European Journal of Applied Sciences – Vol. 13, No. 05
Publication Date: September 27, 2025
DOI:10.14738/aivp.1305.19432.
Kutsman, V. (2025). Program-method for Storing Secret Information Using Vernam Cipher & AES. *European Journal of Applied Sciences, Vol - 13(05).* 182-188.

Program-method for Storing Secret Information Using Vernam Cipher & AES

Vladyslav Kutsman
Computer Engineering & Information Systems Department,
Information Technologies Faculty, Khmelnytskyi National
University, Khmelnytskyi, Ukraine

ABSTRACT

Information security has been and will be one of the key positions both in business and in scientific work or for storing personal information. Reports, receipts, guarantees of information security of clients - all these are very important aspects that ensure the credibility of the company. But also very important is the safety of information during the work of the company itself or personal information of an individual or employee, for example, this can be an algorithm or program code that needs to be placed on github, but in such a form that only a few people can read it. Or certain financial statements of an accountant of a financial company, which should be seen only by the accountant himself and, at most, the owner. And in the case of a financial company, data leakage can be a reason for at least a loss of credibility, profit, or even the possibility of further business activity. This article presents software that implements an algorithm for storing file data on a permanent storage device of a computer using Vernam encryption and AES encryption, which can be used when working both in the workplace and in the private area of life, allowing you to conveniently and safely store information without fear of further data leakage. It is also very important that the proposed software does not require additional hardware elements in a computer or work laptop and, of course, can work with both local data and data from the network.

Keywords: Data security, Golang, Vernam & AES Encryption, Fintech.

INTRODUCTION

The fact of creating cryptographic software may not seem entirely justified in view of the existence of such programs that have undergone a certain path of testing and improvements, and in view of the complexity of the topic of cryptography by definition, but the presented idea is a completely new method of storing data that is based on such algorithms as the Vernam encryption algorithm [1] and AES [1] encryption and can become an indispensable tool to guarantee the security of "sensitive" information. For example, there is data that should be seen by a maximum of several people, these can be financial reports, or in the process of developing a program or some new algorithm, it may be necessary to exchange data via GIT (to maintain versioning) or via a file sharing service with colleagues, but the transmitted information must be encrypted and the recipient must be able to process the information in the same way as the sender. Another example may be information / algorithm that is used by an accountant to calculate a certain kind of company performance indicators, but the algorithm for calculating these indicators and the result should be available only to the accountant of the enterprise and the owner, but at the same time there should be the ability to

exchange this information. It is also very important that the above examples are taken from real life, when using BitLocker can cary in a complete loss of information in the case of a key loss. Another important factor is that the same BitLocker does not provide the ability to exchange information, for example, via github, and in cases where versioning is a very important argument or critical, this encryption format will not work. The software proposed in this article does not require additional hardware modules and is not limited exclusively to the computer's hard drive, because it is possible to exchange information between computers. This method storing information will be integrated in DFSboRD[3].

The basic idea is that certain types of "sensitive" information can be stored encrypted on a permanent storage device on the same computer. And access to the information is possible only through a special encoder/decoder program that processes the information, "on the fly". As shown in Fig. 1, the program is connected to the hard drive bidirectionally. The handler program itself is launched from a flash drive, then the flash drive can be disconnected, and after loading into RAM, the program begins to process the data, data exchange occurs using the WebDAV protocol. That is, for example, on disk "D" a folder INFO is created, in the program this folder is installed for processing, and in the explorer interface via the WebDAV protocol this folder is connected as a kind of "virtual" network disk "Z" Fig. 2. And when trying to read a file directly from the folder D:\INFO the user will receive an encrypted unreadable file Fig. 3., but when this information gets through the handler already to the virtual network disk "Z" the user will receive a decrypted file Fig. 4. and can work with it as he needs.

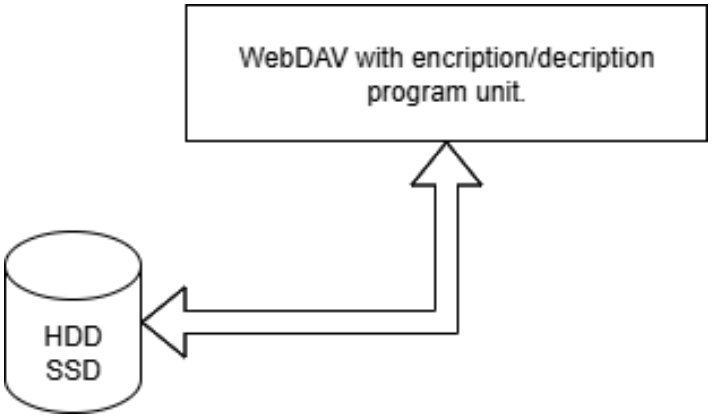

Fig. 1: Schematic representation of program interaction.

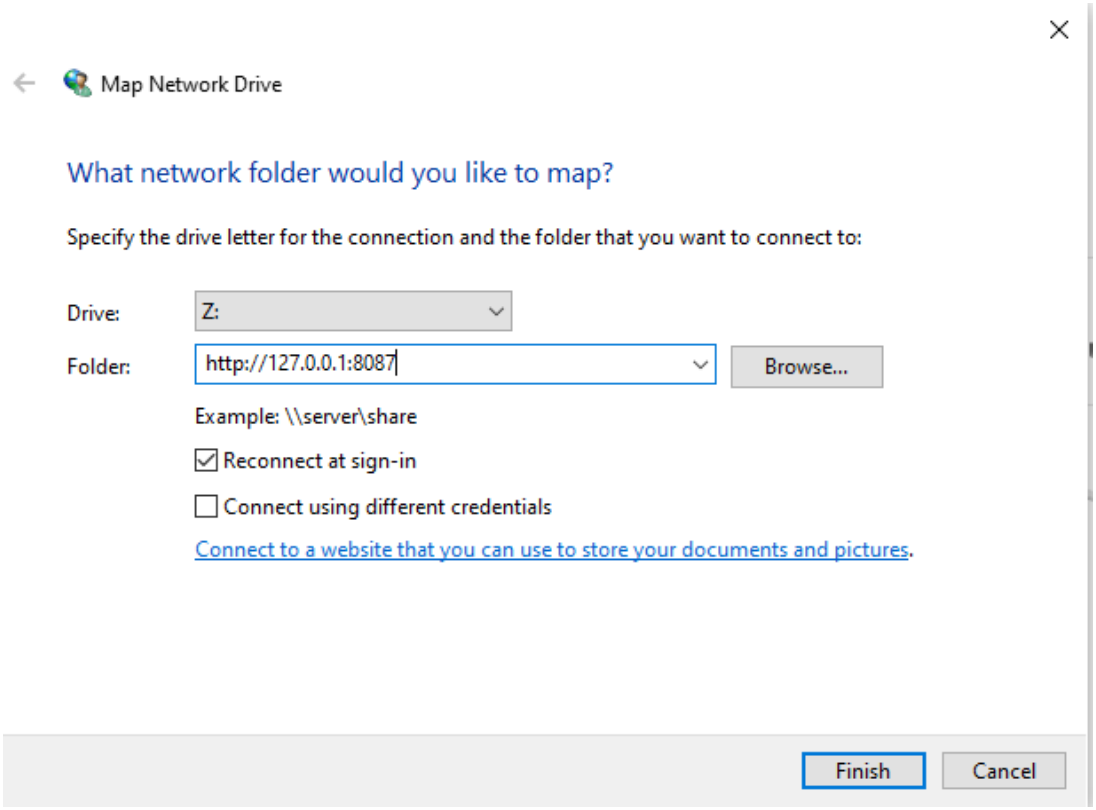

Fig. 2: Connecting a folder as a network drive via WebDAV.

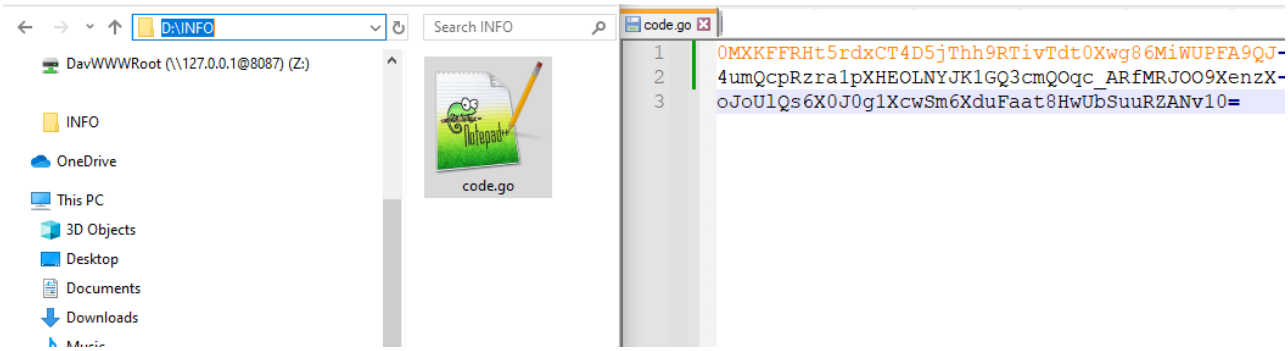

Fig. 3: Encrypted file in the D:\INFO folder.

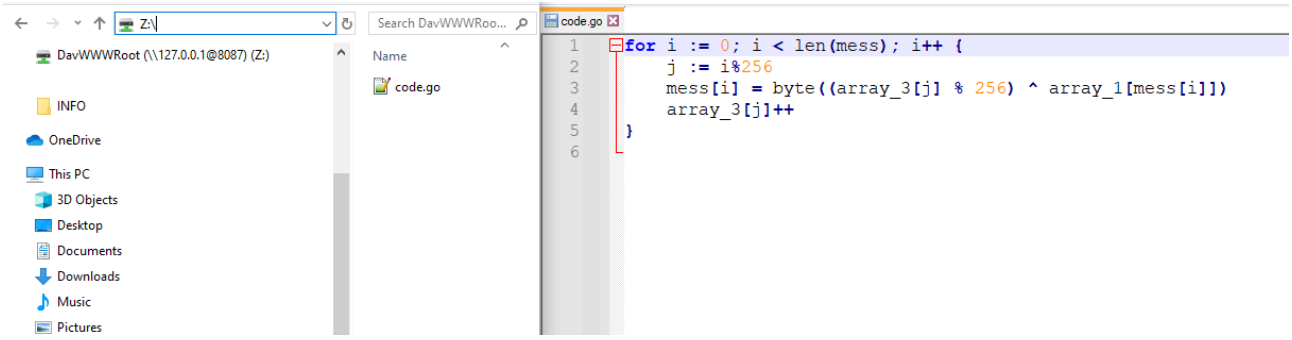

Fig. 4: The same file but already on disk Z.

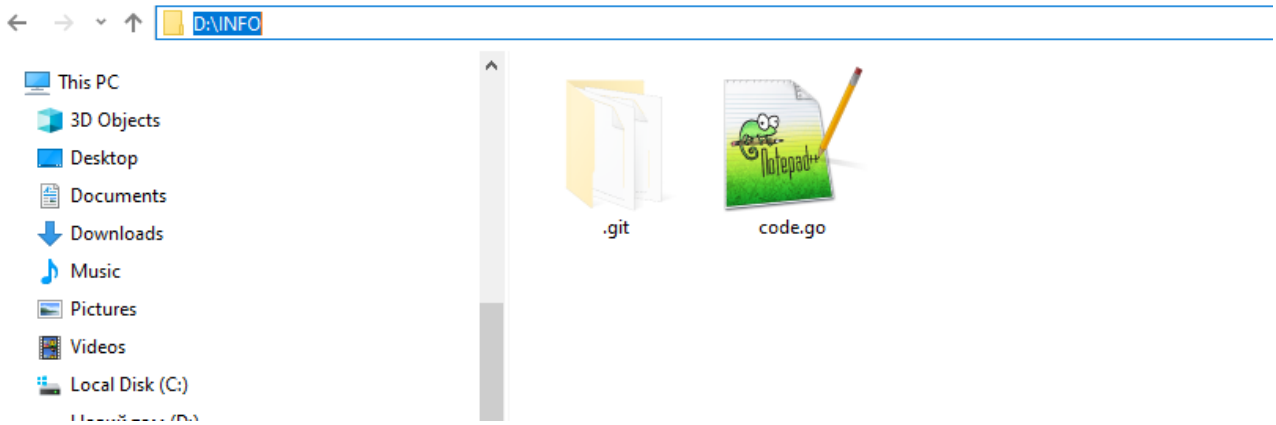

Fig 5: Initializing Git versioning in the encrypted folder D:\INFO.

Also, in Fig. 5. It is shown that in the D:\INFO folder GIT is initialized to handle file versioning. That is, you can work with the file through the virtual disk Z Fig. 4 save data and handle file versioning using the GIT repository, and the data remains completely encrypted.

TECHNICAL DESCRIPTION OF THE SOFTWARE ENCODER/DECODER ALGORITHM

Description of the Main Part of the Program

In this section we will consider both the program as a whole and the encryption/decryption algorithm itself that is used. Each byte of the original information sequence is encoded separately. For encoding and decoding, the program uses 3 arrays [256]array_1, [256]array_2, [256]array_3. The array_1 array is used to replace the original byte value with a random one, the array_2 array is used to return values where the value for the zero index from array_1 will be the index in array_2 and vice versa, and array_3 is used to obtain keys for imposing an XOR operation on the received byte value and contains the starting values of the keys for the elements of the array of received bytes.

The array_1 array contains random, unique values for each possible numeric byte value in the range 0 to 255:
Array_1 [0] = 126
Array_1[1] = 15
……
Array_1[255] = 200

The array array_2 contains the reverse "original" values for each random value from array_1:
Array_2[15] = 1
Array_2 [126] = 0
……
Array_2[200] = 255

The array_3 array contains random, unique values for each possible numeric byte value in the range 0 to 255:Array_3 [0] = 26
Array_3[1] = 150
Array_3[2] = 15

……
Array_3[255] = 1

Encryption

At start, the algorithm looks through each byte of the message, replacing the numeric value of the original byte with the value from array_1, where the numeric value of the byte of the original message is used as an index in array_1, then the resulting value is subject to the XOR operation of the corresponding key value from array_3. To obtain the key from array_3, the remainder of the division numeric value of the byte of the original message by 256 is taken (because array_3 contains only 256 elements), then the remainder of the division by 256 is taken from the resulting value, since the key length cannot exceed the message length, after obtaining the key from array_3, the key value for this index is incremented by 1, which creates unique keys. That is, if the message index is "0", then the key value from array_3 will be equal to 26 (from the example above), but when the message index is equal to 256, then the key value will be equal to 27, etc. Also, for a visual example of the encryption algorithm, Program 1 is given. It is written in the Golang programming language, where mass is an array of bytes of the original, open, message that needs to be encoded, i is the byte index in the message, array_1 is an array of values for replacing the original byte value, array_3 is an array of key values. Then the message obtained in this way is encoded with the AES cipher.

```
for i := 0; i < len(mess); i++ {
        j := i%256
        mess[i] = byte((array_3[j] % 256) ^ array_1[int(mess[i]])
        array_3[j]++
}
```

Program 1. Byte-by-byte encryption of the original mess message, which is an array of bytes, the program fragment is implemented in the Golang language.

Decryption

Decryption of the encrypted message occurs in the reverse order. First, the message is decoded by the AES algorithm, after which the key value is taken from the array_3 array, where the array index is the remainder of dividing the message byte index by 256, and then the resulting value, after applying the XOR operation, serves as an index in the array_2 array to obtain the original byte value. Next, the key value from the array_3 array for the resulting index is incremented in order to obtain the correct key value for the next consecutive index value. Also, for a visual example of the decryption algorithm, Program 2 is given.

```
for i := 0; i < len(mess); i++ {
        j := i%256
        mess[i] = byte(array_2[(array_3[j]%256)^int(mess[i])])
        array_3[j]++
}
```

Program 2. Byte-by-byte decryption of the encrypted message mess, which is an array of bytes.

Description of the Main Configuration Elements of the Program

Fig. 6 shows the program configuration file, which contains only 3 parameters that are needed to launch the program (they can also be specified in the parameters when launching the program without a configuration file). PORT - is the port on which WebDAV will be enabled to enable connection to the program as a network drive, FOLDER - is the directory used to store encrypted files, BIND - is the parameter responsible for which network interface can be connected (if the parameter is empty, then only the local address 127.0.0.1, if "*" - then everything, or you can specify specific IP addresses that are allowed to connect).

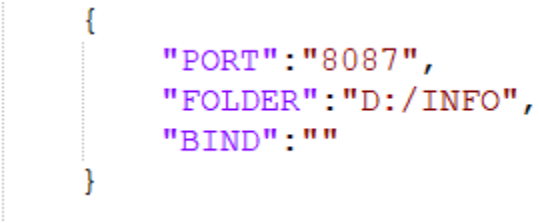

Fig. 6: Program configuration file.

DISCUSIONS

The developed program-method is new, easy to use, and without exaggeration an indispensable tool for ensuring the security of information. As mentioned earlier, the program allows you to encrypt/decrypt data "on the fly", which complicates the "interception" of information, because you need to work only with RAM to obtain information, which in turn is more difficult. Also, after running the program from a USB drive, the latter can be disconnected from the computer since the program runs from RAM. Also, the program written in the Golang version go1.22.2 language takes up about 7.5 Mb. And in comparison with BitLocker, it allows you to exchange information through the same Git repository as shown in Fig. 5., the sender of the file makes a commit, and the recipient, having a copy of the program in his arsenal, having received the latest changes on his computer, can decrypt the file and work with the data, it is possible to add addresses to the BIND parameter Fig. 6. at start, and a colleague or office employees will be able to work with the information without problems. In the same way, as a storage location, you can select a remote data storage by first connecting it to the computer and writing it in the FOLDER parameter, Fig. 6., and you do not need to think about the security of information on a remote source, since the data from the computer where the program is installed will already be transmitted in encrypted form. Also, according to Bruce Schneier [3], the program does not rely on the uniqueness of the encryption algorithm, which was previously unknown and is a secret. The most reliable algorithms are used in encryption and decryption, tested by time and the best cryptographers in the world. It should be noted that, the main purpose of the program is to work with file data on a work computer. In order to work with a certain type of information, save the file and close the program, and the file that will be saved through this program without the program will be information garbage that has no meaning for an attacker or for a person trying to gain unwanted access to the data. Another important aspect is the simplicity of implementation, which in turn provides high performance when working with files, that is, the difference when working with files on a local disk and through a "virtual" is imperceptible. Also, ease of use, the user does not need to have special knowledge in order to run the program, he only needs a USB drive and the desire to create something unique, without thinking that his developments can simply be stolen. But

it should also be mentioned that in the case of loss of a compiled program, it is impossible to restore the data, but in this case, sale of this program must be centralized like the sale of SSL certificates. And buyer could buy a version of the program with unique arrays of randomly distributed values and at the same time the buyer can be given a unique key assigned specifically to his program, which in the future, in the case of loss of a USB drive or program, will make it possible to get the user's program by key on the website of the software seller.

CONCLUSIONS

Advantages

- The proposed encoding method implements the Vernam algorithm, which is one of the most powerful tools for encrypting information, and the encrypted message is “sealed” with an AES cipher on top, which makes decryption impossible without an encryption program.
- Without a decoder program, decryption of data is impossible. This point is highlighted separately since this is the main task of the program - to make it impossible to decrypt data without an encoder program.
- High performance when working with both small files of 10 - 100 MB, and with files of 1 GB and more. When working through a "virtual" disk, the user does not feel any delays when working with files, which in modern conditions, plays one of the key indicators when choosing software.
- Low cost with high reliability and data security. Due to the fact that the software is very simple but at the same time extremely reliable in terms of its intended purpose, the buyer will be more than satisfied after purchasing the software.

Flaws

One of the main disadvantages is that for each user arrays array_1 and array_3 must be unique, but according to the probability theory of such combinations there can be (250!)2 and the key for AES encryption must also be unique. And as was said earlier, for sale need compile a program for each user with their unique arrays and a unique key value for AES encryption.

References

1. Mao W. Modern Cryptography: Theory and Practice 2005. ISBN 978-5-8459-0847-6
2. Bruce Schneier. Applied Cryptography. ISBN 0-471-11709-9
3. Kutsman Vladyslav. Distributed File System Based on a Relational Database. Open Journal of Applied Sciences 2023(05). http://doi:10.4236/OJAPPS.2023.135051.