Jurnal Media Teknik Elektro dan Komputer
Vol. 02, No.02 (July - December, 2025)
E-ISSN: 3063-2927

Incoming Mail Security Using AES-128 Encryption with QR Code Verification at the North Sumatra Provincial Inspectorate

Dodyk Fahlome[1], Dea Alya[2]

[1] Universitas Islam Negeri Sumatera Utara; dodyk0701232093@uinsu.ac.id
[2] Universitas Islam Negeri Sumatera Utara; alya0701232102@uinsu.ac.id

ABSTRACT

Incoming mail security is essential for maintaining information integrity within the North Sumatra Provincial Inspectorate. This study developed a mail metadata security system using the AES-128 algorithm as the encryption mechanism and QR Code as a verification medium. The encryption input consists of mail metadata representing document identity, including sender, receiver, document number, and message description, which are concatenated, converted into 128-bit blocks, and encrypted through ten rounds of AES-128 transformation. The resulting ciphertext is generated from the encrypted metadata output, transformed into a binary bitstream, and encoded into a QR Code to support secure distribution and authenticity validation. System performance stability was evaluated through Black Box testing of functional components, while encryption accuracy was validated through repeated encryption–decryption trials on multiple metadata samples. The results show that the system can generate valid ciphertext, accurately restore metadata, and provide a QR Code–based verification mechanism for government mail management.

Keywords: Data Security, Encryption, QR Code, AES-128, Documents

Corresponding Author:
Dodyk Fahlome
Universitas Islam Negeri Sumatera Utara; dodyk0701232093@uinsu.ac.id

1. INTRODUCTION

Data security is a crucial element in governance, especially in the process of managing incoming mail, which often contains sensitive information that should not be accessed or manipulated by unauthorized parties (Raflika et al., 2025). Threats such as data leaks, illegal access, and document modification have been increasingly reported in public sector information systems, with national cybersecurity reports detecting 207 data breach incidents in various Indonesian government sectors in 2023, leading to administrative losses and damage to institutional credibility at both national and regional levels (Setyadi et al., 2024). Manual systems that are still used in some administrative processes are unable to provide adequate mechanisms for detecting and preventing security breaches, and are therefore no longer in line with current digital security requirements (Wulandari, 2025). A technology-based approach is needed to ensure that the confidentiality, integrity, and authenticity of government data are maintained.

In an effort to strengthen document protection, various agencies have begun adopting encryption as a primary layer of security, combined with the use of QR codes as a means of quickly verifying encrypted data (Rahman & Erzed, 2024). Encryption ensures that only authorized parties can read the contents of the document, while QR Codes enable the validation process to be carried out efficiently without error-prone manual checks (Rahman & Erzed, 2024). The Advanced Encryption Standard (AES), based on studies reported in the literature (Akwukwuma et al., 2024), has been shown to be effective in securing text-based data, including digital documents and coded textual information, due to its strong cryptographic properties and stable performance. The integration of QR codes into the verification mechanism also increases the speed of inspection, data consistency, and document authentication accuracy (Kusumah et al., 2024).

This research aims to support the improvement of security and effectiveness of incoming mail management at the North Sumatra Provincial Inspectorate through the application of AES-128 encryption combined with QR codes as a verification medium. The developed system is expected to minimize the risk of data leakage, prevent document manipulation, and ensure that the metadata of received letters remains authentic and unchanged. In addition, the implementation of a QR Code-based verification mechanism is designed to provide greater transparency and accountability in the mail administration process. The main focus of this research is to design and implement a reliable and systematic AES-128–based encryption–decryption workflow integrated with a QR Code–based verification mechanism, aligned with the institution's operational requirements, to improve information security in the government environment.

2. LITERATURE REVIEW

2.1. Data Security

Data security in incoming mail management is a crucial aspect for public organizations to maintain the authenticity, integrity, and confidentiality of documents (Alda & Rifki, 2022). With the increasing threat of cybercrime, the implementation of digital systems that rely on encryption and QR Code-based verification has become an important strategy to prevent manipulation and ensure that access is only granted to authorized parties (Alam et al., 2025). These efforts must be aligned with national regulations such as Law No. 27 of 2022 concerning Personal Data Protection and international standards such as ISO/IEC 27001:2022, which emphasize data confidentiality, integrity, and availability (Nadine et al., 2025). In practical implementation, data security is applied through a combination of technical and administrative controls to protect digital information from unauthorized access and misuse (Rifki et al., 2023). The implementation generally includes access control mechanisms, authentication processes, encryption techniques, and system monitoring to ensure data security throughout its lifecycle, particularly in administrative information systems that manage official documents and sensitive institutional data (Gemawaty & Yuliani, 2024).

Encryption is one of the primary methods used in data security to protect digital information by transforming plaintext into ciphertext using cryptographic algorithms (Almadira et al., 2024). Based on key usage, encryption methods are commonly classified into symmetric and asymmetric encryption, where symmetric encryption uses a single secret key for both encryption and decryption processes, making it more efficient for securing large volumes of textual data in administrative systems (Maulana et al., 2025). AES is a symmetric encryption algorithm with key lengths of 128, 192, and 256 bits, widely used for data confidentiality and security in modern information systems (Mufti et al., 2025). AES is widely recognized for its strong level of security, efficient computational performance, and high

resistance to various cryptographic attacks, which makes it particularly suitable and reliable for protecting sensitive digital documents and confidential information in administrative and governmental environments (Rifki & Syamia, 2024).

2.2. Data Security Validation

Data security validation refers to a set of processes used to ensure that digital data and documents remain authentic, intact, and unaltered during storage and transmission (Almadira et al., 2024). In document-based information systems, validation mechanisms are essential to verify the originality of documents, confirm data integrity, and ensure that accessed information corresponds to the authorized source (Raflika et al., 2025). Several types of data security validation are commonly implemented in document security systems, including checksum verification, digital signatures, hash-based validation, and access-based authentication (Mahgafhira et al., 2023). These validation techniques are designed to detect unauthorized modifications and ensure that documents are accessed only by legitimate users, thereby reducing the risk of data manipulation and fraud in digital administrative environments (Alam et al., 2025).

One practical and widely adopted validation method in document security systems is the use of QR Codes as a verification medium (Kusumah et al., 2024). QR Codes enable users to quickly access validation metadata or encrypted references by scanning the code using standard digital devices, without requiring specialized software (Rahman & Erzed, 2024). This approach improves validation efficiency, minimizes human error, and ensures consistency between the original encrypted data and the information retrieved during the verification process (Halim & Pribadi, 2024).

3. METHODS

This research uses a qualitative approach with a descriptive method to describe the incoming mail security system developed at the North Sumatra Provincial Inspectorate. The development process was carried out by applying the AES-128 algorithm to maintain document confidentiality and integrity (Ridho & Romli, 2024). The research stages followed the Waterfall model, which included needs analysis, design, implementation, and structured system testing, where this model emphasized a sequential and systematic development flow from one stage to the next (Risdiansyah & Agustine, 2025). The entire process was visualized through the research flow as shown in Figure 1.

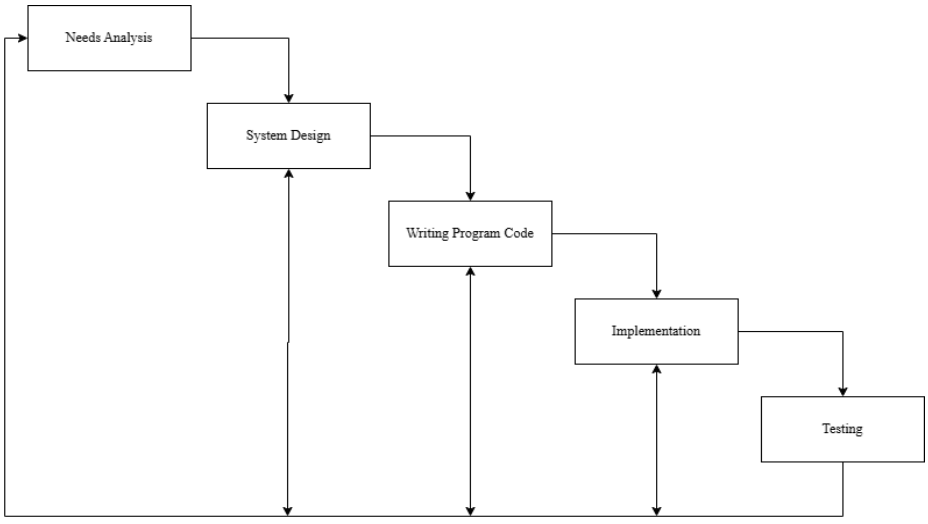


Figure 1. Waterfall Method

3.1. Data Collection

The initial stage of the research began with a needs analysis to identify the main problems in the incoming mail management system at the North Sumatra Provincial Inspectorate. The problems found included potential data leaks and document manipulation that could threaten the integrity of the system. To obtain a more comprehensive picture, the researchers collected data through:

1. Interviews: Speaking directly with relevant parties in the inspectorate, such as system administrators and administrative staff.
2. Observation: Directly observe existing problems and identify the need for a more secure system.
3. Documentation: Collecting and analyzing relevant documents, such as administrative records, letter contents, and operational procedures, to gain a clearer understanding of how the current system works (Wulandari, 2025).

3.2. Encryption Method

This stage explains the encryption method used to secure incoming mail metadata at the North Sumatra Provincial Inspectorate. The AES-128 algorithm converts plaintext into 128-bit blocks and processes them through ten rounds of cryptographic transformations (Baru, 2022). The resulting ciphertext is then encoded into a QR Code for practical, verifiable encrypted storage. The complete encryption process flow is shown in Figure 2 below.

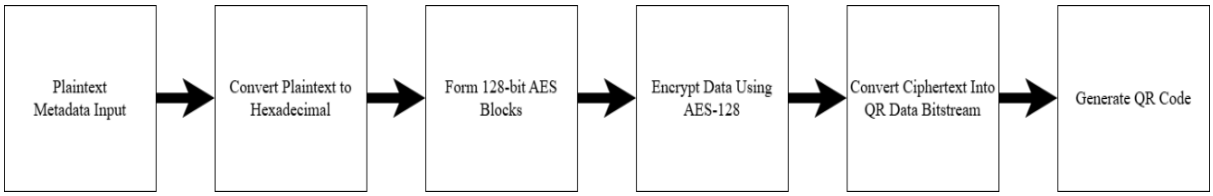


Figure 2. Algorithm Encryption Mechanism Diagram

The AES-128 encryption process consists of cryptographic transformations on 128-bit blocks, enhancing security through non-linear substitution, structural diffusion, and key-based operations. These stages sequentially convert plaintext metadata into unpredictable ciphertext.

1. SubBytes

   The SubBytes stage is a non-linear substitution process that provides high non-linearity to the state (Yusri et al., 2025). Each byte is replaced with a new byte based on the AES standard substitution table, namely S-Box. This transformation strengthens security because the relationship between input and output becomes non-linear. The SubBytes transformation can be represented by equation (1).

   $$State_{i,j} = SBox[State_{i,j}] \quad (1)$$

   Description:

   $State$: AES state matrix containing the input data block.

   $State\ \{i, j\}$: Byte value located at row i and column j of the state matrix, where $i, j \in \{0, 1, 2, 3\}$.

   $SBox[\ ]$: Substitution function defined by the AES standard, which maps each input

byte to a corresponding output byte using a fixed 16×16 substitution table. The output of the function replaces the original byte at position (i, j) in the state matrix.

2. ShiftRows

ShiftRows provides horizontal diffusion to the state matrix by cyclically shifting the second, third, and fourth rows, spreading byte values to different positions while maintaining the column structure (Manurung et al., 2025). This expands the influence of changes in the plaintext across the entire state, as shown in equation (2).

$$State_{r,c} = State_{r,(c+r)\ mod\ 4} \quad (2)$$

Description:

$State$: AES state matrix containing the data block.

$State\ \{r, c\}$: Byte located at row r and column c of the state matrix, where r and $c \in \{0, 1, 2, 3\}$.

The shift operation is performed by shifting the bytes of the second, third, and fourth rows of the matrix.

The number of positions each row is shifted corresponds to its row index r, where the second row is shifted by 1, the third by 2, and the fourth by 3. The $mod\ 4$ operation ensures that the indices are within the bounds of the matrix (0 to 3), making the shift cyclic.

3. MixColumns

MixColumns provides vertical diffusion by mixing each column using $GF(2^8)$ arithmetic operations (Nino, 2023). This process creates a linear combination of bytes in the column, thereby strengthening the distribution of data between columns. This stage is applied in rounds 1–9, but not in the final round. The MixColumns operation is represented by the irreducible AES polynomial shown in equation (3).

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix}$$

$$m(x) = x^8 + x^4 + x^3 + x + 1 \quad (3)$$

Description:

S_0, S_1, S_2, S_3 represent the bytes of the state matrix in the current column.

The matrix multiplication is performed in $GF(2^8)$ using modular arithmetic, where each byte in the column is multiplied by the corresponding value in the matrix, followed by the XOR operation.

The irreducible AES polynomial $m(x) = x^8 + x^4 + x^3 + x + 1$ is used to ensure that the arithmetic is carried out in $GF(2^8)$, which is a finite field with 256 elements.

4. AddRoundKey

AddRoundKey is the stage where the state is combined with the round key (Indraka & Romli, 2025). The XOR operation is applied byte-by-byte so that any change in the key results in a significant change in the ciphertext. This stage is the only process that directly links the data to the key, making it very important in the overall security of AES-128. This stage can be formulated through equation (4).

$$State = State \oplus RoundKey_i \quad (4)$$

Description:
$State$ refers to the current state matrix of the AES algorithm, which contains the data being encrypted or decrypted.
$RoundKey_{\text{i}}$ is the round key derived from the original key using the AES key expansion algorithm. It is used to modify the state matrix in each round.
The XOR operation (denoted as $\oplus$) is applied byte-by-byte to combine the state with the round key. This ensures that even a small change in the round key results in a significant alteration in the output ciphertext.

After obtaining the ciphertext, it is converted into a format suitable for QR Code encoding, allowing it to be represented as a scannable visual pattern for decryption. Each stage is described below.

1. Ciphertext-to-Binary Conversion
   The ciphertext generated in hexadecimal values is converted into an 8-bit binary representation (Gunawan & Rahmi, 2025). This conversion is necessary because QR Codes operate at the bit level, so each byte of data must be presented in binary form. The process of converting bytes to binary and arranging the bitstream is shown in equations (5) and (6).

$$C_i^{bin} = Binary\ (C_i) \quad (5)$$

$$Bitstream = C_0^{bin} \parallel C_1^{bin} \parallel \ldots \parallel C_{31}^{bin} \quad (6)$$

   Description:
   C_{i} represents the ciphertext byte at position i in the sequence, where $i \in \{0, 1, 2, \ldots, 31\}$, assuming the ciphertext is divided into 32 bytes.
   ${C_{\text{i}}}^{bin}$ denotes the binary representation of the ciphertext byte C_{i}. Each byte is converted into an 8-bit binary value.
   $Binary\ (C_{\text{i}})$ is the function that converts the byte C_{i} from its hexadecimal representation into an 8-bit binary format.
   The Bitstream is the concatenation of all 32 binary values from the ciphertext, forming a continuous bit sequence that is ready for QR Code encoding. The operator $\parallel$ denotes the concatenation of binary values.

2. Header Construction (Mode & Character Count)
   QR Codes require a header to inform the scanner about the type of data and the number of characters. This system uses Byte Mode, and the number of ciphertext bytes is calculated to be inserted into the header (Halim & Pribadi, 2024). The mode and data length are represented by equation (7).

$$Header = Mode \parallel Count \quad (7)$$

Description:
$Mode$ refers to the data encoding mode used in the QR Code, which, in this case, is Byte $Mode$. Byte $Mode$ is used to encode binary data, such as the ciphertext.
$Count$ represents the number of ciphertext bytes, which is calculated based on the length of the data that will be encoded into the QR Code. For instance, if the ciphertext is 256 bytes, the $Count$ will be 256 (in binary format).
The concatenation operator ‖ is used to combine the $Mode$ and $Count$ into a single header that is then inserted at the beginning of the QR Code data stream.

3. Bitstream Finalization
Before data is placed into the QR module, the bitstream must be refined to meet the standard QR Code structure (Rizal et al., 2022). This refinement includes terminators, byte alignment, and the addition of pad codewords. The final bitstream is determined by equation (8).

$$FinalStream = Header \parallel Bitstream \parallel Terminator \parallel Padding \quad (8)$$

Description:
$Header$ is the data header that contains information about the encoding mode and the length of the ciphertext, as previously discussed.
$Bitstream$ refers to the continuous stream of binary data obtained from the ciphertext after it has been converted into binary and concatenated.
$Terminator$ refers to a sequence of bits added at the end of the bitstream to signal the end of the data, ensuring that the QR Code reader can correctly interpret the data length.
$Padding$ involves adding extra codewords to make the bitstream align with the QR Code size requirements. This padding ensures that the final bitstream fits the QR Code's size constraints, and it is added according to the specific QR Code format (using predefined padding codewords).

3.3. UML System Design

This cryptographic application system is modeled using UML to describe functional requirements and interactions between users, applications, and QR Code modules. Since the research focuses on encryption, decryption, and QR Code generation, UML modeling uses use case and sequence diagrams to explain system boundaries, actor roles, and message flow without unnecessary complexity.

Use case diagrams map the system's functional requirements from the user's perspective (Ramdany et al., 2024). The diagram shows that the main actor (user) interacts with three core functions—Encryption, Decryption, and Close—and one additional use case, Share, included in the encryption process. This model illustrates the system's simplicity, where users start a session, select encryption or decryption, and close the application, as shown in Figure 3.

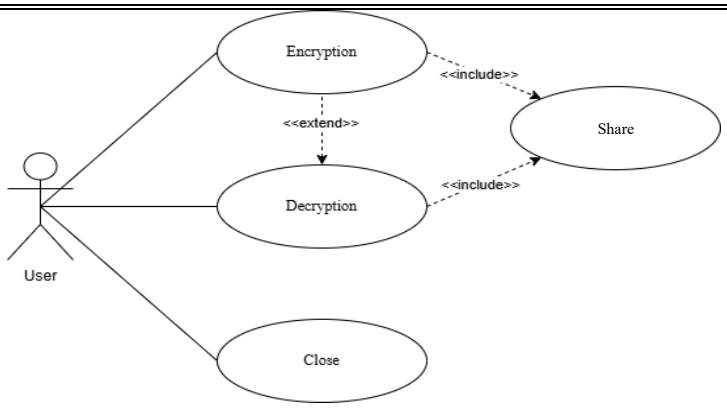


Figure 3. Use Case Diagram Design

Sequence diagrams describe the flow of messages and processes during encryption or decryption (Ramdany et al., 2024). The diagram shows how actors interact with the Menu, Cryptography App, and QR Module, starting from metadata input and AES key, through AES-128 encryption, QR generation from ciphertext, to decryption back to the original plaintext. This confirms the interrelationship between components and ensures consistency in operational steps, as shown in Figure 4.

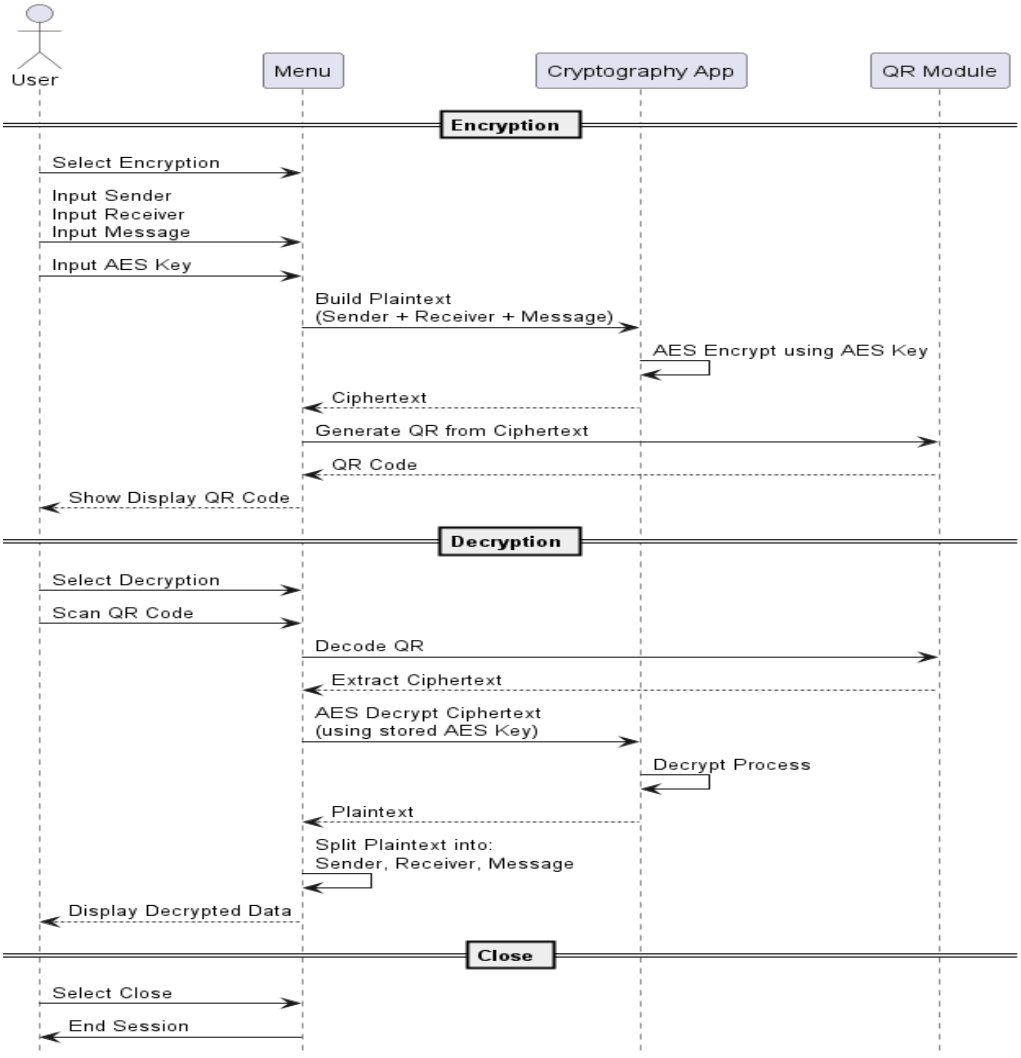


Figure 4. Sequence Diagram Design

4. RESULT AND DISCUSSION

This research produced a series of processes for system development, starting with the Encryption Process that converts letter metadata into ciphertext using AES-128. The next stage, QR Code Data Encoding, compiles the ciphertext into a bitstream ready for QR Code encoding as an encrypted storage medium. The UML System Design outlines the functional architecture, followed by Application Implementation, which applies methods and workflows into the system. All functions are validated through Black Box Testing to ensure each process supports the security objectives for incoming mail.

4.1. *AES-128 ncryption Process*

The first stage in the AES encryption process begins with transforming the plaintext and key from character form into binary data representation. Each character is first mapped to an ASCII value as the basis for determining bytes, then represented in hexadecimal code, which is the operational format for AES (Febriyadi, 2023). The hex representation is essential to model data from text as bytes, which are then arranged into 128-bit blocks for the next encryption stage. This conversion process is shown in Tables 1, 2, 3, and 4, illustrating how plaintext and keys are converted into hexadecimal values.

Table 1. ASCII-to-Hexadecimal Mapping for Sender Plaintext

Plaintext	I	n	s	p	e	k	t	u	r
Hexadecimal	49	6E	73	70	65	6B	74	75	72

Table 2. ASCII-to-Hexadecimal Mapping for Receiver Plaintext

Plaintext	S	e	k	r	e	t	a	r	i	s
Hexadecimal	53	65	6B	72	65	74	61	72	69	73

Table 3. ASCII-to-Hexadecimal Mapping for Message Plaintext

Plaintext	T	i	n	d	a	k		L	a	n	j	u	t
Hexadecimal	54	69	6E	64	61	6B	20	4C	61	6E	6A	75	74

Table 4. ASCII-to-Hexadecimal Mapping for AES-128 Encryption Key

Key	I	n	s	p	e	k	t	o	r	a	t
Hexadecimal	49	6E	73	70	65	6B	74	6F	72	61	74

After converting all plaintext characters into hexadecimal bytes, the AES block assembly process begins by combining the bytes and dividing them into 128-bit blocks (16 bytes per block). Each block is then processed by the AES algorithm. If the plaintext exceeds 16 bytes, multiple blocks are formed. If the final block has fewer than 16 bytes, padding is applied according to AES rules (Fahrizal, 2023).

This stage arranges each 128-bit plaintext block into a 4×4 state matrix, the core structure of AES (Baru, 2022). The 16 bytes from each block are placed into four columns with four rows. This matrix serves as the input for subsequent AES transformations, including SubBytes, ShiftRows, MixColumns, and AddRoundKey. (Baru, 2022). Since the plaintext produces two blocks, each is converted into a separate state matrix and processed individually, as shown in Table 5.

Table 5. Formation of AES-128 Plaintext State Matrix for Block 1 and Block 2

Block 1				Block 2			
49	65	72	72	72	69	6B	6E
6E	6B	53	65	69	6E	20	6A
73	74	65	74	73	64	4C	75
70	75	6B	61	54	61	61	74

The key matrix formation in AES-128 begins by arranging the key bytes into a 4×4 matrix using a column-by-column pattern. AES requires a 128-bit (16-byte) key, but the key for this study ("Inspektorat") only produces 11 bytes after conversion. To meet the AES input length, 00 byte padding is added, which does not alter the key's meaning but ensures it meets the required length for processing through the key schedule mechanism (Febriyadi, 2023). The complete key structure after padding is shown in Table 6.

Table 6. Formation of AES-128 Key Matrix After Padding to 128-bit

Key			
49	65	72	74
6E	6B	61	00
73	74	74	00
70	6F	00	00

The AddRoundKey stage is the first step in AES encryption, where each byte in the state matrix is XORed with a byte in the key matrix (Indraka & Romli, 2025). Since each plaintext block is processed independently, both Block 1 and Block 2 undergo the same process, linking each part of the message to the encryption key. The XOR operation is performed position-by-position on the 4×4 state and key matrices, embedding the key's influence before the block moves to the next stages, such as SubBytes and ShiftRows (Indraka & Romli, 2025). The AddRoundKey results for Round 0 are shown in Table 7.

Table 7. AddRoundKey Operation (Round 0) Output for Block 1 and Block 2

	AddRoundKey (Block 1)					**AddRoundKey (Block 2)**			
	00	00	00	06		3B	0C	19	1A
Round	00	00	32	65	**Round**	07	05	41	6A
0	00	00	11	74	**0**	00	10	38	75
	00	1A	6B	61		24	0E	61	74

The SubBytes stage is the first non-linear transformation in AES, where each byte in the state matrix is replaced using the S-Box (Yusri et al., 2025). The S-Box is designed to resist cryptographic attacks, especially linear and differential cryptanalysis, by ensuring no linear relationship between inputs and outputs (Yusri et al., 2025). The substitution is performed independently on each byte, so both Block 1 and Block 2 undergo changes based solely on the S-Box structure. The results of the SubBytes transformation in the first round for both blocks are shown in Table 8.

Table 8. SubBytes Transformation Output (Round 1) for Block 1 and Block 2

	SubBytes (Block 1)					**SubBytes (Block 2)**			
	63	63	63	6F		2D	6B	D4	C6
Round	63	63	A5	4E	**Round**	82	7C	87	54
1	63	63	82	5A	**1**	63	CA	F5	5A
	63	C6	6F	4B		32	3A	4B	5A

ShiftRows is a diffusion transformation in AES that shifts the bytes in each row of the state matrix (Manurung et al., 2025). The first row remains unchanged, the second row shifts one position left, the third row shifts two, and the fourth row shifts three positions left, with bytes wrapping around cyclically. This transformation enhances data distribution between columns, ensuring that the byte values modified by SubBytes are no longer localized (Manurung et al., 2025). The ShiftRows process is applied independently to Block 1 and Block 2, and the results for Round 1 are shown in Table 9.

Table 9. ShiftRows Transformation Output (Round 1) for Block 1 and Block 2

	ShiftRows (Block 1)					ShiftRows (Block 2)			
	63	63	63	6F		2D	6B	D4	C6
Round 1	63	A5	4E	63	**Round 1**	7C	87	54	82
	82	5A	63	63		F5	5A	63	CA
	4B	63	C6	6F		5A	32	3A	4B

MixColumns is a diffusion transformation in AES applied to each column of the state matrix (Nino, 2023). In this step, four bytes in each column are combined through matrix multiplication with a fixed matrix defined over GF(2^8) (Nino, 2023). This operation ensures that a change in one byte affects all bytes within the same column, thereby increasing data spread and significantly enhancing overall encryption security. The MixColumns process is applied independently to Block 1 and Block 2 using the standard AES constant matrix, and the resulting state matrices after the completion of Round 1 are shown in Table 10.

Table 10. MixColumns Transformation Output (Round 1) for Block 1 and Block 2

	MixColumns (Block 1)					MixColumns (Block 2)			
	F4	6A	1D	41		6C	36	B2	5D
Round 1	E5	8A	39	3C	**Round 1**	5F	8D	8A	7A
	76	8D	56	63		92	5D	F1	58
	01	0B	60	2C		0A	41	6A	1C

In the early AES encryption rounds, the master key is expanded to generate round keys through the Key Expansion process (Indraka & Romli, 2025). For AES-128, the first four words come from the original key, and the following words are derived using the RotWord, SubWord, Rcon addition, and XOR with the previous word. This ensures that each round key has a unique non-linear structure while remaining dependent on the master key. After Round Key 1 is formed, it is used in the AddRoundKey stage to XOR the state resulting from MixColumns with Round Key 1, linking changes in the state directly to the key and enhancing diffusion and security (Indraka & Romli, 2025). The AddRoundKey results for Round 1 in Block 1 and Block 2 are shown in Table 11.

Table 11. AddRoundKey Round 1 Output After Key Expansion (Block 1 & Block 2)

	AddRoundKey (Block 1)					AddRoundKey (Block 2)			
	25	1C	EC	05		DC	9C	41	68
Round 1	DA	8D	F9	5B	**Round 1**	A2	0F	D9	A3
	25	BE	05	74		3A	62	E6	EF

	92	AE	A8	42		7D	17	37	0C

In the AES-128 encryption structure, the AddRoundKey operation combines the state matrix with the round key through XOR (Indraka & Romli, 2025). This ensures that each state change is directly influenced by the encryption key, linking the data to the key throughout the process. AddRoundKey Round 0 uses the master key before the non-linear stages, while Round 1 closes the first round after SubBytes, ShiftRows, and MixColumns. For rounds 2 to 9, the AddRoundKey operation remains the same, though not explicitly shown. Round 10 is the final stage, as MixColumns is no longer used, and XOR with Round Key 10 produces the final ciphertext (Indraka & Romli, 2025). The AddRoundKey results for Round 0, 1, and 10 are summarized in Table 12.

Table 12. Summary of AddRoundKey Results for All Rounds

	AddRoundKey (Block 1)					**AddRoundKey (Block 2)**			
	00	00	00	06		3B	0C	19	1A
Round	00	00	32	65	**Round**	07	05	41	6A
0	00	00	11	74	**0**	00	10	38	75
	00	1A	6B	61		24	0E	61	74
	25	1C	EC	05		DC	9C	41	68
Round	DA	8D	F9	5B	**Round**	A2	0F	D9	A3
1	25	BE	05	74	**1**	3A	62	E6	EF
	92	AE	A8	42		7D	17	37	0C
	…	…	…	…		…	…	…	…
Round	…	…	…	…	**Round**	…	…	…	…
…	…	…	…	…	…	…	…	…	…
	…	…	…	…		…	…	…	…
	C1	A3	9F	72		FE	80	33	44
Round	16	4D	B8	20	**Round**	29	AD	50	7F
10	7A	91	6E	03	**10**	9B	C1	A2	18
	55	C8	14	99		6A	DE	39	01

Based on AES-128 encryption calculations using ten cycles on the combined metadata ("Inspektur", "Sekretaris", and "Tindak Lanjut"), the original text (32 bytes) was fully transformed into encrypted text. AES-128 processes the data in two 128-bit blocks, each undergoing SubBytes, ShiftRows, MixColumns (except in the last round), and AddRoundKey from Round 0 to Round 10 (Indraka & Romli, 2025). The first block contains the first 16 bytes, and the second block contains the remaining 16 bytes. Both blocks are processed independently using the same expanded key schedule. The encrypted output for each block is shown in Table 13.

Table 13. Final AES-128 Encryption Output

Categories	**Hexadecimal Representation**															
Plaintext	49	6E	73	70	65	6B	74	75	72	53	65	6B	72	65	74	61
	72	69	73	54	69	6E	64	61	6B	20	4C	61	6E	6A	75	74
Key	49	6E	73	70	65	6B	74	6F	72	61	74	00	00	00	00	00

Ciphertext	36	1A	F3	F8	51	60	62	8D	7E	4B	1E	3A	B2	00	62	F6
	67	F9	5D	B0	58	1C	81	16	57	EB	80	69	DA	47	9B	6B

4.2. *QR Code Data Encoding Process*

Two blocks of ciphertext resulting from AES-128 encryption are broken down into pairs of 8-bit hexadecimal-binary values. The decimal representation shows the byte values to be processed as numerical data, while the 8-bit binary form describes the bit arrangement that can later be mapped to the module pattern in the QR Code (Gunawan & Rahmi, 2025). With this table, each byte of ciphertext can be clearly tracked before proceeding to the manual design of the data structure and QR pattern, as shown in Table 14.

Table 14. Ciphertext Byte Mapping: Hexadecimal, Decimal, and 8-bit Binary Representation

Hexadecimal	Decimal	8-bit Binary
36	54	00110110
1A	26	00011010
F3	243	11110011
F8	248	11111000
51	81	01010001
60	96	01100000
62	98	01100010
8D	141	10001101
7E	126	01111110
4B	75	01001011
1E	30	00011110
3A	58	00111010
B2	178	10110010
00	0	00000000
62	98	01100010
F6	246	11110110
67	103	01100111
F9	249	11111001
5D	93	01011101
B0	176	10110000
58	88	01011000
1C	28	00011100
81	129	10000001
16	22	00010110
57	87	01010111
EB	235	11101011
80	128	10000000
69	105	01101001
DA	218	11011010
47	71	01000111
9B	155	10011011
6B	107	01101011

In the QR generation stage, the ciphertext (originally ASCII plaintext converted to hexadecimal) is treated as a series of data bytes. Each byte is converted to an 8-bit binary representation, and all bits are sequentially combined into one long bitstream. This bitstream is then mapped to QR modules

following the QR Code encoding structure (Halim & Pribadi, 2024). The bitstream arrangement for the two ciphertext blocks is shown in Table 15.

Table 15. Sequential Bitstream Construction from Ciphertext Bytes

8-bit Binary	bitstream
00110110	000-007
00011010	008-015
11110011	016-023
11111000	024-031
01010001	032-039
01100000	040-047
01100010	048-055
10001101	056-063
01111110	064-071
01001011	072-079
00011110	080-087
00111010	088-095
10110010	096-103
00000000	104-111
01100010	112-119
11110110	120-127
01100111	128-135
11111001	136-143
01011101	144-151
10110000	152-159
01011000	160-167
00011100	168-175
10000001	176-183
00010110	184-191
01010111	192-199
11101011	200-207
10000000	208-215
01101001	216-223
11011010	224-231
01000111	232-239
10011011	240-247
01101011	248-255

After arranging the ciphertext into a bitstream, the QR Code requires an initial structure with a mode indicator and character count to help the scanning device recognize the data type and byte count (Rizal et al., 2022). The mode indicator for Byte Mode is 0100, and the character count for 32 bytes of ciphertext is 00100000 (Rizal et al., 2022). These components form the "head" of the QR before the binary data is inserted, as shown in Table 16.

Table 16. QR Header Structure

Component	Value	Bit
Mode Indicator	Byte Mode	0100
Character Count	32 bytes	00100000

Combining the QR header with all the ciphertext bits from Table 15 produces the QR data bitstream, which is a continuous sequence of bits that forms the core of the information to be encoded in the QR Code symbol. At this stage, the QR Code has not yet added error correction or masking mechanisms—the important thing is to form a valid and well-structured data payload. The initial structure of the QR data bitstream is shown in Table 17 for reference.

Table 17. Initial QR Data Bitstream Structure

Component	Bit
Mode Indicator	0100
Character Count	00100000
Ciphertext Bitstream	Bit 0–255

The QR Code requires the bitstream to be completed within the codeword boundary by adding a terminator, adjusting byte alignment, and adding pad codewords (EC and 11 alternately) (Rizal et al., 2022). These steps do not alter the ciphertext but are necessary for the data to form a valid QR Code that can be processed by the error correction module. The bitstream adjustments are summarized in Table 18.

Table 18. Finalization of QR Bitstream

Component	Value / Bit
Terminator	0000
Byte Alignment	0 / 00 / 000 as needed
Pad Codewords	EC → 11101100 → 11 → 00010001 → EC → 11 ...

The previous description outlines the steps to convert AES-128 ciphertext into a data structure compatible with QR Code encoding. These steps focus on converting ciphertext to a standardized byte representation and forming a bitstream that meets QR format requirements. This approach demonstrates the core process while staying within the scope of the research.

4.3. *Application Implementation*

The application system has been successfully implemented through trials that ensure stable performance and alignment with development objectives. Featuring a simple, user-friendly interface, the application allows easy management of letter metadata requiring protection and confidentiality. The Home Display presents three main options: “Encrypt,” “Decrypt,” and “Close Application,” enabling users to encrypt metadata for secure distribution or decrypt it to retrieve the contents from the QR Code, as shown in Figure 5.

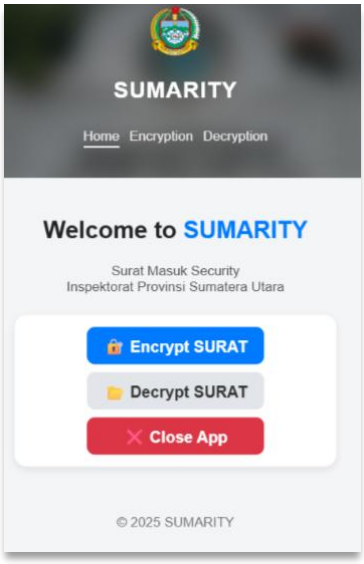
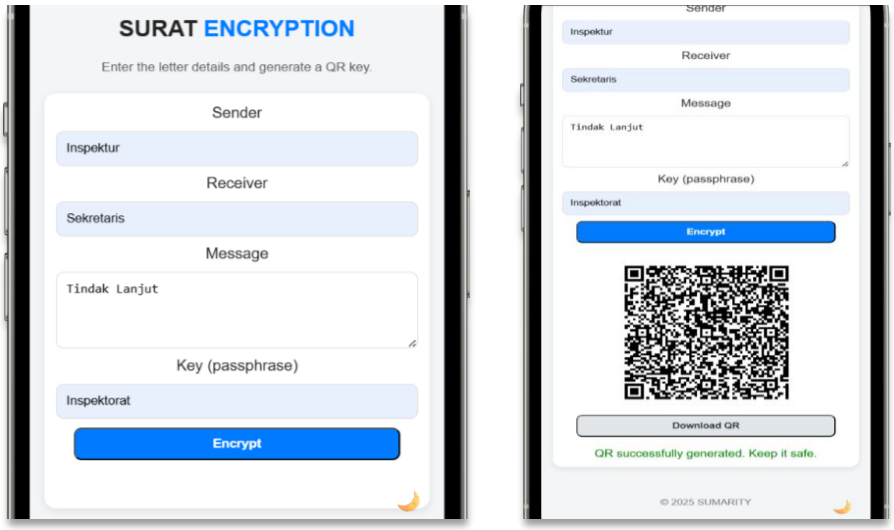


Figure 5. Home Display

On the Encryption Display, users enter the letter metadata (Sender, Receiver, and Message) and the AES-128 secret key. In section (a), users fill in the metadata via the provided form, and in section (b), they click the "Encrypt" button to begin. The system converts the metadata to ASCII and hexadecimal, organizes it into 128-bit blocks, and processes it using AES-128 to generate ciphertext. This ciphertext is then converted into a QR Code and displayed on the screen, ready for secure distribution or download, as shown in Figure 6. This process ensures metadata protection before sending or storage.

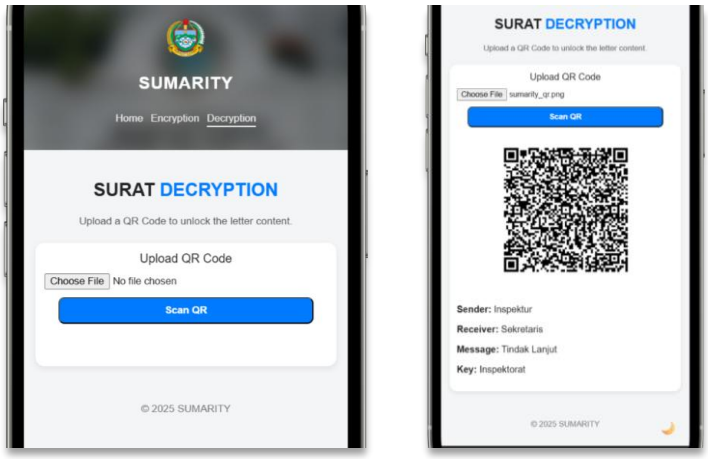


(a) **(b)**

Figure 6. Encryption Display

On the Decryption Display, users recover encrypted metadata by scanning or selecting a QR Code containing the ciphertext. In section (a), users upload or scan the QR Code, and the system automatically extracts the binary data. Using the AES-128 key entered by the user, the system decrypts the data and displays the original metadata—Sender, Receiver, and Message—in section (b), without

showing the ciphertext. This ensures users see only relevant data, while the technical process remains hidden, maintaining interface simplicity and information security, as shown in Figure 7.



(a) **(b)**

Figure 7. Decryption Display

4.4. *Black Box Testing*

This system testing was conducted using the Black Box Testing strategy, which focuses on testing the functionality of the application without regard to its internal implementation (Mardiati & Saputra, 2025). As part of the testing strategy, unit, integration, operational, and system testing stages were implemented to evaluate all aspects of the application comprehensively. Unit testing was performed to verify each application component individually, including encryption and decryption functions, as well as interactions with the user interface (UI), as shown in Table 19.

Table 19. Testing the Incoming Mail Security System with the Black Box Method

Testing	Testing Procedure	Input	Output	Description
Encryption Process	- Open the Application - Select the Encryption Menu - Input Data Fields - Input AES Key (128-bit) - System Encrypts Data - System Generates QR Code	Sender: Inspektur Receiver: Sekretaris Message: "Tindak Lanjut" Key: "Inspektorat"		Success
Decryption Process	- Open the Application - Select the Decryption Menu - Scan QR Code - System Decrypts Using Stored AES Key - System Displays Recovered Data		Sender: Inspektur Receiver: Sekretaris Message: "Tindak Lanjut"	Success

Based on Table 19, the encryption and decryption processes function as expected. The application encrypts metadata (sender, receiver, message) with the AES-128 key ("Inspektorat") and generates a QR Code containing the ciphertext. The QR Code can be downloaded or attached for storage. In decryption, the application reads the QR Code, reconstructs the ciphertext, and decrypts it using the same key to retrieve the original metadata. Both processes are consistent and accurate, ensuring data confidentiality and integrity.

5. CONCLUSION

AES-128 encryption and QR Code encoding are two good ways to protect the metadata of incoming mail. AES-128 is used by this system to encrypt metadata. It first breaks the metadata up into 128-bit blocks, and then it turns them into a QR Code bitstream. The whole process, from encrypting the data to making the QR Code, follows well-known rules for cryptography and encoding. This keeps people who shouldn't be able to see or change metadata from doing so. The system's design, which is based on UML, makes it simple to use and works well. This makes it easy to set up and check on workflows.

The tests show that the system's main functions—encryption, QR code generation, decryption, and metadata restoration—operate correctly and consistently without errors or data inconsistencies. The system demonstrates stable performance and preserves data integrity throughout the process. In addition, its user-friendly interface makes it easy to operate by administrative staff. Overall, the system works effectively and reliably, making it a suitable solution for securing incoming mail and sufficiently robust for use within government information security frameworks.

REFERENCES

Akwukwuma, V. V. N., Chete, F. O., Oshioluamhe, M. N., & Okpako, A. E. (2024). Text Encryption Using Advanced Encryption Standard (AES) Algorithm. *NIPES-Journal of Science and Technology Research*, *6*(2). https://doi.org/10.5281/zenodo.12558923

Alam, R. G. G., Hidayah, A. K., Gunawan, G., Wijaya, A., & Abdullah, D. (2025). *Manajemen Risiko Keamanan Informasi*. PT. Sonpedia Publishing Indonesia.

Alda, M., & Rifki, M. I. (2022). Implementasi Metode Triple Des pada Aplikasi Keamanan Pesan Berbasis Mobile. *Journal of Information Technology and Computer Science*, *7*(1), 17–26.

Almadira, A., Pratama, Y., & Purwani, F. (2024). MELINDUNGI DATA DI DUNIA DIGITAL: PERAN STATEGIS ENKRIPSI DALAM KEAMANAN DATA. *Journal of Scientech Research and Development*, *6*(2), 540–549. https://doi.org/10.56670/jsrd.v6i2.608

Baru, J. B. R. I. R. D. K. (2022). Analisa Metode Kriptografi Modern Advance Encryption Standar (AES) 128 Bit dalam Mengenkripsi dan Mendekripsi File Dokumen Digital. *Jurnal Ilmiah KOMPUTASI*, *21*(3).

Fahrizal, D. (2023). Implementasi Kriptografi Aes 256 Bit Pada Aplikasi Pesan Di Android Dengan Raspberry Pi Server Berbasis Open Source. *TECHSI-Jurnal Teknik Informatika*, *14*(2), 107–123.

Febriyadi, F. (2023). Implementasi AES ECB dan Hashing MD5/SHA-256 Pada Aplikasi Penyuratan Android. *PENERAPAN ALGORITMA K-MEANS CLUSTERING UNTUK MENGETAHUI POLA PENERIMA BEASISWA BANK INDONESIA (BI) DI PROVINSI RIAU*, *5*(1), 113–126.

Gemawaty, C. A., & Yuliani, Y. (2024). MANAJEMEN IDENTITAS DAN AKSES DALAM KEAMANAN SISTEM INFORMASI (PENDEKATAN LITERATURE REVIEW). *Jurnal Manajamen Informatika Jayakarta*, *4*(4), 396–403. https://doi.org/10.52362/jmijayakarta.v4i4.1527

Gunawan, R., & Rahmi, E. (2025). Implementasi Algoritma Blowfish Untuk Pengamanan Data Transaksi Dalam Aplikasi Berbasis Website E-commerce. *Techno. Com*, *24*(2).

Halim, M., & Pribadi, O. (2024). Penerapan Qrcode Sebagai Tanda Tangan Digital Dalam Penerbitan Surat Keluar Pada Stmik Methodist Binjai. *Jurnal TIMES*, *13*(2), 260–268.

Indraka, A. P., & Romli, M. A. (2025). Keamanan Arsip Kelurahan Bumijo Menggunakan Metode Advanced Encryption Standard (AES 128) Berbasis Web: Security of Bumijo Village Archives Using Advanced Encryption Standard (AES-128) Method Based on Web. *MALCOM: Indonesian Journal of Machine Learning and Computer Science*, *5*(1), 232–241.

Kusumah, I. M. Y., Yusmanyah, E. F., Siregar, M. F., & Fatmawati, P. (2024). Implementasi QR Untuk Legalitas Tanda Tangan Digital di Lingkungan STMIK Bandung. *Jurnal Penelitian Dan Pengembangan Teknologi Informasi Dan Komunikasi*, *13*(1), 1–7.

Mahgafhira, I., Wahid, A., & Parenreng, J. M. (2023). Pengembangan Sistem Otentikasi Dokumen Digital Jurusan Teknik Informatika Dan Komputer Fakultas Teknik UNM Berbasis Digital Signature. *Progressive Information, Security, Computer, and Embedded System*, *1*(2), 115–125. https://doi.org/10.61255/pisces.v1i2.151

Manurung, R. A., Sutarman, S., & Efendi, S. (2025). Comparative Analysis of the Performance of Four Symmetric Algorithms on Digital File Security. *JOURNAL OF INFORMATICS AND TELECOMMUNICATION ENGINEERING*, *8*(2), 152–164.

Mardiati, D., & Saputra, Y. (2025). Implementasi sistem informasi manajemen klinik menggunakan metode black box testing. *Jurnal Informatika Dan Teknik Elektro Terapan*, *13*(1).

Maulana, H., Tahir, M., Farrohah, N., Maulana, F., & Aprilianyani, R. R. (2025). PERANCANGAN SISTEM KEAMANAN FILE MENGGUNAKAN HYBRID ENCRYPTION UNTUK PERLINDUNGAN DATA. *Jurnal RESTIKOM: Riset Teknik Informatika Dan Komputer*, *7*(1), 87–96. https://doi.org/10.52005/restikom.v7i1.420

Mufti, M., Wiharto, Y., & Subandi, S. (2025). Penerapan Algoritma Kriptografi Advanced Encryption Standard 128 Pada Laporan Transaksi Di Kopi Tuku. *Journal Software, Hardware and Information Technology*, *5*(1), 48–60. https://doi.org/10.24252/shift.v5i1.147

Nadine, Z. S., Aknuranda, I., & Farisi, H. (2025). Evaluasi Pengelolaan Keamanan Informasi Berbasis Indeks KAMI 5.0 di Satuan Logistik (SLOG) POLRI. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *9*(10).

Nino, B. E. (2023). Perbandingan performa algoritma aes dan twofish menggunakan metode strict avalanche criterion pada nomor induk kependudukan indonesia. *Jurnal Teknologi Informasi*, *9*(1), 19–29.

Raflika, L., Husna, K., Fahrezi, M. A., Andini, S., Annisa, S. R., & Darmansah, T. (2025). Optimalisasi Keamanan Informasi melalui Sistem Pengelolaan Surat yang Aman di Lingkungan Kerja. *RISOMA: Jurnal Riset Sosial Humaniora Dan Pendidikan*, *3*(4), 254–266.

Rahman, A. P., & Erzed, N. (2024). SISTEM PENGESAHAN DOKUMEN DENGAN QR CODE DAN MEMANFAATKAN JSON WEB TOKEN (JWT) UNTUK MENGURANGI PENYIMPANAN. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *8*(6), 12143–12150.

Ramdany, S. W., Kaidar, S. A., Aguchino, B., Amelia, C., Putri, A., & Anggie, R. (2024). Penerapan UML Class Diagram dalam Perancangan Sistem Informasi Perpustakaan Berbasis Web. *Journal of Industrial and Engineering System*, *5*(1), 30–41.

Ridho, A., & Romli, M. A. (2024). SISTEM PENGAMANAN DOKUMEN MENGGUNAKAN ALGORITMA KRIPTOGRAFI ADVANCED ENCRYPTION STANDARD (AES-256). *Jurnal Informatika Teknologi Dan Sains (JINTEKS)*, *6*(4), 1044–1052.

Rifki, M. I., Raditya, M. E., & Hasugian, A. H. (2023). *Text Data Security Application Using a Mobile-Based Base64 Algorithm. Instal: Jurnal Komputer, 15 (02), 224–235*.

Rifki, M. I., & Syamia, N. (2024). Message Security Application Using Mobile-Based AES Algorithm. *Journal of Computer Science, Information Technology and Telecommunication Engineering*, *5*(2), 595–606.

Risdiansyah, D., & Agustine, Lady. (2025). Pengembangan Sistem Informasi Pemesanan Makanan (SIMAKAN) Berbasis Web menggunakan Metode Waterfall. *Reputasi: Jurnal Rekayasa Perangkat Lunak*, *6*(1), 27–36.

Rizal, A., Azhari, A., Susanto, A., Sugiyamto, S., & Wahyudi, E. N. (2022). *ENKRIPSI TEKS DENGAN PENDEKATAN BITSTREAM*.

Setyadi, H. J., Masa, A. P. A., Widagdo, P. P., Irsyad, A., Aulia, S., Nugroho, M. H., & Ananda, N. T. (2024). Edukasi Keamanan Cyber Untuk Melindungi Masyarakat Dari Ancaman Digital. *Pengabdian Kepada Masyarakat Bidang Teknologi Dan Sistem Informasi (PETISI)*, *2*(2), 40–47. https://doi.org/10.30872/petisi.v2i2.2280

Wulandari, R. (2025). *Interview Regarding Incoming Mail Management Procedures*. North Sumatra Provincial Inspectorate.

Yusri, F., Muttaqin, A., Syamil, A. H. A., Luqman, M. N., Citra, C., Mursyid, M., Sidiq, M. F., & Siregar, F. A. (2025). Implementasi Algoritma Advanced Encryption Standard (AES) Secara Manual Menggunakan Python. *JIKUM: Jurnal Ilmu Komputer*, *1*(1), 12–16.