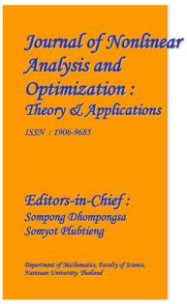
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1 : 2024
ISSN : **1906-9685**



UNLOCKING SECURITY: AN IN-DEPTHANALYSISOFKEY BASED CRYPTOGRAPHIC ALGORITHMSANDTHEIR USES

Swathi Kambhampati, Maramreddy Kusuma, Mandava Hitesh Chandrachowdary, Mirzatabassum Fathema, Nagudu Harsha Vardhan, Department of Electronics & Communication Engineering, NRI Institute of Technology, Pothavarappadu (V), Agiripalli (M), Eluru (Dt)-521212

Abstract:
Information security is the process of safeguarding data. It safeguards its availability, privacy and integrity. The capacity to secure data from attacks, as well as efficiency and speed are the two key features that separate one cryptography algorithm from another. Security is the most difficult issue in the present world, and the various security dangers in data security must be avoided in order to provide consumers with more privacy while still permitting high information availability and Integrity. Data encryption employing various data encryption techniques will increase the security of data transmission. This paper primarily focuses on a comparative analysis of symmetric algorithms (AES, DES).
Keywords: DES, AES, Encryption, Decryption.

1. Introduction

John devoted significant time to exploring various methods of enhancing DES. While considering options like longer key lengths and more encryption rounds, he realized that these approaches might not provide lasting security improvements. Instead, he devised a novel solution that would enable the company to continue using DES while enhancing its security. John's approach involved using a secondary key to further encrypt DES-encrypted data. This secondary key, randomly generated for each encryption operation, would be stored separately from the encrypted data. To decrypt the data, the secondary key would need to be provided along with the primary key used for DES encryption. Implementing this approach required significant changes to the company's encryption system, including modifying the key generation process and adding new storage and retrieval mechanisms for the secondary key. Despite these challenges, John believed that the enhanced security benefits would outweigh the costs of these changes. After months of development and testing, John's enhanced encryption system was deployed in the company's production environment. The results were impressive - the system maintained the security of the company's data while also allowing for the continued use of the legacy DES algorithm. John's innovative approach to enhancing DES's security earned praise from his colleagues and superiors, establishing him as a leader in encryption and security. His work paved the way for further improvements to the company's security systems and demonstrated the importance of innovation and creativity in the constantly evolving world of technology. The paper also presents compact architectures for AES Mix Column and its inverse, aimed at reducing the area cost in resulting AES implementations. In the hardware implementation of AES with direct mapping substitute byte optimization, the Mix Column/Inverse Mix-Column transformation demands the utilization of logic resources, affecting the critical path delay and resulting throughput. The proposed architectures, based on byte and bit-level decomposition, lead to two types of architecture that demonstrate deeper resource sharing within byte and between bytes, along with rearrangement of output terms with respect to FPGA architecture at the bit level. The proposed architectures were investigated on an FPGA-based implementation platform, resulting in a 40% reduction in reconfigurable logic area compared to separate implementation of Mix-Column and Inverse Mix-Column, along with a 9% reduction in path delay. Experimental results demonstrate that the proposed

architecture can significantly reduce the area cost compared with previous implementations. This paper introduces an outer-round only pipelined architecture for FPGA implementation of the AES-128 encryption processor. The design utilizes Block RAM to store S-box values and exploits two types of Block RAM. By consolidating operations into a single round, critical delay can be reduced. As network transmission speeds increase to gigabits per second (Gbps), software-based implementations of cryptographic algorithms become inadequate. Hardware-based implementations, using optimization techniques like pipelining and look-up tables, can significantly improve throughput and reduce key generation time. Additionally, packaging cryptographic algorithms and key generation in a chip enhances physical security, as they cannot easily be read or altered by external attackers.Some implementations use field programmable gate arrays (FPGA), while others use application-specific integrated circuits (ASIC). ASICs lack flexibility and have high development costs and long development cycles. The AES algorithm has become the default choice for various security services due to its security and versatility. This paper proposes a high-speed, non-pipelined FPGA implementation of the AES-CCMP cipher for wireless LAN, utilizing Xilinx development tools and Virtex-It Pro FPGA circuits. The AES CCMP core aims to provide high speed with sufficient security, operating at 194/148MHz for encryption/decryption, resulting in throughputs of 2.257 Gbits/sec and 1.722 Gbits/sec, respectively. Compared to software implementations, hardware-based approaches offer higher security and faster encryption speeds. The paper includes a comparison with similar existing implementations. The AES S-box, comprising a 256-entry table, replaces each input byte of the State matrix independently. The S-box performs two transformations: first, replacing each input byte with its multiplicative inverse in GF(28), with the element {00} being mapped onto itself; second, applying an affine transformation over GF(28). For decryption, the inverse S-box is obtained by applying the inverse affine transformation followed by multiplicative inversion in GF(28).The increasing need for data protection in computer networks has led to the development of several cryptographic algorithms. Hardware implementations of these algorithms are more physically secure than software implementations, as they cannot be easily modified by outside attackers. Hardware implementation offers better speed and reliability, making it a wise choice for achieving higher performance in today's heavily loaded communication networks. This paper presents a hardware implementation of the AES algorithm using Xilinx– Virtex5 Field Programmable Gate Array (FPGA). To achieve higher speed and lower area, operations like Sub Byte, Inverse Sub Byte, Mix Column, and Inverse Mix Column are designed as Look Up Tables (LUTs) and Read Only Memories (ROMs). Encryption is typically performed just before data transmission to fully utilize channel resources, requiring the encryption algorithm to match the data transmission speed. Achieving high throughput for encryption in a communication channel with a high data rate is challenging. The AES algorithm (also known as the Rijndael algorithm) is widely used in wired and wireless digital communication networks for secure data transmission, especially over public networks. This paper presents a hardware implementation of the AES Rijndael Encryption and Decryption Algorithm using Xilinx Virtex-7 FPGA. The hardware design is based on pre-calculated look-up tables (LUTs), resulting in a less complex architecture with high throughput and low latency. The AES has three formats: AES-128, AES-192, and AES-256, and the encryption and decryption blocks for all three formats are efficiently designed using Verilog-HDL and synthesized on a Virtex-7 XC7VX690T chip. Power analysis is conducted using Xilinx XPower Analyzer, and the proposed architecture demonstrates good efficiency in terms of latency, throughput, speed/delay, area, and power.The Advanced Encryption Standard (AES) is an approved cryptographic algorithm that can protect electronic data and can be programmed in software or implemented in pure hardware. Field Programmable Gate Arrays (FPGAs) offer a faster and more customizable solution for AES implementation. This paper presents the implementation of AES on FPGA using VHDL and ModelSim SE PLUS 5.7g software for simulation and optimization of synthesizable VHDL code. The code is synthesized and implemented on Xilinx - Project Navigator, ISE 8.2i suite, using an iterative design approach to minimize hardware consumption. The proposed architecture integrates the AES encrypter and decrypter, resulting in a low-complexity architecture suitable for hardware-critical applications like smart cards, PDAs, and mobile phones. Additionally, a speech encryption algorithm based on a 4D hyper chaotic system is proposed to protect speech security in the cloud, showing good discrimination, robustness, recall, precision, and retrieval efficiency compared to existing methods[1-7].

2. Proposed Method

2.1 INTRODUCTION OF DES

DES (Data Encryption Standard) algorithm purpose is to provide a standard method for protecting sensitive commercial and unclassified data. In this same key used for encryption and decryption process.

2.2 ENCRYPTION AND DECRYPTION

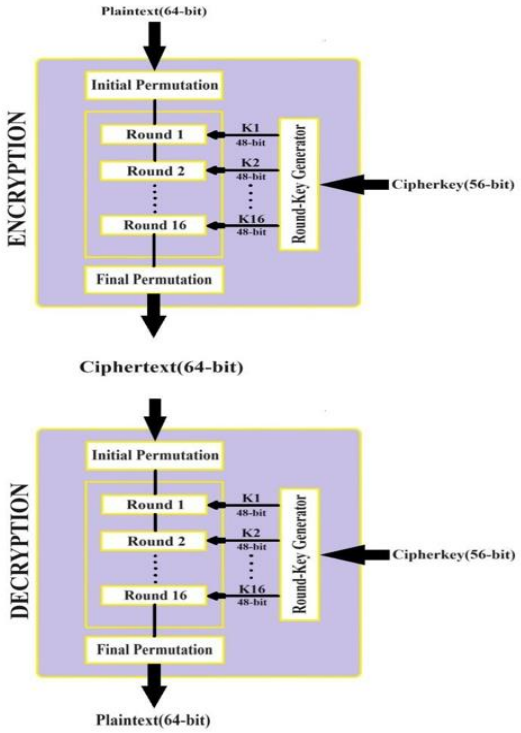


Figure.1 DES Algorithm

DES processes a 64-bit plaintext input and a 56-bit key (with 8 parity bits), producing a 64-bit output block.

2.3 STEPS INVOLVED IN DES ALGORITHM

The plaintext block undergoes bit shifting operations.
The 8 parity bits are removed from the key using a key permutation.
The plaintext and key undergo the following process:
The key is divided into two 28-bit halves.
Each half of the key is rotated by one or two bits, depending on the round.
The rotated halves are combined and subjected to a compression permutation to reduce the key to 48 bits, which is used to encrypt the current round's plaintext block.
The rotated key halves from the previous step are used in the next round.
The data block is divided into two 32-bit halves.
One half undergoes an expansion permutation to increase its size to 48 bits.
The result of step 6 is XORed with the 48-bit compressed key from step 3.
The output of step 7 is passed through an S-box, which substitutes key bits and reduces the block back to 32 bits.
The output of step 8 undergoes a P-box permutation.
The result from the P-box is XORed with the other half of the data block. The two halves are then swapped and become the input for the next round.

2.4 ADVANCED ENCRYPTION STANDARD

In the AES method, we implement AES-128, using a 128-bit key for encrypting 128-bit data with the same S-box. The encryption process involves 14 rounds, with each round comprising Add Round Key, SubBytes, ShiftRows, and MixColumns operations. Round 0 includes only the Add Round Key operation, while Round 14 includes SubBytes, ShiftRows, and Add Round Key operations, requiring 3 clock cycles. Rounds 1 to 13 encompass all four operations, with each operation executed in a distinct clock cycle. Consequently, once the hardware is implemented for Add Round Key, SubBytes,

ShiftRows, and MixColumns, the same hardware can be used for all 14 rounds, as none of the operations share the same clock cycle.

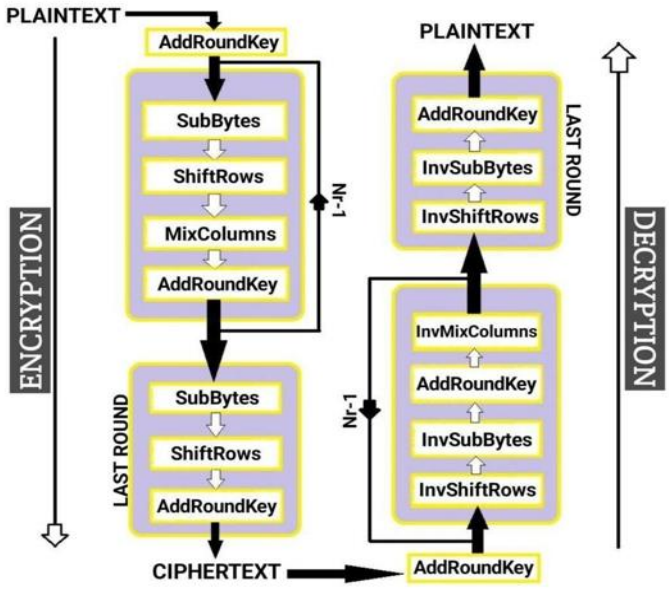


Figure.2. AES Algorithm

The AES encryption process consists of a specific sequence of four operations, namely AddRoundKey, SubBytes, ShiftRows, and MixColumns. This process is serial, meaning the output of one round becomes the input for the next round. Consequently, the same hardware can be utilized for each round. The data is structured in a 128-bit matrix, where each column contains four elements of 8 bits each, totaling 32 bits per word. For the conventional AES algorithm, a total of one S-box and one MixColumns operation are required.

2.5 IMPLEMENTATION OF AES ALGORITHM

The AES algorithm implementation involves four operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. The architecture for the 256-bit AES algorithm is depicted above. There are a total of 14 rounds for both encryption and decryption. After encryption, the ciphertext is transmitted across the channel and decrypted using the same key.

In the 128-bit AES algorithm, the key size is 128 bits, and all data sizes are 128 bits, including the message to be encrypted, the ciphertext, and the decrypted message. The internal data structure for 128-bit data is a 4x4 matrix, where each element is 8 bits. Since all operations are performed on a column basis, the 128-bit data is converted into a 4x4 matrix with each element being 8 bits.

The 128-bit AES encryption block is implemented in 14 rounds, with each round consisting of AddRoundKey, SubBytes, ShiftRows, and MixColumns. Round 0 involves only the AddRoundKey operation, while Round 14 includes SubBytes, ShiftRows, and AddRoundKey operations, requiring 3 clock cycles. Rounds 1 to 13 include all four operations. Each operation is performed in a distinct clock cycle. Therefore, once the hardware is implemented for AddRoundKey, SubBytes, ShiftRows, and MixColumns, it can be used for all 14 rounds without sharing clock cycles. The sequence of round operations with the specific sequence of operations is shown in the figure above, illustrating the serial process of the AES algorithm where the output of one round is the input to the next, allowing for the use of the same hardware for each round.

Relation between Key size and number of rounds is given by the mathematical expression

$$Nr = (Nk/32) + 6$$

Nr = Number of rounds

Nk = Key size

Figure.3. Mathematical Expression

Round 0	01 cycle
Round 1 to 13	52 cycles
Round 14	03 cycles
Total	**56 cycles**

Figure.4. Cycles required in Each Round

2.6 5.3 FOUR STAGES OF EACH ROUND

SubBytes: The SubBytes transformation, used in encryption, involves substituting each byte of the state with a value from a lookup table called the S-Box. This is a non-linear byte substitution applied independently to each byte of the state. For decryption, the inverse operation, InvSubBytes, is used. ShiftRows: The ShiftRows operation in encryption involves shifting the bytes in each row of the state matrix to the left. The number of shifts is based on the row number: no shift for row 0, 1 shift for row 1, 2 shifts for row 2, and 3 shifts for row 3. MixColumns: The MixColumns transformation operates on the columns of the state matrix, transforming each column into a new column using matrix multiplication with a constant square matrix. This transformation is performed in the Galois Field, treating the bytes as polynomials rather than simple numbers. AddRoundKey: The AddRoundKey operation is applied one column at a time, similar to MixColumns. It involves adding a round key to each column matrix, performing a matrix addition operation.

2.7 5.4 ENCRYPTION AND DECRYPTION

In encryption, SubBytes, ShiftRows, MixColumns, and AddRoundKey are performed in all rounds except the last round. The MixColumns transformation is omitted in the final round of encryption. For decryption, the process mirrors encryption, but with nine rounds of InverseShiftRows, InverseSubBytes, InverseAddRoundKey, and InverseMixColumns transformations.

2.8 5.5 RIJNDAEL S-BOX

The Rijndael S-Box is a square matrix used in the Rijndael cipher, functioning as a look-up table. It is generated through a process that involves determining the multiplicative inverse of a given number in GF(28) and then applying an affine transformation to the result.

Multiplicative Inverse Phase: In this phase, the input byte is inverted by substituting a value from the multiplicative inverse table.

Affine Transformation: The selection of the irreducible polynomial and the designated byte are crucial in this phase. For Rijndael AES, the irreducible polynomial x^8 + x^4 + x^3 + x + 1 is used, and the designated byte 0x63 is chosen. The affine transformation consists of two operations: multiplication of an 8x8 square matrix and addition of an 8x1 constant column matrix.

3.Results and Discussion

3.1 RTL SCHEMATIC OF AES

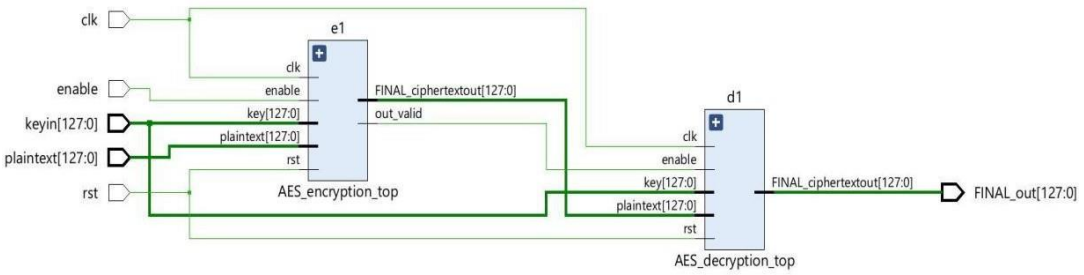


Figure.5. RTL Schematic of AES

3.1.1 SIMULATION RESULTS OF AES

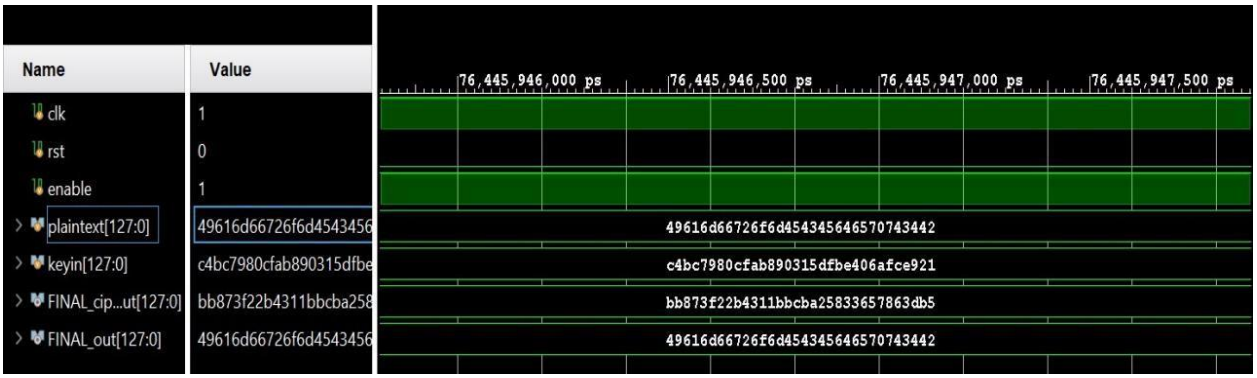


Figure.6. Simulation results of AES

3.2 AREA IN AES

Name	Slice LUTs (134600)	Slice Registers (269200)	Bonded IOB (400)	BUFGCTRL (32)
top	4636	3249	387	3
d1 (AES_decryption_top)	2302	1688	0	0
e1 (AES_encryption_top)	2334	1561	0	0

Figure.7. Area in AES

3.3 DELAY IN AES

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay
Path 1	∞	9	9	15	e1/ins2/u0/w_reg[3][21]/C	e1/ins2/u0/w_reg[1][30]/D	6.548	1.401	5.147
Path 2	∞	9	9	15	e1/ins2/u0/w_reg[3][21]/C	e1/ins2/u0/w_reg[0][30]/D	6.545	1.398	5.147
Path 3	∞	9	9	15	e1/ins2/u0/w_reg[3][21]/C	e1/ins2/u0/w_reg[2][30]/D	6.545	1.398	5.147
Path 4	∞	9	9	15	e1/ins2/u0/w_reg[3][21]/C	e1/ins2/u0/w_reg[3][30]/D	6.545	1.398	5.147
Path 5	∞	9	9	15	d1/ins2/u0/w_reg[3][21]/C	d1/ins2/u0/w_reg[0][30]/D	6.543	1.401	5.142
Path 6	∞	9	9	15	d1/ins2/u0/w_reg[3][21]/C	d1/ins2/u0/w_reg[1][30]/D	6.540	1.398	5.142

Figure.8. Delay in AES

3.4 RTL SCHEMATIC OF DES:

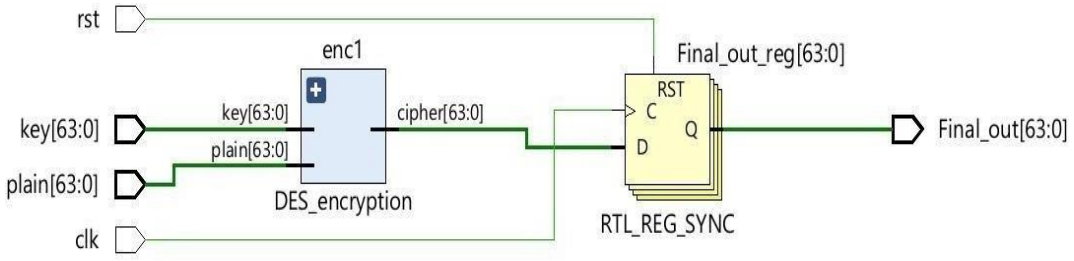


Figure.9. RTL Schematic of DES

3.5 SIMULATION RESULTS OF DES

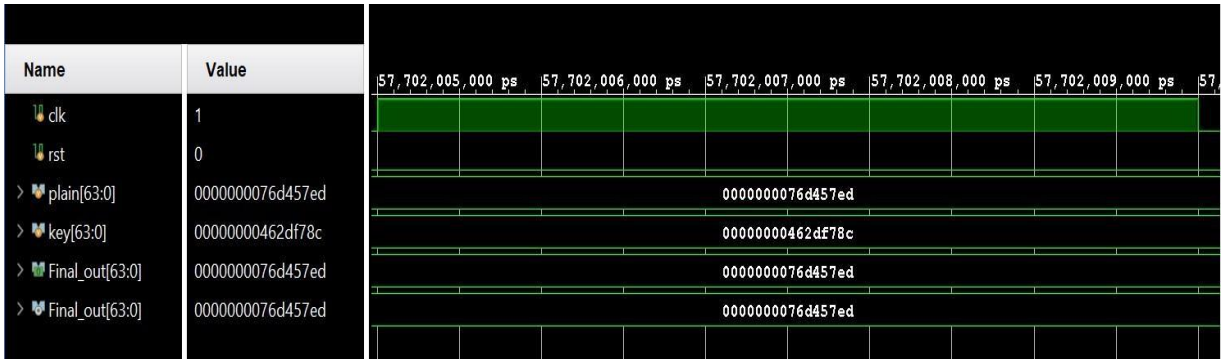


Figure.10. Simulation results of DES

3.6 AREA IN DES

Name	Slice Registers (269200)	Bonded IOB (400)	BUFGCTRL (32)
DES_TOP	64	130	1

Figure.11. Area in DES

3.7 DELAY IN DES

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement
Path 1	∞	2	2	1	Final_out_reg[0]/C	Final_out[0]	3.521	2.876	0.646	∞
Path 2	∞	2	2	1	Final_out_reg[10]/C	Final_out[10]	3.521	2.876	0.646	∞
Path 3	∞	2	2	1	Final_out_reg[11]/C	Final_out[11]	3.521	2.876	0.646	∞
Path 4	∞	2	2	1	Final_out_reg[12]/C	Final_out[12]	3.521	2.876	0.646	∞
Path 5	∞	2	2	1	Final_out_reg[13]/C	Final_out[13]	3.521	2.876	0.646	∞
Path 6	∞	2	2	1	Final_out_reg[14]/C	Final_out[14]	3.521	2.876	0.646	∞

Figure.12. Delay in DES

3.8 EVALUATION TABLE FOR AREA, DELAY

	Area (LUT's)	**Delay (ns)**
AES TOP(128-bit)	4636	6.548
DES TOP(64-bit)	688	3.521

3.9 **Advantages- Dis-Advantages and Applications**

ADVANTAGES: AES ADVANTAGES:

- High security.
- Resistance to attacks.
- Speed of Operation.

DES ADVANTAGES:

- Simplicity.
- Compatibility.

DISADVANTAGES: AES DISADVANTAGES:

- 192 and 256 have far less security.
- Hard to implement with software.

DES DISADVANTAGES:

- Weakly secured algorithm.
- It is slower than AES.
- It is not flexible.

Cloud Storage: Data stored in the cloud is encrypted before transmission and stored securely. Encryption keys can be managed by the user or the cloud service provider, and access to encrypted data is controlled through authentication mechanisms. Decryption involves retrieving the encrypted

data and using the appropriate decryption key. VLSI-based cryptographic hardware accelerators can improve decryption performance, enabling fast and secure access to cloud-stored data.
VPNs and Wi-Fi Security Protocols: VPNs and Wi-Fi security protocols use encryption algorithms to secure data during transmission. Data is encrypted at the source and decrypted at the destination. This process ensures data confidentiality and security while traveling over the network.
Mobile Apps (e.g., WhatsApp and LastPass): Some mobile apps implement end-to-end encryption, where data is encrypted on the sender's device and decrypted only on the recipient's device. This approach prevents intermediaries from accessing or intercepting the data. Robust encryption mechanisms and secure communication protocols in mobile apps protect user data from unauthorized access.

4. Conclusion

After conducting a comparative analysis of AES and DES for security applications, it can be concluded that AES is more secure and efficient than DES. AES is a symmetric encryption algorithm that utilizes larger key sizes (128-bit, 192-bit, or 256-bit) compared to DES, which uses a fixed 56-bit key. AES offers superior resistance to cryptographic attacks, such as brute-force attacks and differential cryptanalysis, due to its larger key size and more complex key scheduling algorithm. Conversely, DES has been shown to be vulnerable to various attacks, including brute-force attacks, differential cryptanalysis, and linear cryptanalysis. Additionally, DES is slower than AES due to its simpler key scheduling algorithm. In terms of implementation, AES is widely utilized in a variety of security applications, including online banking, data encryption, and military applications, while DES is seldom used in modern systems due to its vulnerabilities and limitations. Overall, AES is a more secure and efficient encryption algorithm than DES and is the preferred choice for most security applications.

Reference

1. Prashanti.G, Deepthi.S & Sandhya Rani.K. ”A Novel Approach for Data Encryption Standard Algorithm”. International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958, Volume-2, Issue-5, June 2013,pp. 264).
2. Nalini C. Iyer ; Deepa ; P.V. Anandmohan ; D.V. Poornaiah “Mix/InvMix- Column decomposition and resource sharing in AES”.
3. Yulin Zhang ; Xinggang Wang; “Pipelined implementation of AES encryption based on FPGA” 2010 IEEE International Conference on Information Theory and Information Security.
4. C. Sivakumar ; A. Velmurugan ; “High Speed VLSI Design CCMP AES Cipher for WLAN (IEEE 802.11i)” 2007 International Conference on Signal Processing, Communications and Networking.
5. P. S. Abhijith ; Mallika Srivastava ; Aparna Mishra ; Manish Goswami ; B. R. Singh ; “High performance hardware implementation of AES using minimal resources” 2013 International Conference on Intelligent Systems and Signal Processing (ISSP).
6. N. S. Sai Srinivas ; Md. Akramuddin; “FPGA based hardware implementation of AES Rijndael algorithm for Encryption and Decryption” 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT).
7. Ashwini M. Deshpande ; Mangesh S. Deshpande ; Devendra N. Kayatanavar; “FPGA implementation of AES encryption and decryption” 2009 International Conference on Control, Automation, Communication and Energy Conservation