JOURNAL OF BASIC SCIENCE AND ENGINEERING

# An intelligent model of symmetric key encryption and decryption applied to south Indian language: Kannada

**Vivekananda[1], K C Ravishankar[2]**

[1]Research Scholar, Department of Computer Science and Engineering, Government Engineering College, Hassan, Research Center Affiliated to Visvesvaraya Technological University, Belagavi, Karnataka, India.
Assistant Professor, Department of Artificial Intelligence and Data Science, Navkis College of Engineering, Hassan, Karnataka, India.
[2] Professor and Head, Department of Computer Science and Engineering, Government Engineering College, Mosale Hosahalli, Hassan, Karnataka, India.

DOI: https://doie.org/10.0824/Jbse.2024535627

## Abstract

Alphasyllabic language text is represented in computers using Unicode format. The ASCII character text processing cryptographic models require converting the Unicode text into ASCII byte streams. Also, most cryptographic models have encryption keys supplied externally. The same key encrypts all the plain text symbols. These methods may be susceptible to frequency analysis to break encryption. The present article proposes a language processing-based symmetric cryptographic model. This model is capable of processing Unicode text without intermediate byte conversions. The encryption key generation is driven by extracting the features of the letters in plain text. This article proposes an attribute annotation process with the help of a syntax-directed translation scheme. The extracted and annotated features would contribute to generating an
encryption key set. The proposed model uses a context-sensitive encryption key. This paper explores the capabilities of alphasyllabic language processing to guide cryptographic applications. The proposed algorithm takes Kannada textual data for processing.

***Keywords:*** ***Context-Dependent Encryption Key, Text feature-based cryptography, Indic Language Processing, Syllabic Annotation of Kannada Letters, Context Free Grammar***.

## 1 Introduction

Exploration of potential applications of natural language processing in computers is still underway. The volume of linguistic knowledge utilised in computer applications is still significantly less than the vast accumulated linguistic knowledge in human history. Natural languages have subtle and hidden capabilities suitable for modern information-processing applications. Some capabilities can provide alternate solutions to existing problems or evolve as orthogonal solutions. The present work attempts to use the characteristics of the syllables of the alpha-syllabic language in the cryptography domain.
Cryptography is one of the oldest technologies relevant today. Cryptography is also evolving by absorbing new possibilities in response to conquering new challenges. The lifestyles of Common men are the driving forces for the contemporary impactful technological growth. This growth is no longer limited to some serious experiments bound to laboratories. In the present era, the measure of success for a technology is the level of acceptance by the people. Therefore, acceptance by people is as important as being innovative. These facts now influence the

technologies to grow beyond the linguistic barriers to reach common men in common men's Languages. A large population of the world speaks English. it is the lingua franca of the era. English is the language used for technological advancements, accumulation, and dissemination of knowledge. The worldwide acceptance and popularity of English encouraged many cryptographic techniques to evolve. Historically, ancient civilizations like India, Chinese, Greek, Roman, and others used cryptography independently to hide information. Cryptography sprouted in languages other than English. It gained momentum through the contributions of Arabic (Islam civilization) and matured in English. English is also predominantly used in computer and communication technologies. Here, there is a need for data security and authentication to protect from unauthorized access to the data. The computer and communication technologies are standardized based on the ASCII character set. Resulting in the development of cryptography around the ASCII set data. Also, the ancient cryptographic techniques are very trivial compared to computer-based techniques. The non-English character set cryptographic methods gradually lost their focus. The efforts in standardization of Unicode technology made it easy to use non-ASCII languages in computers,
communications, and other related fields. Increased usage of such languages offers new possibilities in processing, securing, and application developments in information processing and communication domains. Revoking cryptography for non-English languages is now prompting a new beginning.
According to Noor R. Al-Kazaz [1], the evolution of cryptographic techniques started with manual methods and shifted to electromechanical, then to fully electronic, and finally to computer-based approaches. Substitution, Transposition, and Play-fair are the classical encryption techniques used before the arrival of computer-based methods. The invention of computers helped to develop encryption methods that are hard to implement manually. Key-based cryptographic methods marked the beginning of modern cryptography and gained more popularity over classical approaches because of their robustness. The classical methods are still sublime, making them the basis for many modern cryptographic techniques. The key-based encryptions are of three types. Namely, symmetric key encryption, asymmetric key encryption, and hash function methods [2]. According to Al-qdah and Hui [3], asymmetric methods require high mathematical manipulations. Bit rotation, bit shifting, and logical operations on bits are simple and good encryption techniques. The proposed work introduces a cryptographic model where syllabic attributes extracted from plain text assist in generating cipher text. A context-free grammar guides the attribute extraction process. The proposed method conducts experiments on Kannada text. This approach is suitable for any alpha-syllabic language in general.

## 2 Literature Survey

Wagh J D [4] worked on encrypting the Indian regional language. The proposed method in the paper is very trivial. The algorithm begins with transliteration of the given regional language plain text to English. Then, the text is encrypted using the substitution method, and reversal will do the decryption. Bhadri Raju MSVS et.al.[5] presents a method to encrypt the Indic scripts considering the Telugu language (the Indian Language used in the states of Andra Pradesh and Telangana). The security model presented starts with segmenting the plain text into syllables. The next step is
to convert a syllable byte stream into a code point byte stream. Then, perform the conversion of a code point byte stream into a bit stream representation. Finally, bit steam will get encrypted. The encryption key stream generator makes use of a random number generator function. Transformation techniques are applied to generate the cipher text. The decryption

follows the same steps in reverse. Frequency distribution attack reveals around 25% to 50% of plain text. E C Papakitsos [6] proposes the idea of adapting ancient syllabaries as substituting text to encrypt the plain text. The developed application works on the Greek text. The method generates a Latin alphabet-based encryption key grid with 148 substitution characters. During substitution, the key grid substitutes single letters in some cases and syllabic pairs in other cases. This mechanism protects the cipher from cryptanalysis through frequency distribution. The same key grid dictates the decryption process. Bushra and Hyder [2] identified Ibn Dunaynir as the first cryptologist to describe arithmetical cipher. Arabian cryptologist Ibn Dunaynir's arithmetical cipher generation model works with each plain text letter substituted with an equivalent decimally weighted numerical value. The numerical symbol assigned for a letter is unchanged throughout the cryptogram. In Dunaynir's method, numerical processing involves making the representative numbers of times great in value where there is no scope for a secret key.
A literature survey shows few works published in the cryptography domain for non-English languages, including Japanese efforts during World War. Though many other language works are studied, the Arabic language is in the lead since the cultural roots of Islamic civilization nourished the cryptanalysis. In the Indian subcontinent context, few publications have unconventional approaches, such as using the Fibonacci series to select the substitution cipher letter [7]. Another work published by Prasad et al.[8] presented the steganographic method for the Telugu language. The Steganography approach selects the consonant character modifiers and the punctuation marks to hide the information. Linguistic properties of the Telugu language letters decide the embedding position. This work attempted to process the text for adopting linguistic properties for security purposes but was limited only to conjuncts.

## 3 Introduction to the Kannada Language

Kannada is a member of the Dravidian language family, spoken predominantly in Karnataka, a state of in Southern India. According to data from Ethnologue, Kannada speaking population across the world is more than 58.69 million. The literary activities in Kannada began around the first century B.C. to the third century A.D. in the form of folk songs. According to an estimation, the written form of literature in Kannada began around 1500 years ago. But, the first available literary work is estimated to be from 850 A.D [9]. Kannada is an abugida or alpha-syllabic language with 'Akshara' or 'orthographic syllable' as the basic unit. Each Akshara has a consonantal core. Depending on the number of consonant symbols present, we categorise Aksharas into three types. The first set is a set of independent vowels (Swara) having no consonant in the consonantal core. There are two subgroups in the set of Swara (Vowels). They are Hraswa Swara (Short Syllable/ Vowel) and Deerga Swara (Long Syllable/Vowel), depending on the time taken to pronounce. Anuswara and Visrga are symbols of the Swara set, known as Yogavahas (part vowel, part consonant). Dependent vowel signs (Matras) represent one-to-one correspondence to the vowels. Matras do not have an independent appearance in the scripts. They are augmented to a single consonant or cluster of consonants to form an Akshara. A set of consonants (Vyanjana) have a single consonant sound augmented with an inherent vowel. Vargeeya Vyanjanas and Avargeeya Vyanjanas are two types of consonants in Kannada. Vargeeya Vynajana set has the subclass Anu-nasika (Nasal) symbols. Finally, consonant conjuncts (Samyuktakshras) are part of the Kannada language. Consonant conjuncts are a cluster of consonants constructed with a series of dead consonants (ending with a halanth or without an inherently augmented vowel denoted by consonant Cd) followed by a consonant and then by an optional dependent vowel.

## 4 Methodology

The cryptographic model proposed in this paper uses a syllabic properties-based encryption key generation approach. Arabian cryptologist Ibn Dunaynir's arithmetical cipher generation model [2] is the inspiration for the proposed symmetric cryptographic model. The proposed cryptographic model differs from Dunaynir's keyless encryption in the way this approach uses a symmetric key. In the proposed model, the plain text and the corresponding context-dependent key are operated to generate a cipher text. For every plain-text character, using the attributes and the consonant core of a look-ahead letter produces a key to encrypt. This encryption-key generation process characterises the context-adoptive key generation. The context-sensitive key generated for each symbol nourishes the idea of a context-sensitive cryptographic model. To the best of our knowledge, it is the first attempt to consider text features to generate the encryption keys. The proposed cryptographic model is shown pictorially in Fig.1 and Fig.2.

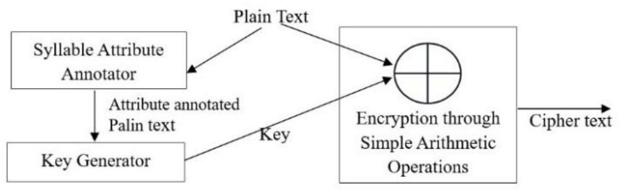

Plain Text
Syllable Attribute Annotator
Attribute annotated Palin text
Key Generator
Key
Encryption through Simple Arithmetic Operations
Cipher text

Fig. 1: Encryption Process

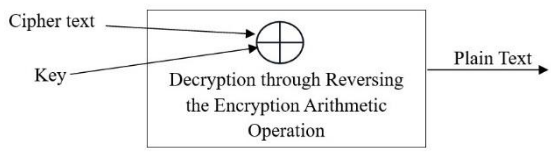

Cipher text
Key
Decryption through Reversing the Encryption Arithmetic Operation
Plain Text

Fig. 2: Decryption Process

## 4.1 Plain Text and Kannada Letter Attributes

Kannada Unicode text is the input for experiment conduction. The Kannada Language has an alphabet with 15 vowels and 34 consonants. There are 49 symbols in total. Table1 illustrates the Kannada letters and corresponding attributes.

### 4.1.1 Grammar for Syllable Attribute Annotator

The Kannada syllable structure extraction process uses the intrinsic code point structure of the letters. A novel Syntax-Directed Translation Scheme with semantic actions augments the attributes. The set production rules of designed grammar are as follows:

(Rule 1) consonant → C H C H C m y | C H C m y | C m y | CH
(Rule 2) vowel→ V y
(Rule 3) m → M | ϵ
(Rule 4) y → Y | ϵ

Table 1: Overview of Kannada letters and Attributes

| Alphabet | | Code point Representation |
|---|---|---|
| Short Vowels | ಅ ಇ ಉ ಎ ಒ | V |
| Long Vowels | ಆ ಈ ಊ ಏ ಐ ಓ ಔ | |
| Yogavahas | ಅಂ ಅಃ | |

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

| | | |
|---|---|---|
| Consonants | ಕ ಖ ಗ ಘ ಙ ಚ ಛ ಜ ಝ ಞ ಟ ಠ ಡ ಢ ಣ ತ ಥ ದ ಧ ನ ಪ ಫ ಬ ಭ ಮ ಯ ರ ಲ ವ ಶ ಷ ಸ ಹ ಳ | C |
| Dependent vowels/ Matras (Modifiers) | | |
| Dependent vowel sign corresponding to Vowels | ◌ಾ ◌ಿ ◌ೀ ◌ು ◌ೂ ◌ೃ ◌ೆ ◌ೇ ◌ೈ ◌ೊ ◌ೋ ◌ೌ | M |
| Dependent vowel sign corresponding to Yogavahas | ◌ಂ ◌ಃ | Y |
| Consonant Conjuncts/Clusters (Otthakshara/Samyuktakshra) | | |
| Example for level one Consonant Conjunct | ಕ್ಕ (k+k+a) | CHC |
| Example for level Two Consonant Conjunct | ಸ್ತ್ರ (s+th+r+a)_ | CHCHC |
| Consonants/Consonant Conjuncts with Dependent Vowels (Gunithakshara) | | |
| Consonant Conjunct with Dependent Vowel | ಕ್ಕು (k+k+u) | CHCM |
| | ಸ್ತ್ರೀ (s+th+r+i) | CHCHCM |
| Consonant Conjunct with Dependent Vowel and Yogavahas | ಕ್ಕುಂ (k+k+u+m) | CHCMY |
| | ಸ್ತ್ರೀಂ (s+th+r+i+m) | CHCHCMY |

### 4.1.2 Key Generation Algorithm

Algorithm 1, presented in this section, uses the above production rules to remove non-Kannada text symbols and generates the annotated plain text. An attribute bit vector records the annotated properties of the plain text. Positional values of attribute bit vectors shown in Table 2 contribute to computing the corresponding numerical values of attribute combinations. Each bit of the attribute vector represents the particular property of a letter. Setting a bit indicates the presence, and resetting the bit informs the absence of a property. Building the bit vector for a plain text symbol is summarized in Equation (1). The key generated for each plain text character depends on the single look-ahead letter that appears in the immediate next position. If the letter next to the symbol in the given text changes, it results in a different encryption key. Which means to have multiple encryption keys for the same plain text letter depending on the look-head symbols. Also, the key varies if the look-ahead letter has the same consonant core with different attribute sets or the same attribute set with other consonants. The key generation algorithm in the section returns KEY VECTOR having a collection of ordered pairs (key, length) as elements. The number of elements in the KEY VECTOR is the same as that of plain text. Equation (2) computes the entries into KEY VECTOR.

Table 2: Attribute bit vector composition

| Bit Position | Property Name | Numeric Value |
|---|---|---|
| 0 | Visarga | 1 |
| 1 | Anuswara | 2 |
| 2 | Long Dependent vowel Modifiers | 4 |
| 3 | Long Vowels | 8 |

| | | |
|---|---|---|
| 4 | Short Dependent vowel Modifiers | 16 |
| 6 | Short vowels | 32 |
| 7 | Consonants | 64 |
| 8 | Single Consonant Conjunct cluster | 128 |
| 9 | Double Consonant Conjunct cluster | 256 |
| 10 | Half Consonant Conjunct cluster (Ardhakshara) | 512 |

$$Bit\_Vector[AttributeBitPosition] = \begin{cases} 1 & if\ property\ exists \\ 0 & otherwise \end{cases}$$ ……………….. Eq. (1)

$$\text{KeyVector} = \left\{ (\text{key}, \text{length}) \mid \begin{array}{l} \text{key} = \text{valueofBit}_{\text{V}}\text{ector} + \text{valueofnextletters}'\text{consonantcore} \\ \text{length} = \text{ number of code points in current letter} \end{array} \right.$$ ………… Eq. (2)

**Algorithm 1:** Single Letter Lookahead based Context Sensitive Symmetric Key Generator
**Input:** Plain text
**Output:** A Pair has a Key for encryption and a sequence of points representing the letter that gets enciphered.
**Method:** Repeat the following steps until all the symbols in plain text are processed.

```
Step 1: Read the Kannada Plain Text character in Unicode format.
Step 2: Tokenize to identify the Syllables
Step 3: Annotate Attributes for the identified Syllables and Store them in numerical form using the Attribute bit Vector.
Step 4: Using the Syllable Attributes and Code Length of the letter generate a (key, length) pair and store it in KEY_VECTOR. The first entry in the KEY_VECTOR has the key computed using attributes of the second letter and the count (length) of code points to represent the first letter. The other entries follow the same order. The last entry in the KEY_VECTOR has the key generated using the properties of the first letter and the count (length) ) of code points to represent the last letter
```

### 4.1.3 Encryption algorithm

The encryption algorithm described in this section performs an addition operation between the code points of the Kannada letter and the key. For every code point used to represent a letter in plain text, the encryption process shifts the code point value to a different language code space. This process brings in a new substitution symbol (a cipher symbol) generated for each code point representing a plain text character. Hence, the generated cipher symbols are more than the original text letters. But it is the same as the number of code points in the plain text. The Equation (3) is responsible for code point shift.

$$Cipher\ symbol[j] = \{c \mid \forall\ palin\ text\ letters\ (\forall\ Code\ points\ \in current\ letter + i^{th}\ \text{key})\}$$ …Eq. (3)

**Algorithm 2:** Cipher text generation algorithm for Kannada Plain Text
**Input:** Kannada Plain text and (Key, Length) pair

```
Output: Encrypted Kannada Text.
Method: Repeat the following steps for each letter in the plain text
        Step 1: Read a Character from the remaining input.
        Step 2: Collect the sequence of code points to represent the letter
        Step 3: Select an individual code point and add it with the key to generate the
    cipher symbol.
```

### 4.1.4 Decryption algorithm

During the decryption process of the proposed algorithm, a subtraction operation reverses the encryption. The decryption process involves groping length (available along with the key) the number of cipher symbols and subtracting the key from each to regain code points of letters. Grouping recovered code points renders the original plain text letter. The process is formalized in the Equation (4) and in Equation (5). The algorithm 3 demonstrates the steps involved in the process.

$$ncipher = \{c | c_i \in cipher\ text, CurrentPosition \leq i \geq (CurrentPosition + length)\} \quad \text{……… Eq. (4)}$$

$$recoveredLetter = \{r | (c - i^{th}\ key)\ \ \forall c \in nciphder\} \quad \text{………..Eq. (5)}$$

```
Algorithm 3: Reconstructing Kannada plain Text from Cipher text
Input: Cipher text and (Key, Length) pair
Output: Decrypted Kannada Text.
Method: Repeat the following steps for symbols in the cipher text
    Step 1: Select the chunk of symbols from cipher text by referring to KEY_VECTORs'
    length component and also select the corresponding key present in the pair.
    Step 2: Repeat the following steps on each cipher symbol present in the chunk
            Step 2.1: inverse the arithmetic operation performed in encryption to get
            numerical equivalent (ordinal) representing recovered plain text.
            Step 2.2: Add recovered plain text to the set of recovered symbols.
```

## 5 Results and Analysis

The world's first online magazine and the website in Kannada http://vishvakannada.com/ is referred to as a text source for conducting the experiments. Different articles in different genres written by various authors and topics are considered individually and in combinations.

The group of cipher symbols generated for any character in the plain text is context-dependent. For example, consider Table 3 showing results of running algorithms on input with Kannada words having the same first letter followed by different second letters and attributes and consonant core combinations. The first column is the letter enciphered. The second column enlists the look-ahead character dictating the context for the key. The third column is the list of generated keys, the next column shows the different numeric codes generated, and the last column lists the corresponding cipher symbols.

This paragraph compares the behavior of the proposed method with the Vigenere poly-alphabetic cipher. The major unlikeliness is the presence of an external key. The proposed method generates a plain-text-dependent key for encryption. Another difference is that the character code space of plain and cipher text is not identical in the proposed technique. Whereas in Vigenere's approach, both fall in the same code space, this may reveal the language of the text encrypted. The proposed method protects the cipher text from known-text-attacks or dictionary attacks by safeguarding it from early language identification. Vigenere's approach performs a modulo addition operation to generate the cipher, whereas the proposed method carries out addition without a modulo operation. Table 4 shows the cipher text generated by the proposed method. One more difference worth mentioning is the length of the cipher text. In Vigenere's approach, the length of text in the plain and the cipher texts. However, in the proposed method, cipher text length is always more than plain text. It will help the algorithm to defeat frequency analysis attacks, overcoming the limitations of Vigenere's methods.

Table 3: Example for context sensitive Key generation and resulting cipher Symbol

| INPUT TEXT | ಅಮ್ಮಅಪ್ಪಅಣ್ಣಅನ್ನಅಕ್ಕಅನ್ನುಅರಅತ್ತೆಅಡ್ಡಅದುಅವಅಸೆಅವ್ವಅಜ್ಜ ಅಜ್ಜಿಅಪ್ಪುಅತ್ಯಅರ್ಧಅರ್ಥಅಲ್ಪಅಮೃ | | | |
|---|---|---|---|---|
| Encrypted Letter | Next Letter | Key Generated | Cipher symbols' numeric code | Cipher Symbol for ಅ |
| ಅ | ಮ್ಮ | 3374 | 6579 | ᦳ |
| ಅ | ಪ್ಪ | 3370 | 6575 | □ |
| ಅ | ಣ್ಣ | 3363 | 6568 | ᦨ |
| ಅ | ನ್ನ | 3368 | 6573 | □ |
| ಅ | ಕ್ಕ | 3349 | 6554 | ᦚ |
| ಅ | ನ್ನು | 3384 | 6589 | ᦽ |
| ಅ | ರ | 3312 | 6517 | □ |
| ಅ | ತ್ತೆ | 3380 | 6585 | ᦹ |
| ಅ | ಡ್ಡ | 3361 | 6566 | ᦦ |
| ಅ | ದು | 3318 | 6523 | □ |
| ಅ | ವ | 3317 | 6522 | □ |
| ಅ | ಸೆ | 3336 | 6541 | ᦍ |
| ಅ | ವ್ವ | 3381 | 6586 | ᦺ |
| ಅ | ಜ್ಜ | 3356 | 6561 | ᦡ |
| ಅ | ಜ್ಜಿ | 3372 | 6577 | ᦱ |
| ಅ | ಪ್ಪು | 3386 | 6591 | ᦿ |
| ಅ | ತ್ಯ | 3364 | 6569 | ᦩ |
| ಅ | ರ್ಧ | 3376 | 6581 | ᦵ |
| ಅ | ರ್ಥ | 3376 | 6581 | ᦵ |
| ಅ | ಲ್ಪ | 3380 | 6585 | ᦹ |
| ಅ | ಮೃ | 3326 | 6531 | ᦃ |
| Cipher Text | ᦳᦰᦀᦰ□□ᦀ□ᦨ□ᦀ□□□ᦀ□ᦚ◌□ᦀ◌□ᦽ□ᦀ□ᦻ□ᦇᦹ□ᦀ□ᦼᦦ□ᦀ□□□ᦻ□ᦎᦍ ᦀᦼᦺᦎᦀᦎᦡ□ᦀ□ᦱ□ᦀ□ᦾᦿ□ᦀ□ᦻᦩ□ᦀᦴᦵᦇᦀ□ᦵᦇᦀ□ᦹᦒᦀ□□ᦃᦰᦸ | | | |

Table 4: Sample Input/Output

| **Result of Encryption with Addition operator** | |
|---|---|
| **Plain Text** | **Cipher text Generated** |

| | |
|---|---|
| ಜೋಗದ ಸಿರಿ ಬೆಳಕಿನಲ್ಲಿ<br>ತುಂಗೆಯ ತೆನೆ ಬಳುಕಿನಲ್ಲಿ,<br>ಸಹ್ಯಾದ್ರಿಯ ಲೋಹದದಿರ<br>ಉತ್ತುಂಗದ ನಿಲುಕಿನಲ್ಲಿ<br>ನಿತ್ಯ ಹರಿದ್ವರ್ಣವನದ<br>ತೇಗ ಗಂಧ ತರುಗಳಲ್ಲಿ<br>ನಿತ್ಯೋತ್ಸವ, ತಾಯಿ ನಿತ್ಯೋತ್ಸವ<br>ನಿನಗೆ ನಿತ್ಯೋತ್ಸವ, ತಾಯಿ ನಿತ್ಯೋತ್ಸವ | ဖပ□□ဧရှု□ချလဗိဘ□ခွဗိုဂွဋဂွဂဒွဂွပဂြဂရှဖွ္ဃပဗ□ဘဏွ<br>□ခွဗိုသွဏှသွင်ၡ္ဗိုဗိုၳၡ္ဗိုၡ္ဏယျလ□ပ်ဂွမွလဖွဏွ□ပရှု□ဒ်□<br>ဘပ□ဝသွဇွဏှဏွ□ခွဗိုသွဏှသွင်□ဗိုကြဒွဂွဗဖြၡ္ဗိုဖြ၇ၢၡ္ဗိုပ်ဂှဘ<br>ကြတတ□ဝπ□ပဒွဒ်ဂှဘၡ္ဗိုသွဏှသွင်ဝဗို၆ၣ္ဂဗိုမှဂှ□ကြရဈ<br>ခွင်ဝဗို၆ၣ္ဂဗိုမှဂှ□□ဖွခွဗွဗွဃ်ဝဗို၆ၣ္ဂဗိုမှဂှ□ကြရဈခွင်ဝဗို<br>၆ၣ္ဂဗိုမှဂှ□ဏ |
| **Result of Decryption with Subtraction operator** | |
| ಜೋಗದಸಿರಿಬೆಳಕಿನಲ್ಲಿತುಂಗೆಯತೆನೆಬಳುಕಿನಲ್ಲಿಸಹ್ಯಾದ್ರಿಯಲೋಹದದಿರಉತ್ತುಂಗದನಿಲುಕಿನಲ್ಲಿನಿತ್ಯಹರಿದ್ವರ್ಣವನದತೇಗಗಂಧತರುಗಳಲ್ಲಿನಿತ್ಯೋತ್ಸವತಾಯಿನಿತ್ಯೋತ್ಸವನಿನಗೆನಿತ್ಯೋತ್ಸವತಾಯಿನಿತ್ಯೋತ್ಸವ | |

## 5.1 Analysis

Cipher text and the corresponding original text have undergone frequency analysis text. The results demonstrate that the frequency analysis fails to extract the original text. Fig. 3 illustrates an example. Fig. 3a shows that the unique code points present in cipher text are more than that of plain text. The example shows that 43 more distinct symbols appear in cipher text compared to the original text. The 1-,2-, and 3-gram pairing of original and cipher text is carried out. Fig.3b presents uni-gram, bi-gram, and tri-grams paring of plain and cipher texts. Since it is the case of known plain text, pairing the corresponding symbols at a position should break the cipher. This algorithm behaves as expected, with few outliers. The multiple instances of cipher symbols can correspond to one plain text symbol and vice-versa. The one-to-one correlation of the cipher and plain-text characters does not help to reveal the original text. Examples are shown in Fig.3c and Fig.4d. The Advanced Encryption Standard (AES) algorithm is the most popular and secure symmetric algorithm. So, the data tabulated in Table 5 and Table 6 are used to captures the comparison between the performance of the proposed algorithm and AES algorithm. The AES model used for comparison refers to the code available at [10] with basic ECB (Electronic Code Block) mode and block size set to 16 bytes. Kannada letter structure is in syllabic style. Hence, a sequence of code points (Unicode Symbols) represents the single letter.

Table 5 shows the number of code points in text before and after encryption. Also, it shows the number of distinct symbols (Unique code points) present in plain text and cipher text generated by the proposed model and AES model.

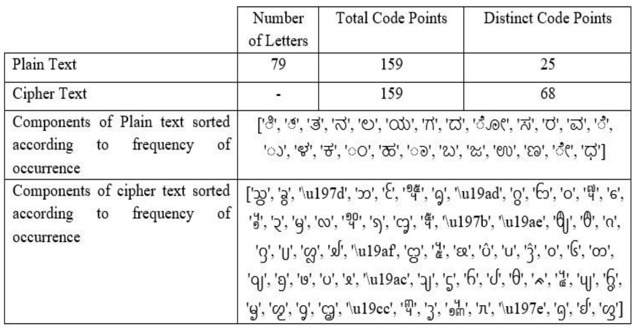
Table 5: Number of code points to represent plain text and Code points in cipher

| File Size in KBs | Number of Kannada Letters | Plain text Code Points | | Proposed algorithm Code Points | | AES algorithm code points | |
|---|---|---|---|---|---|---|---|
| | | Total | Unique | Total | Unique | Total | Unique |
| 5 | 79 | 159 | 25 | 159 | 68 | 960 | 16 |
| 8 | 1722 | 3332 | 54 | 3332 | 224 | 20000 | 16 |
| 19 | 3880 | 7816 | 54 | 7816 | 327 | 46912 | 16 |
| 21 | 4341 | 8542 | 56 | 8542 | 275 | 51264 | 16 |
| 25 | 5395 | 10141 | 55 | 10141 | 294 | 60864 | 16 |
| 27 | 5956 | 11662 | 56 | 11662 | 298 | 69984 | 16 |
| 29 | 6561 | 12387 | 56 | 12387 | 242 | 74336 | 16 |

Vol. 21, No. 1, (2024)
ISSN: 1005-0930

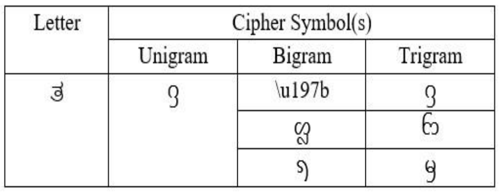
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 49 | 11024 | 21061 | 60 | 21061 | 354 | 126368 | 16 |
| 136 | 30315 | 58125 | 60 | 58125 | 443 | 348768 | 16 |
| 147 | 32816 | 63067 | 60 | 63067 | 456 | 378432 | 16 |
| 167 | 37157 | 71609 | 61 | 71609 | 467 | 429664 | 16 |
| 306 | 67472 | 130924 | 61 | 130924 | 526 | 785568 | 16 |

The comparative analysis graph in Fig.4a shows the difference between the unique code points of plain text and that of AES cipher text. The flat curve of AES and the rising curve of the proposed algorithm infer that the proposed algorithm has a varied and increasing number of diverse cipher symbols. The number of keys that get generated and the sensitivity towards context is the reason for this behavior of curves. The above reason also provisions the encryption algorithm to represent plain text symbols with different cipher symbols based on the context. The comparative analysis graph in Fig.4a shows the difference between the unique code points of plain text and that of AES cipher text. The flat curve of AES and the rising curve of the proposed algorithm infer that the proposed algorithm has a varied and increasing number of diverse cipher symbols. Table 6 summarizes the time consumed by the proposed algorithm and the AES algorithms for key generation, encryption, and decryption operations.



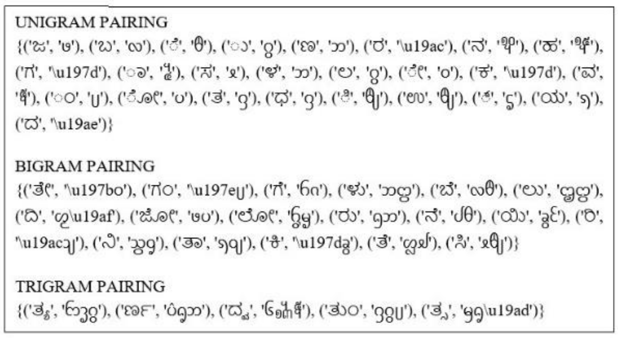
| | Number of Letters | Total Code Points | Distinct Code Points |
|---|---|---|---|
| Plain Text | 79 | 159 | 25 |
| Cipher Text | - | 159 | 68 |
| Components of Plain text sorted according to frequency of occurrence | ['ಿ', 'ೆ', 'ತ', 'ನ', 'ಲ', 'ಯ', 'ಗ', 'ದ', 'ೋ', 'ಸ', 'ರ', 'ವ', 'ೀ', 'ು', 'ಳ', 'ಕ', 'ಂ', 'ಹ', 'ಾ', 'ಬ', 'ಜ', 'ಉ', 'ಣ', 'ೇ', 'ಧ'] | | |
| Components of cipher text sorted according to frequency of occurrence | ['[illegible]', '[illegible]', '\u197d', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '\u19ad', '[illegible]', '[illegible]', 'o', '[illegible]', '6', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '\u197b', '\u19ae', '[illegible]', 'θ', 'ɑ', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '\u19af', '[illegible]', '[illegible]', '[illegible]', 'ô', 'u', '[illegible]', 'o', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '\u19ac', '[illegible]', '[illegible]', '[illegible]', '[illegible]', 'θ', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '[illegible]', '\u19cc', '[illegible]', '[illegible]', '[illegible]', 'π', '\u197e', '[illegible]', '[illegible]', '[illegible]'] | | |

3(a) Test Case for Frequency Analysis.



| Letter | Cipher Symbol(s) | | |
|---|---|---|---|
| | Unigram | Bigram | Trigram |
| ತ | [illegible] | \u197b | [illegible] |
| | | [illegible] | [illegible] |
| | | [illegible] | [illegible] |

3(c) Different cipher symbols for one plain text symbol

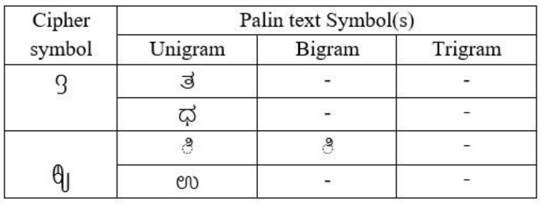


UNIGRAM PAIRING
{('ಜ', '[illegible]'), ('ಬ', '[illegible]'), ('ೀ', 'θ'), ('ು', '[illegible]'), ('ಣ', '[illegible]'), ('ರ', '\u19ac'), ('ನ', '[illegible]'), ('ಹ', '[illegible]'), ('ಗ', '\u197d'), ('ಾ', '[illegible]'), ('ಸ', '[illegible]'), ('ಳ', '[illegible]'), ('ಲ', '[illegible]'), ('ೇ', 'o'), ('ಕ', '\u197d'), ('ವ', '[illegible]'), ('ಂ', '[illegible]'), ('ೋ', '[illegible]'), ('ತ', '[illegible]'), ('ಧ', '[illegible]'), ('ಿ', '[illegible]'), ('ಉ', '[illegible]'), ('ೆ', '[illegible]'), ('ಯ', '[illegible]'), ('ದ', '\u19ae')}

BIGRAM PAIRING
{('ತೇ', '\u197bo'), ('ಗಂ', '\u197e[illegible]'), ('ಗೆ', '[illegible]'), ('ಳು', '[illegible]'), ('ಬೆ', '[illegible]'), ('ಲು', '[illegible]'), ('ದಿ', '[illegible]\u19af'), ('ಜೋ', '[illegible]'), ('ಲೋ', '[illegible]'), ('ರು', '[illegible]'), ('ನೆ', '[illegible]'), ('ಯಿ', '[illegible]'), ('ರಿ', '\u19ac[illegible]'), ('ಲಿ', '[illegible]'), ('ತಾ', '[illegible]'), ('ಕಿ', '\u197d[illegible]'), ('ತೆ', '[illegible]'), ('ಸಿ', '[illegible]')}

TRIGRAM PAIRING
{('ತ್ಯ', '[illegible]'), ('ಣ್ಣ', '[illegible]'), ('ದ್ವ', '[illegible]'), ('ತುಂ', '[illegible]'), ('ತ್ನ', '[illegible]\u19ad')}

3(b) Uni-gram, Bi-gram and Tri-gram pairings.



| Cipher symbol | Palin text Symbol(s) | | |
|---|---|---|---|
| | Unigram | Bigram | Trigram |
| [illegible] | ತ | - | - |
| | ಧ | - | - |
| [illegible] | ಿ | ಿ | - |
| | ಉ | - | - |

3(d) Same cipher symbols for more than one plain text symbol

Fig. 3: Frequency Analysis Scenario

Table 6: Time taken for the operation measured in Mili Seconds

| File Size in KBs | Time taken for the operation measured in Mili Seconds | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Proposed Algorithm | | | | AES Algorithm | | | |
| | Key Generation* | Encryption | Decryption | Number of Keys[2] | Key Generatio[3] | Encryption | Decryption | Number of Keys[4] |
| 5 | 0.0451 | 0.0540 | 0.0562 | 79 | 0.0031 | 0.5870 | 0.0806 | 1 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 8 | 1.1535 | 1.6048 | 1.5996 | 1722 | 0.0026 | 0.7396 | 0.0877 | 1 |
| 19 | 2.6934 | 3.5240 | 3.4063 | 3880 | 0.0023 | 0.6348 | 0.0960 | 1 |
| 21 | 3.1435 | 3.8566 | 4.0199 | 4341 | 0.0026 | 0.6751 | 0.1161 | 1 |
| 25 | 3.8792 | 4.5628 | 4.8760 | 5395 | 0.0022 | 0.7602 | 0.1156 | 1 |
| 27 | 5.1929 | 5.2727 | 6.0000 | 5956 | 0.0026 | 0.6503 | 0.1477 | 1 |
| 29 | 5.0819 | 6.4441 | 6.7320 | 6561 | 0.0023 | 0.9304 | 0.1863 | 1 |
| 49 | 17.454 | 12.361 | 14.935 | 11024 | 0.0023 | 0.8850 | 0.1956 | 1 |
| 136 | 50.167 | 54.742 | 79.390 | 30315 | 0.0032 | 1.2415 | 0.4411 | 1 |
| 147 | 61.646 | 61.796 | 102.62 | 32816 | 0.0035 | 1.2180 | 0.4175 | 1 |
| 167 | 69.171 | 76.066 | 115.54 | 37157 | 0.0021 | 1.1488 | 0.4058 | 1 |
| 306 | 194.41 | 212.14 | 341.17 | 67472 | 0.0022 | 2.2496 | 1.0446 | 1 |

1.size of key set is equal to the size of plain text 2. Each key of size 12 bits 3. Single key 4. Single key of size 128 bits

Fig.4b infers that the AES algorithm has more cipher symbols than the proposed algorithm. However, the diversity in substitution is lacking, as there are a fixed number of unique code points. Also, AES converts the Unicode plain text into a group of ASCII codes. Hence, cipher text generated by applying the AES algorithm has only English alphabet symbols and numerals with fewer unique substitution symbols. The number of keys that get generated and the sensitivity towards context is the reason for this behavior of curves. The above reason also provisions the encryption algorithm to represent plain text symbols with different cipher symbols based on the context.

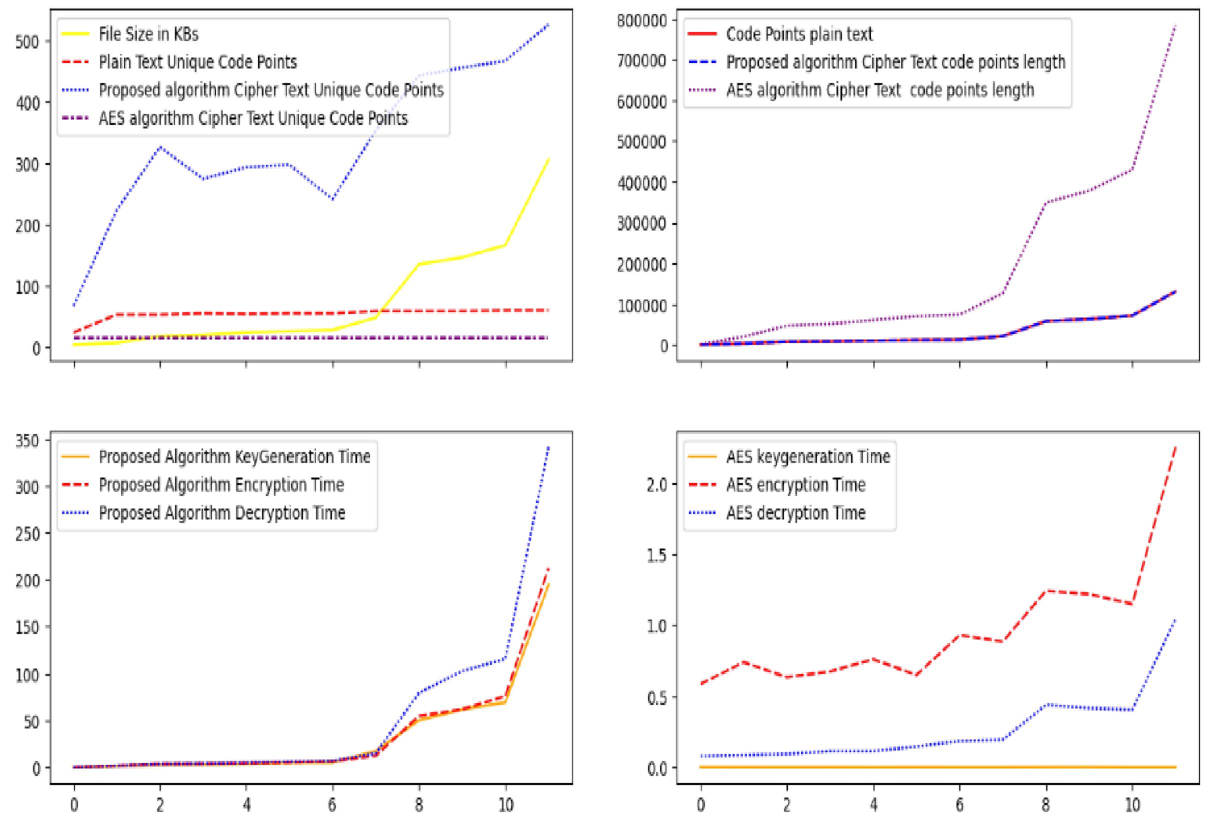


Fig. 4: Comparison between Proposed Algorithm and AES algorithm

The proposed algorithm opts for the substitution to code point symbols of the plain text rather than for an individual letter. The length of the plain text is always less than the number of code points because a group of code points forms a letter. Hence, the proposed algorithm produces the cipher text with a length equal to the number of code points representing the entire input plain text. Fig.4c and Fig.4d are the graphs showing the time for the cryptographic models to operate on the input. The time taken by the proposed algorithm is very high compared to the AES algorithm. It is expected behaviour because the algorithm has processing overhead at each step. In the AES model, the secret key is generated once and remains the same for all the plain text blocks. The proposed algorithm generates a set of keys (KEY_VECTOR) depending on the input size. The proposed algorithm encodes a single plain text letter in a context-dependent manner. To encipher the present input letter, the proposed model will be consulting KEY_VECTOR for the corresponding key and to know the length to group the code points representing a letter. Hence, encryption is not uniform across the text and needs more processing time. The advantage of this model is that the same plain text letter gets replaced with different substitution symbols in different positions depending on the letter that appears next. The proposed cryptographic model processes a single code point one at a time, a stream-based approach. The AES algorithm processes the plain text bytes blockwise to achieve comparative speed.

## 6 Conclusion

The cipher code space of the proposed algorithm varies depending on the context of the text. In the AES algorithm, the key is invariant to the context. The proposed algorithm operates in the Unicode domain. The encryption key, Plain, Cipher, and deciphered text are in Unicode format. The intermediate conversion of Unicode to ASCII is not required. The approach is also novel in annotating plain text letters and using annotated information encryption keys generated. The encryption key is usually inserted externally into the encryption models. The proposed model generates its own adoptive, context-sensitive look ahead-based encryption key. The proposed algorithm exhibits its strength by not revealing the secret text during frequency analysis using 1, 2, and 3-gram pairings. It is distinct from polyalphabetic cipher methods.
The recovered text in the proposed decryption process is identical to the original text. Compared to the AES algorithm, the proposed algorithm's performance is slow due to the reasons discussed in the previous section. There are many ways to improvise the proposed model' in several aspects. One such enhancement can be adapting mechanisms to encipher partial bytes in Unicode by skipping the known redundant bytes information. The approach presented in the paper should work for all syllabic based languages. Further extension of the proposed work can design a unified text attribute-based cryptographic model for related languages, such as the Indic family languages.

## References

[1] Noor R. Al-Kazaz (2019), Compression-based Methods for the Automatic Cryptanalysis of Classical Ciphers, Ph.D. Dissertation, Bangor University
[2] Bushra Mohamed Elamin Elnaim, Hayder Abood S.Wsmi Al-Lami (January 2017), Arab Contributions in Cryptography, Case Study: Ibn Dunaynir Effort. International Journal of Computer Science and Information Security (IJCSIS),Vol.15, No.1.
[3] Majdi Al-qdah, Lin Yi Hui (2007), Simple Encryption/Decryption Application. International Journal of Computer Science and Security, Volume (1): Issue (1).
[4] Wagh J.D.(2015), Cryptography on regional languages, Advances in Computational Research ISSN: 0975-3273 & E-ISSN: 0975-9085, Volume 7, Issue 1, 2015, pp.-268-270.

[5] Bhadri Raju MSVS, Vishnu Vardhan B et al(2009), A Noval Security Model for Indic Scripts - A Case Study on Telugu, International Journal of Computer Science and Security, (IJCSS) Volume (3): Issue (4).
[6] Evangelos C.Parkitsos (2018), A software application of ancient syllabaries to cryptography, Journal of Software Engineering & Intelligent Systems, ISSN 2518-8739, 31st December 2018, Volume 3, Issue 3.
[7] Prachi Agarwal et al (2015), Data Encryption through Fibonacci Sequence and Unicode Characters, MIT International Journal of Computer Science and Information Technology, Vol. 5, No. 2, August 2015, pp. 79-82 ISSN 2230-7621©MIT Publications
[8] Ramineni Siva Ram Prasad, Kalavathi Alla (2011) A New Approach to Telugu Text Steganography, IEEE Symposium on Wireless Technology and Applications (ISWTA), September 25-28,2011, Langkawi, Malaysia.
[9] A Hand Book of Karnataka 2015, Chief Editor Karnataka Gazetteer Department,Government of Karnataka, India.
https://gazetteer.karnataka.gov.in/info2/Special+Publications/A+Hand+Book+of+Karnataka+2015/en. Accessed on11 Oct 2023.
[10] user: 16570963 (2022), Example code to implement AES,
https://stackoverflow.com/questions/48595705/cant-encrypt-strings-withspecial-characters-with-pycrypto-aes/73432276#73432276. Accessed on 13 Oct 2023.
[11] Chris Christensen (Spring 2015), Cryptanalysis of the Vigen`ere Cipher: The Friedman Test.
https://www.nku.edu/˜christensen/1402%20Friedman%20test%202.pdf. Accessed on 11 Oct 2023.
[12] Editor,vishvakannada online magzine, http://vishvakannada.com. Accessed on July 26, 2023, at 4.11 PM
[13] Ibrahim A. Al-Kadit (1992) ORIGINS OF CRYPTOLOGY: THE ARAB CONTRIBUTIONS, Cryptologia, 16:2, 97-126, DOI: 10.1080/0161-119291866801
[14] Noor D. AL-Shakarchy et al., (2018) Cryptographic system based on Unicode. The Sixth Scientific Conference "Renewable Energy and its Applications", IOP Conf. Series: Journal of Physics: Conf. Series 1032 (2018) 012049 DOI:10.1088/1742-6596/1032/1/012049, IOP Conf. Series: Journal of Physics: Conf. Series 1032 (2018) 012049 DOI:10.1088/1742-6596/1032/1/012049.