A Formal Tripartite Framework for Symmetric Ciphers: Shared Symbol Set, Symbol–Space Mapping, and Permutation Function

Okereke Chukwudi Donald
Department of Computer Science
University of Windsor
Windsor, ON, Canada

***Abstract*—Classical treatments of simple symmetric ciphers often leave foundational agreements implicit, obscuring both their structure and security dependencies. We introduce a unified, *tripartite framework* in which any symmetric cipher is defined by three components: (i) a *shared symbol set* Σ, (ii) a bijective *symbol-to-space mapping* $g\colon \Sigma \to \mathcal{S}$, and (iii) a secret *permutation/transformation* $f\colon \mathcal{S} \to \mathcal{S}$. We demonstrate this framework on shift (Caesar) and monoalphabetic substitution ciphers, embedding precise mathematical definitions and PGFPlots figures for g and f. We then analyze how legitimate key agreement and an attacker's knowledge (or ignorance) of each component affect cryptanalysis. Finally, we discuss broader implications for cipher design and pedagogy.**

I. Introduction

Early symmetric ciphers hid crucial structural assumptions—alphabet conventions, numeric encodings, and key-dependent transformations—beneath procedural descriptions. Shannon's foundational work characterized secrecy systems as transformations on a message space, implicitly invoking a symbol set, a numerical mapping, and a permutation [1]. Modern block ciphers continue this pattern: plaintext symbols are encoded into mathematical domains (e.g., $\mathbb{F}_{2^8}$), then permuted via key-dependent operations [2]. We formalize these agreements explicitly as the triad $(\Sigma,\, g,\, f)$, providing a clear lens for analysis and teaching.

The remainder of this paper is organized as follows. Section II defines the formal framework. Section III applies it to classical shift and substitution ciphers, including step-by-step examples and two PGFPlots figures. Section IV examines how key agreement and attacker knowledge of Σ, g, and f shape security. Section V discusses broader design and pedagogical implications.

II. Formal Framework

Let Σ be a finite *symbol set* (e.g., letters, bytes) agreed upon by sender and receiver [4]. This corresponds to component (i) from the abstract. This set dictates the valid characters for messages. Define a bijection (component (ii)):

$$g : \Sigma \longrightarrow \mathcal{S},$$

mapping each symbol $\sigma \in \Sigma$ to a unique element $s \in \mathcal{S}$, where $\mathcal{S}$ is a suitable mathematical domain (e.g., indices $\{0, \ldots, N-1\}$ or vectors in $\mathbb{Z}_m^n$) [5]. The inverse mapping $g^{-1} : \mathcal{S} \to \Sigma$ must also exist. Finally, let (component (iii)):

$$f : \mathcal{S} \longrightarrow \mathcal{S}$$

be a secret, bijective transformation (permutation) whose specific form depends on shared secret key data k [7]. We often write this as f_k to emphasize key dependence. Since f_k is bijective, its inverse f_k^{-1} exists.

Encryption of a message symbol $m \in \Sigma$ to a ciphertext symbol $c \in \Sigma$, and the corresponding decryption, compose these functions as:

$$\begin{aligned} \text{Encrypt:} \quad & c = g^{-1}\big(f_k\big(g(m)\big)\big), \\ \text{Decrypt:} \quad & m = g^{-1}\big(f_k^{-1}\big(g(c)\big)\big). \end{aligned}$$

This abstraction captures the core structure of many symmetric ciphers, from classical examples to elements within modern designs [2]. The framework inherently processes messages symbol by symbol, based on the domain Σ of the mapping g.

III. Illustrative Examples

We demonstrate the framework using two classical ciphers.

A. Shift (Caesar) Cipher

(i) Symbol Set: The standard English lowercase alphabet. This set Σ defines the characters that can be encrypted or decrypted.

$$\Sigma = \{\mathtt{a}, \mathtt{b}, \ldots, \mathtt{z}\}.$$

Here, $|\Sigma| = N = 26$.

(ii) Mapping: A standard mapping to integers modulo 26.

$$g : \Sigma \to \{0, \ldots, 25\}, \text{ where } g(\mathtt{a}) = 0, g(\mathtt{b}) = 1, \ldots, g(\mathtt{z}) = 25.$$

The mathematical space is $\mathcal{S} = \mathbb{Z}_{26} = \{0, 1, \ldots, 25\}$.

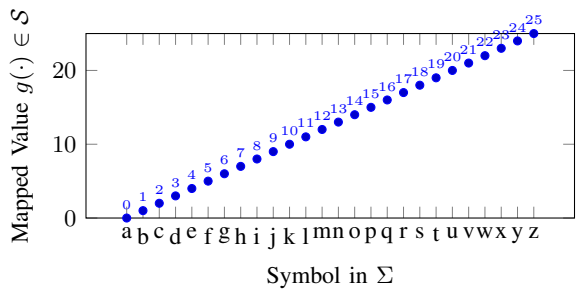

Fig. 1. Mapping g for the shift cipher ($\Sigma \to \mathbb{Z}_{26}$).

(iii) Permutation: Additive shift modulo 26, determined by the secret key $k \in \{0, \ldots, 25\}$.

$$f_k(s) = (s + k) \bmod 26, \quad \text{for } s \in \mathcal{S}.$$

The inverse is $f_k^{-1}(s') = (s' - k) \bmod 26$.

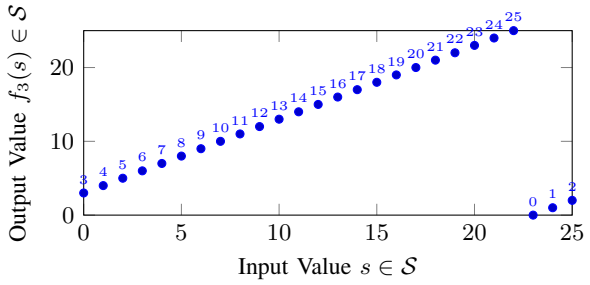

Fig. 2. Permutation f_k for Caesar cipher with key $k = 3$.

The cipher operates on messages composed of symbols from Σ. Because g and g^{-1} map individual symbols, the encryption/decryption process defined in Section II naturally applies symbol by symbol. For instance, to encrypt the message "hello" (note that 'h', 'e', 'l', 'o' are all members of Σ) with key $k = 3$, we process each allowed symbol sequentially. Let's demonstrate with the first symbol, 'h':

1. Map the symbol $\mathtt{h} \in \Sigma$ to the space $\mathcal{S}$: $g(\mathtt{h}) = 7$.
2. Apply the permutation f_3 within $\mathcal{S}$: $f_3(7) = (7 + 3) \bmod 26 = 10$.
3. Map the result $10 \in \mathcal{S}$ back to the symbol set Σ: $g^{-1}(10) = \mathtt{k}$.

Repeating this procedure for the subsequent symbols 'e', 'l', 'l', 'o' yields the complete ciphertext "khoor". This symbol-by-symbol processing ("splitting" the message) is a direct consequence of the framework operating on the basis of the defined symbol set Σ.

B. Monoalphabetic Substitution

(i) Symbol Set & (ii) Mapping: Same Σ and g as for the shift cipher, mapping letters to $\mathcal{S} = \mathbb{Z}_{26}$.

(iii) Permutation: An arbitrary permutation π of the set $\{0, \ldots, 25\}$, determined by the secret key (which essentially *is* the permutation).

$$f_\pi(s) = \pi(s), \quad \text{where } \pi \in S_{26},$$

and S_{26} is the symmetric group of all permutations on 26 elements. The key space size is 26!. The inverse function f_π^{-1} is simply the inverse permutation π^{-1}. Substitution-permutation networks, fundamental to modern ciphers like AES [3], [6], use permutations related to this concept.

IV. Key Agreement and Cryptanalysis

For secure communication, legitimate parties must agree on all three components: (Σ, g, f_k). In many practical scenarios, particularly with standardized algorithms, the symbol set Σ (e.g., ASCII, Unicode bytes) and the mapping g (e.g., standard numerical representation) are publicly known or defined by standards [4]. Security therefore relies primarily on the secrecy of the key k, which determines the specific permutation f_k from a family of possible permutations.

An adversary's goal is typically to recover the secret key k, or equivalently, to determine the function f_k (or its inverse f_k^{-1}) without prior knowledge of k. If Σ and g are known (Kerckhoffs's principle assumption), cryptanalysis often involves analyzing plaintext-ciphertext pairs (m, c) to deduce properties of f_k. This is feasible for simple ciphers like the shift or monoalphabetic substitution via frequency analysis, especially when operating in the mapped space $\mathcal{S}$.

However, if an adversary is uncertain about Σ or g, the task becomes significantly harder. Uncertainty about the symbol set or how symbols are mapped to the mathematical space $\mathcal{S}$ acts as an additional layer of obscurity, potentially hindering statistical attacks or attempts to directly infer f_k. While security through obscurity is generally discouraged as the *sole* means of protection, non-standard choices for Σ and g can supplement the security provided by a strong f_k.

V. Implications and Future Work

Explicitly separating the cipher structure into the (Σ, g, f) triad offers several benefits:

- **Pedagogy:** It provides a clear, structured way to introduce fundamental concepts in symmetric cryptography, distinguishing between representational choices (Σ, g) and the core cryptographic transformation (f).

- **Design:** It encourages designers to consciously consider each component. While f receives the most attention regarding cryptographic strength, choices for Σ and $\mathcal{S}$ (via g) can impact implementation efficiency and interaction with surrounding protocols.
- **Analysis:** It clarifies the assumptions made in security analysis (e.g., which components are considered known to an attacker).

This framework suggests potential research avenues:

- *Obfuscating g*: Systematically exploring the practical security benefits and trade-offs of using secret or non-standard symbol-to-space mappings g, beyond simple obscurity.
- *Algebraic Design of f*: Leveraging group theory, finite field arithmetic, and other algebraic structures for designing families of strong permutations f_k resistant to known attacks (as is done in modern cipher design [6]).
- *Framework Extension*: Adapting or extending the (Σ, g, f) model to more complex scenarios like authenticated encryption schemes or different modes of operation, identifying analogous components [7].

ACKNOWLEDGMENT

The author thanks Salman Youssef for helpful discussions and feedback during the development of this framework.

REFERENCES

[1] C. E. Shannon, "Communication Theory of Secrecy Systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, Oct. 1949.

[2] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996, ch. 7.

[3] NIST, "Advanced Encryption Standard (AES)," *FIPS PUB 197*, Nov. 2001.

[4] D. R. Stinson and M. B. Paterson, *Cryptography: Theory and Practice*, 4th ed. CRC Press, 2019.

[5] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. CRC Press, 2007.

[6] J. Daemen and V. Rijmen, *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer-Verlag, 2002.

[7] R. Lampe and Y. Seurin, "How to Construct an Ideal Cipher from a Small Set of Public Permutations," IACR Cryptology ePrint Archive, Report 2013/255, Apr. 2013. [Online]. Available: https://eprint.iacr.org/2013/255