Journal of Information Hiding and Multimedia Signal Processing ©2025 ISSN 2073-4212
Ubiquitous International Volume **16**, Number **4**, December **2025**

A Novel Framework Integrating Rough Set Theory with Arabic Text Steganography for Improving Data Security

Assist Prof. Dr. Farah R. Shareef

Education Development Department, Iraq
sabahaliraq2014@gmail.com

*Corresponding author: Assist Prof. Dr. Farah R. Shareef

Received May 25, 2025; revised August 25, 2025; accepted September 2, 2025.

Abstract. *Existing Arabic text steganography techniques often suffer from limited hiding capacity, susceptibility to detection, and low robustness. To address these issues, this study introduces a novel two-phase framework that integrates Rough Set Theory (RST) with Arabic text steganography to enhance data security. In the first phase, RST is applied to a decision table constructed from key Arabic letter characteristics (such as pointed vs. Not pointed, solar vs. lunar classification, and Noorani vs. dark categorization) to derive optimized classification rules. In the second phase, these rules guide a new steganographic technique that conceals secret information in Arabic text by embedding two secret bits per character and dynamically adapting across five distinct linguistic contexts to maximize concealment and security. Experimental results demonstrate that the proposed method significantly improves key performance metrics, including embedding accuracy, robustness, and hiding capacity, compared to existing techniques, thereby making it highly suitable for secure data communication applications utilizing the Arabic language.*
Keywords: Rough Set Theory; Arabic text steganography; decision table; data security; embedding techniques.

1. **Introduction.** Steganography is the practice of concealing secret information within an innocuous digital carrier (e.g., text, images, audio, or video) such that the existence of the hidden content is not apparent to unintended recipients. The hidden payload (plaintext or ciphertext) is embedded into a cover to produce a stego object, and extraction typically requires a stego key analogous to a password [1], [2]. In text steganography, common strategies include statistical, random, and format-based manipulations, such as inserting invisible characters or subtle typographic changes. Despite their practicality, many methods still face three persistent limitations: capacity (how much data can be hidden), security (resistance to detection), and robustness (resilience against editing or degradation) [3].

Rough Set Theory (RST), introduced by Pawlak, offers a principled framework for reasoning under uncertainty by approximating sets via their lower and upper bounds and enabling attribute reduction to retain only the most discriminative features. RST has been widely used for pattern discovery, feature significance assessment, and dimensionality reduction with minimal information loss [4]. These properties make RST a natural candidate for deriving compact, data-driven rules that guide text steganography decisions.

The central concept in RST involves the approximation of sets, represented through lower and upper approximations. The lower approximation includes elements that definitively belong to a set, while the upper approximation encompasses all elements that potentially belong to the set. The difference between the lower and upper approximations is referred to as the boundary region, representing uncertainty regarding membership status [5], see Figure 1.

In this research, Rough Set Theory is utilized to enhance Arabic text steganography by systematically deriving decision rules based on specific linguistic attributes of the Arabic language. These rules serve

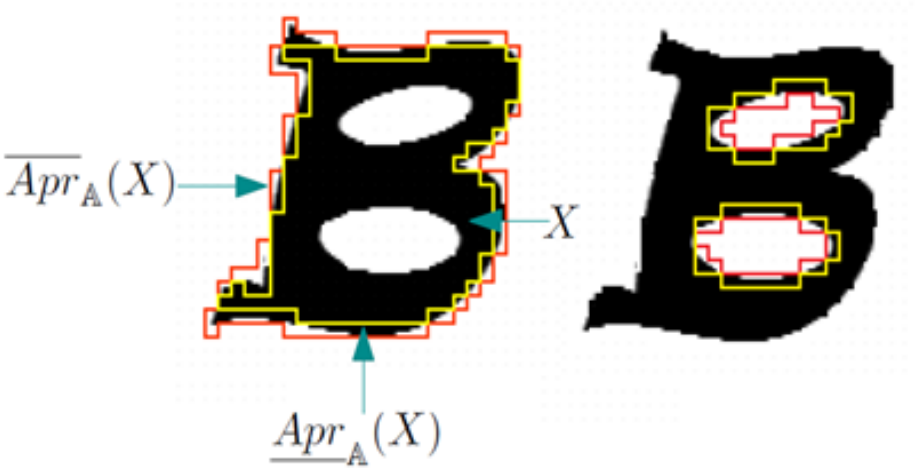

FIGURE 1. The lower and upper approximations for an object X.

as guidelines in developing a novel steganographic approach capable of embedding secret bits efficiently and securely within Arabic textual covers. Rough Set Theory (RST) has been used for a number of things, including quickly analysing binary data, feature reducibility (the ability to easily find the best or worst subsets of attributes), less complex computations, and more accurate results because it is naturally insensitive to image noise [6]. Steganography, on the other hand, is the act of hiding private or secret information in a harmless medium. The secret information can be text, pictures, binary data, or videos. The cover medium can also be text, pictures, or videos. The thing that comes out of the process for embedding is called stego media [7].

This work leverages RST to exploit distinctive linguistic properties of Arabic script such as pointed vs. unpointed letters, solar vs. lunar letters, and Noorani vs. dark groupings to generate classification rules that drive a Unicode-based embedding mechanism. The proposed system encodes two secret bits per character and adapts across five linguistic contexts, including a dedicated strategy for isolated letters that cannot accept Kashida, thereby improving both capacity and concealment while maintaining visual fidelity of the cover text.

1.1. **Related work.** Arabic script comprises 28 letters written in a cursive style (similar to Urdu and Persian). Letter shapes vary by position (initial, medial, final, isolated), and many letters carry one to three dots above or below. Fifteen letters are dotted, and five of these have more than one dot—properties not present in English orthography [8]. In addition, Arabic uses diacritics ("Harakat") that encode short vowels and other phonetic cues—Shaddah, Sukun, Kasrah, Tanwin Kasrah, Fathah, Tanwin Fathah, Dammah, and Tanwin Dammah—which are crucial for accurate interpretation in texts such as the Holy Qur'an and educational materials [9]. Leveraging these linguistic features (dotted vs. undotted letters, solar vs. lunar letters, and Noorani vs. dark classes), the literature has proposed several text-based steganographic strategies:

(1) **Kashida-based embedding [10]:** Methods that insert the Arabic elongation character (Kashida) can encode up to two bits per insertion, improving capacity over single-bit schemes. Some approaches partition the cover text and embed separately in each segment. However, bit ordering and Kashida placement can affect both capacity and detectability.
(2) **Solar/Moon letter encoding [11]:** These techniques map bit patterns to Kashida insertions conditioned on the solar (sun) vs. lunar (moon) classification of the preceding letter. For example, a single Kashida after a solar letter may denote "00," while patterns following lunar letters encode other combinations (e.g., "01," "10," "11").
(3) **Noorani/Dark classes with Unicode spaces [12]:** By combining Kashida with zero-width or non-printing Unicode spaces and exploiting Noorani vs. dark letter groupings, these methods expand the embedding codebook and offer as many as five placement options, thereby improving concealment flexibility.
(4) **Rough Set Theory (RST) for feature selection [13]:** RST has been used to reduce high-dimensional steganalysis features (e.g., via alpha-positive domain reduction), maintaining detection

accuracy while lowering computational cost. Although not an embedding method, this line demonstrates RST's utility for rule extraction and dimensionality reduction.

(5) **Broader RST applications [14]:** Surveys show RST's integration with fuzzy, probabilistic, and deep-learning frameworks for large-scale analytics, suggesting potential for hybrid, data-driven rule design in steganography.

(6) **RST in Arabic text processing [15]:** RST-based approaches have improved Arabic text direction detection, reporting notable accuracy across unigram, bigram, and trigram models, and indicating that RST can discriminate among linguistically motivated feature sets.

Hence from analysis of related works, we can identify limitations, one that the existing Arabic text steganography largely focuses on Kashida patterns and letter classes but does not systematically derive embedding rules using RST, nor does it consistently handle isolated letters (which cannot accept Kashida) with a unified Unicode-based strategy. Our work addresses these gaps by using RST to generate compact decision rules over Arabic letter attributes and by operationalizing them in a two-bit, multi-context embedding scheme.

1.2. **Motivation and contribution.** To overcome the capacity, security, and robustness gaps noted above, this method propose an RST-guided, Arabic-aware steganography framework. The main contributions are:

- **RST-derived decision rules.** This method construct a decision table over Arabic letter attributes (pointed/unpointed, solar/lunar, Noorani/dark, isolated) and apply RST to obtain lower/upper approximation rules that minimize attributes while maximizing classification reliability for embedding actions.
- **Two-bit, Unicode-based embedding across eight contexts.** Using Kashida, Zero-Width Joiner (ZWJ), Zero-Width No-Break Space (ZWNBS), Zero-Width Space (ZWS), word joiner (WJ), Right-to-Left mark (R-T-L), Arabic Letter Mark, and Hair Space, the method encode 2 bits/character and adapt to five linguistic scenarios, including a dedicated Rule 5 for isolated letters that cannot take Kashida. This expands the usable alphabet and reduces detectability.
- **Practical pipeline and evaluation.** We detail embedding and extraction algorithms, and evaluate on Arabic covers with secret messages in Arabic, English, and Persian. Results show improved hiding capacity and favorable secret-to-cover ratios compared with representative baselines, with optional Gzip compression further increasing effective capacity.

2. Arabic Text Steganography. Preliminaries.

2.1. **Secret Message Pre-processing.** Before embedding data, the secret message is prepared to maximize capacity and maintain cover text quality. The message is first encoded into a binary bitstream (sequence of 0s and 1s). If the message is large, compression can be applied (e.g., using Gzip) to reduce its length before embedding [16]. Compressing the secret message decreases the number of bits to hide, thereby improving effective embedding capacity and reducing the impact on the cover text. After optional compression, the binary secret data is ready to be concealed two bits at a time in the cover text. Figure 2 demonstrates how the secret message is compressed before being passed to the embedding module.

FIGURE 2. The framework for the compression of confidential communications.

2.2. **Arabic Cover Text Characteristics and Features.** Arabic script has unique linguistic features that we exploit for steganography. It is written cursively, and many letters can connect to adjacent letters. The Kashida (Arabic Tatweel, U+0640) is an elongation character that can be inserted into certain connected letters without altering the text's meaning. However, not all Arabic letters accept a Kashida insertion letters that cannot connect to a subsequent letter (such as alef " ", dal " ", etc.) will

not visually accommodate a Kashida. We refer to such non-connecting letters as isolated letters, since they break the cursive connection (appearing isolated in writing).

Additionally, Arabic letters can be classified by other attributes that affect how we embed data:

1. **Pointed vs. Not Pointed:** Some letters have one or more dots (e.g. " " has a dot, " " has a dot below) while others have none. Prior work has used this distinction in Kashida-based schemes;
2. **"Noorani" vs. "Darkness" Letters:** This is a linguistic grouping of Arabic letters (Noorani appears in specific contexts such as Al-Fatiha, while Darkness letters do not). In our context, we divide the alphabet into two groups termed Noorani and Dark letters (each group containing certain characters). This grouping, along with the letter's connectivity, will guide different embedding rules;
3. **Solar vs. Lunar Letters:** The Arabic alphabet is also categorized into sun (shamsiyyah) letters and moon (qamariyyah) letters based on how the definite article is assimilated in pronunciation. This attribute can influence where subtle markers might be less noticeable. (This feature is conceptually related to Noorani/Dark classification in our scheme.); Table 1 show the Pointed and Not Pointed letters, Noorani and Darkness letters, (Solar/sun) and (Lunar/Moon) letters and Isolated letters.

Table 1. Attributes of Arabic Letters

Attribute	**Arabic letters**
Pointed	ب-ت-ث-ج-خ-ذ-ز-ش-ض-ظ-غ-ف-ق-ن-ي
Not Pointed	أ-ح-د-ر-س-ص-ط-ع-ك-ل-م-ه-و
Noorani	ح-س-ص-ط-ع-ق-ك-ل-م-ن-ه-ي
Darkness	ب-ت-ث-ج-خ-ش-ض-ظ-غ-ف
(Solar/sun)	ت-ث-س-ش-ص-ض-ط-ظ-ل-ن
(Lunar/Moon)	ب-ج-ح-خ-ع-غ-ف-ق-ك-م-ه-ي
Isolated	ء-ا-أ-إ-آ-ئ-ى-ذ-د-ر-ز-ة-ه-و-ؤ

Using these attributes, we identify seven distinct contexts in the cover text that can carry hidden bits.

- **Attributes:** These represent the linguistic features of Arabic letters, including:
  - Whether the letter is pointed or not pointed.
  - Whether it is a solar (sun) or lunar (moon) letter.
  - Whether it belongs to the Noorani or dark group.
  - Whether it is an isolated letter that cannot accommodate a Kashida extension.
- **Decision Attribute:** This defines the classification or embedding action to be applied, based on the combination of the above attributes.
- **The Decision Rules definition.** Using Rough Set Theory, decision rules are systematically extracted from the decision table as Table 2. This process involves three key components:
  - ∅ **Indiscernibility Relation:** This step identifies data entries (Arabic letters) that are indistinguishable from one another based on the selected set of attributes. It establishes equivalence classes of objects sharing identical feature values.
  - ∅ **Attribute Reduction:** This stage determines the minimal subset of attributes that preserves the classification ability of the full attribute set. The resulting reduct eliminates redundant or non-essential features while retaining classification integrity.
  - ∅ **Rule Generation:** Based on the reduced data, a set of decision rules is derived. These rules enable accurate classification of new, unseen instances by mapping combinations of linguistic features to specific embedding actions.

Each context will be handled with a tailored embedding strategy to optimally hide two bits per character while preserving the natural appearance of the text. Table 2 shows the features of Arabic letters.

3. **Improved Method for Arabic Text Steganography.** The proposed steganographic method leverages Rough Set Theory (RST) to systematically derive embedding rules from the Arabic linguistic features above. By using RST, we ensure that the chosen features (letter attributes) lead to a minimal and effective set of rules, improving embedding capacity and security. The scheme embeds 2 secret bits per character by using a combination of Kashida and Unicode zero-width characters, adapting the technique across five

Table 2. The features of Arabic letters

Arabic letters	Pointed	Un-pointed	Noorani	Darkness	Sun	Moon	Decision_start embedding
ا	NO	YES	YES	NO	NO	YES	SKIP
ب	YES	NO	NO	YES	NO	YES	ZWJ
ت	YES	NO	NO	YES	YES	NO	KASHIDA1
ث	YES	NO	NO	YES	YES	NO	KASHIDA1
ج	YES	NO	NO	YES	NO	YES	ZWJ
ح	NO	YES	YES	NO	NO	YES	K+ZWS
خ	YES	NO	NO	YES	NO	NO	ZWJ
د	NO	YES	NO	YES	YES	NO	SKIP
ذ	YES	NO	NO	YES	YES	NO	SKIP
ر	NO	YES	YES	NO	YES	NO	SKIP
ز	YES	NO	NO	YES	YES	NO	SKIP
س	NO	YES	YES	NO	YES	NO	ZWJ
ش	YES	NO	NO	YES	YES	NO	KASHIDA1
ص	NO	YES	YES	NO	YES	NO	ZWJ
ض	YES	NO	NO	YES	YES	NO	KASHIDA1
ط	NO	YES	YES	NO	YES	NO	ZWJ
ظ	YES	NO	NO	YES	YES	NO	KASHIDA1
ع	NO	YES	YES	NO	NO	YES	K+ZWS
غ	YES	NO	NO	YES	NO	YES	ZWJ
ف	YES	NO	NO	YES	NO	YES	ZWJ
ق	YES	NO	YES	NO	NO	YES	SKIP
ك	NO	YES	YES	NO	NO	YES	K+ZWS
ل	NO	YES	YES	NO	YES	NO	ZWJ
م	NO	YES	YES	NO	NO	YES	K+ZWS
ن	YES	NO	YES	NO	YES	NO	SKIP
ه	NO	YES	YES	NO	NO	YES	K+ZWS
و	NO	YES	YES	YES	NO	YES	SKIP
ي	YES	NO	YES	NO	NO	YES	SKIP

linguistic contexts (including a special handling for isolated letters that cannot accept Kashida). This results in higher payload capacity and improved invisibility compared to traditional one-bit methods.

3.1. **RST-Based Derivation of Embedding Rules.** Rough Set Theory is used to generate decision rules that map each letter to an appropriate embedding action based on its attributes. We construct a decision table where each object is a character in the cover text (categorized by its linguistic features), and the decision outcome is the embedding strategy for that character (e.g. which character to insert, if any). RST is adept at reducing attributes and extracting minimal rules, which helps identify the most significant features for our steganography scheme. By applying RST, we obtained a set of if-then decision rules guiding the hiding process. In essence, the rules partition the problem into the four cases mentioned (Pointed vs. Not Pointed, Noorani vs. Dark letters, Sun vs. Moon letters, Isolated vs. connected letters), determining how to encode two bits in each case.

For example, RST analysis revealed that a letter's group (Pointed/Not Pointed) and its connectivity (isolated or not) are sufficient to decide the embedding method for that letter. This led to rules such as: "IF letter is Pointed-group AND not isolated THEN...; IF letter is Dark-group AND Sun-group THEN

use pattern X; IF letter is isolated THEN use pattern Y," and so forth, with a separate rule for spaces. These rules form the basis of our improved embedding algorithm, ensuring that each cover character is treated optimally according to its type. The use of RST guarantees that we are not using any superfluous features – the rule set is minimal and focused, which simplifies the embedding and enhances reliability.

3.2. **Two-Bit Embedding Scheme Design.** Using the rules derived above, the proposed method designs an embedding scheme that hides 2 bits in each eligible character or space by adding zero-width or Kashida characters. The method employs a variety of Unicode invisible characters to represent different bit values without visibly altering the text. Specifically, the method utilizes:

(1) **Kashida (Tatweel, `U+0640`):** a visible elongation character inserted into connectable letters;
(2) **Zero-Width Joiner (ZWJ, `U+200D`), Word Joiner (WJ, `U+2060`) and Zero-Width Non-Joiner (ZWNJ, `U+200C`):** invisible formatting marks that affect text joining behavior but have no visible glyph, used in certain contexts (e.g., around spaces or non-connecting letters).
(3) **Non-printing thin spaces:** a set of very narrow space characters (such as Hair Space `U+200A`, Right-to-Left Mark (R-T-L, `U+200F`), Zero-Width Space (ZWS, `U+FEFF`), Arabic Letter Mark, `U+061C`). These appear almost imperceptible or not at all in text, especially when used sparingly.

By combining these characters, we create a mapping for each 2-bit pair in each context. In general, for a given character: two of the bit combinations are encoded by inserting one of the above characters, while the remaining two combinations are encoded by leaving the text unchanged, relying on the absence of an insertion as a meaningful state. This approach ensures that every possible 2-bit value (00, 01, 10, 11) has a representation either as a specific hidden character or as a "no embed" condition, which maintains the cover text if those bits are encountered.

For instance, in the case of an Un-Pointed-group letter that can accept a Kashida, our scheme might define: insert a Kashida after the letter to encode "00", insert a ZWJ to encode "01", and for bit-pairs "10" and "11", the insertion are Kashida+ZWS and ZWJ+ZWS, respectively. This complementary design balances the use of Kashida across different letter types, reducing any bias in where elongation appears.

For letters that cannot accept a Kashida (isolated letters), for example, an isolated letter might use a R-T-L (`U+200F`) to encode "00" and ZWS (`U+FEFF`) to encode "01" and Arabic letter mark to encode "10", while using Hair Space (`U+200A`) to encode "11".

By adapting the encoding strategy to each context (letter type), the scheme ensures that two bits are hidden per character on average, significantly boosting capacity. Importantly, the use of Kashida and zero-width characters is spread out and context-aware, which preserves the visual fidelity of the cover text – e.g., Kashidas are only added where they naturally could appear for justification, and other markers are invisible or negligibly small. This minimizes the risk of detection or degradation of the text. The RST-derived rules govern these decisions, making the embedding process both automated and based on solid feature selection.

4. Embedding Framework: Rough Set and Steganography Processes.

4.1. **Phase 1: Rough Set Rule Extraction Process. Input:** A .csv file containing the decision table, where each row represents an Arabic letter sample and each column denotes an attribute such as:

- Pointed / Not Pointed
- Dark / Noorani
- Sun (Solar) / Moon (Lunar)

Output: Two sets of decision rules:

- Upper Approximation Rules (Uncertain cases)
- Lower Approximation Rules (Certain cases)

Process:

1. Collect Arabic letter characteristics based on their linguistic features.
2. Design the decision table using Microsoft Excel with rows as letter samples and columns as feature attributes.
3. Apply Rough Set Theory to compute the indiscernibility relation and group equivalent entries.
4. Calculate the lower and upper approximations for each class (decision attribute).
5. Extract and record the resulting decision rules for later use in the embedding phase.

4.2. **Phase 2: Embedding Process (Steganography).** Using the above scheme, the method implements an embedding algorithm that conceals the secret bitstream into an Arabic cover text. Pseudo-code for the procedure is outlined below:

1. **Input:** Secret message S; Cover text C (in Arabic).
   Output: Stego text with hidden message (Arabic-Stego).
2. **Prepare Data:** Convert the secret message S into a binary bit sequence. If S is large, apply compression beforehand to reduce its size. Let *bitstream* represent the sequence of secret bits to embed. Initialize a pointer j for *bitstream* at the beginning.
3. **Traverse Cover:** Iterate through each character $C[i]$ in the cover text:
   - **Skip Non-letters:** If $C[i]$ is a harakat (Arabic vowel diacritic), punctuation, digit, or white-space that is not to be used for embedding, skip it (i.e., copy it unchanged to output). These characters do not carry hidden data, so they remain as is.
   - **Otherwise,** determine the feature category of $C[i]$: Is it a Pointed or Not Pointed letter? Can it connect to the next letter (i.e., not one of the isolated forms) or is it an isolated letter?
4. **Apply Embedding Rule:** Based on the category of $C[i]$, apply the corresponding embedding rule (from the RST-derived rules) to hide the next two secret bits. Let (b_1, b_2) be the next two bits.
   → **Pointed Letter Group:**
   - **Rule 1 (Dark + Sun letters):**
     - If $(b_1b_2) = 00 \rightarrow$ Insert Kashida + WJ after $C[i]$.
     - If $(b_1b_2) = 01 \rightarrow$ Insert ZWJ after $C[i]$.
     - If $(b_1b_2) = 10 \rightarrow$ Insert Kashida after $C[i]$.
     - If $(b_1b_2) = 11 \rightarrow$ Insert ZWJ + ZWS after $C[i]$.
   - **Rule 2 (Dark + Moon letters):**
     - If $(b_1b_2) = 00 \rightarrow$ Insert ZWJ + ZWS.
     - If $(b_1b_2) = 01 \rightarrow$ Insert Kashida1 + WJ.
     - If $(b_1b_2) = 10 \rightarrow$ Insert ZWJ.
     - If $(b_1b_2) = 11 \rightarrow$ Insert Kashida1.

   → **Unpointed Letter Group:**
   - **Rule 3 (Noorani + Sun letters):**
     - If $(b_1b_2) = 00 \rightarrow$ Insert Kashida.
     - If $(b_1b_2) = 01 \rightarrow$ Insert ZWJ.
     - If $(b_1b_2) = 10 \rightarrow$ Insert Kashida + ZWS.
     - If $(b_1b_2) = 11 \rightarrow$ Insert ZWJ + ZWS.
   - **Rule 4 (Noorani + Moon letters):**
     - If $(b_1b_2) = 00 \rightarrow$ Insert ZWJ + ZWS.
     - If $(b_1b_2) = 01 \rightarrow$ Insert Kashida.
     - If $(b_1b_2) = 10 \rightarrow$ Insert ZWJ.
     - If $(b_1b_2) = 11 \rightarrow$ Insert Kashida + ZWS.

   → **Isolated Letter Group:**
   - **Rule 5 (Non-joinable characters):**
     - If $(b_1b_2) = 00 \rightarrow$ Insert Right-to-Left Mark (R-T-L) after the character.
     - If $(b_1b_2) = 01 \rightarrow$ Insert Zero-Width Space (ZWS).
     - If $(b_1b_2) = 10 \rightarrow$ Insert Arabic Letter Mark.
     - If $(b_1b_2) = 11 \rightarrow$ Insert Hair Space.

   Copy the original cover character $C[i]$ (and any inserted symbol) to the output stego text.
5. **Termination:** Continue the loop until either all secret bits have been embedded (j reaches the end of the bitstream) or the cover text is exhausted. If the cover text ends but there are still secret bits left, the algorithm cannot embed the entire message (this situation is avoided by pre-checking capacity and possibly compressing or choosing a larger cover). If the secret bits end before cover text is done, the remaining cover characters are just copied over with no changes.
6. **Output Stego Text:** The result is the stego-text where the secret message bits are hidden in the form of Kashidas and zero-width characters embedded as per the rules. This steganographic text should appear visually identical to the original cover text for a human reader, with differences only at the invisible-character level. Table 3 shows the embedding rules.

It is observed that the embedding decision presented in Table 3 corresponds to a single instance of data; for example, in Rule 1 '10', the decision is to embed using the combination (Kashida1 + ZWS). However, in practical steganographic applications, it is essential to accommodate all four binary combinations 00,

TABLE 3. Applying five rules for embedding

Rule	Secret Bits	Action
Rule 1	00	Kashida (K1) (U+0640)
	01	ZWJ (U+200D)
	10	K1 + ZWS
	11	ZWJ + ZWS
Rule 2	00	ZWJ + ZWS
	01	K1
	10	ZWJ
	11	K1 + ZWS
Rule 3	00	K1 + WJ (U+2060)
	01	ZWJ
	10	K1
	11	ZWJ + ZWS
Rule 4	00	ZWJ + ZWS
	01	K1 + WJ
	10	ZWJ
	11	K1
Rule 5	00	R-T-L (U+200F)
	01	ZWS (U+FEFF)
	10	Arabic Letter Mark (U+061C)
	11	Hair Space (U+200A)

01, 10, and 11 for each rule to ensure flexible and comprehensive embedding functionality, as illustrated in Table 3.

For instance, consider the Arabic letter ش appearing in the cover text. If the secret bits to be embedded are '10', and the letter matches the attributes defined in Rule 3 (pointed, dark, solar), then the appropriate embedding action is to insert Kashida1 after the letter. Conversely, if the secret bits are '11', the system applies a different encoding ZWJ + ZWS—to maintain accurate and undetectable data hiding under the same rule.

Additionally, Rule 5 must be defined and applied to handle isolated letters, which cannot accommodate Kashida insertions. In the current version of Table 2, isolated letters were previously marked as "skip," but to enhance embedding capacity and completeness, a dedicated concealment strategy for such letters is necessary in the steganography phase.

4.3. **Extraction Process.** The extraction procedure is the reverse of embedding and is straightforward given the known rules.

Input: Arabic stego cover (Ar-St) contains a secret message.

Output: Secret message(s).

Process:

1. Capture the cover text and convert it into an array of characters.
2. Advance through the list one character at a time, referring to each as $C[i]$.
3. Classify each $C[i]$ into one of three categories: (Pointed, Unpointed, or Isolated).
   - **If $C[i]$ is Pointed Letter Then Examine:**
     - If $C[i]$ is (Darkness + Sun) letters Then:
       - If $C[i+1]$ is (K1 + WJ) Then get bits $(b_1 b_2) = 00$
       - If $C[i+1]$ is (ZWJ) Then get bits $(b_1 b_2) = 01$
       - If $C[i+1]$ is (Kashida1) Then get bits $(b_1 b_2) = 10$
       - If $C[i+1]$ is (ZWJ + ZWS) Then get bits $(b_1 b_2) = 11$
     - If $C[i]$ is (Darkness + Moon) letters Then:

        * If $C[i+1]$ is (ZWJ + ZWS) Then get bits $(b_1b_2) = 00$
        * If $C[i+1]$ is (K1 + WJ) Then get bits $(b_1b_2) = 01$
        * If $C[i+1]$ is (ZWJ) Then get bits $(b_1b_2) = 10$
        * If $C[i+1]$ is (Kashida1) Then get bits $(b_1b_2) = 11$
- **If $C[i]$ is Unpointed Letter Then Examine:**
  - If $C[i]$ is (Noorani + Sun) letters Then:
    * If $C[i+1]$ is (Kashida1) Then get bits $(b_1b_2) = 00$
    * If $C[i+1]$ is (ZWJ) Then get bits $(b_1b_2) = 01$
    * If $C[i+1]$ is (K1 + ZWS) Then get bits $(b_1b_2) = 10$
    * If $C[i+1]$ is (ZWJ + ZWS) Then get bits $(b_1b_2) = 11$
  - If $C[i]$ is (Noorani + Moon) letters Then:
    * If $C[i+1]$ is (ZWJ + ZWS) Then get bits $(b_1b_2) = 00$
    * If $C[i+1]$ is (Kashida1) Then get bits $(b_1b_2) = 01$
    * If $C[i+1]$ is (ZWJ) Then get bits $(b_1b_2) = 10$
    * If $C[i+1]$ is (K1 + ZWS) Then get bits $(b_1b_2) = 11$
- **If $C[i]$ is Isolated Letter Then Examine:**
  - If $C[i+1]$ is Right-to-Left (R-T-L) Then get bits $(b_1b_2) = 00$
  - If $C[i+1]$ is Zero-Width Space (ZWS) Then get bits $(b_1b_2) = 01$
  - If $C[i+1]$ is Arabic Letter Mark Then get bits $(b_1b_2) = 10$
  - If $C[i+1]$ is Hair Space Then get bits $(b_1b_2) = 11$

4. If compression was used, the final step is to decompress the retrieved bitstream to obtain the original secret message.
5. **End.**

Overall, this RST-guided Arabic text steganography method enables efficient hiding of data by encoding two bits per character in multiple contexts. By intelligently leveraging Arabic-specific features and zero-width characters, it achieves high capacity and remains inconspicuous – the inserted characters are virtually invisible and linguistically appropriate, thus preserving the cover text's appearance and integrity. The use of decision rules (from RST) adds a layer of adaptability and ensures that the embedding logic is both optimal and generalized, making the steganographic technique robust against detection while maximizing the payload. Figure 3 illustrates the proposed method diagram.

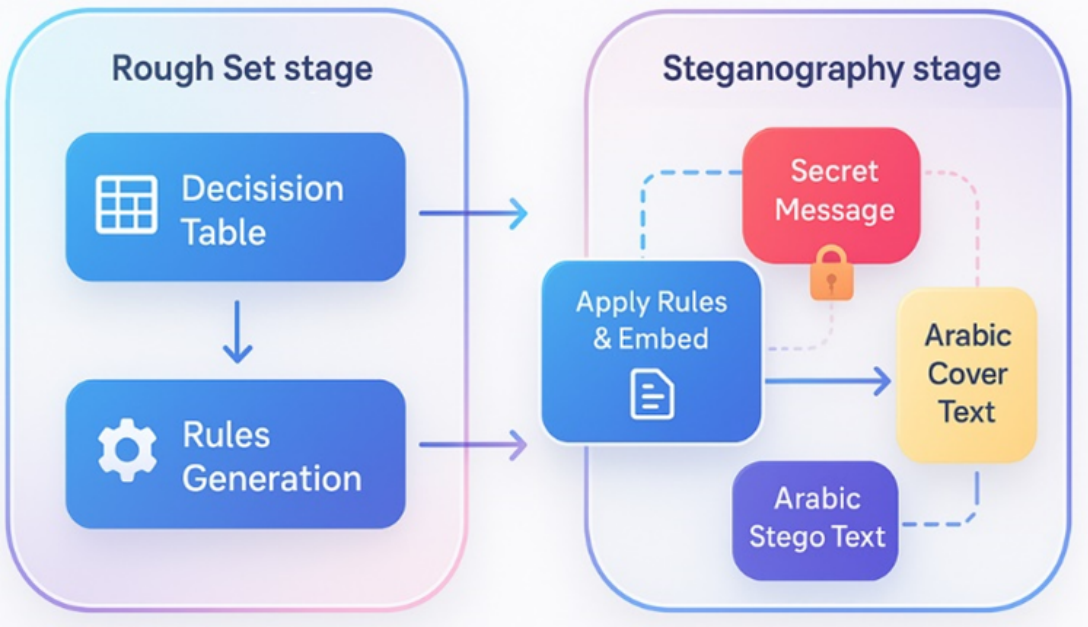

FIGURE 3. Diagram for the proposed method.

5. **Evaluation.** A key objective of this research is to enhance the effectiveness of the steganographic process by improving both the capacity and concealment performance. To accurately evaluate the proposed method, several essential metrics and equations are introduced:

- **Actual Character Utilization (Real Use):** This metric refers to the count of meaningful characters in the cover text which are effectively utilized to embed hidden information. It is considered the true measure of usable capacity for embedding [17].
- **Proportion Capacity (PC):** This metric quantifies the percentage of cover text utilized for hiding data. It is defined as:

$$PC = \left(\frac{\text{Actual Utilization of Characters}}{\text{Length of Cover Text}} \right) \times 100 \tag{1}$$

- **Hiding Capacity (HC):** This expresses the density of hidden data relative to the embedding capacity, calculated as:

$$HC = \left(\frac{\text{Size of Secret Message (in bits)}}{\text{Actual Utilization of Characters (in bytes)}} \right) \times 100 \tag{2}$$

- **Secret-to-Cover Ratio (SCR):** This ratio indicates the extent to which the cover text is used to conceal data and is defined as:

$$SCR = \left(\frac{\text{Actual Utilization of Characters}}{\text{Number of Secret Bits}} \right) \times 100 \tag{3}$$

Understanding and optimizing these metrics is critical to ensuring that the steganographic process remains both secure and efficient across various Arabic textual contexts [18].

6. **Experimental Results and Analyses.** The initial set of embedding rules was generated during the Rough Set analysis phase. This process was implemented using the R programming language, which was applied to the decision table (Table 2) formatted as a .csv file created in Microsoft Excel. The execution of the Rough Set operations—including attribute analysis and rule extraction—is illustrated in Figures 4, 5, and 6, corresponding to the processing steps and output results.

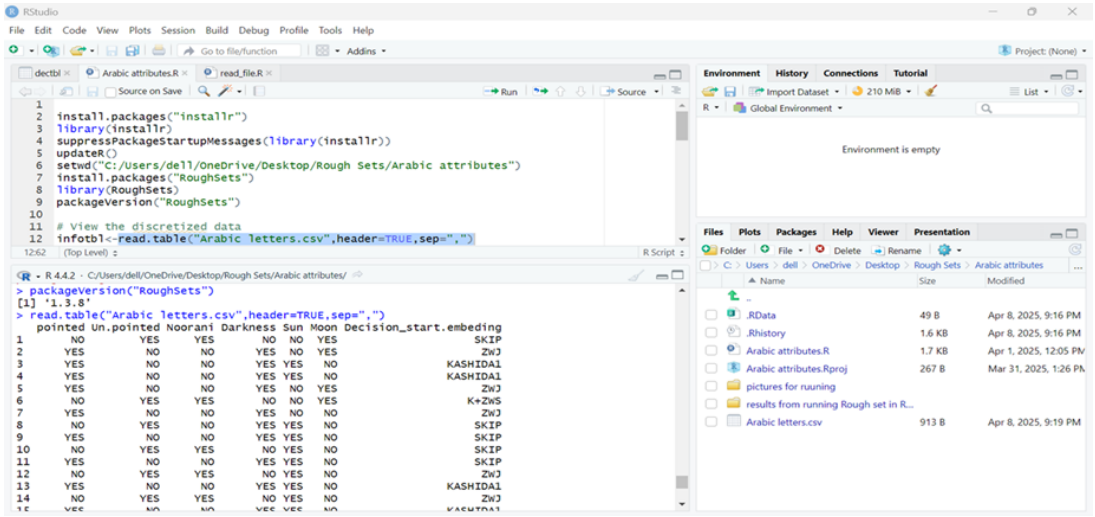

FIGURE 4. Running the code for rough set with R-language.

For instance, the upper approximation rules generated for the embedding action labeled Kashida1 include rule numbers 3, 4, 9, 11, 13, 15, and 17. All of these rules apply specifically to pointed Arabic letters. A representative rule derived from this set is:

If (Pointed = True, Dark = True, and Sun = True), then the decision is Kashida1.

This rule was extracted during the Rough Set phase and it utilized in the steganography phase for embedding the secret bit pattern '10', as illustrated in Table 4.

The following decision rules were extracted using Rough Set Theory, based on the linguistic features of Arabic letters. These rules dictate the specific embedding strategy to be used during the steganography phase:

- **Rule 1:** If a letter possesses the attributes pointed, dark, and belongs to the solar group, then the appropriate embedding marker is Kashida1.
- **Rule 2:** If a letter is pointed, dark, and classified as a lunar letter, then the embedding marker is the Zero-Width Joiner (ZWJ).

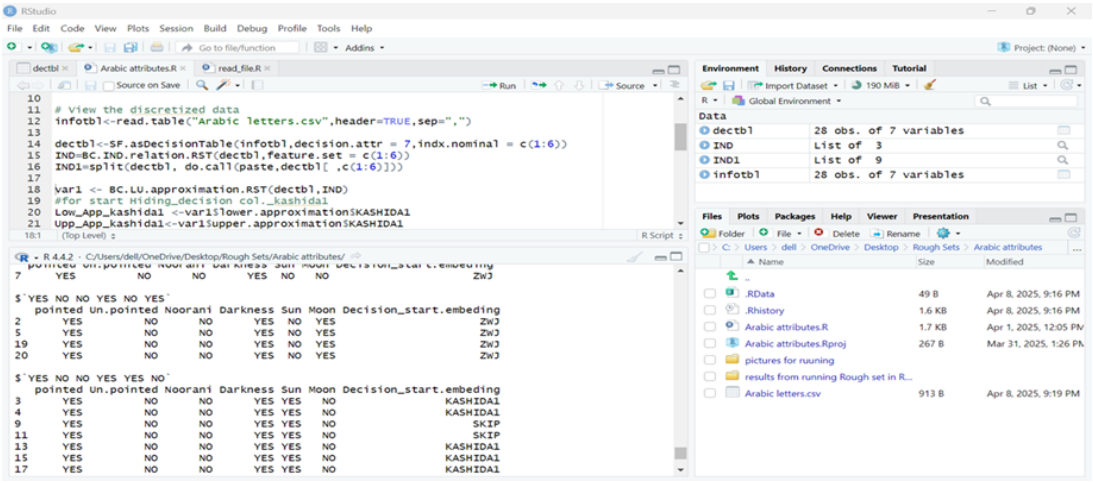

FIGURE 5. The running for the lower and upper approximate for kashida1.

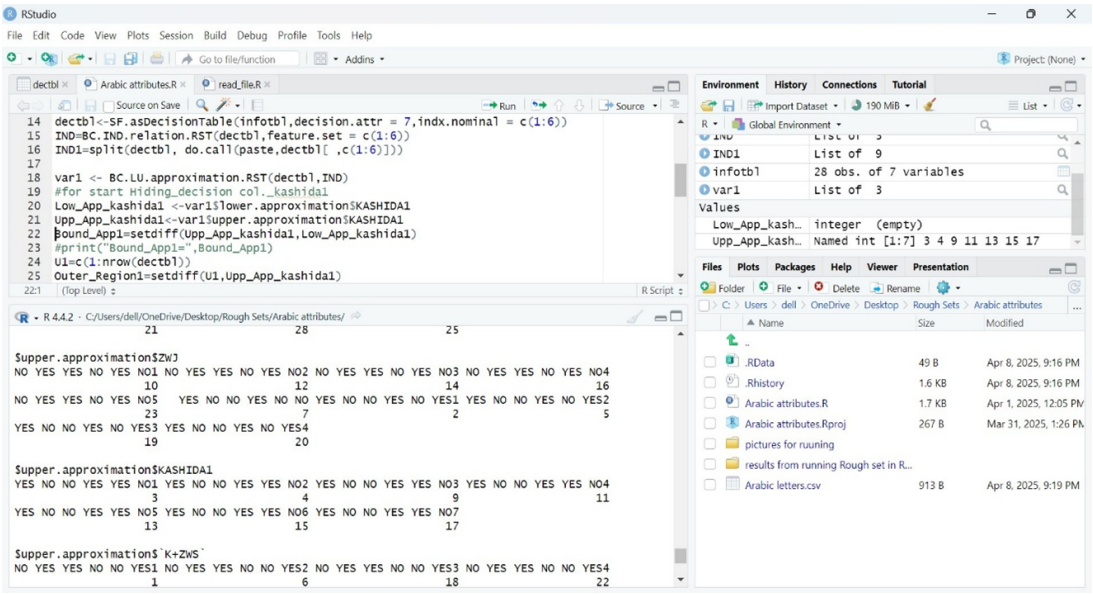

FIGURE 6. The generated rules.

- **Rule 3:** If a letter is unpointed, Noorani, and belongs to the solar group, the embedding marker is also ZWJ.
- **Rule 4:** If a letter is unpointed, Noorani, and classified as a lunar letter, then the embedding strategy is a combination of Kashida1 and Zero-Width No-Break Space (ZWS).

These rules form the core logic for adaptively selecting Unicode-based concealment techniques depending on the structural features of each Arabic letter.

TABLE 4. Four Rules from Rough Set

Rules	Pointed	Unpointed	Noorani	Darkness	Sun	Moon	Embedding
1	✓			✓	✓		kashida1
2	✓			✓		✓	zwj
3		✓	✓		✓		zwj
4		✓	✓			✓	k1+zws

In the second phase of the methodology, the proposed steganographic system was evaluated using secret messages written in three languages: Arabic, English, and Persian, each varying in size. All secret messages were embedded into a common Arabic cover text. The specifications of the secret and cover messages are summarized in Table 5.

TABLE 5. Secret messages with sizes, cover and languages

Cover and Secret Messages	The Languages	The Number of Letters
M1	English	28
M2	English	233
M3	Persian	346
M4	Arabic	995
M5		1678
M6		2391
M7		2425
C	Arabic (Cover)	187782

The testing environment included a laptop equipped with a 2.8 GHz CPU running Windows 11. The entire system was developed and executed using the C# programming language. For evaluation purposes, each secret message is denoted as M, while the Arabic cover message is denoted as C.

Seven secret messages (M1 to M7) were tested against the Arabic cover message to assess the model's performance in real-world text scenarios. The evaluation focused on the model's ability to accurately embed information based on linguistic characteristics, as well as its effectiveness in terms of hiding capacity and cover utilization. The results are illustrated in Figure 7 and detailed in Table 6.

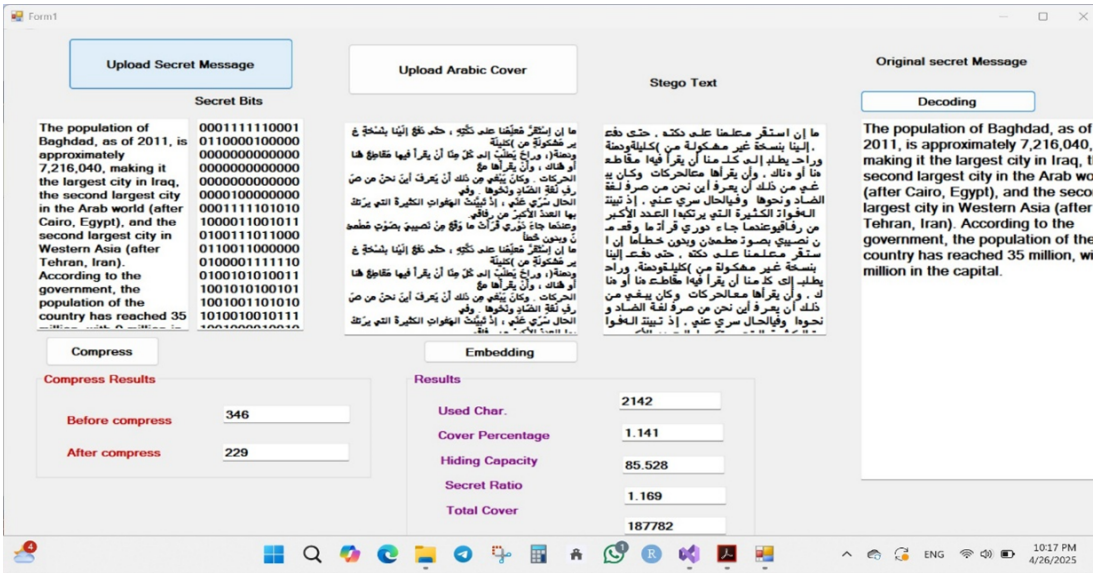

FIGURE 7. An Arabic cover was added to the message 2 implementation through an immediate GUI.

As shown in Table 6, the application of seven secret messages of different lengths reveals notable variations in performance metrics. Specifically, Message M1 yields the lowest cover utilization percentage, indicating minimal use of the available cover space. In contrast, Message M2 achieves the lowest secret-to-cover ratio, suggesting efficient embedding, and simultaneously records the highest hiding capacity percentage, reflecting superior concealment efficiency.

7. **Conclusion.** This study introduces a novel framework that integrates Rough Set Theory (RST) with Arabic text steganography to enhance the accuracy, capacity, and adaptability of information hiding

TABLE 6. The outcomes of real use, hiding capacity and percent coverage of three communications, including a confidential message (both pre- and post-compression).

Secret Message	In Bytes	Cover Size (bytes)	Real Use (char)	Cover Percent	Hiding Percent Coverage	Secret Ratio	Compression Size	
							Before	After
M1	28	187782	264	0.141	84.84	1.179	28	—
M2	233	1598	264	0.851	85.60	1.168	233	171
M3	346	2142	346	1.141	85.52	1.169	346	229
M4	995	4712	995	2.50	85.05	1.176	995	501
M5	1678	5334	1678	2.841	85.18	1.174	1678	568
M6	2391	9295	2391	4.950	84.94	1.177	2391	987
M7	2425	9024	2425	4.80	85.01	1.176	2425	959

techniques. The proposed method is structured into two main phases. In the first phase, RST is applied to a decision table constructed from the linguistic attributes of Arabic letters—such as pointed/not pointed, solar/lunar classification, and membership in Noorani or dark categories—to generate optimized embedding rules. These rules are derived using upper and lower approximations, ensuring both certain and uncertain classification cases are handled.

In the second phase, the generated rules are utilized to embed two secret bits into Arabic cover texts using various Unicode-based techniques (e.g., Kashida, Zero-Width Joiner, ZWS, RTL mark). The technique is designed to react dynamically to five language contexts, including particular treatment for single characters, resulting in increased flexibility and robustness. Experimental evaluation of messages in Arabic, Persian, and English revealed that the suggested system retains excellent hiding capacity, low embedding distortion, and effective cover space utilization across a wide range of message durations. The addition of a compression phase enhances the method's scalability and performance. Overall, the framework provides a secure, adaptable, and linguistically informed steganographic paradigm that is appropriate for Arabic-script environments and multilingual settings.

Future study may investigate additional optimization of feature selection, application to other languages with similar structural qualities, and the incorporation of machine learning to dynamically refine embedding strategies.

References

[1] M. Al-Gailani, "Advanced Cryptographic System: Design, Architecture and FPGA Implementation", Ph.D. thesis, Newcastle University - School of Electrical Electronic and Computer Engineering, 2012.

[2] A. Kumar, "Steganography- A Data Hiding Technique", *International Journal of Computer Applications*, vol. 9, no. 7, 2010.

[3] H. Alshahrani and G. Weir, "Hybrid Arabic text steganography", *International Journal of Computer and Information Technology*, vol. 6, no. 6, 2017.

[4] Z. Pawlak, "Rough Sets and Theoretical Aspects of Reasoning About Data", Institute of Computer Science, vol. 9, 1991.

[5] Z. Pawlak, "Rough set theory and its applications", *Journal of Telecommunications and Information Technology*, vol. 3, no. 3, pp. 7–10, 2002.

[6] U. Chaudhuri, "Rough Set Based Analysis of Document Images", M.Sc. thesis, Advanced Technology and Development Center Indian Institute of Technology, 2017.

[7] M. Almayyahi and R. Sulaiman, "A Review on Text Steganography Techniques", *Mathematics*, vol. 9, no. 21, 2021.

[8] R. Thabit, N. Udzir, and Sh. Yasin, "A comparative analysis of Arabic text steganograph", *Applied Sciences*, vol. 11, no. 15, 2021.

[9] H. Kadhim and M. Mahdi, "Novel steganography scheme using Arabic text features in Holy Quran", *International Journal of Electrical and Computer Engineering*, vol. 9, no. 3, 2019.

[10] A. Alhusban and J. Alnihoud, "A meliorated Kashida-based approach for Arabic text steganography", *International Journal of Computer Science and Information Technology*, vol. 9, no. 2, pp. 99–112, 2017.
[11] A. Shaker and F. Ridzuan, "Text steganography using extension Kashida based on the moon and sun letters concept", *International Journal of Advanced Computer Science and Applications*, vol. 8, no. 8, pp. 286–290, 2017.
[12] F. R. Shareef, "Text steganography based on Noorani and Darkness", *Journal of Information Hiding and Multimedia Signal Processing*, vol. 12, no. 3, 2021.
[13] Y. Ma, X. Luo, and X. Li, "Selection of Rich Model Steganalysis Features Based on Decision Rough Set α-Positive Region Reduction", *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.
[14] P. Pieta and T. Szmuc, "Applications Of rough sets in big data analysis: An overview", *International Journal of Applied Mathematics and Computer Science*, vol. 31, no. 4, pp. 659–683, 2021.
[15] M. K. Hasan and A. Ahmed, "Arabic Text Detection Using Rough Set Theory: Designing a Novel Approach", *IEEE Access*, vol. 11, 2023.
[16] F. R. Shareef, "Arabic Text Steganography based on Arabic Astrology", *Journal of Information Hiding and Multimedia Signal Processing*, vol. 14, no. 3, 2023.
[17] A. Gutub, "High Capacity Steganography Tool for Arabic Text Using 'Kashida' ", *The ISC Int'l J Inf Secur*, vol. 2, no. 3, pp. 107–118, 2010.
[18] F. R. Shareef, "A novel crypto technique based cipher-text shifting", *Egyptian Informatics Journal*, vol. 21, no. 2, pp. 83–90, 2020.