Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

KaTaPaYa Saankhya Sutra Extended as an Encoding Scheme for Secure Communication in IoT Devices

T. SriSivaRamayya[1], D. Lalitha Bhaskari[2]

[1]*Research Scholar, Dept. of Computer Science & Systems Engineering, Andhra University, Visakhapatnam, India*
[2]*Professor, Dept. of Computer Science & Systems Engineering, Andhra University, Visakhapatnam, India*

Modern computers linked to IoT devices constantly produce vast amounts of data and information. Information security is crucial in securing this ever-generating vast volume of data. Modern encryption algorithms significantly enlarge files. Therefore, more bandwidth is needed for safe data transfer. Transforming a message into codes is the act of encoding. The study of ancient Vedic encoding techniques is applied to the contemporary period in the current work. Most of the Vedic literature is in Sanskrit through Poetry rather than narration. This Sanskrit poetry of the Vedic and its contemporary periods not only laud the Gods but also includes numerical codes. The Katapayadi technique of writing numbers in Sanskrit words was used in ancient India. Numerous ancient Indian mathematicians employed this and other similar systems. This tries to reduce the file size while simultaneously encrypting the content using this antique encryption method via Sanskrit as the medium.

In this paper "Katapayadi Sankhya Sutra" (KTPY Rule) was studied and implemented. The "Katapayadi Sankhya Sutra - Extended" (KTPY-E Rule) is the improvement suggested for this KTPY Rule. An effective encryption and decryption approach was proposed using this KTPY-E Rule. In the suggested KTPY-E technique, English text is converted to the Unicode code KTPY encoded form, and then it is stored in its Binary Format indexed by its initial character set.

Keywords: Katapayadi System, Encryption, Decryption, Avalanche Effect, ASCII, Unicode, bandwidth, Cryptanalysis.

1. Introduction

Computers are now widely employed in both our daily lives and in various areas of Science & Technology to solve challenging issues. Later with the introduction of IoT devices in every

walk of life, it has paved the way to replace the large old model computational devices with mini and micro-level IoT devices thus increasing the transmission of data and information among these IoT devices and also between other large computational devices. Consequently, a vast quantity of data was produced. Instead of finding an alternative solution, the growing need for data storage has driven to creation of increasingly complicated gear. An old method known as the Katapayadi technique could be a viable and extremely efficient solution to improve the storage efficiency of computer systems when the data is translated into Sanskrit language and also it offers data security at the same time.

Protecting personal information is an integral and important aspect of security in information. Information security is a lengthy and continuous procedure but not a quick and immediate result [6]. In the exchange of data and information among these IoT devices, information security is a challenging procedure that has to be executed with care beginning with the creation of the security policy and implementing it, as well as the assessment of the security policies' compliance [7]. Security of information must be effective and maintain a proper control of requirements that are in place to function properly and efficiently [8]. Security is dynamic and evolves with technological advancements therefore it's necessary to develop innovative methods via channels of research, defensive and offensive according to current regulations. The need for information security to be addressed in IoT communication protocols. Thus, data encryption models to be implemented on the generated data from different sources.

The process of encoding involves arranging messages in a way that is necessary for data transmission, storage, and compression/decompression. The technique of encryption used to transform data by using a computer algorithm that makes it inaccessible to everyone, except people who have access to specific information, commonly referred to as a key.

A. KaTaPaYa System:

Sanskrit is considered to be the motherly language for all the languages in the world today and its grammar is one of the finest grammars where it is suitable for its application in Modern methods such as Natural Language Processing (NLP) and Machine Learning (ML) [36]. The arrangement depicted in Part-1 of Table-1 is the original arrangement of KaTaPaYa method that conceals numbers by codifying these numbers in alphabets [14]. It was a type of basic cryptography; however, it was not designed to hide the information. It was a simple method of retaining particular numbers, values of mathematical function, numerical formulae, constants that are used in mathematical computations as well as dates associated with specific events and more in the form of poems or prose. The principles of encoding and decoding are straight and simple.

Table 1 Assignment of Values to KaTaPaYa

	1	2	3	4	5	6	7	8	9	0
Part – 1: Devanagari										
	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ
	ट	ठ	ड	ढ	ण	त	थ	द	ध	न
	प	फ	ब	भ	म					
	य	र	ल	व	श	ष	स	ह		
Part – 2: Transliteration - English										
	Ka	Kha	Ga	Gha	Nga	Cha	Cha	Ja	Jha	

	Ta	Tha	Da	Dha	Na	Ta	Tha	Da	Dha	Na
	Pa	Pha	Ba	Bha	Ma					
	Ya	Ra	La	Va	Se	Sa	Sha	Ha		

KaTaPaYa utilizes Sanskrit consonants [14] to represent the numbers on that decimal scale. So, the initial consonant of k ('Ka') is assigned the numerical value of 1. The second consonant ('kh') ('Kha') is assigned the number 2 (and so on) until reaching the consonant j ('jha') which is depicted as 9 and the number ny ('Nja') by 0.

The numbers are given the same number as the following series of ten consonants. This way, we have several alphabets that represent identical numbers. The letter '1' has been assigned to the letters Ka, Ta, Pa as well as Ya and when grouped, gives us KaTaPaYa-adi. The word ‘adi’ refers to various rest of the things. Adi in Sanskrit has dual meanings. One signifies the initial or the first to be listed and the other signifies the meaning of “rest of the things”.

B. KaTaPaYa Number System

The process of encoding is straightforward. A number is assigned to each consonant in Sanskrit. Because of this, the method is as straightforward as writing it from right to left. Let us consider “Jaya”. Here, the letter ‘ya’ is given the number 1, while ‘ja’ is given the number 8 referring to KaTaPaYa Table. When read left to right, it becomes 81. Here is an additional instance, the word "2010" is written as "nata-nara." Take a look at the characters that make up the number 1. Its name, Katapayadi, comes from the letters ka, ta, pa, and ya. Vowels had a value of 0, while vowels after consonants had no significance. The numerical value of the last character was applied to compound letters. 2010 is the result if you write this backward. The fact that the letters were not selected at random is crucial. They have significance. Natanara in this context refers to "a man (nara) who works as an actor (nata)". The mathematician can construct an important phrase from the numbers since there are several alternative letters for each number.

As a result, the author can portray big numbers using simple to-remember terms thanks to the Katapayadi method. Additionally, if it is poetry, it gives the author the freedom to choose a word that matches the meter and the context.

2. Literature Review

Sreeramula Rajeswara Sarma [1] states that the KaTaPaYadi notation is used to mark and represent the degrees of altitude in the astrolabe which occupies an important position in the Indian Astronomical Instruments.

K Lakshmi Priya et al. [18] presented a study on encoding systems in Vedic and Modern Eras of India using Sanskrit as a medium to represent the data in mathematical notations like KaTaPaYadi, Bhutasankhya and Aryabhatiya. They discussed the application of this KaTaPaYadi system to modern encryption techniques like Vigenère Cipher and Affine Ciphers.

R. Anusha et al. [19] have made an extensive study on how to decode the Sanskrit verses to their respective numerals and decrypt the original values behind the verses by developing a code in LabView platform which is only done manually since its inception.

Yashashwini Hegde et al. [20] have modified the base KaTaPaYa table named it V-KTPY and made use of it for Encrypting and Decrypting that best suit to the Indian Languages and other South Asian Languages. They applied this V-KTPY with Prefixed Hashed Trie specifically for Kannada Text and also proved the method is also space efficient.

Jayaganesh Kalyanasundaram and Dr. Saroja T. K. [21] & Raghunathan Anitha, Kandasamy Gunavathiand Finney Daniel Shadrach [22], states that all the existing 72-melakartha ragas of the Carnatic music are arranged accordingly by the application of KaTaPaYadi Sankhya Sutra.

Raman A [23], has compared and illustrated the similarity between the modern hashing technique and the ancient KaTaPaYadi Technique and stated that the KaTaPaYadi scheme can be thought of as the precursor to the modern hashing techniques.

The KaTaPaYādi system is simply a different mapping between digits and letters that consists of only Sanskrit consonants but not Vowels. Also, it is observed that there is an ambiguity while mapping the letters to the digits in it as the same digit refers to more than one consonant which is a strength when data is organized in the form of Sanskrit Stanzas confining to its Prosody but at the same time when we try to apply the same to the English alphabets it results in much ambiguity in mapping. So, there are two choices for each digit, e.g. 2 can be represented by either 'c' or 'p', with the decoding convention that we ignore vowels, and consonants immediately followed (in the same word) by another consonant. Then, to specify a sequence of digits like "314159265", we could write down below each digit the two possibilities for it and use these letters to make words. So, something like "dine rob soul cottage" would stand for DNRBSLCTG = 314159265. Then we could write rhyming poems or memorable prose, to memorize the digits and transmit them accurately in an oral tradition, but makes it very difficult to make it out and leads to lot of confusion even after decoding by the one who is authorized and authenticated to.

With the English alphabet, the above mapping would be hard to remember, but the alphabet in Sanskrit happens to be organized in a more orderly way, so it's easier to remember the mapping. The name Ka Ta Pa Yadi simply states that it is the beginning (ādi) letters of each chunk mapping to 1234567890. So, to resolve the stated issue, a novel arrangement of alphabets is made by allocating the numbers for both rows and columns similar to the matrix form shown in Table 3.

Table. 2

	3	1	4	1	5	9	2	6	5
	d	b	f	b	g	l	c	h	G
	q	n	r	n	s	x	p	t	S

The proposed model is built based on the KaTaPaYādi rules as the letters are divided into two parts constants and vowels. These constants are framed from 1,2,3,4, …., 9,0, and vowels with the same rule as shown in the below table.3

Table 3: Alphabets arranged resolving the ambiguity in Table-1.

	1	2	3	4	5	6	7	8	9	0
0	b	c	d	f	g	h	J	k	l	m
1	n	p	q	r	s	t	V	w	x	y
2	z									
3	a	e	i	o	u					

3. Proposed System

The Proposed algorithm is a Mono-alphabetic substitution symmetric cryptosystem in which K_s is a common key for encryption and decryption of messages that are being exchanged between the two parties. The plain text, P is provided to the encryption algorithm as an input which is subjected to several changes such as replacements, substitutions and rotations. KaTaPaYa like Modelled English Alphabetical Table, Modified KaTaPaYa Table, Semi-Unicode KaTaPaYa Table, ASCII Table and the randomly generated Secret Key, K_s where it is split into individual keys as k_0, k_1, k_2, k_3, k_4, k_5, k_6, k_7 which serves as independent keys for rotating the generated bits and concatenating these rotated bits results in the final cipher text, C. The opposite of the encryption process is the decryption in which the cipher text, C is provided as an input and the same secret key shared among the two parties serves as the key to decrypt the message. The flow of the proposed encryption process is depicted in Fig. 1 and the detailed substitutions, transformations, and rotations using the randomly generated key, K are shown in Fig. 2.

A. Encryption Process

Read the Plain Text, P of length 'n', and generate a random key of size 2^3. Map each of these 'n' characters in P with the KaTaPaYa simulated English table 5 of size (4 x 10) and get their index values. Now map these values to the Original KaTaPaYa table and obtain the Devanagari alphabets. Then these Devanagari alphabets are mapped to a devised Semi-Unicode for Devanagari is presented in table 6 of size (4 x 16) and obtain their index values. Now each of these values is pre-concatenated with '09' so that they become each of size 2^2. Now each digit is converted to its respective ASCII value and finally, its Binary digits are generated each of 8-bits in length makes it a 32-bit long chunk. Continue the process and obtain all the n-chunks of 32-bit binary digits. Now divide these n-chunks into groups of 2^3 and if any groups of chunks less than 2^3 are left as they are. Now intervene in the randomly generated secret key K_s by generating sub-keys k_0, k_1, k_2, k_7. Now apply these sub-keys from k_0 to k_5 on each chunk of the 32-bit binary digits and apply right and left shift rotations those number of bits in a circular mode and then arrange these individual chunks in matrix format which further take k_6 and k_7 and shift the bits of the even and odd columns with number of key values up and down alternatively and finally concatenate all the generated bit patterns that produces the final encrypted cipher text.

B. Required Tables:

Table 4. Sequential Arrangement of KaTaPaYa - Modified Version of size (4 x 10) after removing the ambiguity during mapping, that exists in the original KaTaPaYa Table

	1	2	3	4	5	6	7	8	9	0
0	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ
1	ट	ठ	ड	ढ	ण	त	थ	द	ध	न
2	प	फ	ब	भ	म					
3	य	र	ल	व	श	ष	स	ह		

Table 5. Modified KaTaPaYa Synonymized table for English Alphabets of size (4x10)

	1	2	3	4	5	6	7	8	9	0
0	b	c	d	f	g	h	j	K	L	M
1	n	p	q	r	s	t	v	W	X	Y
2	z									
3	a	e	i	o	u					

Table 6. Semi-Unicode Form of KaTaPaYa table of size (4 x 16)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1						क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट
2	ठ	ड	ढ	ण	त	थ	द	ध	न		प	फ	ब	भ	म	य
3	र		ल			व	श	ष	स	ह						

Encryption Algorithm

Input: PLAIN TEXT MESSAGE

Output: ENCODED STRING (A screenshot is shown in Figure 4)

Step 1. Read the Plain Text P = {P_0, P_1, P_2, ….., P_{n-1}}

Step 2. Generate a Random Secret Key K_s = {k_0, k_1, k_2,…, k_7}

Step 3. (n_i x 2) table = map P_i→ KaTaPaYa synonymized English alphabet table of size (4 x 10). Where n = 0, 1, 2, ...

Step 4. (n_i x 1) table = map each (x_i, y_j) indexes →KaTaPaYa table consisting of Devanagari alphabets.

Step 5. (n_i x 2) table = map each value of (n_ix1) table →Semi-Unicode Form of Devanagari matrix table of size (n_i x 16).

Step 6. (n_i x 1) table = Concatenate(‘09’,(n_i x 2))

Step 7. ASCII values = ASCII values of (n_i x 1) table

Step 8. n_i= Binary(ASCII values) where here each n_i is transformed into 32-bit chunks

Step 9. Circularlyn_{i+2}>>k_j where i = 0 to n-3 and j=0,2,4.

Step 10. Circularlyn_{i+1}<<k_j where j = 1to n-3 and j=1, 3, 5.

Step 11. Circularly shift the bits of col_i upward where i=0 to (32-2) such that ‘i’ is even.

Step 12. Circularly shift the bits of col_j downward where j=1 to (32-1) such that ‘j’ is odd.

Step 13. C = Concatenate(n_i) where i = 0 to n-1.

C. Decryption Process

Read the Cipher text, C. Slice up into chunks of 32 bits each. Input the secretly shared symmetric key K_s and generate the sub-keys k_0, k_1, k_2, ..., k5. Now use these sub-keys on each chunk of 32-bits and shift rotate those chunks in left and right circularly then using the rest of the two subkeys k_6 and k_7 shift rotate the odd and even column bits in the downward and upward directions and finally divide each chunk by 2^2 thus generating four 8-bit slices. Now convert them into respective ASCII values that omit the '09' from the resultant, ending up in the semi-Unicode form of Devanagari alphabets. Now obtain the index values from the semi-Unicode form Devanagari table of size (4 x 16) for these semi-Unicode form Devanagari alphabets. These index values in turn are mapped to the Modified KaTaPaYa table of size (4 x 10) to get the Devanagari alphabets which in turn are mapped to the KaTaPaYa synonymized English table of size (4 x 10) that produces the Original Plain Text, P.

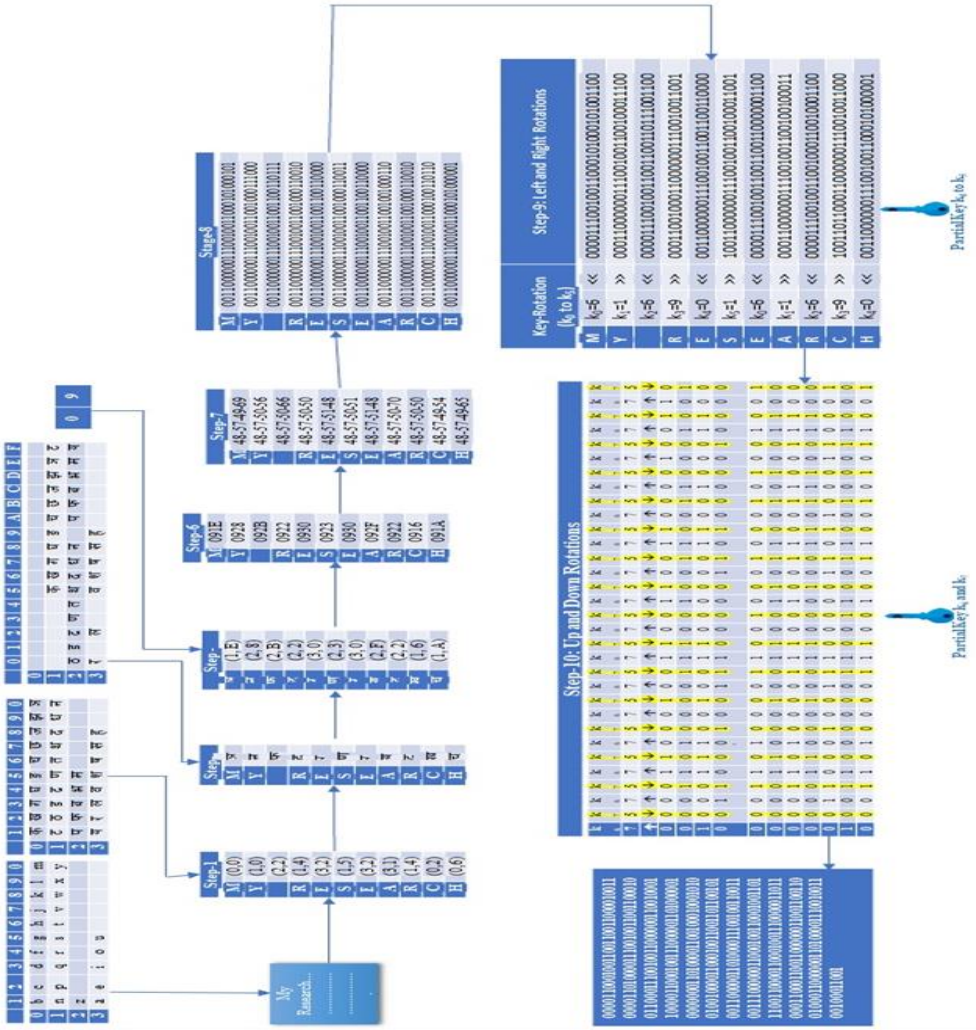

Figure 1. Encryption Process

D. Decryption Algorithm

Input: ENCODED STRING shown in Figure 4

Output: PLAIN TEXT MESSAGE

Step 1. Read the Cipher Text C = {C_0, C_1, C_2, ... , C_{n-1}}

Step 2. n_i = split(C, d); where d = 32-bits long

Step 3. Apply the secretly shared Symmetric Secret Key K_s = {k_6, k_7} on each n_i by rotating as follows

a. Circularly shift the bits of col_j upward where j=1 to (32-1) such that 'j' is odd

b. Circularly shift the bits of col_i downward where i=0 to (32-2) such that 'i' is even.

Step 4. Apply the secretly shared Symmetric Secret Key K_s = {k_0, k_1, k_2,…, k_5} on each n_i by rotating as follows

a. Circularly $n_{i+2} << k_j$ where i = 0 to n-1 and j=0, 2, 4.

b. Circularly $n_{i+1} >> k_j$ where j = 1 to n-1 and j=1, 3, 5.

Step 5. Split each n_i of size 32-bit long such that (n_i, delim=8) forming four octets n_k where k = 0 to 3.

Step 6. Unicode (n_i) = ASCII (Octet (n_k))

Step 7. (x_i, y_j) size table (n_i) = Omit ('09', Split ('09', Unicode (n_i)))

Step 8. (x_i, 1) table = map each (x_i, y_j) → (x_i, y_j) semi-Unicode table.

Step 9. (x_i, yj) table = map each (xi, 1) → (x_i, y_j) KaTaPaYa table

Step 10. (x_i, 1) table = map each (xi, y_j) → (x_i, y_j) KaTaPaYa synonymized English alphabet table.

Step 11. Print the (x_i, 1) which is the final Plain Text, P

Plain Text Message File: Test-case-1.txt

Plain Text Message: Plain Text Message

Generated Random Key: 82675766

Encryption Time: 1.3591551780700684 Secs.

Decryption Time: 1.2092769145965576 Secs.

Figure 2. Screenshot of Randomly Generated Key shown in circle

Figure 3. Screenshot of left, right, up & down shifting of bits in 32-bit chunks

Figure 4. Screenshot of Encoded Message in the form of 1's and 0's

Figure 5. Screenshot of 32-bit chunks, up & down shifting of bits for Decoding

Figure 6. Screenshot of left & right shifting of bits, Original Message & Encryption Time

4. Cryptanalysis

Cryptanalysis is the process of finding out how one can break the cipher text and get back the original text without the knowledge of the secret key that is exchanged between the sending and the receiving parties. The strength of an encryption algorithm against a brute force attack depends on several factors, including the structure of the encryption key, the complexity of the algorithm, and the length and structure of the plaintext and cipher text.

Therefore, our proposed algorithm is not susceptible to brute force attack and can withstand it. Since the key is randomly generated and accordingly the length of PT and CT varies at large that makes the job of the cryptanalyst too difficult to obtain the PT without knowing the key. Similarly, a Cipher text-only attack is also prevented as our algorithm undergoes complex logic and generates the CT of an extremely large length compared to PT.

A. Differential Cryptanalysis

It is a technique of testing an algorithm for its strength by toggling a bit of information in the bit stream of cipher text which causes a lot of change in its plain text when deciphered. This phenomenon is termed an Avalanche Effect. Coming to the proposed algorithm shows its avalanche effect on changing at least more than 5 bits of the bit stream of the cipher text.

B. Cipher Text-Only Attack

It is a method of trying to decrypt the cipher text without the knowledge of either any part of the plain text or the key [25, 26]. The proposed algorithm withstands this Cipher text-only attack as it generates the cipher text after undergoing complex routines of double substitutions and the variable number of horizontal and vertical rotations based on key values.

C. Brute-Force Attack

The strength of any encryption algorithm that withstands the brute force attack should contribute to a number of factors such as the encryption key and its complexity in its algorithm and the lengths of the plaintext and ciphertext. The proposed algorithm majorly targets the Light Weight devices with minimal resources and performs complex logics on the plain text that produces the Cipher Text which no way related to the given Plain Text and even the length of PT and the CT is unmatched thus preserves and guarantees the security of original message but at the same time it is good to use appropriate key management techniques for which it is addressed using the Random Key Generation Method by creating new key randomly for every message.

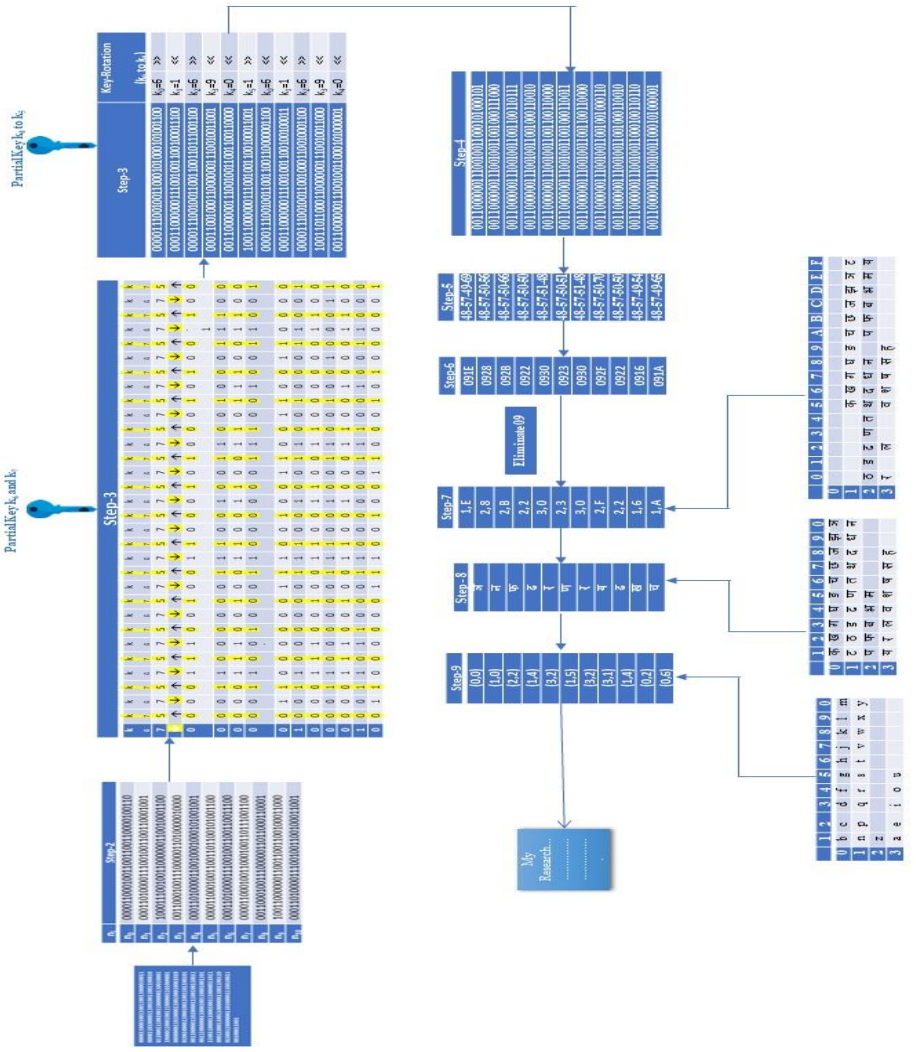

Figure 7. Process of Decryption

5. Results

Rigorous testing is done on the proposed algorithm on two systems with the following configurations:

System-1: Processor: Intel(R) Core (TM) i3-8100F CPU @ 3.60GHz, 64-bit operating system, x64-based processor

System-2: Processor: Intel(R) Core (TM) i5-10400F CPU @ 2.90GHz, 64-bit operating system, x64-based processor

RAM: 8 GB

A. Decryption:

Input: Encoded Message File: Test-case-1-opf.txt

Key: 90513692

Decoded Message: Figures 5 & 6

Decryption Time: 0.8895273208618164

Here the entire data is encrypted to Unicode using the model where all the characters have been converted to another numeric format which helps us in decreasing the size of the memory and improves the performance.

B. Time Complexity

The proposed encryption and decryption algorithms are developed into programs using Python Programming Language and tested for content varying between 1 KB to 10 KB on i5 processors with 8MB of RAM is presented in Table 7 and observed that the algorithm exhibits the following time complexities for various data sizes. The graph representations are shown in Figures 8 & 9

Table 7: Test Cases

	Encryption & Decryption Times & File Sizes			
	File Size (in KB)	Encrypted File Size (in KB)	Encryption Time (in Secs.)	Decryption Time (in Secs.)
Test Case - 1	1	2	2.435569286	1.595797777
Test Case - 2	2	49	61.74484181	46.28572965
Test Case - 3	3	84	128.7315798	79.25636435
Test Case - 4	4	103	165.0729403	94.82317853
Test Case - 5	5	158	157.9727793	130.22663
Test Case - 6	6	166	170.334491	133.8113124
Test Case - 7	7	210	203.4985893	122.1395061
Test Case - 8	8	240	239.505729	207.7781534
Test Case - 9	9	268	273.3522463	232.7629128

Test Case - 10	10	291	282.5673013	254.1524696

C. Observations.

It is observed from the graphs in Figures 3 & 4 generated from the test cases that the time consumed by the proposed algorithm for encryption is more than the time consumed for decryption i.e. the decryption time is approximately half the time of encryption. At the same time, the size of encrypted data is much larger than the original data which is a major strength of this algorithm that leaves the attacker vague in decrypting this large cipher text without knowing the shared secret key between the authorized parties.

Figure 8:

Encryption Time measured in Seconds

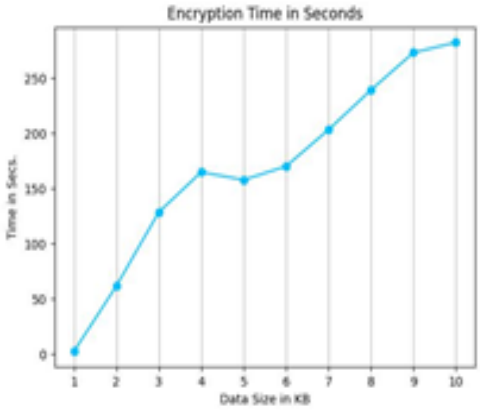

Figure 9:

Decryption Time measured in Seconds

6. Limitations.

This proposed algorithm applies to plain text consisting of purely the English alphabet and does not apply to any numerical values, symbols, or any kind of punctuation marks. One more limitation is that it does not differentiate the capital and small English alphabets.

7. Conclusion

The current encryption methods dramatically boost the size of files by between 20% and 100% or more while the research clearly does not show any significant changes in the size of files when working with Hindi/Sanskrit if they are solely applied the concept of mapping of these alphabets for the alphabets in a particular language. However, file size increases when working with this proposed algorithm as it undergoes many substitutions and rotations at various stages. However, the Sanskrit interpretation of the English phrase is significantly smaller in terms of size. So, through the correct implementation of the Sanskrit system into computers and by using efficient software, it might be possible to shrink the size of files and also the encryption of information that needs to be addressed. Another benefit of this method is during its testing,

it is observed that when some intruder tries to decrypt the message with an unauthorized key or some random key then the original data cannot be retrieved, and even in some cases if at all some part of the message can get decoded there is no use of it as the retrieved message is in some arbitrary arrangement and is meaningless. Thus, it is almost secure and impossible to tell whether data has been encrypted or it is not even in the former case.

References

1. Sreeramula Rajeswara Sarma, "Katapayadi system of numerical notation and its spread outside Kerala," Revue d' histoire des mathematiques, pp. 37-66, 2012.
2. A. A. Krishnaswami Ayaangar, 2010, “The Mathematics of Aryabhata” [Online] Available: www.ms.uky.edul-sohum/aaklpdf..1020files/aryabhatta.pdf
3. George Gheverghese Joseph, "Early Transmission of Indian Mathematics," Kerala Mathematics: History and Its Possible Transmission to Europe, Delhi: B.R. Publishing Corporation, 2009, pp. 205-231.
4. A. A. Hattangadi, "Calculating the Value of pi," Explorations in Mathematics, Hyderabad, Universities Press India Private Limited 2001, ch.l, sec. Appendices!. I, pp. I- I7.
5. Kester Q.A., A cryptographic algorithm based on words database, International Journal of Science, Engineering and Technology Research 2(4) (2013).
6. Paar C., Pelzl J., Understanding cryptography: a textbook for students and practitioners, Springer Science & Business Media (2009).
7. Lakshmi Priya K., Parameswaran R., Encoding systems in Vedic mathematics, National Seminar on Kerala School of Astronomy and Mathematics: Contributions and Contemporary Relevance (2016).
8. R. Prasad, "Sri Ramshalaka: A Vedic Method Of Text Encryption And Decryption," Indian Journal of Computer Science and Engineering, vol. 4, no. 3, p. 10, 2013.
9. "www.britannica.com," Encyclopaedia Britannica, Inc., [Online]. Available: http://www.britannica.com/biography/Varahamihira. [Accessed 05 10 2015].
10. S. Kumar and R. Prasad, "Some Ancient And Modern Concepts Of Cryptography," Ideal Science Review,vol.4, no. 1, pp. 15-20, 10 2013.
11. S. Kak, "Aryabhata’s Mathematics," in RSA Conference, 2006.
12. A. B. Raman, "The Ancient Katapayadi Formula And The Modern Hashing Method," p. 10.
13. A.Raman,http://citeseerx.ist.psu.edu,[Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.9659&rep=rep1&type=pdf.[Accessed 23092015]
14. Frigo, M., Leiserson, C. E., Prokop, H. & Ramachandran, S., 1999, “Cache-oblivious algorithms”, ’IEEE Symp. on the Foundations of Computer Science’, IEEE Computer Society Press, p. 285.
15. Wu DN, Gan QQ, Wang XM (2018) Verifiable public key encryption with keyword search based on homomorphic encryption in multi-user setting. IEEE Access 6:42445–42453. https:// doi. org/ 10. 1109/ ACCESS. 2018. 28614 24
16. Shen T, Wang F, Chen K, Wang K, Li B (2019) Efficient leveled (multi) identity-based fully homomorphic encryption schemes. IEEE Access 7:79299–79310. https:// doi. org/ 10. 1109/ ACCESS. 2019. 29226 85
17. Jun WJ, Fun TS (2021) A new image encryption algorithm based on single s-box and dynamic encryption step. IEEE Access 9:120596–120612.https://doi.org/10. 1109/ ACCESS. 2021. 31087 89
18. K. Lakshmi Priya and R. Parameswaran, 2017, “A Study on Encoding System in Vedic Era Modern Era”, International Journal of Pure and Applied Mathematics, Vol. 114, No. 7, pp. 425-

433.
19. R. Anusha, C. Nitya, R. Venkateswar Pai and C. V. Ramanathan, 2017, "Coding the Encoded: Automatic Decryption of Katapayadi and Aryabhata's system of Numeration", Current Science, Vol. 112, No. 3.
20. Yashashwini Hegde, Padma S. K., June, 2019, "V-KTPY Prefixed Hashed Trie for Indian Languages: A Case Study with Kannada Text Retrieval, International Journal of Engineering Research and Technology, Vol. 8, Issue-6.
21. Jayaganesh Kalyanasundaram and Dr. Saroja T. K. [31], 2019, "Raga Classification in Indian Classical Music-A Generalized Approach", Proceedings of the International Conference on New Music Concepts and Inspired Education, Vol. 6, pp. 116-122.
22. Raghunathan Anitha, Kandasamy Gunavathiand Finney Daniel Shadrach, "Investigation on Musical Features of Carnatic Ragas Using Neutrosophic Logic", First Internation Conference on Advances in Physical Sciences and Materials, Journal of Physics: Conference Series, 1706 (2020) 012051 IOP Publishing doi:10.1088/1742-6596/1706/1/012051
23. Raman A, "The Ancient Katapayadi Formula and the Modern Hashing Method"
24. Dr. T. K. Saroja, 2022, "Application and Representation of Simple Mathematical Ideas in South Indian Classical Music: An Interdisciplinary Approach", International Journal of Music Science, Technology and Art, Vol.4, Issue-2, pp. 125-131, ISSN:2612-2146.
25. R. Prabhakaran and M. Vijay Kumar, 2023, "The Secrets of Universe Behind Vedanta: A Classical Hidden Method by Ancient Indians: An Overview", International Journal of Research Trends and Innovations, Vol.8, Issue-3, pp. 442-445. ISSN.246-3315.
26. Madhav Moole and Flavia Gonsalves, 2024, "Exploring the Application of Sanskrit in Computer Programming", International Journal of Science and Research, Vol.13, pp.594-598, ISSN. 2319-7064.