**1. INTRODUCTION**

**1.1DEFINITION AND ACYRONYM**

The **Vigenère cipher** is a method of encrypting alphabetic text by using a series of interwoven Caesar ciphers, based on the letters of a keyword. It is a form of polyalphabetic substitution.

**1.2 OVERVIEW**

First described in 1553, the cipher is easy to understand and implement, but it resisted all attempts to break it for three centuries until 1863. This earned it the description **le chiffreindéchiffrable** (French for 'the indecipherable cipher'). Many people have tried to implement encryption schemes that are essentially Vigenère ciphers. In 1863, Friedrich Kasiski was the first to publish a general method of deciphering Vigenère ciphers.

The Vigenère cipher was originally described by Giovan Battista Bellaso in his 1553 book *La cifra del. Sig. Giovan Battista Bellaso*, but the scheme was later misattributed to Blaise de Vigenère (1523–1596) in the 19th century and so acquired its present name.

**1.3 HISTORY**

The first well-documented description of a polyalphabetic cipher was formulated by Leon Battista Alberti around 1467 and used a metal cipher disc to switch between cipher alphabets. Alberti's system only switched alphabets after several words, and switches were indicated by writing the letter of the corresponding alphabet in the ciphertext. Later, Johannes Trithemius, in his work *Polygraphiae* (which was completed in manuscript form in 1508 but first published in 1518), invented the tabula recta, a critical component of the Vigenère cipher. The Trithemius cipher, however, provided a progressive, rather rigid and predictable system for switching between cipher alphabets.

What is now known as the Vigenère cipher was originally described by Giovan Battista Bellaso in his 1553 book *La cifra del Sig. Giovan Battista Bellaso*. He built upon the tabula recta of Trithemius but added a repeating "countersign" (a key) to switch cipher alphabets every letter. Whereas Alberti and Trithemius used a fixed pattern of substitutions, Bellaso's scheme meant the pattern of substitutions could be easily changed, simply by selecting a new key. Keys were typically single words or short phrases, known to both parties in advance, or transmitted "out of band" along with the message. Bellaso's method thus required strong security for only the key.

**1.4 APPLICATION**

Cryptography is not only about encrypting and decrypting messages it is also about solving real world problems that require information security.

There are four main objectives that arise;

1. Confidentiality: Eve should not be able to read Alice’s message to Bob the main tools are encryption and decryption algorithms.

2. Data Integrity: Bob wants to be sure that Alice’s message has not been altered/ for example, transmission errors might occur. Also, and adversary might intercept the transmission and alter it before it reaches the intended recipient. Many cryptographic primitive, such as has functions, provide methods to detect data manipulation by malicious or accidental adversaries.

3. Authentication: Bob wants to be sure that only Alice could have sent the message he received.

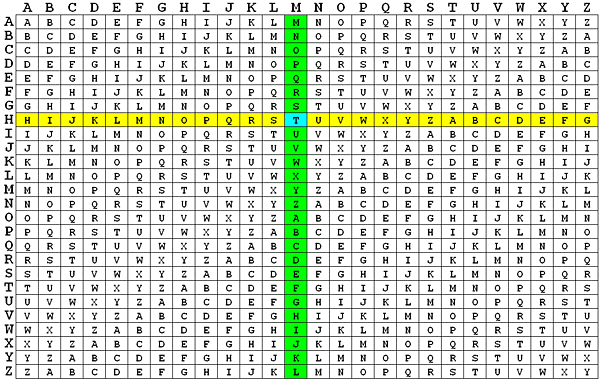
**1.5 WEAKNESS AND FLAWS**

1. It’s a complex process.

2. Time complexity is more.

3. Key has to be shared.

**2. TESTING**

****

**2.1 Description of above figure:**

* First we find the length of the message and then we change the length of key to length of the message. If the length of the key is greater than the length of the message then truncate the extra characters else if the key length is lesser than message length then we repeat the key again until it matches. As a result, the plaintext and keyword become the following:

MICHIGAN TECHNOLOGICAL UNIVERSITY

HOUGHTON HOUGHTONHOUGH TONHOUGNTO

* To encrypt, pick a letter in the plaintext and its corresponding letter in the keyword, and use the keyword letter and the plaintext letter as the row index and column index, respectively, and the entry at the row-column intersection is the letter in the ciphertext. For example, the first letter in the plaintext is **M** and its corresponding keyword letter is **H**. This means that the row of **H** and the column of **M** are used, and the entry **T** at the intersection is the encrypted result.

**3. PERFORMANCE EVOLUTION**

* **Time Complexity**

The time complexity of this algorithm is Θ(n). Where “n” is the length of the message.

**4. CONCLUSION**

In summary, Vigenère’s cipher is a shift cipher based on the use of a key word or phrase for encryption. It works by ‘adding’ the key to the plaintext modulo 26 and thereby returning the encrypted text or ciphertext. This cipher works requires that neither the key nor its length is known to attackers. It is also vulnerable to letter frequency attacks. The most secure implementation would use a long key which is not a known English word to send only short messages. This avoids brute force attacks of common word keys and prevents enough cipher text to be collected for a letter frequency attack to be undertaken.

**REFERENCES**

[1] Robbi Rahim, Ali Ikhwan, “Study of Three Pass Protocol on data Security”

International Journal of Science and Research (IJRS) 5 (11), November

2016.

[2] BoniOktaviana, AndysahPuteraUtamaSiahaan, “Three-Pass Protocol

Implementation in Caesar Cipher Classic Cryptography” IOSR Journal of

Computer Engineering (IOSR-JCE) 18(4) III (Jul-Aug, 2016): 26-29.

[3] Al-Amin Mohammed Aliyu, AbdulrahmanOlaniyan “Vigenere Cipher:

Trends, Review and Possible Modifications” International Journal of

Computer Applications (0975-8887) 135(11), February 2016.

[4] SaloniGarg, SonamKhera, Archana Aggarwal, “Extended Vigenere Cipher

with Stream Cipher” International Journal of Engineering Science and

Computing (IJESC) 6 (5), May 2016.

[5] Nimai Agarwal, PoonamChoudhari, Manivannan S S, “A Cryptosystem

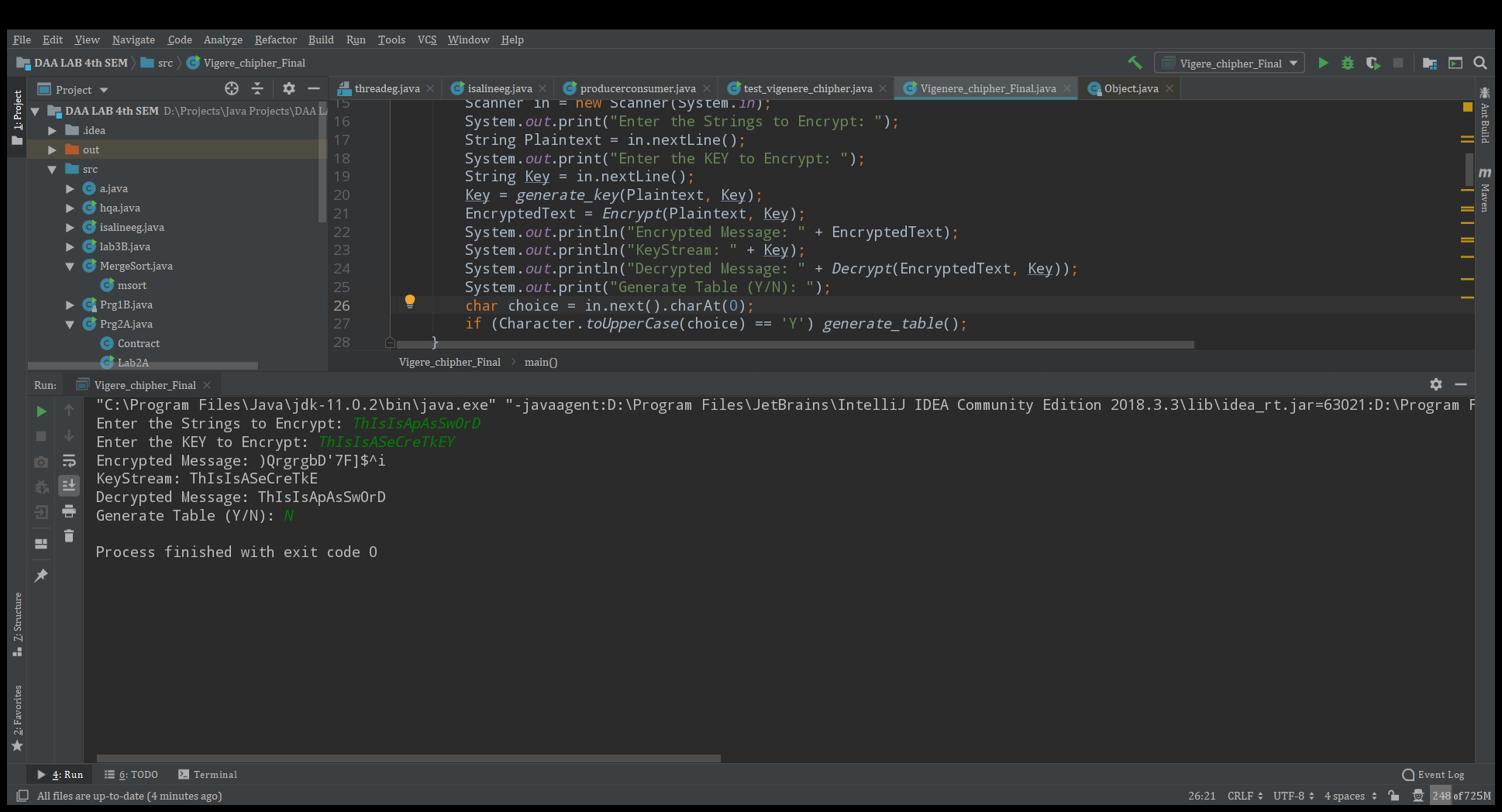
based on Vigenere Cipher with Enhanced Key Mechanism for data Security

a Crypto System Based on Vigenere Cipher to Enhance Data Security”,

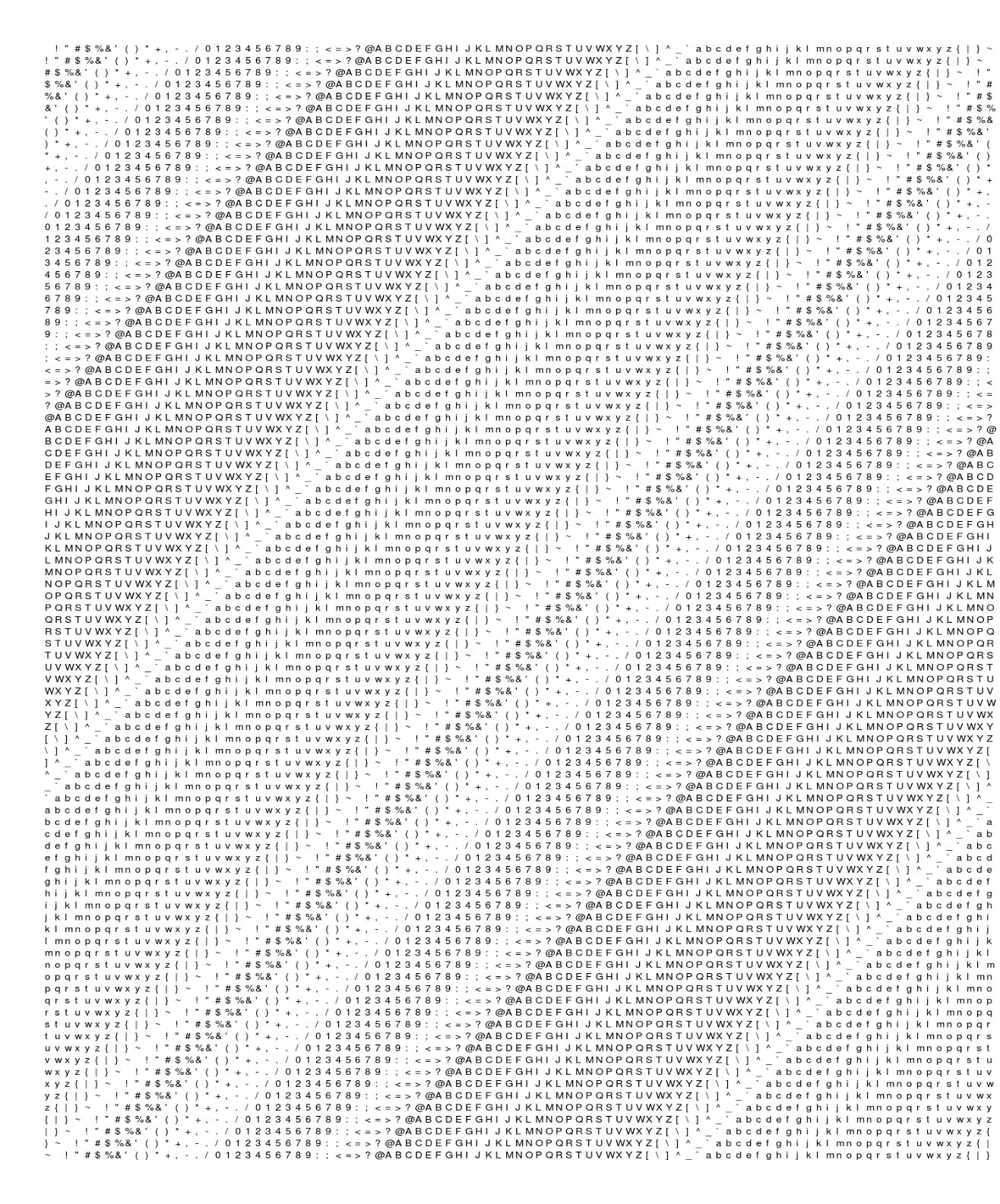
International Journal of Pharmacy & Technology 8 (4), Dec 2016.

**Appendix-A**

* **Sample Output**



* **VigenèreTable**



**Appendix-B**

* **Encryption Code:**

public static String Encrypt(String Message, String key)  
{  
intChipher\_int;  
String Chipher = "";  
 for (inti = 0; i<Message.length(); i++)  
 {  
// -64 coz we have consider the position of the character and not the actual asciii.e( ascii of Chipher -32) and (ascii of key -32) thus -64 in total.  
Chipher\_int = (int) ((Message.charAt(i) + key.charAt(i) - 64) % 95);  
Chipher += (char) (Chipher\_int + 32);// +32 to get the ascii of the Chipher character and store it in Chiper.  
}  
return Chipher;  
}

* **Decryption Code:**

public static String Decrypt(String Chipher, String key)  
{  
intMeassage\_int;  
String Message = "";  
 for (inti = 0; i<Chipher.length(); i++)  
 {  
Meassage\_int = (int) ((Chipher.charAt(i) - key.charAt(i) + 95) % 95); //+95 to get the actual offset of the character.  
Message += (char) (Meassage\_int + 32); // +32 to get the ascii of the Message character and store it in message.  
}  
return Message;  
}