Advances in Computer Science and its Applications (ACSA)
Vol. 1, No. 3, 2012, ISSN 2166-2924

www.worldsciencepublisher.org

An Efficient Implementation of E–MULET with Unicode standard

D.M.V. Sivakarthik[1], M.L. Sunanda[2], A.Venu Madhuri[3], P.Asif Khan[4]

[1]Y8EM226, [2]Y8EM274, [3]L9EM352, [4]Y8EM292
[1,2,3,4] Department of ECM, KLUniversity, Guntur, India

[1]sivakarthik.das@gmail.com, [2]mlsunanda@gmail.com , [3]venumadhuri.avula@gmail.com , [4]asifpathan292@gmail.com

Abstract: The science of cryptology is a boon to the Internet era. In this age of universal electronic connectivity, of viruses and hackers, of electronic eavesdropping and electronic fraud, security matters the most. Ever since Ceasar's time a variety of encryption techniques have used but the cryptanalysis has simultaneously cracked these encryption techniques from time to time. Though complex encryption techniques have been employed in safeguarding data, the use of a multilingual approach is not prevalent. Unicode supports about 100 languages as of now. By using the help Unicode a multilingual approach to cryptology can slowdown the cryptanalysis multifold. Enhancement in security can be brought by localization of encryption technology.

Keywords: cryptology; multilingual approach; Unicode Standard; Perl unicode

I. INTRODUCTION

The current age can in fact be aptly described to be transforming into a digital age. The rapid growth of internet and widespread availability of networks have led to the massive usage of internet based applications. The computers have entered all fields enhancing its utility to the common man in every way possible. The transformation to a digital age can be a boon as well as a curse. Besides the flexibility of data in digital format and the features the digital age has to offer, it is to be understood that too much is at stake. This entire stake has been placed on encryption technology to protect the data and so far it has done quite a job. At this point, it is to be understood that cryptanalysis, which cracks Encryption, has improved a lot and with improvements in computational power, the present day systems are able to crack encryption.

Upon some research, it is found that the Encryption scope is bounded by the limited language acceptance that it has. American Standard Code for Information Interchange (ASCII), the international standard to represent the characters has a character set of only 256 characters, of which only 128 are constant in all versions over the world.

II. THE CHARSET CHAOS

Because the size of a standard ASCII character can represent 256 characters and only 128 have been standardised, everyone thought that they could use the other 128 characters to represent whatever characters that they think would be essential[1]. The additional set of characters formulated is called OEM character set. At first, IBM created this OEM character set with what it imagined to be the characters that would be needed by everyone. As soon as computers started to ship outside America, all kinds of OEM character sets have originated and data transmitted has suddenly different interpretations.

Eventually a OEM free-for-all has been certified by ANSI standard and other OEMs called codepages came into existence which can be used upon specifying that particular codepage (Russian, Israeli, Latin …).

But, what is to be taken from this is that, people found that there are several other characters that are not in ASCII but are thought to be very useful.

III. UNICODE: The proposed system

The people at Unicode Consortium prepared to sort this out by giving each character a unique code, adding all such characters to its character set. Unicode now is a computing industry standard for the consistent encoding, representation and handling of text expressed in most of the world's writing systems [3]. Developed in conjunction with the Universal Character Set standard and published in book form as The Unicode Standard, the latest version of Unicode consists of a repertoire of more than 110,000 characters covering 100 scripts, a set of code charts for visual reference, an encoding methodology and set of standard character encodings, an enumeration of character properties such as upper and lower case, a set of reference data computer files, and a

number of related items, such as character properties, rules for normalization, decomposition, collation, rendering, and bidirectional display order (for the correct display of text containing both right-to-left scripts, such as Arabic and Hebrew, and left-to-right scripts). As of 2012, the most recent version is Unicode 6.1.

With the use of Unicode in Encryption technology, its language acceptance problem can be dealt with easily[4]. Unicode improves the range of language acceptance hundred fold and thereby improves the scope of Encryption technology. Many encodings are present to represent Unicode characters such as UTF, Punycode, GB18030 etc,.[2]

Besides, Unicode even has support in various operating systems and programming languages. Windows NT, Windows 2000, Windows XP, Windows Vista, Windows 7 have UTF 16 as internal character encoding. The other operating systems have Unicode support through Microsoft layer for Unicode. MacOSX and KDE also have Unicode support. The languages that support Unicode character representation standard are Java, Lisp, Ada95, python, perl and many more languages are working on it. Of all these Perl has extensive Unicode feature support integrated and is easy to code.

IV. E-MULET

E-MULET, short for Enhanced Multi Language Encryption Technique, is an encryption algorithm optimized to use characters from multiple languages. The algorithm starts by converting given input into Unicode characters. It involves taking either Unicode input or converting ASCII input to Unicode. For the sake of the project, limited character set from the 100,000 possible characters of Unicode has been selected with characters from over 20 languages. This is stored locally and made available to the program. It then takes user's key that identifies the user. Using this key, it generates a set of values unique to that user and uses them to perform substitution with the characters from the available character set. The encrypted data is sent as output.

The decryption starts by accepting the cipher input and identifying the user with his key. Unique set of values for the user are generated and the substituted characters are replaced with original characters. Thus the original cipher text is returned.

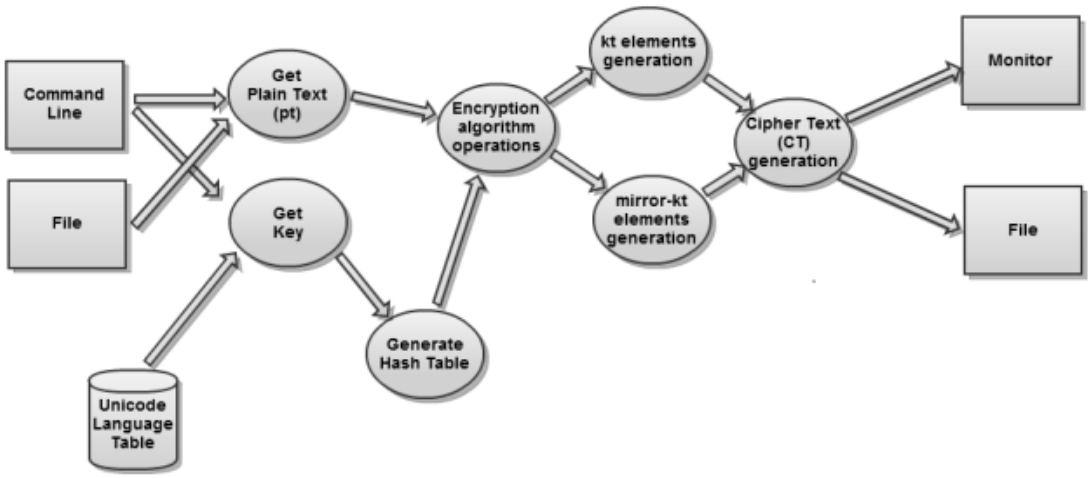

Fig.1. Algorithmic representation of E-MULET

V. Implementation in PERL

Perl is a high-level, general-purpose, interpreted, dynamic programming language. Perl was originally developed by Larry Wall in 1987 as a general-purpose Unix scripting language to make report processing easier. Perl is nicknamed "the Swiss Army chainsaw of programming languages" due to its flexibility and power. It is also referred to as the "duct tape that holds the Internet together", in reference to its ubiquity and perceived inelegance.

Its major features include support for multiple programming paradigms (procedural, object-oriented, and functional styles), reference counting memory management (without a cycle-detecting garbage collector), built-in support for text processing, and a large collection of third-party modules.

Since Perl 5.12, perl supports "Unicode strings" feature which assumes that the data being manipulated is encoded in Unicode. Perl 5.14 has much more support and features but does not exist in any feasible platforms except test platforms. Hence, the project is implemented in Perl 5.12 on LinuxMint 12 (Lisa).

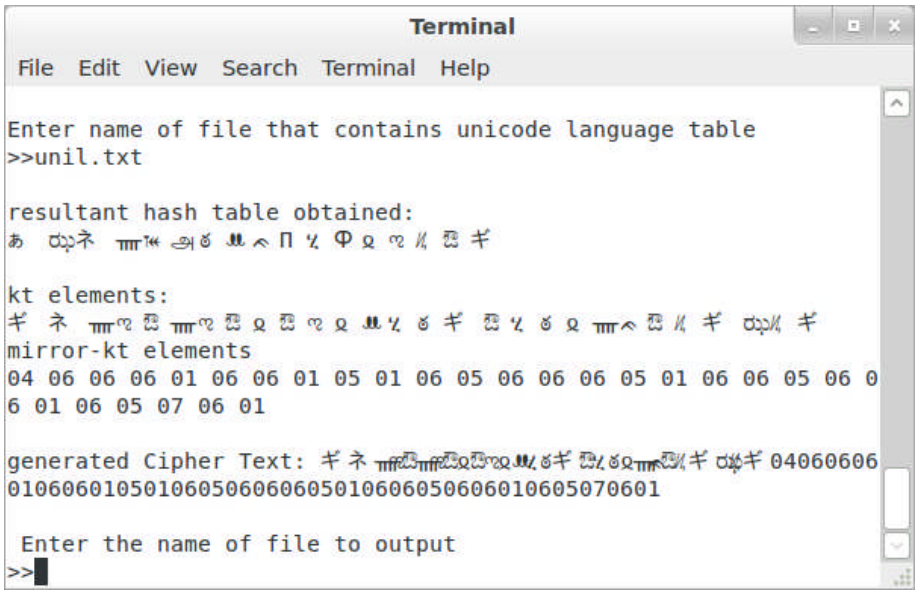

Fig.2: Screenshot showing Cipher text in Unicode

VI. EXPERIMENTAL RESULTS

The code written in Perl is used to implement the E-MULET algorithm. With the support for Unicode standard from version 5.12, the code ran without any errors. The system is subjected to test on various aspects. It is seen that even Unicode based input from different languages such as Telugu, Hindi can be given as input just like ASCII data is given. The program can even withstand large character sets that are supported by the Unicode version being used. The integration of Unicode standard into this algorithm is a success.

VII. CONCLUSION

Unicode support in programming languages as well as Operating systems is increasing by the day. This opened way for many new applications. It is only apt that this is used to increase the scope of encryption technology. The project proves that integration of Unicode standard into Encryption technology can be achieved with little effort and in exchange provides more functionality. The up gradation of presently used algorithms to provide space for Unicode standard requires some work but is worth the upgrade.

REFERENCES

[1]. Roman Czyborra, *Why do we need Unicode?*, czyborra.com [http://goo.gl/w2m8M]

[2]. Joel Spolsky, The absolute minimum every software developer must know about Unicode and Character sets [http://goo.gl/VxUl]

[3]. Unicode, from Wikipedia [http://goo.gl/idxn]

[4]. D.M.V.Sivakarthik, K.Ravi Kumar, Enhancement in Encryption through Localization, IJERA