary fields generated by trinomials. They used the IoT MicroZed FPGA board to deploy their ECC hardware accelerator, validated under a hardware/software codesign of the Diffie-Hellman key exchange protocol (ECDH).

Hamad et al. [18] implemented ECC, a symmetric key algorithm performed on the elliptical planar curve over the finite field and depended on point multiplication, which is followed by a series of addition and point doubling. The problem addressed is determined by the size of the elliptical curve. Kartusube-Hoffman method and modular arithmetic operation were performed to reduce the memory size. Higher throughput and low memory space are achieved by this.

Lightweight ECC for Internet of Things (IoT) to provide better security was proposed by He. and Sherli [19] analyzing the three schemes of existing cryptography algorithm and insisted that for IoT, lightweight schemes are suitable.

Due to the small size of the IoT devices and limited resources, there is a need to develop an algorithm that not only provides the maximum security but with as low as possible use of the hardware, the primary goal is ensuring the authentication and integrity of the message so that the message received by the user can be trusted, and to make sure the data is not being modified.

Millions of devices are connected to the IoT network, which in turn increases the number of communications that are taking place, leading to the issue of scalability and engineering. With this many devices connected to the network, another problem arises the issue of the cloud, making the Security of the devices more difficult. Distributed ledger maintaining a growing number of transactions and data records is called Blockchain, recording the blocks of transactions relating to the network participants, ensuring the correct sequences.

III. Architecture of Cryptographic Algorithms

Different Block ciphers are derived from the fiestel structure. In the encryption of the fiestel structure, the plain text is divided into two halves, the left and right. These two halves then go into the next step of the fiestel structure, called the rounds. In the first round, some special function is performed on the right half of the plain text in which the security key is added to it; then the output is XORed with the left half of the plain text, which is then sent to the next round as the right half. Also, the right half of the plain text is transferred to the next round as it is and as the left half to that round, and then the same operation is performed as above. The round Function is performed a number of times, and different is added for each round. After the last round, the cipher text is produced as shown in Figure 1.

A. Advanced Encryption Standard

This algorithm is based on a permutation-substitution network. It consists of layers of operation which may involve exchanging input by some certain outputs(substitution)and shuffling the bits(permutations). AES algorithm does all its

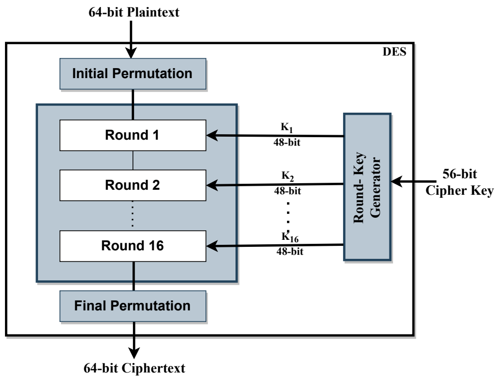


Fig. 3. DES Algorithm Block Diagram.

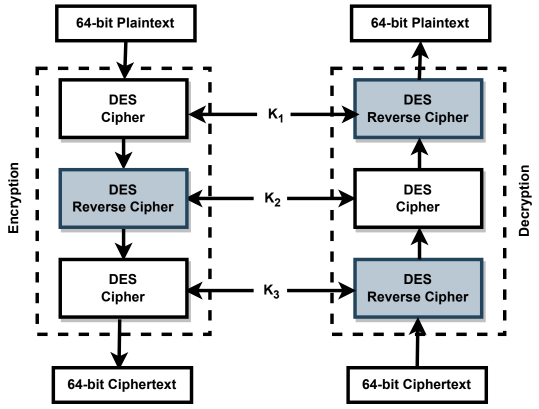


Fig. 4. 3DES Algorithm Block Diagram.

operation on bytes by treating 128 bits as 16 bytes.These bytes are processed as a matrix arranged in four rows and four columns. The block diagram of the AES algorithm is depicted in Figure 2. The length of the key decides the number of rounds in the AES algorithm, for 128 bit key - 10 rounds
192 bit key - 12 rounds
256 bit key - 14 rounds

Each round in the Encryption process comprises four sub-rounds, as discussed below,

- **Byte Substitution** The input is of 16 bytes, and in this process, each bit is substituted by looking into the table specified in the design, which in turn results in $4X4$ matrices.
- **Shift Rows** Each row is shifted towards the left as per the row number; any entries that fall off are reinserted on the right side.
  1) first row - No Shift
  2) Second Row - Shifted towards left by one position
  3) Third Row - Shifted towards left by two positions
  4) Fourth Row - Shifted towards left by three positions
- **MixColumns** The input to this block is now each column of four bytes which is transformed to a set of new bytes using a unique mathematical function to replace the previous entries of the column, which in turn results in a matrix comprising of 16 new bytes. This round is not performed in the 10^{th} round.
- **AddRoundKey** This is the block where the key that we were talking about comes into play; as we have mentioned above, the 128 bits round key is used and is XORed with a matrix of 16 bytes which are considered as 128 bits here in this block. These 128 bits are now again interpreted as 16 bytes to be given as input to the next round, or this can also be the output cipher text if it is the last round.

B. Data Encryption Standard

Published by the National Institute of Standards and Technology (NIST) in 1975, it was a symmetric key block cipher. It is derived from the Feistel structure.

If the block size of 64 bit, i.e., at a time, a 64 bit plain text is converted to the cipher text. The number of Feistel structure rounds used in this process is 16, and the length of key size is 64 bit, but effectively only 56 of its bits are used.

The block diagram of the DES algorithm is depicted in Figure 3. The essential steps for DES are Round functions, Key structure, and initial and final permutations.

The permutation boxes are added to make the algorithm stronger; in this, the bits are permuted randomly, where the bits of the plain text is rearranged. The initial and final permutations are inverse of each other. The round Function is the heart of the DES algorithm; in the DES Algorithm, there are 16 round functions in which some special tasks are performed. In this, a 32 bit output is produced through the rightmost 32 bits by applying a 48 bit key. In the Permutation Box, the Right input needs to be expanded from 32-bit to 48 bits since the key is 48 bits.The expanded correct plain text is XORed with the round key after the expansion is performed. The S-boxes carry out the actual mixing, 8 S boxes are used in DES, and each s box is input with 6 bits and produces 4-bit output. Therefore the output is a total of 32 bits.

C. Triple Data Encryption Standard

In the triple DES algorithm, first, the encryption is performed as per the above DES algorithm using key1, then decryption is performed using the key2, and again the decryption is performed using the key3 to produce the 64 bit cipher text.

D. Blowfish Algorithm

Blowfish contains a block size of 64 bit, and the key length can be any length from 32 bits to 48 bits.It has a huge dependency on the S boxes and follow the 16 round Feistel structure.

Each line in the Blowfish algorithm represents the 32 bits; two subkey arrays are used in this algorithm: four 256 entry S boxes and 18 entry P-array. Input is accepted in the form of 8 bits in the S boxes, and the output is produced in 32 bits. Every round uses one entry of the P array, and after the final

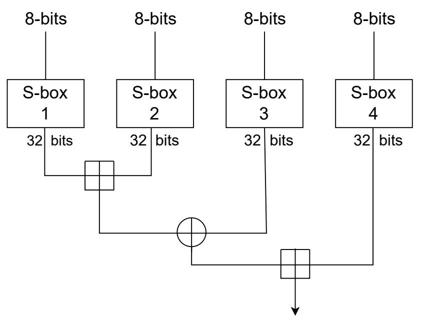


Fig. 5. blowfish Algorithm Block Diagram.

round, every half of the data block is XORed with one of the remaining unused P arrays.

Four 8-bit quarters are formed by splitting the 32 bits input, and each quarter is input to the S-Boxes. Modulo 232 is added to the outputs and XORed, which then produces the final output of 32 bits as shown in Figure 5. Due to the Feistel structure of the blowfish algorithm, it is inverted by using $P17$ and $P18$ to the cipher text block.,then using the reverse-ordered P-entries.

E. RSA (Rivest Shamir-Adleman) Cryptosystem

It was invented in 1977 and is named after Ron Rivest, Adi Shamir, and Len Adleman. RSA is the world's most widely used public key cryptographic algorithm. It can be used not only for public key encryption but also for digital signatures. It is based on the principle that it is easy to calculate the product of two big prime numbers. Still, the inverse is very difficult, i.e., it is nearly impossible to factorize a huge number to its factors which are prime.

1) Generation of RSA key pair:

- Generate the RSA modulus(n):
  Select two large primes, a and b. Calculate the product of these two numbers, $n = a * b$, n should be a minimum of 512 bits for unbreakable solid encryption.
- Derive number (e):
  A number e is chosen such that it is greater than 1 and less than $(a-1)(b-1)$. e and $(a-1)(b-1)$ must not have any factor in common except for 1,that is these two numbers should be co-prime.
- Public key formation:
  RSA Public key is formed by the pair of numbers (n, e) and is made public. The public key has a part n in it, which is difficult to factorize. The RSA algorithm's strength is that it is difficult to find two primes $(aandb)$ used to obtain n.
- Generate private key:
  A unique number d is calculated using a, b, e which is the private key. It inverse of e modulo $(a-1)(b-1)$. This relation is written mathematically as:

$$ed = 1 \mod (a-1)(b-1) \tag{1}$$

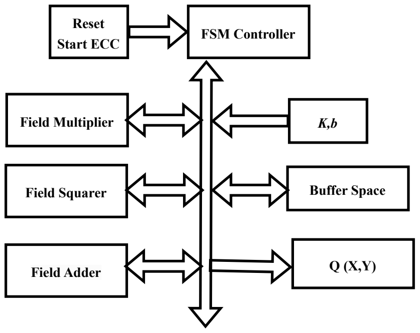


Fig. 6. ECC Algorithm Block Diagram.

2) RSA Encryption: when the sender sends the text message to someone whose public key is (n, e), it is represented as the series of numbers less than n by the sender. To encrypt the plaintext P, which is a number modulo n, a simple the mathematical step is used to represent the encryption process:

$$C = P^e \mod (n) \tag{2}$$

3) RSA Decryption: It is a straightforward process when the cipher text C is received by the receiver of the public key (n, e). Cipher text C is raised by the power of the private key d, and then the plain text is obtained by modulo n the result as:

$$P = C^e \mod (n) \tag{3}$$

F. Elliptical Curve Cryptography

The working principle of Elliptical Curve Cryptography (ECC) is the Trapdoor function. We can get the same level of Security by using the 256-bits key size for ECC as we will get from the 3072-bits of the key length of RSA. US Government's top secrets are secured using the 384 bits of the key length of ECC. The block diagram of ECC algorithm is given in FIg. 6. The curve is represented by the following;

$$y^2 = x^3 + ax + b \tag{4}$$

The number of points N is bounded by:

$$p + 1 - 2\sqrt{p} <= N <= p + 1 + 2\sqrt{p} \tag{5}$$

The elliptical curve should follow the equation;

$$4a^3 + 27b^2 ! = 0 \tag{6}$$

to have the curve 3 different roots. The curve requires its own algorithm for addition and multiplication. If we draw a straight line through the curve, it will intersect no more than 3 points. The elliptical curve is symmetrical about the x-axis as shown in Figure 7.

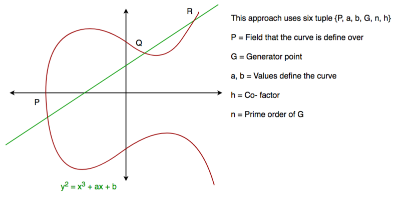

R
Q
P
$y^2 = x^3 + ax + b$
This approach uses six tuple {P, a, b, G, n, h}
P = Field that the curve is define over
G = Generator point
a, b = Values define the curve
h = Co- factor
n = Prime order of G

Fig. 7. ECC Algorithm.

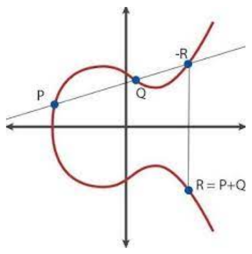


Fig. 8. Addition in ECC.

1) Addition in ECC: If we want to add the two points to the curve, a straight line is drawn through these two points, which intersects the curve at the third point, and a vertical line is drawn, which intersects the curve at another point. This point gives the addition of the given two points as shown in Figure 8. As we know that multiplication is just repeated addition, we can perform the multiplication in the same way.

ECC gets its strength from this repetitive multiplication; we can get to the final point from the initial point in an easy manner, but it is challenging the initial point from the final point; for it, we have to calculate all the points until we find the desired point, but it is nearly impossible when we are using sufficiently large values; it will take a a reasonable amount of time due to increased computational complexity.

G. Diffie-Hellman key exchange algorithm

Diffie-Hellman is a key exchange algorithm that is exponential, which allows the users to exchange a secret key without the requirement of any prior secrets in real time over an unsecured network, discrete computing logarithm for a large number is the principle behind the Diffie Hellman key exchange algorithm. There is not any successful attack found against this algorithm. It requires one prime number and one of its primitive roots; both should be very large, and the prime number should be at least 512 bits. This algorithm can be explained in the Fig. 9.

The public values can be computed by :

$$x = g^a \mod (p) y = g^b \mod (p) \tag{7}$$

where a and b are user picked values, x and y are values to be exchanged. The private key can be computed as:

$$ka = y^a \mod (p) kb = x^b \mod (p) \tag{8}$$

Algebraically it is seen that the user has a symmetric secret key to encrypt i.e.

$$ka = kb \tag{9}$$

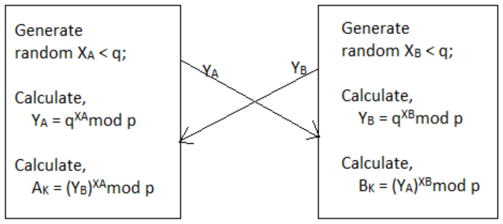


Fig. 9. Diffie Hellman key exchange algorithm.

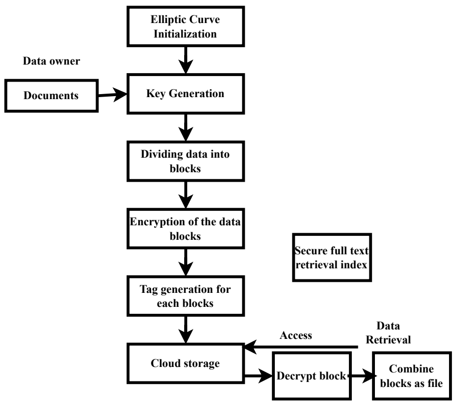

Elliptic Curve Initialization
Data owner
Documents
Key Generation
Dividing data into blocks
Encryption of the data blocks
Secure full text retrieval index
Tag generation for each blocks
Access
Data Retrieval
Cloud storage
Decrypt block
Combine blocks as file

Fig. 10. ECCDH Algorithm Block Diagram.

H. Elliptical Curve Cryptography with Diffie-Hellman key exchange

ECCDH is the modified version of Diffie Hellman of passing the ECC keys; a new public key is generated by passing an old public key and then developing a shared public key. Calculations agreed upon elliptical curve is the basic idea on which the ECCDH public key algorithm. The block diagram of the ECCDH algorithm is given in Fig. 10. The point on the curve is taken as the starting point, which in turn derives a random number for the private key, and the product of all those values is sent to the other users. The curve and the starting point are known to the receiver, common product for both the user can be derived by using the sender's product which can be used as a key.

The key agreement in ECCDH can be understood as follows:
The prime number a, p, q are selected by both the users, which satisfies the elliptical curve group $E_a(p, q)$. The elliptical curve equation is;

$$y^2 = x^3 + px + q(mod(a)) \tag{10}$$

For example, elliptical curve group E_{211} created by $a = 211, p = 0, q = -4$ and the elliptical curve $y^2 = x^3 - 4$. A common coordinate is selected by users on the curve less than p. for example, $P(2, 2)$ is generated by using $x = 2$ and $y = 2$. This point p is the generator point in the Diffie- Hellman algorithm. A random number d (less than p) is selected by the sender, multiplied by the generator point P to generate the multiple points Q on the curve $Q = dP$. The sender sends the value Q, the public key, to the receiver. $S = dQ$ is computed by the receiver with the Q received from the sender. Now the sender and the receiver have the shared and the public key.

The algorithm can be used in the applications in which the bits for the Encryption and Decryption are shared between the devices and the cloud. For any encryption or decryption, the bits from the cloud from that particular device as well as the cloud, are required. If we are using the n-bit key, we can store x-bits in the device and other $n - x$ bits in the cloud. When the cloud and the device want to communicate, bits from both are required for the encryption and decryption of the text.

TABLE I
COMPARISON OF VARIOUS CRYPTOGRAPHIC ALGORITHMS.

Algorithm	BlockSize	Key Size	Resource (LUTs)	Power (in mW)	Frequency (in MHz)
AES	128	128	500	189	51.33
DES	64	64	731	135	140
3DES	64	64	1035	141	107.38
Blowfish	64	64	2540	121	71.14
RSA	256	256	25164	760	100
ECC	233	32	23510	391	212.9
ECCDH	233	32	1030	90	234

TABLE II
COMPARISON WITH EXISTING ARCHITECTURE.

Design	Resource (LUT or Slices)	Frequency (in MHz)
Morales-Sandoval et. al. (2021) [16]	1809	62.5
Our Design	1030	234

IV. CONCLUSION

In this paper, designing and implementing IoT security algorithms, various cryptographic algorithms were implemented on the XILINX FPGA ZedBoard **xc7z020clg484-1** to find the suitable algorithm for the IoT security applications. IoT-enabled products need the algorithm to take the least power and area possible; frequency should be high for faster operations.

As inferred from Table I the power consumption in ECCDH (Elliptical curve cryptography using Diffie-Hellman algorithm) is reduced by approximately 52% and 77% compared to the power consumption in AES and ECC, respectively. The operating frequency in ECCDH is increased by 455% and 9% as compared to the operating frequency in AES and ECC, respectively. Moreover, it is the most secure algorithm to date, as no known attacks have been found against this algorithm. From Table II, we can see that our design uses 43% less resource and 374.4% more frequency than the existing ECCDH architecture [16]. Therefore our proposed fast and resource efficient ECCDH algorithm is suitable for IoT applications. The above results lead us to conclude that ECCDH is superior to other algorithms in terms of area, power, and performance.

ACKNOWLEDGEMENT

The authors acknowledge the partial financial support provided by "The Visvesvaraya Ph.D. Scheme for Electronics and IT," Ministry of Electronics and Information Technology (MeitY). The work has also been partially supported by MeitY's sponsored project Electronics and ICT Academy at IIT Guwahati. The authors acknowledge the infrastructure support provided by the Academy, which helped in the successful and timely conclusion of the proposed work. The support provided by the agencies mentioned above is greatly acknowledged.

REFERENCES

[1] T. A. Pham, M. S. Hasan, and H. Yu, "Area and power optimisation for aes encryption module implementation on fpga," 18th International Conference on Automation and Computing (ICAC), pp. 1–6, 2012.
[2] M. Feldhofer, J. Wolkerstorfer, and V. Rijmen, "Aes implementation on a grain of sand," Information Security, IEE Proceedings, vol. 152, pp. 13– 20, 11 2005.
[3] P. H am al ainen, T. Alho, M. H annik ainen, and T. D. H am al ainen, "Design and implementation of low-area and low-power aes encryption hardware core," 9th EUROMICRO Conference on Digital System Design (DSD'06), pp. 577–583, 2006.
[4] A. Ricci, M. Grisanti, I. De Munari, and P. Ciampolini, "Design of a 2 w rfid baseband processor featuring an aes cryptography primitive," 10 2008, pp. 376 – 379.
[5] K. Rahimunnisa, P. Karthigaikumar, S. Rasheed, J. Jayakumar, and S. SureshKumar, "Fpga implementation of aes algorithm for high throughput using folded parallel architecture," Security and Communication Networks, vol. 7, no. 11, pp. 2225–2236.
[6] S. S. S. Priya, P. Karthigaikumar, and N. R. Teja, "Fpga implementation of aes algorithm for high speed applications," pp. 115–125, Jul 2022.
[7] J. Van Dyken and J. G. Delgado-Frias, "Fpga schemes for minimizing the power-throughput trade-off in executing the advanced encryption standard algorithm," vol. 56, no. 2–3, 2010.
[8] T. M. Kumar, K. S. Reddy, S. Rinaldi, B. D. Parameshachari, and K. Arunachalam, "A low area high speed fpga implementation of aes architecture for cryptography application," Electronics, vol. 10, no. 16, 2021.
[9] R. Farashahi, B. Rashidi, and S. Sayedi, "Fpga based fast and high-throughput 2-slow retiming 128-bit aes encryption algorithm," Microelectronics Journal, vol. 45, p. 1014–1025, 08 2014.
[10] R. Naseer and J. Draper, "Dec ecc design to improve memory reliability in sub-100nm technologies," in 2008 15th IEEE International Conference on Electronics, Circuits and Systems, 2008, pp. 586–589.
[11] S. R. Chatterjee, S. Majumder, B. Pramanik, and M. Chakraborty, "Fpga implementation of pipelined blowfish algorithm," in 2014 Fifth International Symposium on Electronic System Design, 2014, pp. 208–209.
[12] W. N. Chelton and M. Benaissa, "Fast elliptic curve cryptography on fpga," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 2, pp. 198–205, 2008.
[13] M. S. Hossain, E. Saeedi, and Y. Kong, "High-speed, area-efficient, fpga-based elliptic curve cryptographic processor over nist binary fields," in 2015 IEEE International Conference on Data Science and Data Intensive Systems, 2015, pp. 175–181.
[14] G. D. Sutter, J.-P. Deschamps, and J. L. Imana, "Efficient elliptic curve point multiplication using digit-serial binary field operations," IEEE Transactions on Industrial Electronics, vol. 60, no. 1, pp. 217–225, 2013.
[15] V. Kamalakannan and V. Kamalakannan, "Fpga implementation of elliptic curve cryptoprocessor for perceptual layer of the internet of things," ICST Transactions on Security and Safety, vol. 5, p. 155739, 10 2018.
[16] M. Morales-Sandoval, L. A. R. Flores, R. Cumplido, J. J. Garcia-Hernandez, C. Feregrino, and I. Algredo, "A compact fpga-based accelerator for curve-based cryptography in wireless sensor networks," Journal of Sensors, vol. 2021, Jan 2021.
[17] S. S. Roy, C. Rebeiro, and D. Mukhopadhyay, "Theoretical modeling of elliptic curve scalar multiplier on lut-based fpgas for area and speed," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 5, pp. 901–909, 2013.
[18] H. Marzouqi, M. Al-Qutayri, and K. Salah, "Review of elliptic curve cryptography processor designs," Microprocessors and Microsystems, vol. 39, no. 2, pp. 97–112, 2015.
[19] E. Venugopal and T. Hailu, "Fpga based architecture of elliptic curve scalar multiplication for iot," in 2018 Conference on Emerging Devices and Smart Systems (ICEDSS), 2018, pp. 178–182.