INDIAN JOURNAL OF SCIENCE AND TECHNOLOGY

RESEARCH ARTICLE

Design and Implementation of Intelligent Key Cryptography using Time and Tamil Unicode

V Rojasree[1]*, J Gnana Jayanthi[2]

1 Research Scolar, Department of Computer Science, Rajah Serfoji Govt. College(A), (Affiliated to Bharathidasan University), Thanjavur, 613005, Tamil Nadu, India
2 Assistant Professor, Department of Computer Science, Rajah Serfoji Govt. College(A), (Affiliated to Bharathidasan University), Thanjavur, 613005, Tamil Nadu, India

OPEN ACCESS

Received: 22-09-2022
Accepted: 18-12-2022
Published: 16-01-2023

Citation: Rojasree V, Jayanthi JG (2023) Design and Implementation of Intelligent Key Cryptography using Time and Tamil Unicode. Indian Journal of Science and Technology 16(2): 133-145. https://doi.org/10.17485/IJST/v16i2.1917

* **Corresponding author**.

rojasree.v@gmail.com

Funding: Tamil Nadu State Council for Higher Education (TNSCHE)

Competing Interests: None

Published By Indian Society for Education and Environment (iSee)

ISSN
Print: 0974-6846
Electronic: 0974-5645

Abstract

Objectives: To propose a new novel and unique cryptosystem with interfaces for the designed cryptosystem and to implement and test the new designed methodologies. **Methods**: The Tamil Unicode is used to convert the Plain text (Sender's message) to Intelligent Key Cryptography system (IKC) Intermediary code. This intermediary code is converted to IKC cipher text by trans-positioning characters with integer digit values of Time in positive direction. IKC decryption of cipher text (received message) to Intermediary code is done using Tamil Unicode and then this is converted to plain text by trans-positioning characters with integer digit values of Time in negative direction. The IKC algorithm developed thus address the discrepancies for the existing algorithms based on the literary survey done. **Findings**: In the proposed IKC algorithm, the key and key space are generated dynamically from the system Time and the system font Tamil Unicode; thus, reducing the key maintenance problems of the existing algorithms. In current cryptography algorithms, work is based on the assumption that the key value never comes to zero. But all the algorithms are mathematical calculations which will return to zero at an infinite point. This drawback is overcome in IKC algorithm by taking the Time the real infinity. The value of the time once passed will never return. The entire algorithm works in a secure shell developed using C++ and has proved to satisfy the needs of the objective of a robust cryptosystem. Comparative analysis shows that IKC works better compared to the existing system. **Novelty**: The IKC algorithm displays as: (i) keys are generated dynamically, (ii) cipher once formed is never repeated, even if the same plaintext is enciphered again, and (iii) it satisfies NIST policies.

Keywords: IKC Encryption; IKC Decryption; AIRKGS; Dynamic key; Postquantum cryptography

1 Introduction

Quantum computers broke the strength of the entire security world that is dependent on mathematical calculations. The prevailing best known security algorithm uses trapdoor

mathematical calculations to assure security. The devastating impact of Shor's algorithm and Grover's algorithm and the introduction of quantum computers have created a great threat to the world of cyber security[(1)]. The strength of some Asymmetric algorithm namely RSA is the difficulty in finding the factors of the two big prime numbers used to formulate the public key N [(2)]. With the introduction of Shor's algorithm to find the prime factorization of any positive number N, the basement of RSA Shook with the threat of breaking the complexity[(2)]. The ECC algorithm which uses points on elliptical curves to get the keys of cryptographic algorithm has become unsecure thus leading to a new branch, post-quantum cryptography. The discrete-logarithm problems of the Diffie-Hellman and Elliptical-Curve-Cryptography (ECC) are affected as the Shor's algorithm as it directly breaks the cryptographic primitives by solving the equations swiftly.

These being the major challenges, a new kind of cryptography algorithm must be developed such as to help users to migrate from the classical system to new quantum-attack resistant system. Following are the observed facts from the literature studies of the IKC research work,

(i) The symmetric cryptography algorithms even though strong it is prone to attacks due to single key that is more vulnerable[(3)]

(ii) The asymmetric cryptography algorithms mostly adopt mathematical equations to generate key and is very vulnerable to quantum attacks[(4)].

(iii) The hashing algorithms are prone to birthday attacks[(5)].

2 Methodology

The proposed research on cryptography is mainly focused on generating dynamic random keys called intelligent keys both for encryption and decryption and hence this research is referred to as Intelligent Key Cryptography system (IKC) system. The IKC architecture shown in figure, Figure 1 , is a novel and unique, and well accepted by the research forum[(6)].

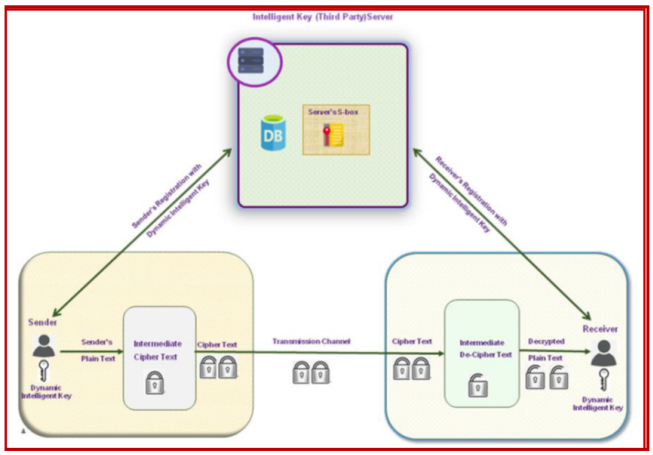

Fig 1. Proposed IKC Architecture Diagram

This newly proposed IKC architecture is highly secured in the sense of generating IKC key at the two ends for, (i) encrypting at the sender side to convert the message as cipher text and (ii) for decrypting at the receiver side to convert the cipher text as plain text. The Figure 2 briefs the initial functioning of the IKC system.

The methodology of the proposed IKC system is divided into four phases namely (i) User Registration, (ii) Generation of Intelligent Keys, (iii) Authentication based Encryption Process and (iv) Authentication based Decryption Process; and these phases are clearly outlined in the following subsections.

2.1 User Registration

A new user to the IKC system must initially register in the IKC system to get access to the IKC secure shell. During the user sign-up process, the user is given a form to fill in where the user details like the username, bio metric details, device from which the user is going to access IKC are all collected and sent to the IKC server. In the server the user details along with the registration time is taken and a secure shell is created for the user.

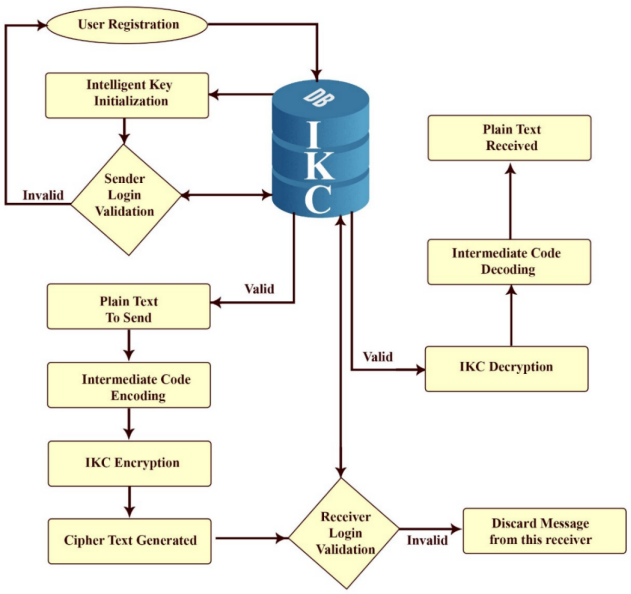

Fig 2. IKC Cryptography System Overview

Through the secure shell the user details are again confirmed by the user to commit the registration process. If the user confirms the details the secure shell created for the user is encrypted using IKC and stored in the core interface (core interface varies based on the area of application of IKC). If the user doesn't confirm the registration details the values created by the secure shell is reset and the control goes to the new user registration page. Figure 3 depicts the flowchart of user registration process.

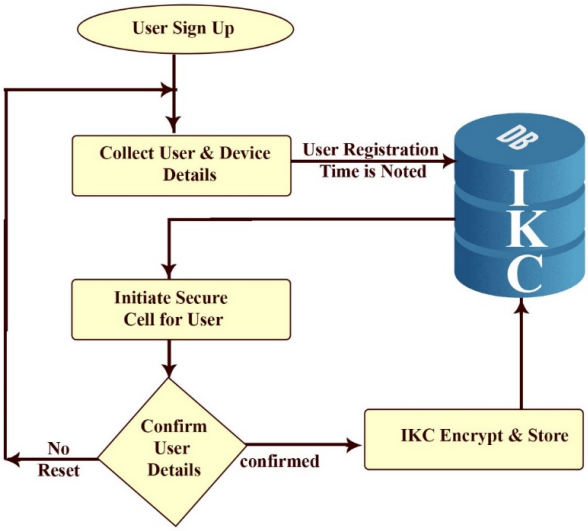

Fig 3. IKC User Registration Diagram

2.2 Generation of Unique Intelligent Keys in IKC

In this IKC architecture, keys for encryption and decryption are generated as per the recommendations stated by NIST for the key management in order to have security[7]. The proposed IKC system satisfies the encryption policy enlisted by NIST as (a) Key lifecycle which includes key generation, pre-activation, activation, expiration, post-activation and destruction; (b) Physical access to the key server(s); (c) Logical access to the key server(s) and (d) User/Role access to the encryption keys. This paper paves way for the reader to understand how IKC system abides to these policies. There are three keys used in IKC system, they are (i) TIME, (ii) Tamil Unicode and (iii) Message of the sender. The TIME is to time at which the sender sends the message. The time constant is universally a unique entity which once passed will never return. All the day-to-day life in interlaced with time. Tamil Unicode is used to encrypt the plain text, this is a novel idea, as all the current cryptographic algorithms uses English alphabets and numerals with the key a limited key space combination. There are 256 Tamil characters including Vada Ezhuthukal, hence provides a larger key space combination, which adds to the strength of the IKC system. In existing algorithms, the message of the sender is just encrypted using any crypt algorithm but in IKC the message of the sender also acts as a key to generate the intermediary code and hence increases the dynamicity of the IKC system. IKC handles the key sharing between connected people from within the algorithm through the server itself, so the role of the people is vanishing in key management. This proves that no overhead is required to specifically maintain the keys in IKC thus satisfying the NIST policies of a robust encryption algorithm.

2.3 Intermediary Code Generation in IKC

Every user of the IKC system must get registered first in the IKC Server where every user and their unique identity gets recorded and stored in the encrypted form. Initially when a user is registering, a secure environment is initiated on the server to record the user details and get the user to register in the server. Once the user is registered then the credentials get analyzed from the server every time the user tries to login to the system. The user login module can be designed by using a pin or biometric detail or a password. These are encrypted using IKC and stored in the user database in the IKC server. The user given pass code / pin / message is received by the IKC secret box wherein every character entered by the user are scanned and then mapped to intermediary Tamil characters by checking the input character is a shifted or un-shifted character. All the scanned characters are converted into intermediary cipher text by checking the occurrence of the special characters. The occurrence of every special character makes a consonant is attached from the Kuzhukuri chakra. At the first occurrence of the special character, the first consonant is attached with the special to form a compound character (uyir-meieluthukkal). The intermediary code created depends on the information sparking in the sender's mind thus the message is itself a primary key in encryption. This is known only to the sender. Based on the repetition of the letters in the message, the intermediary code created varies. These steps are well represented in Figure 4 .

If the message has a word "apple", then "a" is converted to Tamil Unicode "ya" the first "p" is converted as Tamil Unicode of "Ki" and the second "p" is converted as Tamil Unicode "Gni" the "l" as Tamil Unicode "ta" and the "e" as Tamil character "na". The letter "p" in the word apple represents a special symbol in Tamil so compound character is created in the intermediate code generation. This shows precisely that the repeated characters get converted to two different glyphs in intermediate code generator. If a normal un-shifted letter is repeated the intermediary code is the same but at the encryption stage the repeated characters get encrypted as two different characters in cipher.

2.4 Encryption Process in IKC

The intermediary code generated is passed as input to the encryption cycle. The overall encryption process is enumerated in flow diagram depicted in Figure 6 .

The primary chakras, Lipi and thedhiMani are used to encrypt the intermediary code to cipher text. The intermediary code is replaced with a new character that is obtained from the Lipi after trans-positioning the actual character with a character that is after the position equal to the value at the pointer in the thedhiMani chakra.

Encryption is initialized and based on these chakras. The encryption of the intermediary code is done using shift of characters based on the integer values on the chakras[8]. There are possibilities to use 5 secondary chakras in encryption based on which the complexity of the encryption increases thus strengthening the security of the information transferred using the system.

Figure 7 shows the Lipi and the thedhiMani chakras and how the intermediary code of character Tamil character "kaa" is ciphered by trans-positioning the characters in Lipi chakra with the value on the thedhiMani chakra. Once all the 14 values on the thedhiMani chakra is used to transposition the subsequent 14 characters in the intermediary code the next chakra the Mani with the minutes value is taken with the initial reading of the seconds when the user sent the message. Again the 15th character from the intermediate code gets ciphered using the first value in the thedhiMani chakra till the next 14 characters (that is the

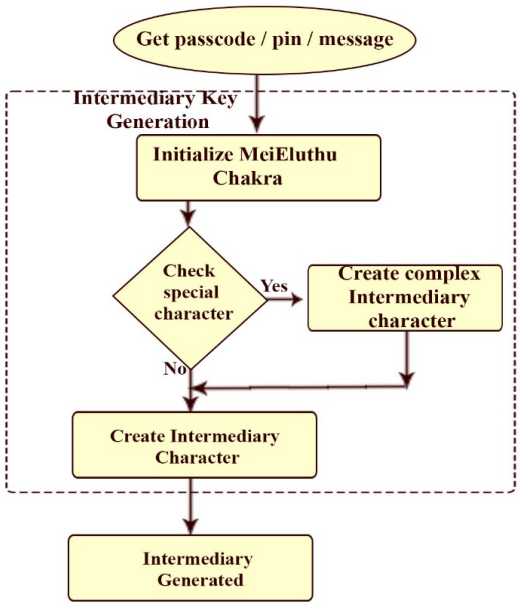

Fig 4. Intermediary Code Generation Cycle Diagram

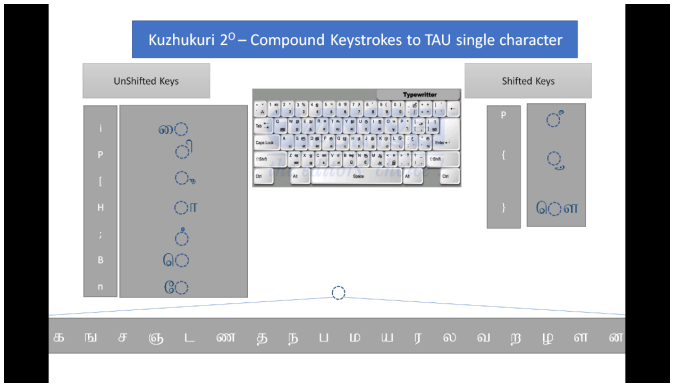

Fig 5. Functioning of Kuzhukuri

29th character), the values in the thedhiMani chakra is used. After which the next value in the vinaadi chakra that is 27 as in the Figure 8 .

The rotation of the font values is done whenever a chakra comes to an end and begins from the beginning. This is done as in counting of numbers from 0-9and then from 10-19, 20-29 and so on. For every full winding of the first chakra there is a single winding in the second chakra, for every full winding in the second chakra there is a single winding in the third chakra and it continues.

2.5 Decryption Process in IKC

The received cipher text is taken and converted to intermediary code by taking the date and time available from the encrypted message. The cipher text received contains the details of the receiver wherein the credentials of the receiver mentioned in the message is counter checked with the logged receiver for validity. If the logged user is not the actual receiver of the message, then the message in the receiver end is discarded.

The trans-positioning is done based on the values on the thedhiMani chakra and the characters in the cipher text. Figure 9 shows how the Tamil character “thaa” is deciphered to Tamil character “kaa” by shifting 6 places in the lipi chakra from the

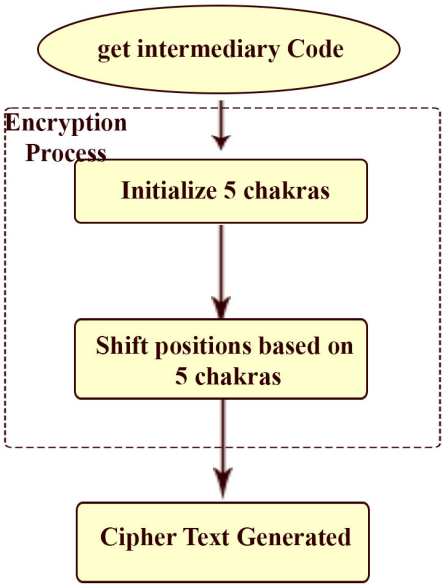

Fig 6. Encryption Cycle Diagram

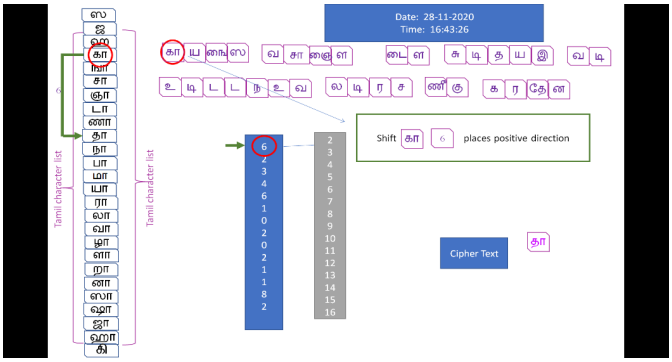

Fig 7. Encryption of Intermediary fh Code to Cipher text

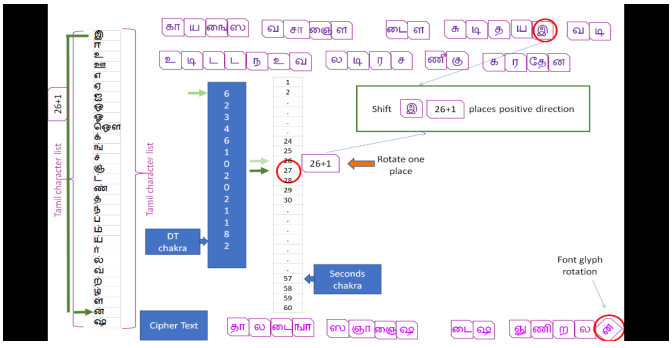

Fig 8. Nimidam Chakra Encryption Diagram

letter “thaa” to get “kaa” which is the intermediary text. The next cipher character is “la” and the next value in the thedhiMani chakra is 2 so first Tamil character “la” is located in lipi chakra and then reverse shifting in negative direction is done to get Tamil character “ya” which is the intermediate code.

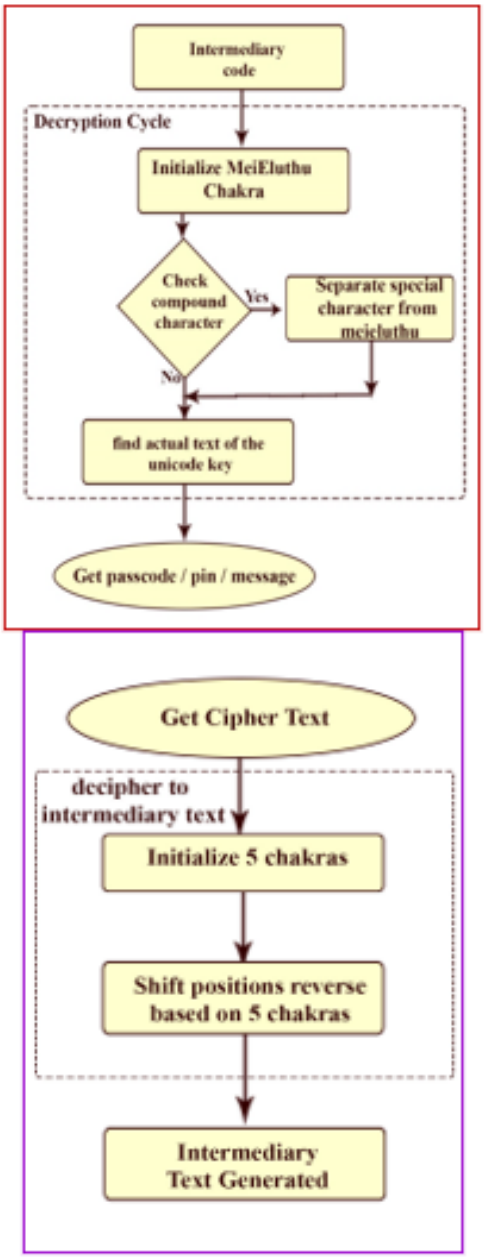

Fig 9. Decryption of Cipher text jh To Intermediary Code Diagram

If the recipient of the message is the actual receiver, then the cipher text is passed to the module where the intermediate code generation is done. The scope of this study is the encryption-decryption algorithm; so the focus is on the cryptography algorithm only. As shown in Figure 8 , the cipher text received is deciphered to intermediary text by initializing the chakras and reverse trans-positioning the characters in lipi chakra. When all the values in the thedhiMani is over, the next vinaadi chakra is considered and incremented one position as shown in Figure 10 .

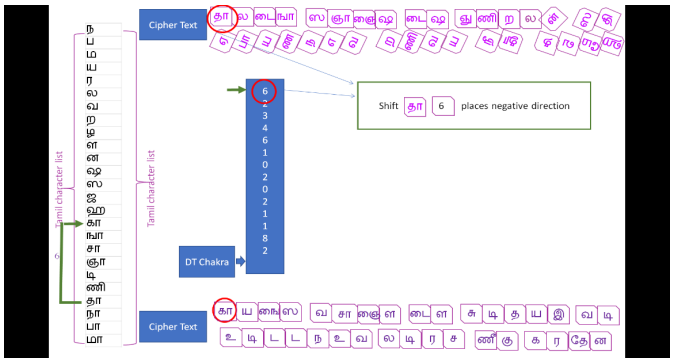

Fig 10. Decryption of Ciphertext jh To Intermediary Code Diagram

The message is an idea that sprouts in the mind of the sender. This message is the first key in IKC. As explained in this section, the plain text is itself a dynamic key and is used to produce the intermediary code in IKC.

3 Result and Discussion

The pseudo code algorithms are well developed for all the three stages of the proposed IKC system namely Intermediate Code Generation cycle, Encryption Cycle and Decryption cycle and they have been implemented for real time execution.

3.1 Implementation Setup for IKC

The proposed research is implemented in visual studio toolkit using C++ programming. The encryption and decryption are done on text files that are created to convert the user entered values to encrypt cipher text. These text files are shared on the internet during login and messages are transferred between the people registered on the IKC server.

All the users registered on the server can only create these text file as the rights are assigned only to the registered users. The user entered keyboard values are fetched and converted using a function called char* IKCcrypt (passcode) in which the user entered password is encrypted to Tamil Unicode and this function returns the character array of the encrypted password which is further used in the communication. On the deciphering end a function char* IKCdecrypt (cryptpasscode) in a similar manner takes the crypt code received and converts it to normal text and compares this new text with that password set in the server by the user. This being a small text it is done using arrays but when it comes to long text messages the encryption and decryption is handled in a slightly different manner wherein the message texted by the user is directly sent to a temporary file which is read from a function IKCcrypt-msg (filename) this reads the message content from the file character by character and simultaneous replaces the individual characters in the file with the encrypted characters. The encrypted text file is transmitted to the receiver's file location. The receivers on their end decrypt this file using a function IKCdecrypt-msg (filename) where the message ciphered text file is opened and read character by character and displayed to the receiver through a temporary textbox on the receiver app. Once the receivers' textbox is closed the temporary decrypted file is also deleted. The ciphered file is transmitted over the Internet, so this filename contains the primary key of encryption and decryption. Thus, the user need not worry about the key. There is neither private nor public it is intelligent key working on the backend for all registered users.

3.2 Implementation Scenario for IKC

The main constraint for using IKC is that all the users must be registered in the IKC server that is hosted in the specified LAN, WAN, or Cloud. Based on usage the server client modules are to be varied. But whatever the usage area the encryption and decryption is the same everywhere.

The encryption is done mainly on the (i) pass code / pin and (ii) on the message. The pass code /pin are normally a small text, so it could be handled by using character arrays to temporarily encrypt and decrypt. On the other end, the messages may vary in size and may be of any length and hence, text files are used to store the encrypted and decrypted messages.

These arrays and files are globally available to the registered user in their login directory in the server. The pass code / pin is tested only in the user's own area, so it is sent to any other user on the server. On the other hand, the message is to be sent to the recipient on the network, so it is stored in the form of text file and then transmitted from one user to another user using quantum entanglement method.

3.3 Implementation Results of IKC

The IKC was implemented using C++ on visual studio Code. The results are tested by allowing the user to input their username and password; the system displayed the intermediary crypt code and then the encrypted cipher text. Similarly, the decryption algorithm was also implemented, and the cipher text is sent to the program and the intermediary crypt code and the plaintext was displayed and this displayed text is checked with the actual primary message sent by the receiver and proved to be working efficiently and effectively.

3.4 Implementation of Users Registration in IKC

The user registration is done by loading the usernames, passwords, device details from a file as in a secure and efficient fashion. This is achieved by encrypting the user details and storing the details in an encrypted file. The time of the user registration is used to encrypt these details so that if anybody else tries to mimic will not be allowed to enter. This user registration is implemented in the core area of the deploying environment so as to enable secure messaging. As of now no collisions have been detected in the implementation of the code. The C++ code snippet of the user registration is shown in Table 1 .

Table 1. Piece of Code for User Registration

```
cout<< ”Please enter a username and password.” <<endl;
cin>>aUsername;
cin>> password;
//check if username is already taken.
for(count=0; count<userList.size(); count++) {while(userList[count].getUsername()==aUsername)
//as long as the username is taken
{cout<< ”Sorry, that username is already taken, please pick another one.” <<endl; cin>>aUsername; count=0;//restart username check
from beginning of the vector.
}
}
newUser.setUsername(aUsername);//store info in a user data type newUser.setPassword(password); newUser.setLoginFlag(false);
userList.push_back(newUser);//make a new user at the end of the set from user data type
i++;//increment user count.
break;
```

3.5 Implementation of Generating IKC Intelligent Keys

The registered users can login using the same encryption method of IKC where the pass code entered by the user is hidden from the time the user clicks the login button. The intelligent part of UDRIK is the primary generation of intermediary code based on the message entered by the use.

Here the 18 vowels of the Tamil language are taken and appended whenever a special character occurs in the user's message. This is done by mapping the QWERTY keyboard keys strokes to a specific Tamil character. The Unicode value of the character entered in identified and stored in a temporary variable (an array in case of pass code / pin and a file in case of message). This intermediary encrypted text is then sent to the encryption module where the transposition of the characters is done based on the various chakras chosen. The piece of code implemented is given in Table 2 .

Table 2. Piece of Code for Generating IKC Intelligent Key

```
case (112):
_setmode(_fileno(stdout),_O_TEXT);
cout<< ”E kurilsymbol :”; _setmode(_fileno(stdout),_O_WTEXT); tamil_char = L’\u0BBF’;
wcout<< ”You entered ”<<tamil_char<<” So intermediary
tamil char is: ”; password_int[j]=mei_eluthu[k]; wcout<<password_int[j]<<””;
j++;
password_int[j]=tamil_char; k++; wcout<<password_int[j]<<” ”; j++; _setmode(_fileno(stdout),_O_TEXT); cout<<endl;
_setmode(_fileno(stdout),_O_TEXT); cout<<endl;
```

The sample screen shots of the implementation and the output of the intermediary code generated are shown as inFigure 11
.

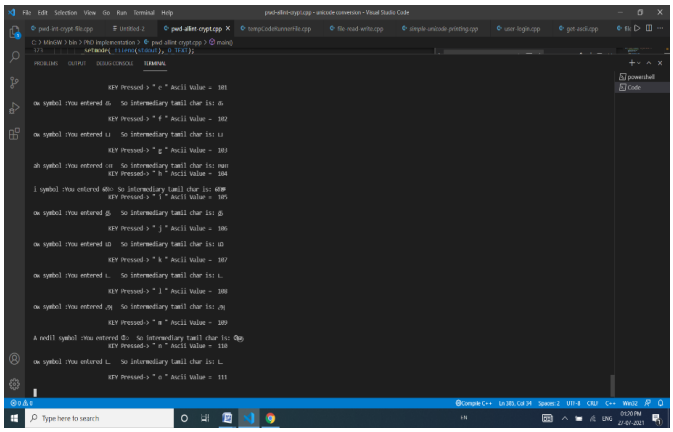

Fig 11. Intermediary Code Generation

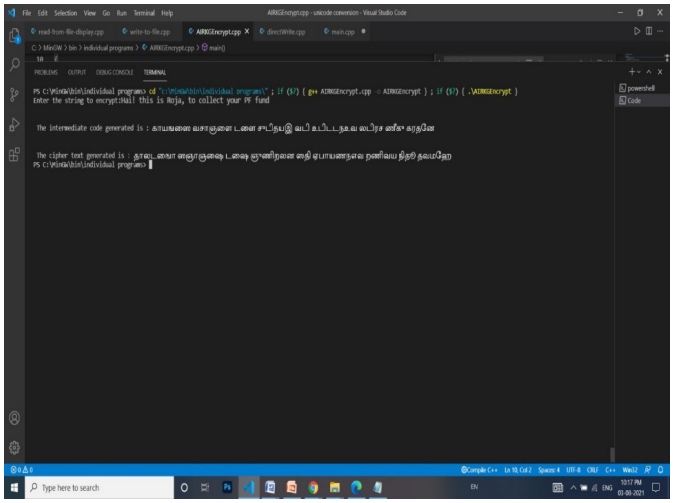

Fig 12. Encryption Screen Shot

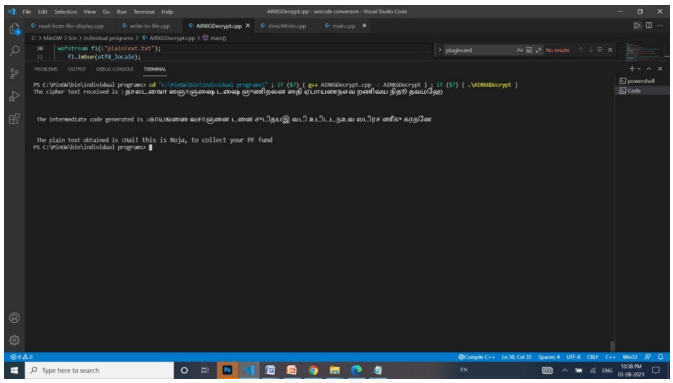

Fig 13. Decryption Screen Shot

3.6 Implementation of Encryption Process in IKC

Once the intermediary code is generated it is sent to the encryption module where the transposition of the intermediary text is carried out till the end of the message. The transposition is done with the help of 5 different data structures called chakras. There are 7 chakras used out of which the lipi (Alphabets) and the thedhiMani (Date-Time) are the 2 mandatory chakras, and the remaining Mani (Hours), nimidam (Minutes), naal (Day), maatham (Month) and varudam (Year) are used as secondary chakras to increase the strength of the algorithm. The lipi chakra contains a set of all Tamil characters and thedhiMani chakra is the list that contains the system date and time at the time the sender pressed the submit button on the application and is taken in the format ”DDMMYYYYhhmmss”. The thedhiMani chakra forms the heart of the IKC algorithm. The intermediary code generated is read character by character and for each character read is located on the Lipi and based on the thedhiMani the transposition is done whatever the value in Lipi is the cipher character for the character read. The entire intermediary code is scanned and converted at the end of this encryption process. The remaining five chakras are implemented based on the need of level of security as explained in the encryption and decryption process. The current date and time are set as the first item in the remaining chakras. That is in chakra Mani the current Hour of submission of the message is set to active. The Mani chakra has entries from 01 to 24 to indicate the 24 hours of the day. Similarly, the current date time value is used to set the initial value of the chakras, nimidam with the current minute (out of 01-60), naal with the current date (out of 01-31 also considering leap year), maatham with the current month (out of 01-12) and varudam with the current year.

During transposition is carried out, when the pointer comes to the end of thedhiMani chakra, the program jumps to the next chakra Mani where the pointer points to the current hour and so the transposition is done to that number and the cipher character is stored, then again, the pointer comes to the thedhiMani chakra as usual and starts from the beginning.

After the completion of reading all the values in the nimidam chakra the next value in the Mani chakra is used to transposition similarly for the entire process. This shows that once one full cycle of thedhiMani is completed, one shift in nimidam chakra takes place; after one full rotation of nimidam chakra, the mani chakra shifts one position; when this finished one cycle, the naal chakra shifts one place; When naal finishes one cycle, maadham rotates one place; and lastly when maadham finishes one cycle, varudam makes one shift. With this design procedure, the entire chakras work as it is done in counting. To add on whenever a shift is done the alignment of the character is also rotated to 4 degrees using the escapement value of the font. Therefore, if the cipher text is sent, it is not possible for the intruder to guess the plain text because the entire encryption is based on the date and time at which the user has sent the message and also primarily dependent on the message itself. The code snippet of implementation of Encryption is given in the Table 3 .

Table 3. Sample Code of IKC Encryption Algorithm

```
if(argc< 3 || argc == 4)
{
cerr<< ”usage: Airkg chakra-file lipi-file (<lipi-file>* lipi-positions)?” <<endl;
return INSUFFICIENT_NUMBER_OF_PARAMETERS;
}
AIRKG *ikc = nullptr;
try
{ ikc = new AIRKG(argc, argv); }
catch(int error)
{ delete Airkg;
return error; }
char letter;
while(!cin.eof())
{ cin>> letter;
if(cin.fail())
{ break; }
Airkg->encryptMessage(letter);
cout<< letter;
}
delete Airkg;
return NO_ERROR;
}
```

3.7 Implementation of Decryption Process in IKC

During the decryption, the cipher text received is first validated for authentication using IKC cryptography and then if accepted then the cipher text opens to get decrypted. At the first stage of decryption the received cipher text is read character by character and converted to intermediary code by using the thedhiMani chakra that is sent from the send through the header of the cipher text. The Lipi is initiated and based on the values in the thedhiMani chakra the reverse transposition of the characters in the Lipi for each and every character read from the cipher text is performed. When the 14 values of thedhiMani chakra have completed one full round the next nimidam chakra is rotated one place to get the next value of transposition.

At the end of the cipher text, all the cipher text gets converted to intermediary code. This intermediary code is sent to a function mapIntCode to get the plain text. Each and every character in the intermediary code is mapped to the corresponding keyboard character after checking the special character, shifted or un-shifted keys. Once the last character in the intermediary code is mapped we get the plain text in the place of the cipher text. Below is the portion of the code used to decrypt in AIRKG algorithm.

Piece of Code from IKC Decryption Algorithm

```
while(!in_stream.eof())
{
in_stream >> ws;
int eof = in_stream.peek();
   if(eof == EOF){
break;   }
in_stream >> num;
   if(in_stream.fail())
{     cerr << "Non-
numeric character in chakra positions file " << path  << endl;
in_stream.close();
throw(NON_NUMERIC_CHARACTER);   }
if(!isNumberRangeCorrect(num))
{     cerr << "The file " << path \
<< " contains a number that is not between 0 and 25" << endl;
in_stream.close();
throw(INVALID_INDEX);   }
counter++;
chakra_positions_.push_back(num);  }
int diff = counter - num_of_chakras_;
 if(diff < 0)
{   cerr << "No starting position for chakra " << num_of_chakras_ + diff \
<< " in chakra position file: " << path << endl;
in_stream.close();
throw(NO_chakra_STARTING_POSITION);  }
in_stream.close();}
bool AIRKG::isNumberRangeCorrect(int num){
return (num < ALPHABET_LENGTH && num >= 0);}
int AIRKG::checkAppearedBefore(vector<int> contacts, int num, int position){
for(int i = 0; i < position; i++ ){
   if(contacts[i] == num){ cerr << "Invalid mapping of input " << position << " to output " << num \
<< " (output " << num << " is already mapped to from input " \
<< i << ")" << endl;
return i;   }
 }  return -1;}
void AIRKG::decryptMessage(char& letter){
int current_index = letter - 'A';
current_index = kuzhukuri_->map(current_index);
 if(num_of_chakras_ > 0){
chakras_[num_of_chakras_-1].rotate();
```

```
 }
 if(num_of_chakras_ > 0){
for(int i = num_of_chakras_ ; i > 0; i–){
// TODO Needs explanation here
current_index = chakras_[i-1].shiftDown(current_index);
current_index = chakras_[i-1].mapForward(current_index);
current_index = chakras_[i-1].shiftUp(current_index);
    if(chakras_[i-1].isCurrentPositionInNotch() && \
chakras_[i-1].getPreviousPosition() != \
chakras_[i-1].getCurrentPosition()){
     if(i-1 > 0){
chakras_[i-2].rotate();
}
}
```

3.8 Result comparisons

The performance analysis of IKC algorithm is compared and contrasted with existing algorithms like Elgamal, RSA and ECC and is published as separate paper entitled "Performance Analysis of intelligent Key Cryptography (IKC) System". All the parameters proved out to be the best and efficient in IKC when compared with the existing algorithms [8].

4 Conclusion

The proposed IKC confirms the following: (i) Integrity as the users and devices are all registered with the AIRKGS and are allotted with a unique IKC encrypted identification. (ii) Data Authentication is assured as the encrypted text, sender-receiver details are sent in secure shell of IKC through quantum entanglement method. (iii) Confidentiality is attained as the entire system is under IKC surveillance. (iv) Nonrepudiation is confirmed as all the users and the devices are working with unique IKC identification and the time stamp is used to double confirm the users. (v) Post-quantum cryptography threats are overcome as there are no fixed mathematical calculations used. The novel IKC architecture is designed that can be used in highly confidential areas wherein data security is of greater importance. In this study, only the text-based cryptography is concentrated using Tamil Unicode; as future enhancement, IKC can be extended over media like image, audio and video. Even though this work is focused on IKC using Tamil Unicode, the same IKC architecture and IKC algorithms can be implemented using other languages with similar logic which will also increase the stability of the algorithm.

5 Acknowledgement

The research was funded by Tamil Nadu State Council for Higher Education (TNSCHE).

References

1) Bernstein DJ, Lange T. Post-quantum cryptography. *Nature*. 2017;549(7671):188–194. Available from: https://doi.org/10.1038/nature23461.
2) Rojasree V, Gnanajayanthi J. Cryptographic Algorithms to Secure Networks - A Technical Survey on Research Perspectives. In: 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT). IEEE. 2020;p. 159–165. Available from: https://doi.org/10.1109/ICSSIT48917.2020.9214259.
3) Rojasree V, Jayanthi JG. Research Intuitions of Block Cipher Techniques. *International Journal of Management*. 2021;XI:14–20. Available from: https://app.box.com/s/dmfcsme24cnmabuf4zxt9ctu8m2tmcju.
4) Rojasree V, Jayanthi JG. Research Intuitions of Asymmetric Cryptography System. *Turkish Journal of Computer and Mathematics Education*. 2021;12(3). Available from: https://doi.org/10.17762/turcomat.v12i3.2016.
5) Rojasree V, Jayanthi JG. Research Intuitions of Hashing Crypto System". *International Journal of Engineering Research & Technology*. 2020;9. Available from: https://www.academia.edu/44786307/IJERT_Research_Intuitions_of_Hashing_Crypto_System.
6) Rojasree V, Jayanthi JG. A Competent Intelligent Key Cryptography (IKC) Architecture. In: 2021 5th International Conference on Computing Methodologies and Communication (ICCMC). IEEE. 2021;p. 166–173. Available from: https://doi.org/10.1109/ICCMC51019.2021.9418233.
7) Barker E. Recommendation for Key Management, Part 1: General. 2015. Available from: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-57pt1r5.pdf.
8) Rojasree V, Jayanthi JG. Performance Analysis of intelligent Key Cryptography (IKC) System. *Mathematical Statistician and Engineering Applications*. 2022;71:67–78. Available from: https://philstat.org.ph/index.php/MSEA/article/view/455.