International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Dynamic Key Camouflage Encryption: A Self-Contained Encryption Structure for Portable Secure Data Storage

Amit Rai

Abstract

Traditional cryptographic systems rely on external key manage- ment mechanisms where encryption keys are stored, exchanged, or managed separately from encrypted data. This creates operational complexity and increases the risk of key leakage through memory dumps, configuration files, or network interception.

This paper proposes a dynamic key camouflage encryption model where encryption keys are generated at runtime and embedded within the encrypted data structure in an obfuscated form. The encrypted structure contains ciphertext and a hidden representation of the en- cryption key that is bound to the ciphertext using cryptographic hash- ing and structural scrambling.

The final encrypted object becomes a portable self-contained con- tainer that can be stored in memory, disk, or transmitted across net- works without requiring separate key storage. The internal layout of the encrypted object is further protected using a family of reversible jumbling functions that obscure the structural boundaries between ci- phertext and key material.

1 Introduction

Secure storage and transmission of sensitive information remain fundamental challenges in modern computing systems. Traditional encryption models depend on external key storage mechanisms such as key servers, hardware security modules (HSM), or password-based key derivation.

In practice, many real-world security incidents occur not because the cryptographic algorithm is weak, but because encryption keys are exposed through operational weaknesses such as:

- Memory disclosure attacks
- Configuration file leaks
- Insecure key distribution
- Key mismanagement

This work explores an alternative design paradigm where encryption keys are generated dynamically and embedded within the encrypted structure in a camouflaged form.

The objective is to produce a portable encrypted object that:

- eliminates external key storage
- supports secure storage in memory or disk
- allows portable encrypted containers
- prevents direct key extraction through structural obfuscation

2 System Overview

The proposed system creates a self-contained encrypted structure called a *Jumbled Object*. This

structure contains both encrypted data and a con- cealed representation of the encryption key.
The encryption pipeline consists of the following stages:

1. Runtime Key Generation
2. AES Encryption
3. Ciphertext Hashing
4. Key Camouflage
5. Structural Composition
6. Multi-Stage Jumbling

3 Encryption Process

3.1 Runtime Key Generation

A random symmetric encryption key is generated at runtime.
$K = Random(256\ bits)$ Each encryption operation generates a unique key.

3.2 AES Encryption

The plaintext is encrypted using AES.
$C = AES(K, P)$
Where:

- P = plaintext
- K = encryption key
- C = ciphertext

3.3 Ciphertext Hashing

A cryptographic hash of the ciphertext is computed.
$H = SHA256(C)$

3.4 Key Camouflage

The key is camouflaged by XOR binding with the ciphertext hash.
$K_h = K \oplus H$
Where:

- K_h = hidden key

3.5 Structure Composition

The base data structure is constructed as:
$D = C \mid K_h$
Where $\mid$ denotes concatenation.

4 Jumbling Function Framework

To obscure the internal layout of the encrypted structure, the system applies a sequence of reversible jumbling functions.
Let:
$F = \{F_1, F_2, \ldots, F_n\}$
be a predefined family of reversible functions. Each function satisfies:
$F^{-1}(F_i(x)) = x$

4.1 Example Jumbling Functions

4.1.1 Function F_1 : Byte Reversal

$F_1(x) = Reverse(x)$

Example:
$F_1(123456) = 654321$

4.1.2 Function F_2 : Block Swap

Split input into two halves and swap them.
$x = AB$
$F_2(x) = BA$
Example:
$F_2(123456) = 456123$

4.1.3 Function F_3 : Circular Rotation

Rotate bytes left by r positions.
$F_3(x) = RotateLeft(x, r)$
Example:
$F_3(123456) = 345612$

4.1.4 Function F_4 : Even-Odd Permutation

Reorder bytes by grouping even indices followed by odd indices.
Example:
$F_4(123456) = 135246$

4.1.5 Function F_5 : XOR Masking

Apply reversible XOR masking using index-based mask.
$F_5(x_i) = x_i \oplus m_i$
where m_i is derived from index.

5 Multi-Stage Jumbling

The system randomly selects a sequence of jumbling functions.
$J = F_{ik}(\ldots F_{i2}(F_{i1}(D)))$
Where:

- D = original data structure
- J = final jumbled object

Each function index is prefixed to the output so that the system can reverse the transformations during decryption.
Example:
$D \rightarrow F_3(D) \rightarrow F_6(F_3(D)) \rightarrow F_2(F_6(F_3(D)))$
The stored encrypted blob becomes:
$Blob = J$

6 Decryption Process

Decryption reverses the encryption pipeline.

6.1 Unjumbling

Functions are applied in reverse order.
$D = F^{-1}(F^{-1}(\ldots F^{-1}(Blob)))$
$i_1 \quad i_2 \qquad i_k$

6.2 Structure Separation

The data structure is split into ciphertext and hidden key.

$C, K_h = Split(D)$

6.3 Hash Reconstruction

$H = SHA256(C)$

6.4 Key Recovery

$K = K_h \oplus H$

6.5 AES Decryption

$P = AES^{-1}(K, C)$

7 Security Considerations

The proposed design introduces multiple protective layers.

- Runtime key generation
- Cryptographic key binding to ciphertext
- Multi-stage structural obfuscation
- Portable encrypted object

The internal structure of the encrypted object becomes difficult to inter- pret without knowledge of the algorithm and function sequence.

8 Advantages

- Eliminates external key storage
- Enables portable encrypted containers
- Reduces operational key management complexity
- Supports secure memory storage
- Protects against simple memory disclosure

9 Limitations

Despite its advantages, the system must be implemented carefully.

- Security partially depends on algorithm secrecy
- Structural obfuscation must avoid predictable patterns
- Formal cryptographic analysis is required

10 Future Work

Future research directions include:

- Hardware-bound encryption using TPM
- Integration with confidential computing
- Feistel-based structural scrambling
- entropy analysis of jumbling functions
- formal security proofs

11 Conclusion

This paper presented a dynamic key camouflage encryption architecture designed to simplify encrypted data storage by embedding a concealed rep- resentation of the encryption key within the encrypted structure.

By combining runtime key generation, cryptographic binding, and multi- stage structural jumbling,

the system creates a portable self-contained en- crypted object suitable for secure storage in memory, disk, or network trans- mission.