International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 07 | July – 2025 DOI: 10.55041/ISJEM04733
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

High Security Encryption Using AES and Visual Cryptography

K. TULASI KRISHNA KUMAR, G. MADHU NISHA
Assistant Professor, Training & Placement Officer, MCA Final Semester,
Master of Computer Applications,
Sanketika Vidya Parishad Engineering College, Vishakhapatnam,
Andhra Pradesh, India.

Abstract
In the digital communication era, ensuring the security and confidentiality of sensitive information is crucial. This project proposes a robust encryption scheme that combines Advanced Encryption Standard (AES) with Visual Cryptography (VC) to achieve high security, reliability, and efficiency.The application allows users to encrypt and decrypt files through a user-friendly GUI built with Python's Tkinter library. Users provide a key to XOR each byte of the file's content for encryption, and the process is reversed for decryption. The tool includes comprehensive error handling to manage file access permissions, key validation, and other potential issues. Designed for users with varying technical expertise, this solution enhances data security by protecting sensitive information from unauthorized access while ensuring data integrity during storage and transmission.

INDEXING TERMS: Secure Data Transmission, Advanced Encryption Standard (AES), Visual Cryptography, Image Security, Dual-layer Encryption, Cryptographic Shares, Confidentiality

1. Introduction
In today's digital landscape, protecting sensitive data is more important than ever due to increasing threats from cyber-attacks and data breaches. Encryption plays a vital role by converting data into a secure format accessible only with the correct decryption key. This project introduces a File Encryption and Decryption Tool based on the Advanced Encryption Standard (AES), a widely trusted and efficient encryption method [1]. Developed using Python and a graphical interface built with Tkinter, the tool is designed to be user-friendly, enabling users of all technical backgrounds to encrypt or decrypt files with ease. It uses AES in Cipher Block Chaining (CBC) mode and applies a password-based key derivation function (PBKDF2) with SHA-256 hashing to generate secure keys from user-provided passwords [2]. To enhance security, each encryption session includes a unique salt and initialization vector (IV), ensuring that even identical inputs produce different encrypted results. The application allows users to select files, input passwords, and choose output locations, while preserving the original files. It also includes robust error handling to manage issues such as incorrect passwords or file access problems, providing helpful feedback to users. This tool offers a practical and secure solution for safeguarding personal, financial, or professional data.

1.2 Literature Survey
Early foundational work by Xuejia Lai and James L. Massey (1991) introduced concepts like substitution-permutation networks (SPNs) and iterative block ciphers, which laid the groundwork for AES. Joan Daemen and Vincent Rijmen (1998) later developed the Rijndael algorithm, selected as the AES standard due to its flexibility, simplicity, and strong resistance to cryptographic attacks. Zhang and Parhi (2002) focused on high-speed VLSI architectures for AES, proposing pipelining and loop unrolling techniques to improve encryption throughput in hardware. Prasithsangaree and Krishnamurthy (2003) compared AES and RC4 in wireless LANs, revealing AES's energy efficiency for small packets, which is crucial for mobile applications[4] . Rajalakshmi et al. (2010) explored a hardware-software co-design of AES on FPGAs, optimizing performance for low-cost embedded systems [5]. Garg et al. (2013) and Atha et al. (2013) enhanced AES implementations on FPGAs, focusing on the trade-offs between power, speed, and area, with pipelined designs achieving high throughput [7]. Finally, Subhadra et al. (2013) proposed a VHDL-based AES implementation for high-security applications, efficiently handling both encryption and decryption on a single FPGA. Collectively, these studies provide a comprehensive understanding of AES from theoretical design to practical, energy-efficient hardware implementations.

1.3 Existing System

Encryption systems vary in complexity and usability. Command-line tools like OpenSSL and GPG offer powerful encryption but require technical expertise. Integrated solutions, such as BitLocker on Windows or FileVault on macOS, are easier to use but may lack flexibility. Third-party applications offer varying levels of security and usability, with some potentially using weak encryption algorithms.File encryption tools include desktop applications like VeraCrypt and command-line utilities like OpenSSL, while commonly used algorithms include AES with CBC mode for security. Key derivation functions like PBKDF2 protect against brute-force attacks, and padding schemes like PKCS7 handle varying data lengths.However, existing systems often struggle with balancing security and usability, making them either too complex or less secure, and they may lack seamless integration with other software.

1.3.1 Challenges

1. Varying Security Levels in Third-Party Tools:
Some third-party applications may use outdated or weak encryption algorithms, posing potential security risks if users are unaware of underlying implementations.
2. Lack of Cross-Platform Support:
Some tools are platform-dependent, limiting their usability across different operating systems (e.g., Windows, macOS, Linux).
3. Insufficient Error Feedback:
Many tools do not provide meaningful error messages when issues like incorrect passwords or corrupt files occur, leaving users confused and frustrated.
4. Lack of Seamless Integration:
Existing tools often don't integrate well with other applications or workflows, which can disrupt user productivity and hinder widespread adoption.

1.4 Proposed System

The proposed system is a user-friendly file encryption and decryption tool developed using Python and Tkinter, designed to make data protection simple and accessible for all users. It features an intuitive graphical interface that guides users through the encryption and decryption process step by step. Users can easily select files using file dialogs and enter a key (password) to secure or unlock their files, eliminating the need for complex technical steps. The tool simplifies key management by using the same key for both encryption and decryption. It also includes robust error handling to notify users of any issues, such as incorrect keys or file access problems. Additionally, the integration of the tqdm library provides a visual progress bar, allowing users to track the status of their task in real-time. Overall, the system offers a simple, reliable, and cross-platform solution for securing files.

1.4.1 Advantages of proposed system

Ease Of Use : The tool's intuitive GUI and straightforward process make it accessible to users with varying levels of technical expertise. This ease of use encourages the adoption of encryption practices, enhancing data security.
Simplified Key Management : By allowing users to enter a key value directly and use it for both encryption and decryption, the tool simplifies the key management process. This approach reduces the risk of key interception and unauthorized access.
Robust Error Handling : The tool includes comprehensive error handling mechanisms that inform users of any issues, ensuring a smooth and reliable experience.
Visual Progress Feedback : The use of the 'tqdm' library to display progress bars provides users with real-time feedback on the status of their tasks, enhancing the user experience and transparency of the process.
Cross-Platform Compatibility : Built with Python and Tkinter, the tool is compatible with various operating systems, making it a versatile solution for users across different platforms.

2. Architecture

The architecture of the File Encryption and Decryption Tool is organized into four key layers, each with a specific role to ensure usability, security, and maintainability [3]. The **User Interface Layer**, built using

Python's Tkinter, offers a simple and intuitive graphical interface that allows users to select files, input passwords, and initiate encryption or decryption tasks. It also integrates the tqdm library to visually display task progress.

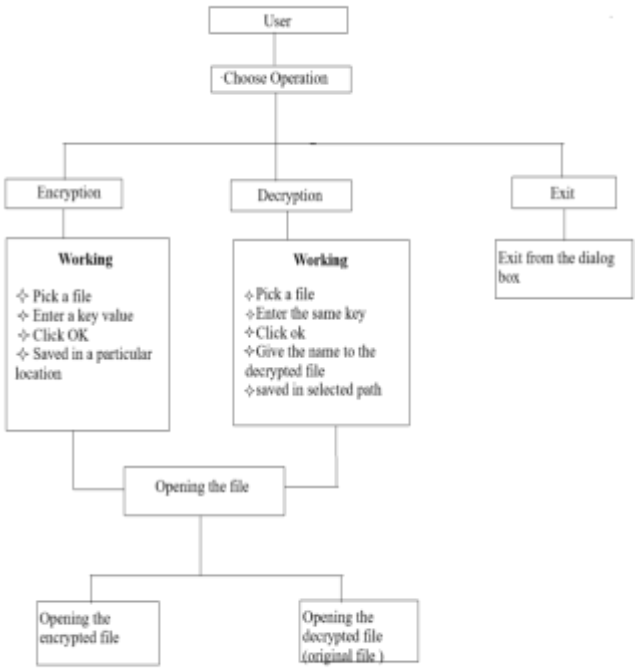


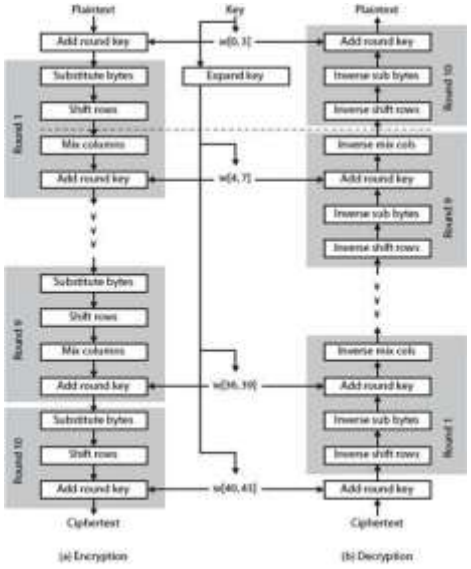
Fig: : [1] Architecture of File Encryption and Decryption

The **Application Logic Layer** controls the overall workflow, processes user inputs, manages file paths and passwords, and bridges communication between the GUI and the encryption/decryption functions. It includes strong error handling to ensure smooth operation and clear guidance for users in case of issues. The **Cryptographic Processing Layer** performs the core security functions, including AES encryption and decryption in CBC mode. It uses PBKDF2 with HMAC-SHA256 for secure key derivation and applies PKCS7 padding to align data with AES block sizes. Each session generates a random salt and IV to enhance security. Finally, the **File System Layer** manages all file operations, ensuring original files remain unmodified and that encrypted or decrypted output files are saved correctly to the user-specified locations [15]. Together, these layers provide a robust, secure, and user-friendly file encryption system.

2.1 Algorithm

The encryption and decryption tool uses the AES algorithm in CBC mode combined with PBKDF2 and HMAC-SHA256 for secure key derivation[6]. During encryption, the user selects a file and inputs a password through the GUI. A random 16-byte salt and initialization vector (IV) are generated. The password and salt are used to derive a 256-bit AES key using PBKDF2. The selected file's content is read and padded using PKCS7 to match AES block size.

BASIC STRUCTURE OF AES AlGORITHM :



The data is then encrypted using AES-CBC with the derived key and IV, and the resulting encrypted data, along with the salt and IV, is saved in a new file prefixed with "cipher_". For decryption, the user selects the encrypted file, inputs the same password, and specifies a name for the output file. The tool extracts the salt, IV, and encrypted content from the file, re-derives the AES key, and decrypts the data using AES-CBC. After removing the PKCS7 padding, the original content is restored and saved to the specified output file. This approach ensures both security and ease of use through password-based encryption and decryption.

2.2 Tools

The project utilizes a combination of tools and technologies to achieve high-security encryption through AES and Visual Cryptography. The core development is done in **Python**, chosen for its simplicity and extensive library support. **PyCryptodome** is used to implement AES encryption and decryption in CBC mode, along with secure key derivation using **PBKDF2** with HMAC-SHA256 [17]. **Hashlib** assists in hashing operations, and **base64** is used for encoding encrypted data. For the graphical user interface, **Tkinter** provides a user-friendly platform that allows users to select files, input passwords, and view process feedback. To integrate Visual Cryptography, libraries like **OpenCV**, **NumPy**, and **Pillow** (PIL) are employed for image processing, share generation, and decoding [9]. **Tqdm** is used to show real-time progress during encryption and decryption tasks, enhancing user experience. The tool is cross-platform, functioning on Windows, Linux, and macOS, and development is supported in IDEs like **VS Code**, **PyCharm**, or **Jupyter Notebook**. Together, these tools ensure the application is secure, efficient, and accessible to users across different environments.

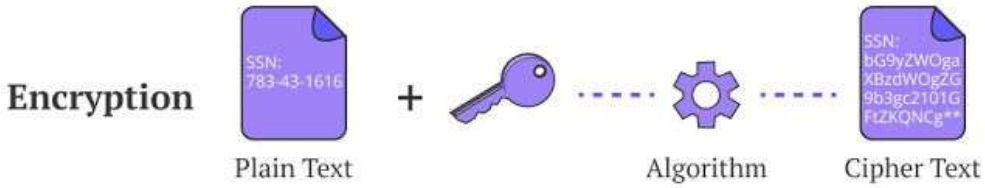
3. Methodology

The file encryption and decryption tool follows a structured methodology using Python and Tkinter. A simple GUI is created with options for encryption, decryption, and exiting the application. Users can select files and input passwords through the interface. For encryption, the tool reads the file, generates a random salt and IV, derives a key using PBKDF2, and encrypts the content using AES in CBC mode. The salt, IV, and encrypted data are saved together. During decryption, the tool extracts the salt and IV from the file, re-derives the key, decrypts the data, and saves it to an output file. Comprehensive error handling ensures that users receive clear messages for issues like missing files or permission errors.

Encryption:

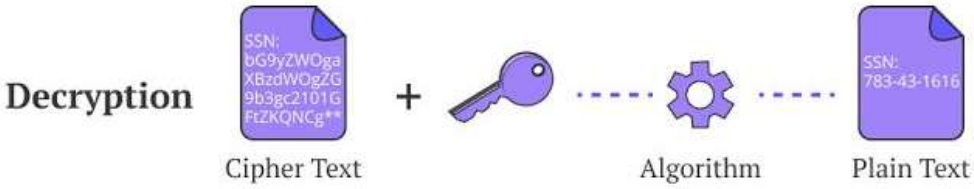
Encryption is the process of converting plain, readable data (plaintext) into an unreadable format (ciphertext) using an algorithm and a secret key. This ensures that only authorized parties with the correct decryption key can access the original information [12]. It is a fundamental technique used to protect sensitive data from unauthorized access during storage or transmission.

ISSN: 2583-6129
DOI: 10.55041/ISJEM04733



Decryption:

Decryption is the reverse process of encryption. It involves converting the encrypted data (ciphertext) back into its original readable form (plaintext) using the appropriate decryption key. Decryption restores access to the original information for authorized users, ensuring data confidentiality and security.



3.1 Output

Encryption:

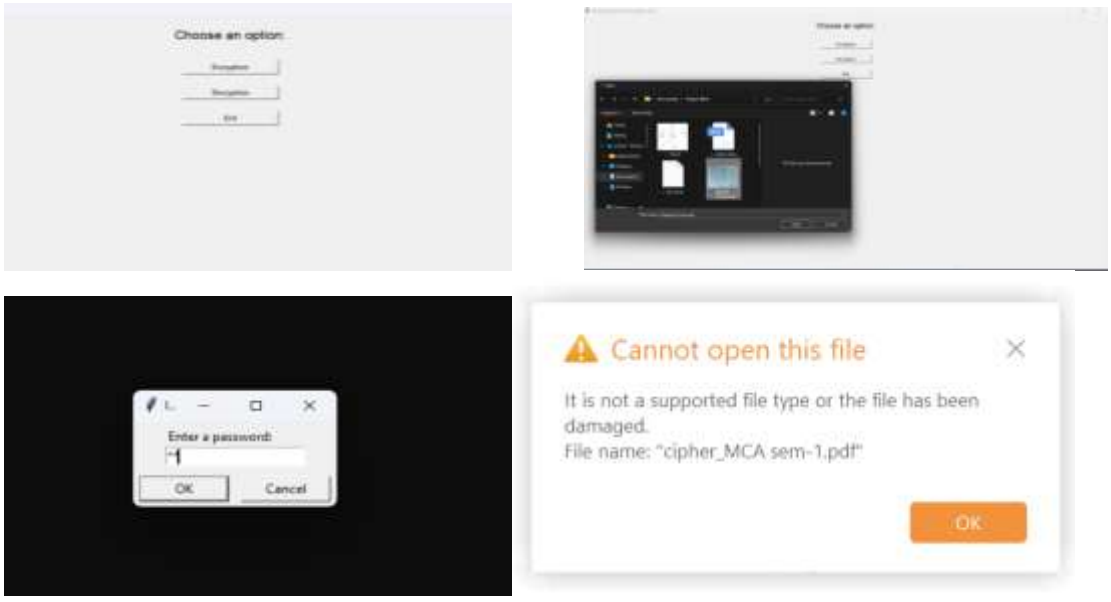


Fig : [1],[2],[3],[4] Encryption of text using AES

Decryption:

International Scientific Journal of Engineering and Management (ISJEM)
Volume: 04 Issue: 07 | July – 2025
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata
ISSN: 2583-6129
DOI: 10.55041/ISJEM04733

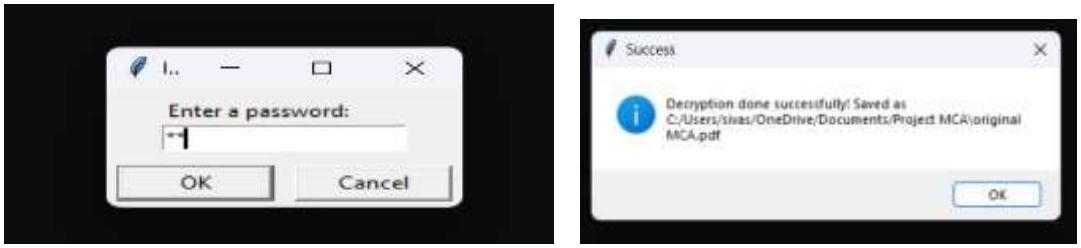
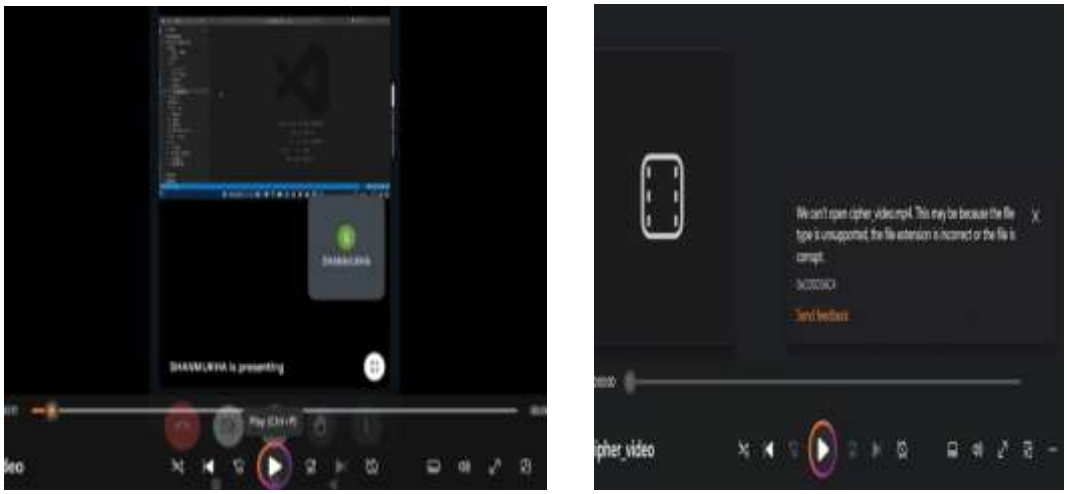


Fig : [5],[6],[7],[8] Decryption of text using AES

Before Encryption **After Encryption**

1. Image Encryption

2. Video Encryption



3. Audio Encryption

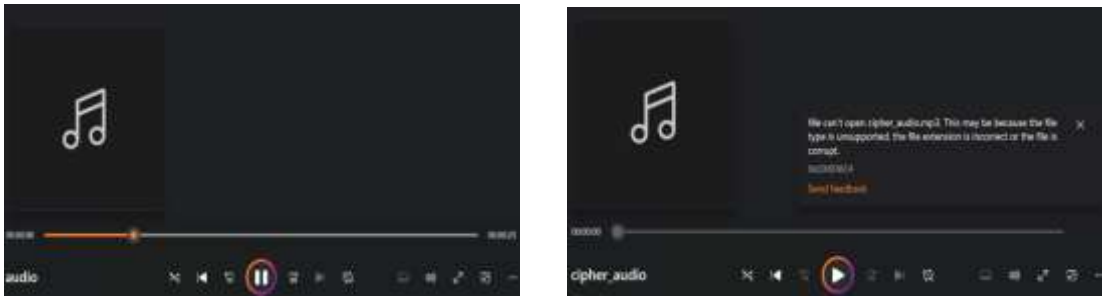


Fig : [1],[2],[3] Image,Video and Audio Encryption using Visual Cryptography

4. Conclusion

This project successfully delivers a secure and user-friendly file encryption and decryption tool using AES in CBC mode. It incorporates strong cryptographic practices such as PBKDF2-based key derivation, random salt and IV generation, and PKCS7 padding. The Tkinter-based GUI ensures ease of use for all users, while robust error handling enhances reliability.

5. Future Scope

The tool can be extended to support other algorithms like RSA or ChaCha20 and integrated with cloud services for secure file handling in online environments. Future improvements may include batch file processing, advanced key management, cross-platform enhancements, and integration with enterprise security systems. Adding user authentication and access control would further strengthen security.

6. ACKNOWLEDGEMENT

Kandhati Tulasi Krishna Kumar: Training & Placement Officer with 15 years' experience in training & placing the students into IT, ITES & Core profiles & trained more than 9,700 UG, PG candidates & trained more than 450 faculty through FDPs. Authored various books for the benefit of the diploma, pharmacy, engineering & pure science graduating students. He is a Certified Campus Recruitment Trainer from JNTUA, did his Master of Technology degree in CSE from VTA and in process of his Doctoral research. He is a professional in Pro-E, CNC certified by CITD He is recognized as an editorial member of IJIT (International Journal for Information Technology & member in IAAC, IEEE, MISTE, IAENG, ISOC, ISQEM, and SDIWC. He published 6 books, 55 articles in various international journals on Databases, Software Engineering, Human Resource Management and Campus Recruitment & Training.

Gandi Madhu Nisha is pursuing his final semester MCA in Sanketika Vidya Parishad Engineering College, accredited with A grade by NAAC, affiliated by AndhraUniversity and approved by AICTE. High security encryption by using AES and visual cryptography has taken up to his PG project and published the paper in connection to the project under the guidance of Kandhati Tulasi Krishna Kumar, Training & Placement Officer, SVPEC.

References

1. Daemen, J., & Rijmen, V. (1998).
The Design of Rijndael: AES – The Advanced Encryption Standard.
NIST AES finalist paper.
arxiv.org | nist.gov

2. Xuejia Lai & James L. Massey (1991).
A Proposal for a New Block Encryption Standard.
ieeexplore.ieee.org | researchgate.net

3. Zhang, X., & Parhi, K. K. (2002).
High-Speed VLSI Architectures for the AES Algorithm.
ieeexplore.ieee.org

4. Prasithsangaree, P., & Krishnamurthy, P. (2003).
On the Energy Efficiency of AES and RC4 in Wireless LANs.
ieeexplore.ieee.org

5. Rajalakshmi, P., et al. (2010).
Hardware-Software Co-Design of AES on FPGA for Low-Cost Embedded Systems.
sersc.org

6. Garg, A., et al. (2013).
Efficient FPGA Implementation of AES Algorithm.
ijera.com

7. Atha, P., et al. (2013).
AES Algorithm Implementation on FPGA for Secure Communication.
ijera.com

8. Komala Subhadra, M., et al. (2013).
FPGA Implementation of AES for High Security Applications.
sersc.org

9. M. S. Hossain, M. S. Hossain, & M. Rahman (2021).
Visual Cryptography for Secure Image Transmission.
sciencedirect.com

10. Naor, M., & Shamir, A. (1994).
Visual Cryptography.
researchgate.net

11. Khan, A., et al. (2020).
An Enhanced Visual Cryptography Scheme for Color Image Sharing.
springer.com

12. Alani, M. M. (2016).
AES Encryption Algorithm in Secure Data Communication.
researchgate.net

13. Prateeksha, D., et al. (2022).
Secure File Storage Using AES and Visual Cryptography.
ijert.org

14. Islam, M. R., et al. (2021).
Combining AES and Visual Cryptography for Enhanced Security in Data Sharing.
ijrte.org

15. PyCryptodome Documentation (2024).
Secure Python Cryptography Toolkit.
pycryptodome.readthedocs.io

16. Goyal, V., & Bhatnagar, P. (2020).
Secure Data Transmission Using AES and Visual Cryptography.
ijeat.org

17. Chanda, S., & Dasgupta, D. (2016).
A Novel Approach to File Security Using AES and Visual Cryptography.
ijcaonline.org

18. Zafar, S., & Khan, M. (2018).
A Hybrid Model for Image Encryption Using AES and Visual Cryptography.
sciencedirect.com

19. Tiwari, R., & Mishra, R. (2015).
Data Hiding Using Visual Cryptography with AES.
ijarcsse.com

20. National Institute of Standards and Technology (NIST).
FIPS 197 – Advanced Encryption Standard (AES).
csrc.nist.gov