Security Based on Matrix of Keys in Header and Transpose for Physical Token Authentication Device

Binu C T[*], Rubini P[**]
*https://orcid.org/0009-0001-5042-9177
SOET, CMR University

DOI: 10.29322/IJSRP.16.02.2026.p17049
https://dx.doi.org/10.29322/IJSRP.16.02.2026.p17049

Paper Received Date: 15th January 2026
Paper Acceptance Date: 19th February 2026
Paper Publication Date: 24th February 2026

***Abstract*-**The cipher is created by transpose and interchange of rows and columns based on keys. Key is generated based on transpose and interchange of matrix and square matrix also with empty value. header contain the size of matrix. “This authentication technique uses matrix multiplication and transpose operations combined with a unique-password to generate a verification token. The method improves confidentiality by avoiding direct password storage. Header is validated in both sides and is applicable to all the security services.

***Index Terms*-** cryptography, algorithm, physical authentication device

I. INTRODUCTION

This article guides security services for physical tokenizer. Security services are mechanisms designed to protect information and systems from unauthorized access, misuse, modification, and disruption. In modern computer networks and information systems, security services ensure that data remains protected throughout its lifecycle—during storage, processing, and transmission.
The fundamental security services include Confidentiality, Authentication, Integrity, and Availability. Together, these services form the foundation of secure communication and system reliability.

1. Confidentiality
Confidentiality ensures that information is accessible only to authorized users and is protected from unauthorized disclosure.It prevents sensitive data from being read by attackers.Common techniques include encryption, access control, and data masking.Example: Encrypting emails so that only the intended recipient can read them.

2. Authentication
Authentication verifies the identity of a user, system, or entity before granting access to resources.It answers the question: *“Who are you?”*Authentication methods include passwords, biometric systems, digital certificates, and multi-factor authentication (MFA).Example: Logging into a system using a username and password.

3. Integrity
Integrity ensures that data remains accurate, complete, and unaltered during storage or transmission.It detects unauthorized modification of data.Common techniques include hash functions, checksums, and digital signatures.Example: Verifying that a downloaded file has not been tampered with.

4. Availability
Availability ensures that information and services are accessible to authorized users whenever needed.It protects against system failures and attacks such as Denial of Service (DoS). Techniques include redundancy, backups, load balancing, and fault tolerance. Example: Ensuring a website remains online even during high traffic.

The token authentication device is not only for authentication and it can be used for confidentiality,integrity and availability

All the data passed through the tokenizer to improve security. This device help to provide high security. All the security algorithms are loaded in the device.

II BACKGROUND AND RELATED WORK

1. Key Management & Authentication — Core to Security Services

1.1 Authentication and Secure Key Management

- "A Comprehensive Survey on Authentication and Secure Key Management in Internet of Things: Challenges, Countermeasures, and Future Directions" — Patruni & Deebak (2023)
  • Systematic review of state-of-the-art techniques for authentication and key management in IoT environments. Discusses key agreement schemes, vulnerabilities, and future directions relevant for CIA security services.

1.2 Key Management & Authentication in Smart Grids

- "A Survey on Key Management and Authentication in Smart Grid Systems" — Abdalzaher et al. (2023)
  • Reviews the main key-management techniques and authentication mechanisms used in smart grid (SG) cyber-physical infrastructures, with security analysis and research challenges.

1.3 Satellite Communication Authentication

- "Comprehensive Survey towards Security Authentication Methods for Satellite Communication Systems" — Meng et al. (2025)
  • Categorizes authentication methods (cryptography, Blockchain, AKA protocols, etc.) and evaluates them on security, cost, applicability, and performance — essential for maintaining confidentiality, integrity, and availability.

2. Recent Security Protocol & Key Exchange Advances

2.1 Authentication & Security in 5G Networks

- Patel (2025): "Advances in Authentication and Security Protocols for 5G"
  • Explores authentication and security mechanisms in 5G systems — including how authentication protocols contribute to confidentiality, integrity, and availability in next-gen networks.

2.2 Formal Security Verification

- Reaz & Wunder (2024): "Formal Verification of Permission Voucher"
  • Formal evaluation of a security protocol's confidentiality, integrity, and authentication using symbolic verification methods — showing how key protocols can be modeled and proven secure.

3. Cryptographic Key Exchange & Matrix-Like Methods

3.1 Quantum-Resistant Key Exchange

- Staat et al. (2024): "Key Exchange in the Quantum Era"
  • Evaluates hybrid public key and physical-layer security for secure key exchange — with implications for confidentiality and integrity in future security frameworks.

3.2 Post-Quantum Key Management Trends

- Quantum Cryptography Surveys (e.g., Shapna Akter 2023)
  • Reviews key distribution and post-quantum mechanisms that support confidentiality and integrity beyond classical cryptosystems.

4. Supporting Standards and Protocol Frameworks

4.1 Key Management Interoperability Protocol (KMIP)

- Standard protocol defining secure key manipulation and storage — foundational for building key-based security services across platforms.

4.2 Double Ratchet & Dynamic Secrets

- Double Ratchet Algorithm: ongoing key renewal mechanism enhancing confidentiality and forward secrecy in secure messaging protocols — relevant for modern key-based service models.

5. Cross-Cutting Themes Across Literature
Security Characteristics & CIA Matrix

- MDPI systematic review finds that most security pattern research addresses CIA characteristics, and also increasingly authentication and related controls (authorization, non-repudiation), indicating a need for matrix-style analysis frameworks.

Cryptographic Key Management General Survey
A 2023 ScienceDirect comprehensive survey on cryptographic key management systems proposes a taxonomy of key stages (generation, exchange, storage) — useful when building a matrix tying key management to confidentiality/authentication/integrity/availability requirements.

III PROPOSED SYSTEM WITH RANDOM KEY SIZE

The tokenizer uses security mechanisms like confidentiality, authentication, integrity and availability. Here matrix of keys is used to map and decrypt the cipher to plain text. The header of the tokenizer contains the size of the matrix of keys and it must be a square matrix and empty spaces in the matrix of plain text help to identify text and numbers. The key size of matrix is random and it defines whether its 2X2 or 3X3 .The empty spaces fill with 0 to indicate the complexity of the matrix of text and number.

Matrix Transpose–Based Cipher Algorithm
Overview

- Plaintext → numeric matrix
- Apply **key matrix multiplication**
- Apply **matrix transpose**
- Convert back to ciphertext

1. Definitions
Alphabet Mapping
A = 0, B = 1, ..., Z = 25
Inputs

- Plaintext message
- Key matrix **K** of size n × n
- Plaintext length must be a multiple of n (pad if needed)

2. Encryption Algorithm
Step 1: Plaintext to Matrix
Split plaintext into blocks of size n^2.
Example (n = 2):
Plaintext: **HELP**
Mapping:
H=7, E=4, L=11, P=15
Plaintext matrix **P**:

$$P = \begin{bmatrix} 7 & 4 \\ 11 & 15 \end{bmatrix}$$

Step 2: Key Matrix Selection
Choose an invertible key matrix **K**.
Example:

$$K = \begin{bmatrix} 3 & 5 \\ 2 & 7 \end{bmatrix}$$

Step 3: Matrix Encryption
Multiply key matrix with plaintext matrix:

$$E = (K \times P) mod 26$$
$$E = \begin{bmatrix} 3 & 5 \\ 2 & 7 \end{bmatrix} \begin{bmatrix} 7 & 4 \\ 11 & 15 \end{bmatrix} = \begin{bmatrix} 76 & 87 \\ 91 & 113 \end{bmatrix} mod 26 = \begin{bmatrix} 24 & 9 \\ 13 & 9 \end{bmatrix}$$

Step 4: Matrix Transpose

$$C = E^T = \begin{bmatrix} 24 & 13 \\ 9 & 9 \end{bmatrix}$$

Step 5: Matrix to Ciphertext'
Read row-wise:
24 = Y, 13 = N, 9 = J, 9 = J
Ciphertext:
YNJJ

3. Decryption Algorithm (Brief)

1. Convert ciphertext into matrix **C**
2. Transpose: $E = C^T$
3. Multiply by inverse key matrix:

$$P = (K^{-1} \times E) mod 26$$

4. Convert numbers back to letters

4. Pseudocode
INPUT plaintext, key matrix K
PAD plaintext to fit matrix size
CONVERT plaintext to matrix P

E = (K × P) mod 26
C = transpose(E)

CONVERT C to ciphertext
OUTPUT ciphertext
Authentication
Authentication Using Transpose Matrix and Uni-Password
Goal
To **authenticate a user** by verifying whether the submitted password, when transformed using a **key matrix and transpose operation**, matches the stored authentication value.

1. Preliminaries
Alphabet Mapping
A = 0, B = 1, C = 2, …, Z = 25
Shared Secret

- **Uni-password (UP)**: known only to user and server
- **Key matrix K (n × n)**: secret authentication key

2. Registration Phase
Step 1: Convert Uni-Password to Matrix
Example:
Uni-Password: PASS
P = [15 0
18 18]

Step 2: Key Matrix

$$K = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}$$

Step 3: Authentication Matrix Generation

$$A = (K \times P) mod 26$$
$$A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 15 & 0 \\ 18 & 18 \end{bmatrix} = \begin{bmatrix} 84 & 54 \\ 87 & 72 \end{bmatrix} mod 26 = \begin{bmatrix} 6 & 2 \\ 9 & 20 \end{bmatrix}$$

Step 4: Transpose Operation

$$T = A^T = \begin{bmatrix} 6 & 9 \\ 2 & 20 \end{bmatrix}$$

Step 5: Store Authentication Token

- Store **T** (not the password)
- Store **key matrix ID**

3. Login / Authentication Phase
Step 1: User Inputs Uni-Password
User enters:
PASS

Step 2: Convert Password to Matrix P'

Step 3: Compute Authentication Matrix

$$A' = (K \times P') mod 26$$

Step 4: Transpose

$$T' = (A')^T$$

Step 5: Verification
If:

$$T' = T$$

→ **Authentication Successful**
Else → **Authentication Failed**

4. Pseudocode
REGISTRATION:
Input UP
Convert UP to matrix P
A = (K × P) mod 26
T = transpose(A)
Store T

AUTHENTICATION:
Input UP'
Convert UP' to matrix P'
A' = (K × P') mod 26
T' = transpose(A')
If T' == T
Authenticate user
Else
Reject

INTEGRITY

Integrity of Security Using C_{ij}Matrix with Transpose
Objective
To ensure **data integrity** by detecting any modification in transmitted or stored data using **matrix transformation and transpose comparison**.

1. Notations

- Plain data block → D
- Character–number mapping:
- A = 0, B = 1, ..., Z = 25
- Data matrix → M_{ij}
- Integrity matrix → C_{ij}
- Transpose matrix → C_{ji}
- Modulus → 26

2. Integrity Generation Phase (Sender)
Step 1: Data to Matrix Conversion
Example data:
DATA
Numeric conversion:
D = 3, A = 0, T = 19, A = 0
Data matrix M_{ij}:

$$M = \begin{bmatrix} 3 & 0 \\ 19 & 0 \end{bmatrix}$$

Step 2: Generate Integrity Matrix C_{ij}

$$C_{ij} = (M_{ij} \times M_{ij}^{T}) mod 26$$
$$M^{T} = \begin{bmatrix} 3 & 19 \\ 0 & 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 9 & 57 \\ 57 & 361 \end{bmatrix} mod 26 = \begin{bmatrix} 9 & 5 \\ 5 & 23 \end{bmatrix}$$

Step 3: Transpose for Integrity Tag

$$T = C_{ij}^{T} = \begin{bmatrix} 9 & 5 \\ 5 & 23 \end{bmatrix}$$

(Here symmetric, transpose shown for formal integrity step.)

Step 4: Transmit

- Send **data** D

- Send **integrity tag** T

3. Integrity Verification Phase (Receiver)
Step 1: Receive Data and Tag

- Received data → D'
- Received integrity tag → T'

Step 2: Recompute Integrity Matrix

$$M' \rightarrow C'_{ij} = (M' \times M'^{T}) mod 26$$
$$T'' = (C'_{ij})^{T}$$

Step 3: Integrity Check
If:

$$T'' = T'$$

→ **Integrity Verified**
Else → **Integrity Violation Detected**

4. Pseudocode

```
SENDER:
  Convert data to matrix M
  C = (M × M^T) mod 26
  T = transpose(C)
  Send (Data, T)

RECEIVER:
  Convert received data to matrix M'
  C' = (M' × M'^T) mod 26
  T" = transpose(C')
  If T" == T
    Integrity OK
  Else
    Integrity Failed
```

5. Why This Ensures Integrity

- Any bit/character change alters M
- Changes propagate through C_{ij}
- Transpose ensures structural consistency check
- Acts like a **matrix-based checksum**

"The proposed integrity mechanism generates a C_{ij}matrix using matrix multiplication and transpose operations. Any modification in the original data results in a mismatch during verification, thus ensuring integrity."

AVAILABILITY

To ensure **data availability** by storing data in **redundant matrix forms** using **key matrices and transpose**, so data can be recovered even if part of it is lost or corrupted.Size of the matrix is varied based on random number.

Notations

- Data block → D

- Data matrix → M_{ij}
- Key matrix → K_{ij}
- Transposed backup matrix → M_{ji}
- Modulus → 26

2. Availability Generation Phase (Storage)

Step 1: Convert Data to Matrix

Example data:

INFO

Numeric conversion:

I = 8, N = 13, F = 5, O = 14

Matrix:

$$M = \begin{bmatrix} 8 & 13 \\ 5 & 14 \end{bmatrix}$$

Step 2: Generate Key-Based Storage Matrix

$$S = (K \times M) mod 26$$

Example key matrix:

$$K = \begin{bmatrix} 1 & 2 \\ 3 & 5 \end{bmatrix}$$
$$S = \begin{bmatrix} 18 & 41 \\ 39 & 109 \end{bmatrix} mod 26 = \begin{bmatrix} 18 & 15 \\ 13 & 5 \end{bmatrix}$$

Step 3: Transpose-Based Redundant Backup

$$B = S^T = \begin{bmatrix} 18 & 13 \\ 15 & 5 \end{bmatrix}$$

Step 4: Distributed Storage

- Store **S** in primary storage
- Store **B** in secondary / backup storage
- Store **key matrix K** securely

AVAILABILITY RECOVERY PHASE

Case 1: Primary Storage Failure

Recover original matrix:

$$S = B^T$$

Case 2: Data Matrix Recovery

$$M = (K^{-1} \times S) mod 26$$

Convert Mback to plaintext.

Iv CONCLUSION

This security services are suitable and best for physical token authentication device. It provides 99.9 percent security by using tokenizer. The security algorithms are installed in the tokenizer. The size of matrix for keys and plain text is stored in the header and it may varies based on random number algorithm. So it's the best security algorithm using tokenizer.

REFERENCES

[1]Patruni & Deebak (2023). *Authentication and Secure Key Management in IoT* — Ad Hoc Networks.

[2]Abdalzaher et al. (2023). *Key Management and Authentication in Smart Grids*.

[3]Meng et al. (2025). *SatCom Security Authentication Survey*.

[4] Patel (2025). *Authentication & Security Protocols for 5G*.

[5].Reaz & Wunder (2024). *Formal Verification of Permission Voucher*.

[6]Staat et al. (2024). *Key Exchange in the Quantum Era*.

[7]Akter (2023). *Quantum Cryptography Survey*.

[8] Comprehensive Key Management Systems Survey (2023).

[9].MDPI Security Pattern Review (CIA focus).

AUTHORS

Binu C T, ME, Research scholar CMR University binuct143@gmail.com
Rubini P, PhD, Professor CMR University rubini.p@cmr.edu.in