APPLYING LABVIEW FOR ANALYSING SOME CRYPTOGRAPHIC ALGORITHMS IN TERMS OF SECURITY ENHANCEMENT OF ELECTRONIC TRANSACTIONS

Florim IDRIZI
Faculty of Natural Sciences and Mathematics
State University of Tetova
e-mail: florim.idrizi@unite.edu.mk
Ilia NINKA
Department of IT, Faculty of Natural Sciences, University of Tirana
e-mail:ilia.ninka@fshn.edu.al

Abstract
Cryptography and data security are becoming a required necessity and standard of the present time. There is not a total security on Internet, everyday our information is exposed to a certain threat or risk of misuse. In this paper we explain the concepts of security enhancement during the process of electronic data transfer, emphasizing the security enhancement of electronic transactions. Then, we are going to introduce some of recent algorithms which support and enable this electronic communications by using the LabView software. Further we will analyze and measure the speed of some cryptographic algorithms in terms of number of bits of words that are being encrypted or decrypted by using LabView, and at the end we will conduct some comparisons and evaluations of security issues for those algorithms.
Keywords: cryptographic algorithms, encrypted/decrypted, LabView.

1. Introduction

The main problem and challenge among current cryptographic system remains reducing the time for encryption and decryption.In order to achieve the secrecy requirements and not reproducing the exchanged information, the process of transfering sensitive data through communication channels has revealed the necessity of increasing the network speed and security.

Data security is an essential part of an organization; it can be achieved by the using various methods. In order to maintain and upgrade the model still efforts are required and increase the marginally overheads. The encrypted data is safe for some time but never think it is permanently safe. After the time goes on there is chance of hacking the data by the hacker. Fake files are transmitted in the same manner as one can sends the encrypted data. The information about the key is present in the encrypt data which solves the problem of secure transport of keys from the transmitter to receiver [1,2]. In case of practical system encrypted data is passed through the various stations which are capable to re-encrypt the data by their own key. At the time the previous keys are discarded, this will make the system more secure. There are many algorithms available in the market for encrypting the data. Encryption is the process in which plaintext has been converted into the encoded format cipher text with the help of key [4].

The security of a symmetric cryptosystem is a function of the length of the key. The longer the key, the more resistant the algorithm is to a successful brute force attack. For this reason, key length was chosen as the first parameter for specifying cryptographic algorithms. Key Length is an easy objective, numeric metric to adopt since key size is universally expressed as a number of bits. For example, the standard key length for the Data Encryption Standard (DES) is 56 bits. Assuming there is no better way to break the cryptosystem, other than to try every possible key with a brute force attack, the longer the key, the longer it will take to make the number of attempts necessary to find the correct key. In fact, every extra key bit generally doubles the number of possible keys and therefore increases the effort required for a successful brute force attack against most symmetric algorithms. A key length of *N* bits has 2^N possibilities. Adding an extra key bit does not always exactly double the effort required to break public key algorithms because some public key algorithms may have short-cut attacks such as factoring and computing the discrete log[3].

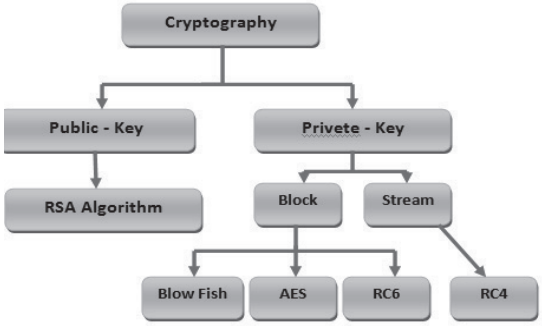


Figure 2. Overview of the field of cryptography

They can be categorized into Symmetric (private) and Asymmetric (public) keys en- cryption. In Symmetric keys encryption or secret key encryption, only one key is used to encrypt and decrypt data. In Asymmetric keys, two keys are used; private and public keys. Public key is used for encryption and private key is used for decryption. Public key encryption is based on mathematical func tions, computationally intensive and is not very efficient for small mobile devices [5,6]. There are many examples of strong and weak keys of cryptography algorithms like RC2, DES, 3DES, RC6, Blowfish, and AES. RC2 uses one 64-bit key. DES uses one 64-bits key. Triple DES (3DES) uses three 64-bits keys while AES uses various (128,192,256) bits keys. Blowfish uses various (32-448); default 128bits while RC6 is used various (128,192,256) bits keys. The most common classification of encryption techniques can be shown in Figure 1 [7].

2. Blowfish Algorithm

Blowfish is a symmetric block cipher that can be used as a drop-in replacement for Data Encryption Standard or

International Data Encryption Algorithm. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic and exportable use. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Since then, it has been analyzed considerably, and is slowly gaining acceptance as a strong encryption algorithm. Blowfish is not patented, is license-free, and is available free for all uses.

As far as I know, this is the first implementation of the Blowfish Algorithm in LabVIEW. With this set of sub-VIs, one can encrypt data in LabVIEW without the need for external software. This can be used to send data securely over Data Socket as well as TCP and UDP communications. It can also be used to protect remote control systems from unauthorized access by encrypting the control communications.

I have added compatibility for the basic blowfish functions with other implantations, and fixed a couple of bugs. The "LabVIEW Blowfish Encryption.vi" uses header info to return an encrypted message back to its original length; because of this, it will likely not be directly compatible with other encryption software even though the software may use the same blowfish encryption method. Without knowing what other software uses for header info, I just used what was convenient for me in LabVIEW [8].

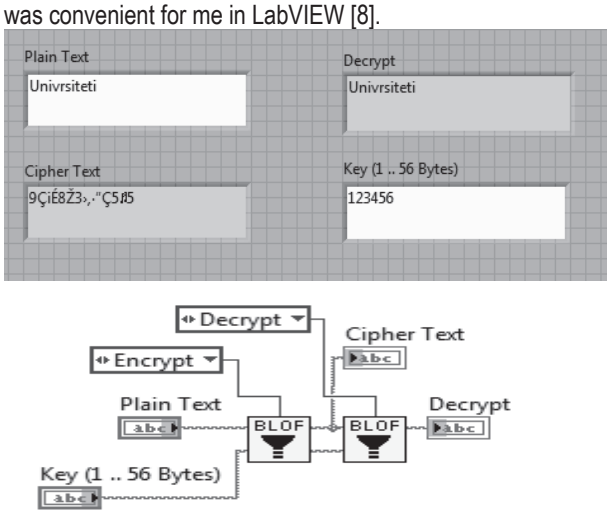


Figura 1. Relizimi i Blowfish algoritmit duke shfrytëzuar LabView

3. Tiny Encryption Algorithm (TEA)

The Tiny Encryption Algorithm (TEA) is a symmetric (private) key encryption algorithm created by David Wheeler and Roger Needham of Cambridge University and published in 1994. It was designed for simplicity and performance, while seeking an encryption strength on par with more complicated and resource-intensive algorithms such as DES (Data Encryption Standard). Wheeler and Needham summarize this as follows: "it is hoped that it can easily be translated into most languages in a compatible way… it uses little set up time and does… enough rounds to make it secure… it can replace DES in software, and is short enough to write into almost any program on any computer."[9]

Use XTEA whenever memory and processing requirements are limited, or when you want basic fast encryption without requiring pre-requisite code that may be licensed or patented, or carry

export restrictions. It should not be used for extremely sensitive or long-lived data. Choose algorithms carefully. Understand the purpose and benefits of a particular algorithm before choosing it. TEA/XTEA is a private key encryption algorithm; do not repeat Microsoft's mistake of using it as a hash function. Many security weaknesses have come from the simple fact that an otherwise good algorithm was used for the wrong purpose. Choose keys carefully and protect them. TEA/XTEA is a private key algorithm, and its key must be protected. Do not repeat Microsoft's mistake of letting a critical private key be easily detected. Seek to understand the true significance of "weakness" claims. A "weakness" may have no real practical impact other than to spread "fear, uncertainty, and doubt." Cryptanalysis algorithms should be published and independently verified. Encryption authors should publish test suites to verify implementations, in addition to complete reference implementations[10].

The Tiny Encryption Algorithm (TEA) is a block cipher encryption algorithm that is very simple to implement, has fast execution time, and takes minimal storage space. The included example is to be compiled and used on a LabVIEW FPGA target.

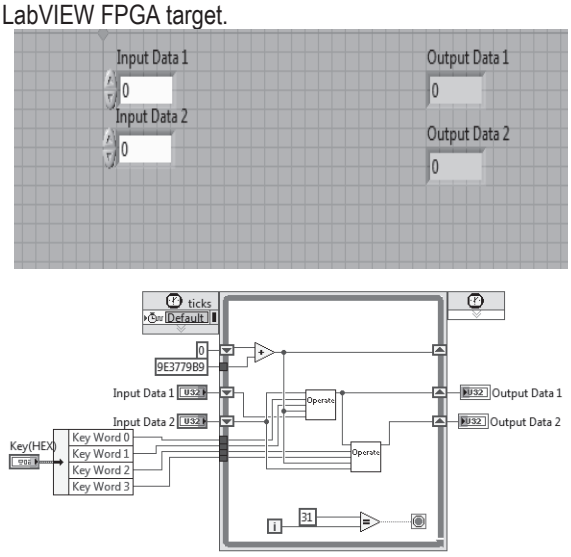
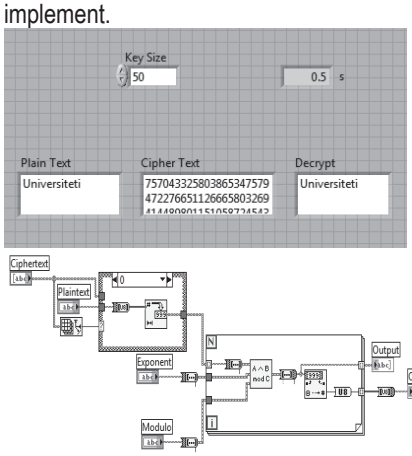


Figura 2. Relizimi i **TEA** algoritmit duke shfrztëzuar LabView

4. RSA Algorithm

RSA has two keys one private and one public. RSA is implemented as a software. With computers its hard to code plain text and to decodecipher text using RSA. If the key is sent separately encrypted with RSA, then the recipient can use it to decrypt messages encrypted with DES. AES is faster and more secure. For its encoding and decoding is used the same key. If AES is used for communication (encoding of the message), the receiver must have the key, so there appears the problem of key exchange. This problem is solved by using asymmetric algorithms such as RSA. Here the sender sends the message using the private key. Receiver decodes the message with the public key. Anyone can use this public

key. So, the receiver can decode the message. Asymmetric algorithms are much slower but the key exchange can baccomplished with asymmetric algorithms and encryption of data with symmetric algorithms. Both are very easy to implement.



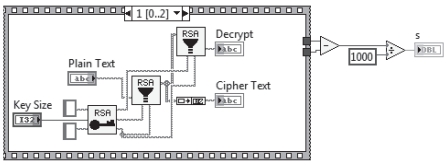


Figure 3.The realization of RSA algorithm using LabVIEW

The table below shows the measure speed of RSA algorithm according to the length of the key and the length of the sentence. Seen from the table the algorithm speed depends of two factors: the length of the Key and the length of the sentence which is encoded and decoded

Table 1.The algorithm speed according to the key and the length of the used sentence- RSA

	Word length				
Key size	**10 bit**	**20 bit**	**30 bit**	**40 bit**	**50 bit**
32 bit	0.1s	0.2s	0.2s	0.3s	0.3s
64 bit	0.5s	0.6s	1.0s	1.2s	2.2s
128 bit	2.2s	3.1s	4.0s	4.8s	5.4s
256 bit	18.4s	22.1s	27.2s	27.5s	34.7s

5. Advance Encryption Standard (AES)

During 1990s NIST started the trials for AES development, which is an encryption algorithm with symmetric key. This algotithm proved to be more successful than DES because DES security is based in the key length being used, but the usage of key with 56 bits is not enough.

AES algorithm is a subset of Rijndael algorithm. AES uses bloc of 128 bits and three different key sizes with 128, 196, and 256 bits, where Rijndael allows multiple bloc sizes with 128, 196, and 256 bits. For each of them allows different sizes of the keys, again 127, 196, and 256 bits. AES algorithm is symmetric, meaning that the same key can be used for both encrypting and decrypting the message.Moreover, the gained cyphertext by AES algorithm has similar size with the plaintext message

AES is very fast and secure. For its coding and decoding we use the same key. If it is used for message coding, the receiver must posses the key, which reflects the problem of exchanging the key. This problem is solved by applying asymmetric algorithms, such as RSA. Here, the sender sends the message by using private key. The receiver decodes the message with public key. Everyone in the traffic can make use of the public key. Thus, the receiver can decode the message. The asymmetric algorithms are very slow, but the key exchange is done with asymmetric algorithms, and the data encoding with symmetric algorithms.

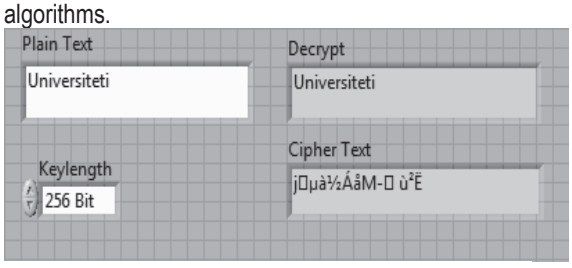


Figure 4. The realization of AES algorithm using LabVIEW

From the table can be concluded that the speed of AES algorithm is not depended on the length of key (128,192,256) nor from the length of the sentence which is encrypted or decrypted.

Table 2.The algorithm speed according to the key and the length of the used sentence- AES

	Word length				
Key size	10 bit	20 bit	30 bit	40 bit	50 bit
128 bit	0.1s	0.1s	0.1s	0.1s	0.1s
192 bit	0.1s	0.1s	0.1s	0.1s	0.1s
256 bit	0.1s	0.1s	0.1s	0.1s	0.1s

6. DES Algorithm

Purpose is to provide a standard method for protecting sensitive commercial and unclassified data. IBM created the first draft of the algorithm, calling it LUCIFER. DES officially became a federal standard in November of 1976. It is based on a symmetric-key algorithm that uses a 56-bit key for encrypting 64-bits plaintext. DES uses Feistel network and it has 16 rounds in its structure. The notable feature of DES is using S-Box, a table-driven nonlinear substitution operation in which input size and output size both can vary either randomly or algorithmically for increasing diffusion [11].

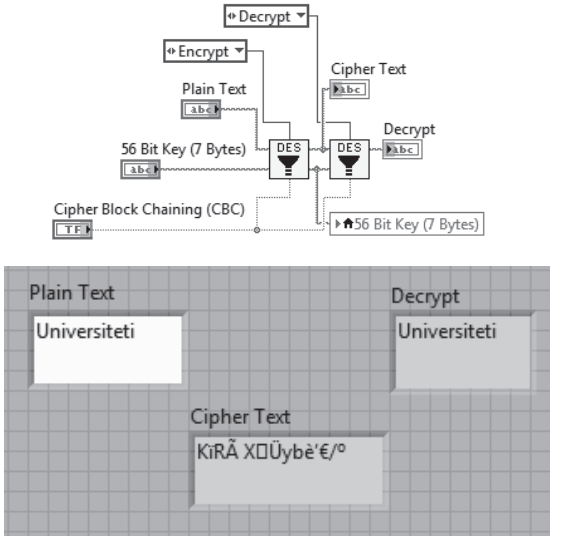


Figure 5. The realization of DES algorithm using LabVIEW

7. RSA - DSA Algorithm

About RSA and DSA is that RSA is less secure than DSA but authenticates faster.But the speed difference in authentication is so small (if you talk about ssh'ing into a system with a regular size password of say 8 characters (or a few more if you use MD5) that you will not notice a difference.

Using DSA with SHA-256 in DNSSEC has some advantages and disadvantages relative to using RSA with SHA-256 when using 2048-bit keys. DSA signatures are much shorter than RSA signatures; at this size, the difference is 512 bits verus 2048 bits. On typical platforms using 2048-bit keys, signing DSA is about three times faster than for RSA, but verifying RSA signatures is more than ten times faster than for DSA.

The cryptographic strength of DSA is generally considered to be equivalent to RSA when the DSA public key and the RSA public keys are the same size. Such an assessment could, of course, change in the future if new attacks that work better with one or the other algorithms are found[12].

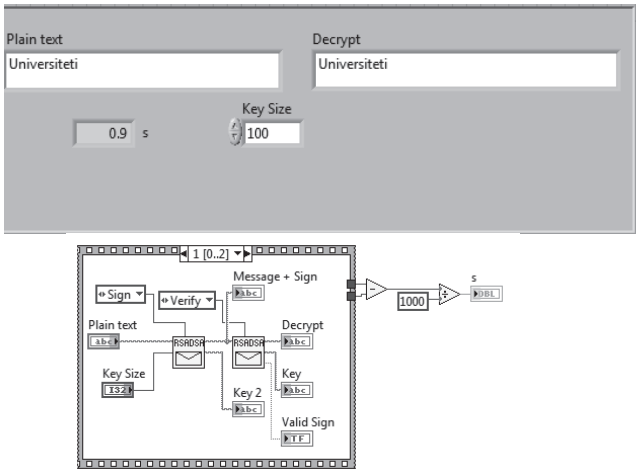


Figure 6. The realization of RSA-DSA algorithm using LabVIEW

The table below shows the measure speed of RSA-DSA combined algorithm according to the length of the key and the length of the sentence. Seen from the table the algorithm speed depends mostly by the length of the Key used rather then the length of the sentence which isencrypted and decrypted.

Table 3.The algorithm speed according to the key and the length of the used sentence - RSA-DSA

	Word length				
Key size	10 bit	20 bit	30 bit	40 bit	50 bit
32 bit	0.1s	0.2s	0.2s	0.2s	0.3s
64 bit	0.5s	0.5s	0.6s	0.7s	0.7s
128 bit	1.2s	1.4s	1.6s	2.3s	2.5s

Conclusions

The Internet allowed the idea of an idealized market to seem reliable. However, there are still uncertainties and concerns regarding the secureexchange of money through the Internet. This paper was undertaken to explore and analyzes the benefits of using some of the most known cryptographic algorithms and reports the measurements of the speed of some new encryption algorithms depending on the number of bits of words that are encrypted / decrypted, as a method of ensuring the data transmitted electronically. According to the analysis and the algorithms figured in the program LabVIEW can be seen that some

algorithms depend more from the length of the key used and the length of the sentence which is encrypted and decrypted. During the encryption of transferred data the speed depends from the system more specifically from the system performances that implements cryptography.

.

References

[1] Ajay Kakkar, Dr. M. L. Singh, Dr. P. K. Bansal, "Efficient Key Mechanisms in Multinode Network for Secured Data Transmission", International Journal of Engineering Science and Technology, Vol. 2, Issue 5, 2010, pp.787-795.
[2] Davis, R, "The data encryption standard in perspective", Communications Society Magazine, IEEE, 2003, pp. 5 – 9.
[3] Norman D. Jorstad Director, Cryptographic algorithm metrics, , Technology Identification and Analyses Center
[4] Ajay Kakkar, M. L. Singh, P.K. Bansal, Comparison of Various Encryption Algorithms and Techniques for Secured Data Communication in Multinode Network, International Journal of Engineering and Technology Volume 2 No. 1, January, 2012
[5] Hardjono, Security In Wireless LANS And MANS, Artech House Publishers, 2005.
[6] P. Ruangchaijatupon, and P. Krishnamurthy, "Encryption and power consumption in wireless LANsN," The Third IEEE Workshop on Wireless LANs, pp. 148-152, Newton, Massachusetts, Sep. 27-28, 2001
[7] Evaluating The Performance of Symmetric Encryption Algorithms, International Journal of Network Security, Vol.10, No.3, PP.216–222, May 2010
[8] http://zone.ni.com/devzone/cda/epd/p/id/3473
[9] Wheeler, David J. and Needham, Roger M. TEA, a Tiny Encryption Algorithm. Computer Laboratory, Cambridge University, England. November, 1994.
[10] Derek Williams ,The Tiny Encryption Algorithm (TEA), CPSC 6128 – Network Security,Columbus State University April 26, 2008
[11] Harsh Kumar Verma,Performance Analysis of RC5, Blowfish and DES Block Cipher Algorithms, International Journal of Computer Applications (0975 – 8887) Volume 42– No.16, March 2012
[12] http://superuser.com/questions/13164/what-is-better-for-gpg-keys-rsa-or-dsa