Evaluating the Performance Parameters of Cryptographic Algorithms for IOT-based Devices

Umar Iftikhar
Computer and Information Systems Engineering Dept.
NED University of Engineering and Technology
Karachi, Pakistan

Kashif Asrar
Computer and Information Systems Engineering Dept.
NED University of Engineering and Technology
Karachi, Pakistan

Maria Waqas
Computer and Information Systems Engineering Dept.
NED University of Engineering and Technology
Karachi, Pakistan

Syed Abbas Ali
Computer and Information Systems Engineering Dept.
NED University of Engineering and Technology
Karachi, Pakistan

Abstract-Nowadays, terabytes of digital data are generated and sent online every second. However, securing this extent of information has always been a challenging task. Cryptography is a fundamental method for securing data, as it makes data unintelligible for attackers, offering privacy to authorized clients. Different cryptographic algorithms have different speeds and costs that make them suitable for different applications. For instance, banking applications need outrageous security amenities, as they utilize superior algorithms having greater requirements, while gaming applications focus more on speed and cost reduction. Consequently, cryptographic algorithms are chosen based on a client's prerequisites. This study compared DES, AES, Blowfish, and RSA, examining their speed, cost, and performance, and discussed their adequacy for use in wireless sensor networks and peer-to-peer communication.

Keywords-DES; AES; Blowfish; RSA

I. INTRODUCTION

Cryptography is the study of utilizing math to conceal messages and has been around since the inception of human civilizations. From the hand-ciphers used thousands of years ago, to electronic ciphers used in modern commercial applications and war, mankind utilized cryptography to ensure the secrecy of information. The advance of programmable computers necessitated more complex cryptographic algorithms and expanded their application domain to secure data confidentiality and integrity, user authentication, and prevention of cyber-attacks [1]. Today, there is a wide variety of powerful encryption algorithms, offering different combinations of speed, security, and computational resources. All efforts in this area converge to the single goal of making the encryption secure at the best cost to performance ratio [2-6]. Cryptographic algorithms are characterized as symmetric or asymmetric, depending on the keys they utilize.

Data Encryption Standard (DES) was developed in the '70s and is considered to be the pioneer symmetric algorithm, using fixed key lengths. Its most prominent successors are Triple-DES (3DES) and Advanced Encryption Standard (AES), with the latter being the most preferred today. Blowfish is another popular symmetric scheme that uses variable key length, while River Ciphers also belong in this group. In the asymmetric category, Rivest-Shamir and Adleman (RSA) and Elliptic Curve Cryptography (ECC) are the undisputed kings [7]. Many researchers investigated these strategies to track down the best in terms of execution cost [2-21]. Authors in [10] compared the strengths and weaknesses of each algorithm. They compared experimentally the most significant encryption algorithms on metrics like memory usage, time consumption, and the avalanche effect, concluding that each one has a separate set of domain-wise strengths. Blowfish was the most economical in terms of time and memory usage, AES had the most cryptographic strength, and DES was the most bandwidth-efficient.

This study extends the analysis with exclusive emphasis on decryption efficiency, as it can be a useful factor for selecting an encryption algorithm in certain domains. Additionally, these algorithms were examined for their adequacy on wireless sensor networks and peer-to-peer communication.

II. THE INVESTIGATED AND IMPLEMENTED CRYPTOGRAPHY ALGORITHMS

During this study the DES, AES, RSA, and Blowfish algorithms were implemented in Java. The source code used for the evaluation of each algorithm including its description, methodology, and execution pattern is exhibited along with the block diagrams of the main functions of each algorithm.

A. DES

DES is based on symmetric key square code and uses a 56-bit key in 64-bit blocks [7]. DES has 16 phases, also named as rounds [2], and uses the following functions:

- A zor function that takes a stream and generates an integer result

Corresponding author: Umar Iftikhar (umariftikhar@neduet.edu.pk)

- The En_box function that generates an array
- The S_box that takes a 48-bit input and returns a 32 bit output
- The f_permute that uses a loop to return permuted data
- The fn_function that uses xor to return a xorstream
- A user_input function to get key bits and data bits

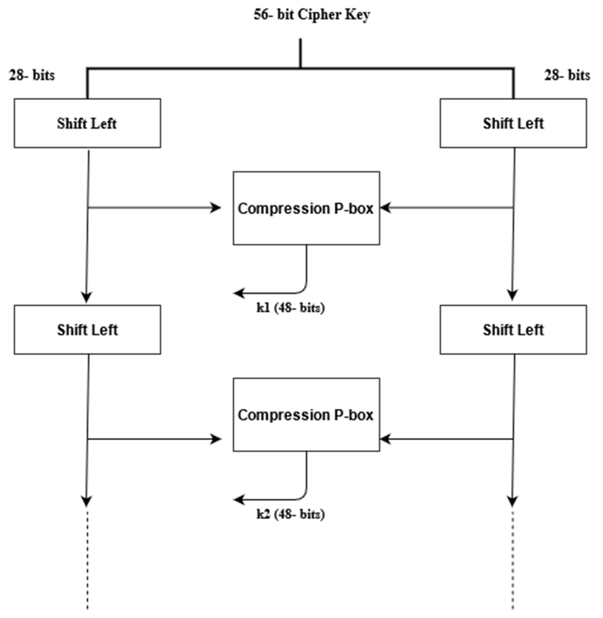


Fig. 1. DES sub-key generation.

The function definitions are:

```
def zor(left,xorstream):
xorresult = np.logical_xor(left,xorstream)
xorresult  = xorresult.astype(int)
return zorresult
def En_box(right_data):
expanded_data = np.empty_data(48)
j = 0
for i in EBox:
expanded_data[j] = right[i-1]
j+=1
expanded_data = list(map(int,expanded_data))
expanded_data = np.array(expanded_data)
return expanded_data
def f_permute(topermute):
permuted_data= np.empty_data(32)
e = 0
for f in F_PBox:
permuted_data[j] = topermute[f-1]
e+=1
return permuted_data
def fn_function(right_data,rkey):
expanded_data = E_box(right_data)
xored = xor(expanded_data,rkey)
sboxed = sbox(xored)
xorstream = f_permute(sboxed)
return xorstream
```

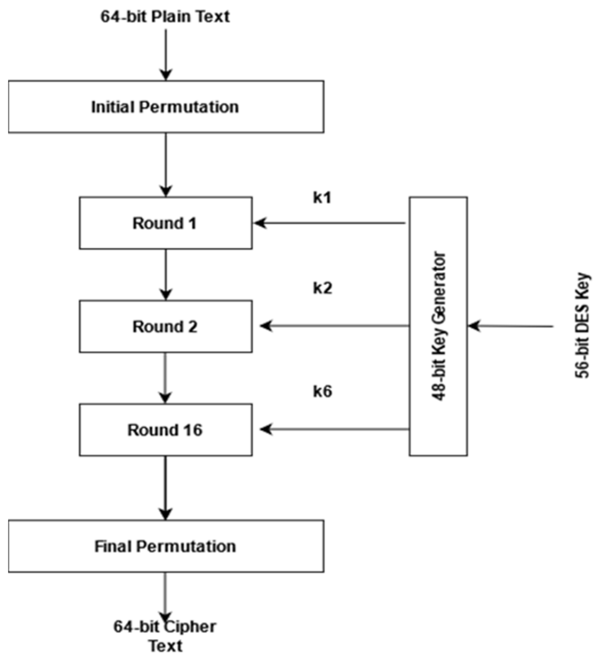


Fig. 2. DES block diagram.

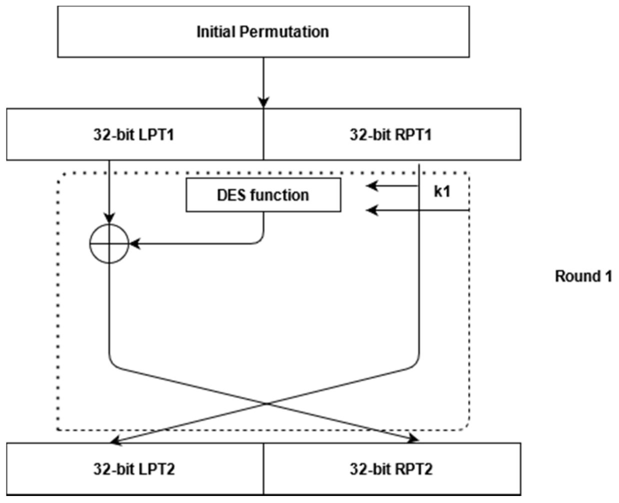


Fig. 3. DES round block diagram.

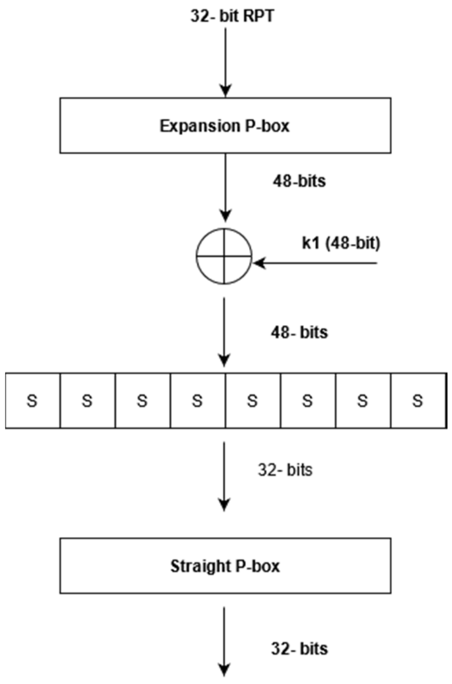


Fig. 4. DES function block diagram.

B. AES

AES is a symmetric key square code algorithm that was developed as a replacement for DES. AES utilizes three key sizes of 128, 192, and 256 bits, and uses 128-bit blocks that can be encrypted in one go, as it is much faster than DES. Encryption and decryption in AES are performed in multiple rounds. The number of rounds used for the encryption or decryption varies with the key size between 10, 12, and 14 for 128, 192, and 256 keys respectively. AES consists of 3 steps that contain different combinations of specified operations [4]. The step by step procedure for AES algorithm is:

- Initialize the constructor function
- Change _of _key uses a text to matrix function along with a control structure to generate a list of round keys
- The encryption gets a plain state from text to matrix and returns the matrix to text
- The decryption gets the cipher state object and uses multiple returns matrix to text
- round_key_addition, _rounding_ encrypt, and a few more functions are used to generate the final output

The function definitions are:

```
def text_to_matrix(text_data):
matrix_data = []
for e in range(16):
byte = (text >> (8 * (15 - e))) & 0xFF
if e % 4 == 0:
matrix_data.append([byte_data])
else:
matrix_data[e / 4].append(byte_data)
return matrix_data
def matrix_to_text(matrix_data):
text_data = 0
for e in range(4):
for f in range(4):
text |= (matrix_data [e][f] << (120 - 8 * (4 * e + f)))
return text
def round_key_addition(self, r, k):
for e in range(4):
for f in range(4):
s[e][f] ^= k[e][f]
```

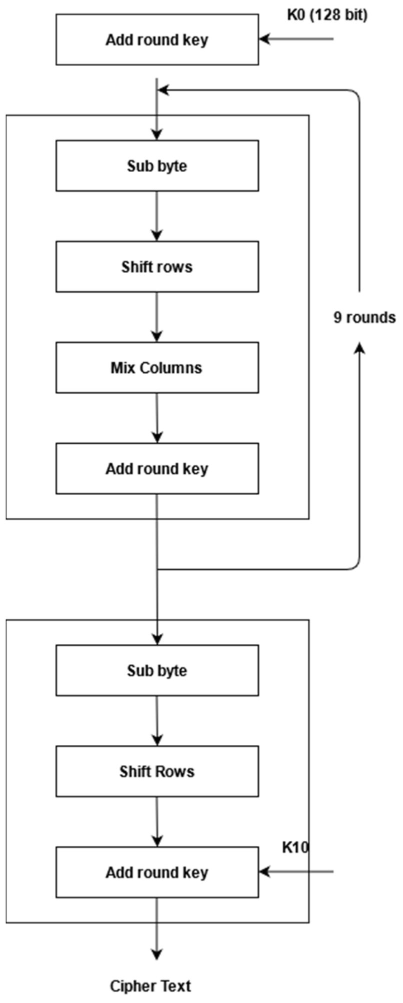


Fig. 5. AES block diagram.

def _rounding_encrypt(self, state_matrix_data, key_matrix_data):

self.__sub_bytes_data(state_matrix_data)

self.__shift_rows_data(state_matrix_data)

self.__mix_columns_data(state_matrix_data)

self.__add_round_key_data(state_matrix_data, key_matrix_data)

The encryption steps are:

- ADD ROUND KEY: Data are straightforwardly XORed with the round key or the code key of 128 pieces.
- SUB BYTES: Breaks the input into bytes and then passes it via a substitution box. Unlike DES, AES has the same s-box for all bytes. Returns a 4×4 matrix.
- SHIFT ROWS: The rows of the obtained matrix from the sub bytes are shifted left. Since it's a 4×4 network, the fourth column is moved threefold, the third is moved twice, the second is shifted once, and the first remains unshifted. Hence, each n-th row will be shifted (n-1) times.
- MIX COLUMNS: Interchanges the columns via a mathematical function to get the output.

The above steps may be applied in reverse to get the deciphered text.

C. RSA

RSA was named after its 3 donors, Rivest–Shamir–Adelman. The key length must be greater than 1024 bits and the plain text to encrypt must be at least 512 bits. Since it utilizes mathematic prime number calculations, it has very high encryption and decryption times and consumes a lot of computational power [20]. The private key is generated in the PKCS#8 format, whereas the public key uses the X.509 format.

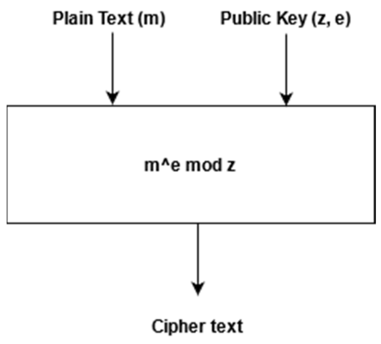


Fig. 6. RSA encryption.

The step by step procedure of RSA is:

- Two large prime numbers are selected
- A pair of public and private keys is generated.
- The user enters a message to be encrypted
- The encrypt_data_message function takes two arguments (public key and message) to generate the encrypted text.
- The private key can be used to decrypt the data message.

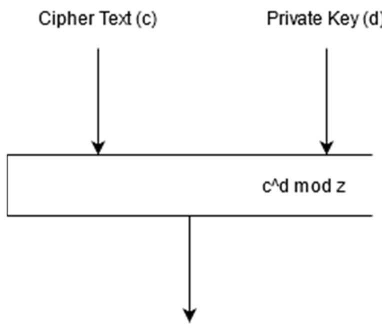


Fig. 7. RSA decryption.

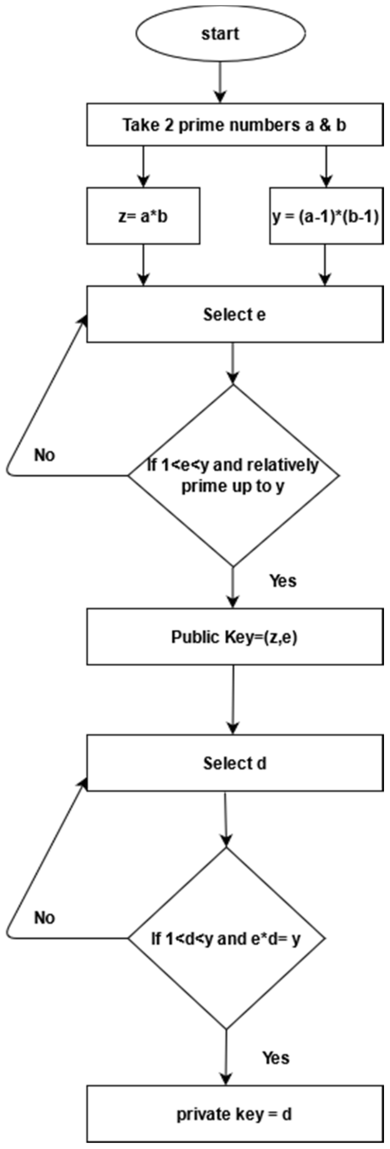


Fig. 8. RSA key generation.

The function definitions are:

```
def encrypt_data_message(pk, plaintext):
key, n = p_k
cipher_text = [pow(ord(char), key, n) for char in plaintext]
return cipher_text
def decrypt_data_message(pk, cipher_text):
key, n = p_k
aux = [str(pow(char, key, n)) for char in ciphertext]
plain = [chr(int(char2)) for char2 in aux]
return join(plain)
```

D. Blowfish

Blowfish is a symmetric key square code with a variable key length from 32 to 448 bits and a block size of 64 bits. Blowfish is faster than DES and is considered a promising encryption algorithm. Blowfish's advantages are its speed, the fewer memory requirements, whereas it is straightforward and more secure as it uses variable key length [5]. Blowfish is a block cipher having a variable order_of_byte which finds out the number of bytes and uses the big-endian byte order. The functions Per_array and Sub_boxes are used to define array and substitute boxes. Two static methods are used, namely data_encrypt and data_decrypt, to get the encrypted and decrypted data.

The function definitions are:

```
def data_encrypt(L, R, P, S1, S2, S3, S4, u4_1_pack, u1_4_unpack):
for p1, p2 in P[:-1]:
L ^= p1
a, b, c, d = u1_4_unpack(u4_1_pack(L))
R ^= (S1[a] + S2[b] ^ S3[c]) + S4[d] & 0xffffffff
R ^= p2
a, b, c, d = u1_4_unpack(u4_1_pack(R))
L ^= (S1[a] + S2[b] ^ S3[c]) + S4[d] & 0xffffffff
p_penultimate, p_last = P[-1]
return R ^ p_last, L ^ p_penultimate
def data_decrypt(L, R, P, S1, S2, S3, S4, u4_1_pack, u1_4_unpack):
for p2, p1 in P[:0:-1]:
L ^= p1
a, b, c, d = u1_4_unpack(u4_1_pack(L))
R ^= (S1[a] + S2[b] ^ S3[c]) + S4[d] & 0xffffffff
R ^= p2
a, b, c, d = u1_4_unpack(u4_1_pack(R))
L ^= (S1[a] + S2[b] ^ S3[c]) + S4[d] & 0xffffffff
p_first, p_second = P[0]
return R ^ p_first, L ^ p_second
def __init__(self, key, byte_order = "large", P_array = PI_P_ARRAY, S_boxes = PI_S_BOXES):
```

1) Sub-Key Generation

Keys should be ready before data encryption or decryption. The variable-length key is produced using K1, K2, K3 ⋯ Kn, where [$1 \leq n \leq 14$] and every K has 32-bits. Key length can be anything from 32 to 448 bits. The P-array has eighteen 32-cycle sub-keys: P1, P2… P18. Four S-boxes or replacement boxes are used, each having 256 entries, and all 256 elements are of 32bits.

S1 $\rightarrow$ S0, S1, … S255

S2 $\rightarrow$ S0, S1, … S255

S3 $\rightarrow$ S0, S1, … S255

S4 $\rightarrow$ S0, S1, … S255

The sub-keys are produced as:

- Instate the P-exhibit and four S-boxes with a static string. This string contains the hexadecimal digits of π.
- The main component in the P-exhibit (P1) is XORed with the initial 32 pieces of the key K1, the second component in the P-cluster (P2) is XORed with the second 32 pieces of the key K2. Rehash until every P-cluster component is XORed with the key pieces.
- Encode every string by blowfish encryption calculation utilizing the sub keys from P1 to P18.
- Change P1 and P2 with the yield of the third step.
- Utilize the adjusted sub keys to encode the yield of the third step.
- Change P3 and P4 with the yield of the fifth step. This interaction proceeds until the whole of the P-cluster is changed.

2) Data Encryption

The following steps are used for data encryption in the Blowfish algorithm:

- The 64-bit plain content is isolated into two sections, left 32-bit and right 32-bit.
- The left 32-bit is XORed with P1 and the yield is shipped off. Blowfish Function F is additionally utilized as right 32-digit for the next round
- The yield of capacity F is XORed with the right 32-digit and the result is utilized as the left 32-bit of the next round.
- This cycle is continued for 18 rounds, and the left and right output of round 18 are joined to give a 64-bit figure text.

3) Blowfish Function

Blowfish Function (F) uses the following process:

- The 32-bit input is divided into four 8-bit data, one for each S-box.
- The S-boxes substitute 8-bit data into 32-bits.
- The output of the first two S-boxes is added.
- The result of the third step is XORed with the output of S-box 3.
- The result of the fourth step is added to the yield of S-box 4.
- The result of the fifth step is the final output.

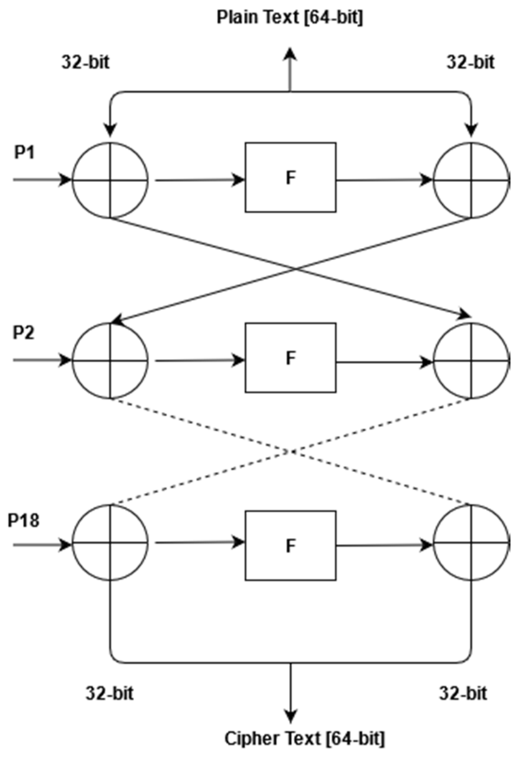


Fig. 9. Blowfish block diagram.

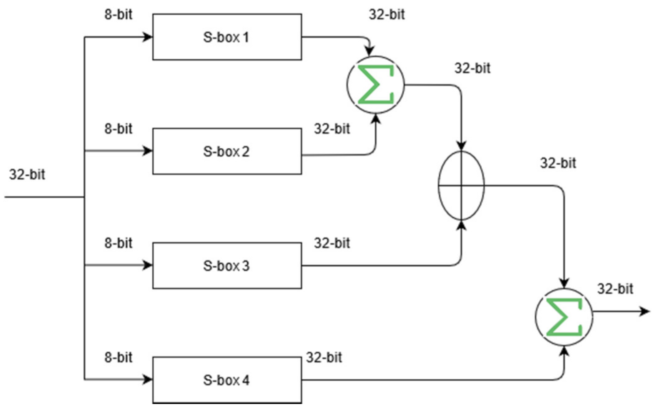


Fig. 10. Blowfish function F block diagram.

III. Evaluation Parameters

Each encryption strategy has advantages and disadvantages. Specific details, such as speed and cost requirements, should be examined before choosing an appropriate cryptographic algorithm for an application. This section describes the parameters used to evaluate those algorithms.

A. Encryption Time

Encryption time is the time needed to transform a plaintext to ciphertext and depends on three parameters: key size, plaintext block size, and mode. In this study, encryption time was estimated in milliseconds (ms). Encryption time impacts a framework's implementation, as shorter encryption time makes the framework more quick and responsive.

B. Decryption Time

Decryption time is the time needed to convert ciphertext into plaintext. Like encryption time, shorter decryption time increases a framework's responsiveness and speed. Decryption time is also measured in ms.

C. Memory Used

Each algorithm requires a different amount of memory space to execute. The memory size needed depends upon the number and kind of calculations, key size, and presentation vectors and affects system's cost.

D. Avalanche Effect

A desirable property of cryptographic algorithms is when a slight change in input changes the output significantly. This property is called dissemination or torrential slide impact. This is assessed using hamming distance, via the following equation:

$$Avalanche\ effect = hamming\ distance \div file\ size \quad (1)$$

E. Entropy

Irregularity or vulnerability is a significant property in cryptography since data should not be speculated by an attacker. Entropy is a measure of irregularity in data. Higher entropy is essential for a more intricate connection between key and ciphertext. This property is called disarray and is determined by Shannon's formula.

IV. Implementation

The encryption and decryption algorithms of AES, DES, RSA, and Blowfish algorithms were implemented in Java. Java has packages that provide extensive support for AES and DES. Three parameters were identified for each algorithm: memory used in KBs, encryption and decryption times in ms. Four different plain texts were used for each algorithm.

V. Results and Discussion

A. Encryption Time

The result comparison is exhibited in Figure 11. It can be noticed that Blowfish has the sortest encryption time.

B. Decryption Time

As it can be noted from Figure 12, decryption time is generally lesser than encryption time, while the RSA decryption requires longer and the Blowfish requires lesser time.

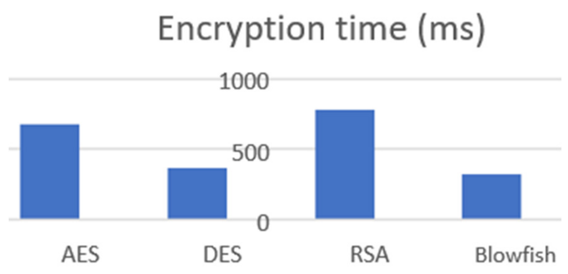


Fig. 11. Encryption time in ms for each algorithm.

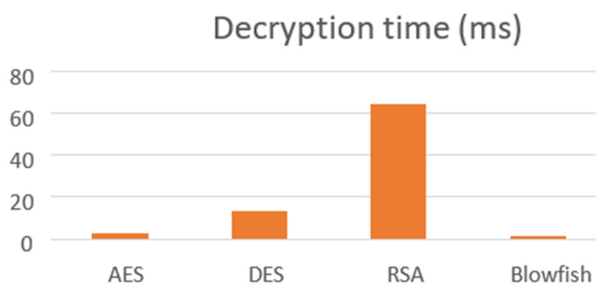


Fig. 12. Encryption time in ms for each algorithm.

C. *Memory Used*

Memory utilization specifies the memory usage in both the encryption and decryption processes. Figure 13 shows that RSA consumed more memory, due to its longer key length and two-step process nature, as it generated a pair of public and private keys before encryption.

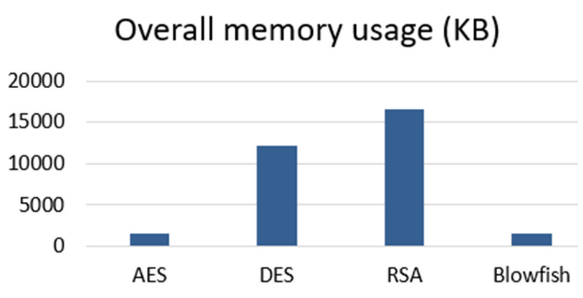


Fig. 13. Memory used by each algorithm.

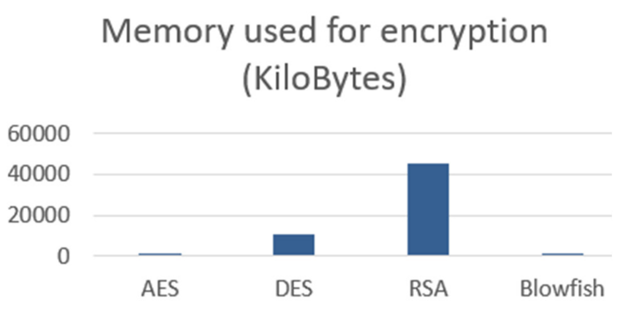
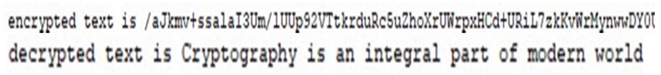


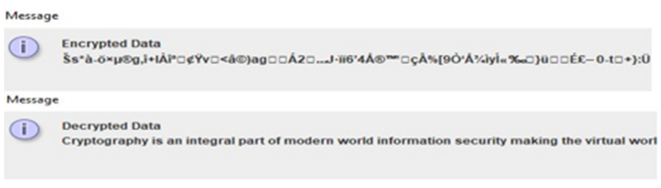
Fig. 14. Memory used in encryption by each algorithm.

The decryption memory usage comes to be negligible (almost 0) when compared to encryption. The snapshots in Figures 15-19 show the outputs observed after encrypting and decrypting a text.



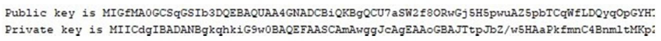
```
encrypted text is /aJkmv+ssalaI3Um/1UUp92VTtkrduRc5uZhoXrUWrpxHCd+URiL7zkKvWrMynwwDY0
decrypted text is Cryptography is an integral part of modern world
```

Fig. 15. AES algorithm output.



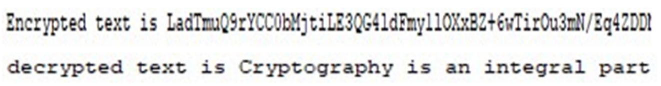
```
Message
Encrypted Data
Šs˚à-ö×µ®g,ì+lÀl³□¢Ÿv□<å©)ag□□Á2□…J·ïï6'4Å®™□çÅ%[9Ò'Å¾ìyÍ«‰□}ü□□É£–0·t□•}:0
Message
Decrypted Data
Cryptography is an integral part of modern world information security making the virtual worl
```

Fig. 16. DES algorithm output.



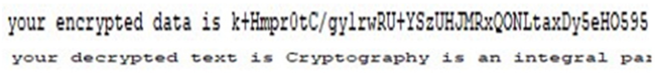
```
Public key is MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCU7aSW2f8ORwGj5H5pwuAZ5pbTCqWfLDQyqOpGYH
Private key is MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAJTtpJbZ/w5HAaPkfmnC4BnmltMKp
```

Fig. 17. RSA key generation step.



```
Encrypted text is LadTmuQ9rYCC0bMjtiLE3QG4ldFmyl1OXxBZ+6wTirOu3mN/Eq4ZDD
decrypted text is Cryptography is an integral part
```

Fig. 18. RSA output.



```
your encrypted data is k+Hmpr0tC/gylrwRU+YSzUHJMRxQONLtaxDy5eHO595
your decrypted text is Cryptography is an integral pa
```

Fig. 19. Blowfish output.

VI. Conclusion

Every cryptographic algorithm has its strengths and weaknesses. RSA is comparatively costly in terms of time and power consumption. Blowfish is effective for applications that require fast and secure communication, as it has lesser encryption and decryption times. AES is a very secure algorithm with a tradeoff of more memory usage and encryption time, whereas DES has a lesser memory footprint. Moreover, since decryption time is considerably lesser than encryption time, it requires less computational power making it highly effective in electronic devices having power consumption and battery life as crucial parameters [21]. For example, the control signals sent to sensors in wireless sensor networks that must be encrypted to avoid attacking or eavesdropping [6]. Since those sensors' operation depends heavily on battery life, decryption algorithms can be used without significantly increasing power consumption or memory usage.

References

[1] P. Patil, P. Narayankar, D. G. Narayan, and S. M. Meena, "A Comprehensive Evaluation of Cryptographic Algorithms: DES, 3DES, AES, RSA and Blowfish," *Procedia Computer Science*, vol. 78, pp. 617–624, Jan. 2016, https://doi.org/10.1016/j.procs.2016.02.108.

[2] A. Jeeva, D. V. Palanisamy, and K. Kanagaram, "Comparative Analysis of Performance Efficiency and Security Measures of some Encryption Algorithms," *International Journal of Engineering Research and*, vol. 2, no. 3, pp. 3033–3037, May 2012.

[3] H. O. Alanazi, B. B. Zaidan, A. A. Zaidan, H. A. Jalab, M. Shabbir, and Y. Al-Nabhani, "New Comparative Study Between DES, 3DES and AES within Nine Factors," vol. 2, no. 3, p. 6, Mar. 2010.

[4] R. Tripathi and S. Agrawal, "Comparative Study of Symmetric and Asymmetric Cryptography Techniques," *International Journal of Advanced Foundation and Research in Computer*, vol. 1, no. 6, Jun. 2014.

[5] M. S. Mahindrakar, "Evaluation of Blowfish Algorithm based on Avalanche Effect," *International Journal of Innovations in Engineering and Technology*, vol. 4, no. 1, pp. 99–103, Jun. 2014.

[6] R. Pahal, "Efficient Implementation of AES," International Journal of Advanced Research in Computer Science and Software Engineering, pp. 290–295, Jul. 2013.

[7] Y. Kumar, R. Munjal, and H. Sharma, "Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures," *International Journal of Computer Science and Management Studies*, vol. 11, no. 3, pp. 60–63, Oct. 2011.

[8] C. P. Mandal, "Superiority of Blowfish Algorithm," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 9, pp. 196–201, Sep. 2012.

[9] S. Karthik and A. Muruganandam, "Data Encryption and Decryption by Using Triple DES and Performance Analysis of Crypto System," *International Journal of Scientific Engineering and Research*, vol. 2, no. 11, pp. 24–31, Nov. 2014.

[10] G. Prashanti, S. Deepthi, and K. Sandhya Rani, "A Novel Approach for Data Encryption Standard Algorithm," *International Journal of Engineering and Advanced Technology*, vol. 2, no. 5, pp. 264–267, Jun. 2013.

[11] D. P. Mahajan and A. Sachdeva, "A Study of Encryption Algorithms AES, DES and RSA for Security," *Global Journal of Computer Science and Technology*, vol. 13, no. 15, Jul. 2013.

[12] N. Kaur and S. Sodhi, "Data Encryption Standard Algorithm (DES) for Secure Data Transmission," *IJCA Proceedings on International Conference on Advances in Emerging Technology*, vol. ICAET 2016, no. 2, pp. 31–37, Oct. 2016.

[13] M. R. Asassfeh, M. Qatawneh, and F. M. ALAzzeh, "Performance Evaluation of Blowfish Algorithm on Supercomputer IMAN1," *International journal of Computer Networks & Communications*, vol. 10, no. 2, pp. 43–53, Mar. 2018, https://doi.org/10.5121/ijcnc.2018.10205.

[14] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, "Wireless sensor networks: a survey on recent developments and potential synergies," *The Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, Apr. 2014, https://doi.org/10.1007/s11227-013-1021-9.

[15] M. A. Al-Shabi, "A Survey on Symmetric and Asymmetric Cryptography Algorithms in information Security," *International Journal of Scientific and Research Publications (IJSRP)*, vol. 9, no. 3, pp. 576–589, Mar. 2019, https://doi.org/10.29322/IJSRP.9.03.2019.p8779.

[16] D. P. Joseph and M. Krishna, "Cognitive Analytics and Comparison of Symmetric and Asymmetric Cryptography Algorithms," *International Journal of Advanced Research in Computer Science*, vol. 6, no. 3, pp. 51–56, May 2015.

[17] A. Gupta and N. K. Walia, "Cryptography Algorithms: A Review," *International Journal of Engineering Development and Research*, vol. 2, no. 2, pp. 1667–1672, 2014.

[18] S. Kumari, "A research Paper on Cryptography Encryption and Compression Techniques," *International Journal of Engineering and Computer Science*, vol. 6, no. 4, pp. 20915–20919, Apr. 2017.

[19] J. Thakur and N. Kumar, "DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis," *International Journal of Emerging Technology and Advanced Engineering*, vol. 1, no. 2, pp. 5–12, Dec. 2011.

[20] E. S. I. Harba, "Secure Data Encryption Through a Combination of AES, RSA and HMAC," *Engineering, Technology & Applied Science Research*, vol. 7, no. 4, pp. 1781–1785, Aug. 2017, https://doi.org/10.48084/etasr.1272.

[21] A. S. Alshammari, "Comparison of a Chaotic Cryptosystem with Other Cryptography Systems," *Engineering, Technology & Applied Science Research*, vol. 10, no. 5, pp. 6187–6190, Oct. 2020, https://doi.org/10.48084/etasr.3745.