

ANALYSIS AND USE OF CRYPTOGRAPHY TECHNIQUES IN PROGRAMMING LANGUAGE C#



SCAN ME

Naim BAFTIU 

University "Ukshin Hoti" Prizren, Faculty of Computer Science, naim.baftiu@uni-prizren.com

Article history:

Submission 04 August 2020

Revision 13 September 2020

Accepted 27 November 2020

Available online 31 December 2020

Keywords:

Cryptography,

Algorithm,

Substitution,

Cipher,

C #.

Abstract

Cryptography is an old idea and science, but its approach exists and plays a large role in modernization today. Conventional cryptographic techniques form the basis of today's cryptographic algorithm. The different categories of algorithms have their respective features; internally, in performance and implementation. Cryptographic schemes and mechanisms have undergone continuous improvement. The application of cryptography has grown increasingly, ranging from limited use in state institutions to widespread use by private individuals and companies. The increased use of the Internet has significantly influenced the nature of applications and the way we communicate. Data security dictates the use of different cryptographic techniques. For this reason, we analyze in detail the various coding techniques by evaluating their performance and efficiency. Regarding the new paradigms in cryptography there are also new cryptographic schemes whose application requires detailed study and analysis. The classical cryptography algorithm is the oldest algorithm that was used long before the cryptographic system was discovered. Currently, the system has been widely applied to secure data, and using new methods in a way to improve existing methods. In this thesis the use of cryptographic methods using the C# programming language will be discussed.

1. Introduction

The purpose of this research is to analyze Cryptography techniques in a more convenient way to do more, where you will find more techniques than I teach in theoretical practice, try to introduce them through C# programming languages, and use Visual Studio. The field of analyzing large amounts of data is current because the number of data is increasing every day. Comparing and finding differences in cryptography techniques is also important when analyzing. Each of the techniques has its own characteristics, and therefore Cryptography is a very broad field of research. It involves algorithms and techniques from different disciplines. First, to make the selection of techniques, it is important to specify the data to be analyzed in order to know how to make the algorithm selection.

This study will use comparative methods in order to reach conclusions regarding the performance of cryptography techniques. All the research will be done in a practical way by implementing different code in Visual Studios C# programming

language. The increasing use of the Internet has significantly influenced the nature of applications and the way we communicate. Data security dictates the use of different cryptographic techniques. For this reason, we analyze in detail the various coding techniques by evaluating their performance and efficiency. To understand the importance of applying cryptographic techniques it is enough to know the wide range of applications and services where we have sensitive data. Data storage is one of the basic personal but also operational requirements in ensuring the success of a business initiative such as different banks or companies. The new encryption forms dictated by the computational difficulties of specific schemes consist of data processing without having a private key to the data, including format encoding, symmetric search encryption, functional encryption and homomorphism encryption. Regarding the new paradigms in cryptography there are also new cryptographic schemes whose application requires detailed study and analysis. Quantum cryptography and Turing-complete encryption programs are really new forms and currently have a

good theoretical basis. What poses a challenge in many perfect coding schemes is closely related to the compositionality that is the functional axis of applying a cryptographic algorithm.

2. Cryptography Today

Cryptography, in general, is the science of art to preserve the confidentiality of data. Furthermore, there is also a sense of understanding the study of mathematical techniques related to information security aspects such as data confidentiality, and data validation and not all aspects of information security are addressed by cryptography. There are four basic purposes of cryptography which is also an aspect of information security, namely:

1. Confidentiality is a service used to keep information content from anyone who receives it with a secret key to unlock the information that is encrypted.
2. Data Integrity in order to maintain data integrity, the system must have the ability to determine data manipulation by parties who are not entitled to do, inter alia, enter, delete and sign data in current data.
3. Authentication, which relates to identification as a whole system and to the information itself. The parties must communicate on their own. The information presented through the channel must be verified, the authenticity, the content of the data and the time of delivery.
4. Non-repudiation is an attempt to prevent the denial of remittances and the creation of information that transmits or makes.

In cryptography, substitution is a type of identification method, in which each character in plaintext replaces the cipher text with the regular system. The recipient of the message can read the message after performing the decoding process on a message using a method identical to the method used by the sender. The replacement method only changes the characters without changing the structure of the message itself, in contrast to the shift method which changed the wording but did not change the character in the message.

3. The use of Methods

The replacement method is divided into several types, namely:

1. Simple substitution is a method of substitution for character.
2. Substitute polygraph is a method that replaces two characters or more.

3. Mono-alphabetical replacement is a method that uses a fixed pattern (eg: Galih password)
4. Poly replacement is the method that the pattern was different as long as the message was.

3.1. Coding and Decoding

If we want to keep information confidential, we have two options: to hide the existence of the information or to make the information incomprehensible. Coding is nowadays widely used and is one of the most commonly encountered techniques in various applications. In electronic money schemes, encryption is used to protect symbolic transaction data such as account numbers and transaction quantities, digital signatures can replace handwritten signatures or credit card authorizations, and public key encryption can provide confidentiality. There are a large number of systems that cover such applications, from classic transactions to complex bank payment schemes. Before we get to the basic coding forms and schemes let's look at a number of concepts that we will come across.

- Plain Text - The original or original message to be sent is known as open text.
- Cipher Text - The message on which the cipher is applied is known as the cipher text. In cryptography the initial message "hello" can be converted to such incomprehensible form "Ajd672 # @ 91uk".
- Encoding - The process of converting an open text into encoded text is known as encoding. Cryptography uses encryption techniques to send confidential messages over an unsecured line of communication. The coding process requires two things: a coding algorithm and a key. The encryption algorithm means the technique used and the encryption occurs on the sender side.
- Decoding - An inverse process of decoding is known as decoding, where the encoded text is converted to the original text. Cryptography uses decryption techniques on the recipient's side to retrieve the original message from the encoded text.
- Key - A key is an alphanumeric text or can be a special symbol.

Private keys are used for signature; public keys are used for verification: For example, to sign something digitally, we encrypt it with our private key (usually a hash is created and encrypted). Anyone can decode this data (deciphering the hash value and comparing their hash value to the previous value) and verify that since it is signed by our private key then the data belongs to us. Key selection in cryptography is quite important as it directly affects the security of the encryption algorithm. For example, if

Anna uses key 3 to encode the original text "President" we will have the text as follows "Suhvlghqw".

3.2. Cryptographic Algorithms in Use

The most commonly used algorithms in data storage are: 3DES, RSA, Blowfish, Two fish and AES. Also a trend regarding the use of encryption algorithms is their presence in security certificates, various protocols and public-key infrastructures. DES is the first coding standard recommended by NIST and operates with a key (56 bits) and a 64-bit data block. DES has a Festal structure, operates with a bit of bits and is no longer considered a secure algorithm. The 3DES is a DES upgrade, operating with a 64bit block and 192 bit (168 bit) switch. In this algorithm the cipher is applied three times to increase the level and average security time. It is a known fact that 3DES is the slowest method among other block chains. AES (Rijndael) is a block chain and usually operates at 256 bits out of three possible 128, 192 or 256-bit key lengths. It encodes 128-bit data block in 10, 12 and 14 rounds depending on the size of the key. Encryption in AES is fast and flexible; it can be implemented on different platforms especially on small devices, which is an advantage already. The Blowfish algorithm created by Schneider and known since 1993, does not turn out to be broken, at least not completely.

This algorithm is optimized in hardware applications, though like all other figures it is often used in software applications. It is a 64-bit block and receives a variable length key, from 32 to 448 bits: usually 128 bits. Bluefish has a very good performance compared to AES, DES and 3DES. The RSA is named after its creators (Rivets, Shamir, and Adelman) have some operational limitations. With the most commonly used variant (PKCS # 1 v1.5), with a 1024-bit RSA key size, this algorithm can encode a message up to 117 bytes, and receive a 128-byte encoded message. Hash functions are good "randomizers" (the output of a hash function does not display known and expected structures) and this makes it quite suitable for building more complex schemes with good security features and moreover hash functions do not they don't even have keys. SHA is a common term for a cryptographic family of hash functions. Four SHA functions were then added (SHA-224, SHA-256, SHA-384 and SHA-512, known as 'SHA2'). SHA-256 and SHA-512 are relatively new and well known functions. SHA-256 is used by the DKIM (Domain Keys Identified Mail) framework for email signatures in controlling spam and phishing phenomena. SHA-512 is supported by True Crypt software for encrypting disk space and virtual images. Also SHA-256 and SHA-512 are recommended for DNSSEC (Domain Name System Security Extensions) regarding security services that can be added to the DNS protocol. Also hash functions can be used in SSL / TLS technologies that

are standard for encrypting connections between servers and web browsers.

3.3. Cryptography Techniques

Cryptography is a broad field but we have analyzed some of the classical encryption techniques such as:

- Caesar Cipher,
- Monoalphabetic Ciphers,
- Playfair Cipher,
- Hill Cipher,
- Polyalphabetic Ciphers,
- Vigenere
- And Rail Fence.

While some of the analog encryption techniques are: Data Encryption Standard (DES) as well Advanced Encryption Standard (AES).

3.4. Caesar Cipher

It is a technique in which the letters of the original text are replaced by letters, numbers or other symbols. We can divide traditional symmetric figures into two broad categories: replacement figures and transposition figures.

If the symbols in plain text are alphabetical characters, we replace one character with another. For example, we can replace the letter A with the letter D and the letter T with the letter Z. If the symbols are digits (0 to 9), we can replace 3 with 7 and 2 with 6.

The implementation of the said algorithm in C # follows:

```
namespace Algoritmet
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    using System.ComponentModel.Composition;
    using System.Data.Common;
    public class Ceaser : SecurityAlgorithm
    {
        readonly int key;
        #region Constructor
        public Ceaser(int key)
        {
            this.key = key;
        }
        #endregion
        #region Public Methods
        public override string Encrypt(string plainText)
        {
            return Process(plainText, Mode.Encrypt);
        }
    }
}
```

```

publicoverride string Decrypt(string cipher)
{
    return Process(cipher, Mode.Decrypt);
}
#endregion
#region Private Methods
private string Process(string message, Mode mode)
{
    string result = string.Empty;
    foreach (char c in message)
    {
        var charposition = alphabet[c];
        var res = Common.GetAlphabetPosition(charposition, key, mode);
        result += alphabet.Keys.ElementAt(res % 26);
    }
    return result;
}
#endregion
}

```

3.5. Mono Alphabetic Ciphers

In a mono-alphabetic cipher, a character (or symbol) that is plaintext is always changed to the same character (or symbol) in the cipher text regardless of its position in the text. For example, if the algorithm says that the letter A in plaintext is changed to letter D, then every letter A is changed to letter D. In other words, the relationship between letters in plaintext and cipher text are one-to-one. The simplest mono-alphabetic cipher is the extra digit (or change digit). Assume that plaintext consists of lowercase letters (a to z) and that cipher text consists of uppercase letters (from A to Z). To be able to apply mathematical operations to plaintext and cipher text, we assign numeric values to each letter.

Implementing the said algorithm in C# does the following:

```

namespace Algoritmet
{
    using System;
    using System.Collections.Generic;
    using System.Linq;
    public class Monoalphabetic : SecurityAlgorithm
    {
        readonly Dictionary<char, char> _alphabetShuffled;
        readonly Dictionary<char, char> _alphabetShuffledReverse;
        public Monoalphabetic()
        {
            _alphabetShuffledReverse = new Dictionary<char,
char>();

```

```

        _alphabetShuffled = new Dictionary<char, char>();
        ShuffleAlphabet();
    }
    #region Public Methods
    publicoverride string Encrypt(string plainText)
    {
        return Process(plainText, Mode.Encrypt);
    }
    publicoverride string Decrypt(string cipherText)
    {
        return Process(cipherText, Mode.Decrypt);
    }
    #endregion
    #region Private Methods
    private string Process(string token, Mode mode)
    {
        string result = "";
        for (int i = 0; i < token.Length; i++)
        {
            switch (mode)
            {
                case Mode.Encrypt:
                    result += _alphabetShuffled[token[i]];
                    break;
                case Mode.Decrypt:
                    result += _alphabetShuffledReverse[token[i]];
                    break;
            }
        }
        return result;
    }
    private void ShuffleAlphabet()
    {
        Random r = new Random(DateTime.Now.Millisecond);
        var alphabetCopy = alphabet.Keys.ToList();
        foreach (var character in alphabet.Keys)
        {
            int characterPosition = r.Next(0, alphabetCopy.Count);
            char randomCharacter = alphabetCopy[characterPosition];
            _alphabetShuffled.Add(character, randomCharacter);
            _alphabetShuffledReverse.Add(randomCharacter, character);
            alphabetCopy.RemoveAt(characterPosition);
        }
    }
    #endregion
}

```

3.6. Poly Alphabetic Ciphers

In poly alphabetic substitution, each appearance of one character may have another substitution. The relationship between a character in plaintext and a character in cipher text is one-to-many. For example, "a" may be coded as "D" at the beginning of the text, but as "N" in the middle. Poly alphabetic figures have the advantage of concealing the frequency of basic language paper. To create a poly alphabetic cipher, we need to make each character cipher text, which depends on both the corresponding plaintext characters and the position of the plaintext in the message. This implies that our key must be a stream of sub keys, in which each sub key depends somewhat on the position of the simple character that that sub key uses for encryption.

Implementing the said algorithm in C # does the following:

```
public class AutoKey : SecurityAlgorithm
{
    #region Member Variables
    string _key;
    #endregion
    #region Constructor
    public AutoKey(string key)
    {
        this._key = key;
    }
    #endregion
    #region Public Methods
    public override string Encrypt(string plainText)
    {
        return Process(plainText, Mode.Encrypt);
    }
    public override string Decrypt(string cipher)
    {
        return Process(cipher, Mode.Decrypt);
    }
    #endregion
    #region Private Methods
    private string Process(string message, Mode mode)
    {
        _key = DuplicateKey(message);
        return Common.Shift(message, _key, mode, alphabet);
    }
    private string DuplicateKey(string message)
    {
        if (_key.Length < message.Length)
```

```
        {
            int length = message.Length - _key.Length;

            for (int i = 0; i < length; i++)
            {
                _key += message[i];
            }
        }
        return _key;
    }
    #endregion
}
```

3.7. Playfair Cipher

Known as multi-letter cipher. It treats diagrams in the original text as a single unit and translates these units into encoded diagrams. The Play fair algorithm is based on using a 5X5 matrix of letters built using a word. Now the question is how to fill that 5x5 matrix? - To fill it, we need a keyword or a message, after that you fill in the 5x5 word letters from left to right and from top to bottom and then fill in the remaining parts of the matrix with the remaining letters in alphabetical order. The letters I and J are treated as one letter and they are placed in the same matrix box as this - I / J. Plaintext is encoded two letters at a time, so you must first put together plain text. Discovered by British scientist Charles Wheatstone in 1854, it was used as a standard system by the British Army in World War I and the US Army and other allied forces during World War II.

Implementing the said algorithm in C # does the following:

```
public class PlayFair : SecurityAlgorithm
{
    string key;
    public PlayFair(string key)
    {
        this.key = key;
    }
    #region Public Methods
    public override string Encrypt(string plainText)
    {
        return Process(plainText, Mode.Encrypt);
    }
    public override string Decrypt(string cipherText)
    {
        return Process(cipherText, Mode.Decrypt);
    }
    #endregion
    #region Private Methods
```

```

private string Process(string message, Mode mode)
{
//Key:Charcater
//Value:Position
Dictionary<char, string> characterPositionsInMatrix =
new Dictionary<char, string>();
//Key:Position
//Value:Charcater
Dictionary<string, char> positionCharacterInMatrix =
new Dictionary<string, char>();
FillMatrix(key.Distinct().ToArray(),
characterPositionsInMatrix, positionCharacterInMatrix);
if (mode == Mode.Encrypt)
{
message = RepairWord(message);
}
string result = "";
for (int i = 0; i < message.Length; i += 2)
{
string substring_of_2 = message.Substring(i, 2); //get characters
from text by pairs
//get Row & Column of each character
string rc1 = characterPositionsInMatrix[substring_of_2[0]];
string rc2 = characterPositionsInMatrix[substring_of_2[1]];
if (rc1[0] == rc2[0]) //Same Row, different Column
{
int newC1 = 0, newC2 = 0;
switch (mode)
{
case Mode.Encrypt: //Increment Columns
newC1 = (int.Parse(rc1[1].ToString()) + 1) %
5;
newC2 = (int.Parse(rc2[1].ToString()) + 1) %
5;
break;
case Mode.Decrypt: //Decrement Columns
newC1 = (int.Parse(rc1[1].ToString()) - 1) %
5;
newC2 = (int.Parse(rc2[1].ToString()) - 1) %
5;
break;
}
newC1 = RepairNegative(newC1);
newC2 = RepairNegative(newC2);
result +=
positionCharacterInMatrix[rc1[0].ToString() +
newC1.ToString()];
result +=
positionCharacterInMatrix[rc2[0].ToString() +
newC2.ToString()];
}
}

```

```

}
elseif (rc1[1] == rc2[1]) //Same Column, different Row
{
int newR1 = 0, newR2 = 0;
switch (mode)
{
case Mode.Encrypt: //Increment Rows
newR1 = (int.Parse(rc1[0].ToString()) + 1) %
5;
newR2 = (int.Parse(rc2[0].ToString()) + 1) %
5;
break;
case Mode.Decrypt: //Decrement Rows
newR1 = (int.Parse(rc1[0].ToString()) - 1) %
5;
newR2 = (int.Parse(rc2[0].ToString()) - 1) %
5;
break;
}
newR1 = RepairNegative(newR1);
newR2 = RepairNegative(newR2);
result +=
positionCharacterInMatrix[newR1.ToString() +
rc1[1].ToString()];
result +=
positionCharacterInMatrix[newR2.ToString() +
rc2[1].ToString()];
}
else //different Row & Column
{
//1st character: row of 1st + col of 2nd
//2nd character: row of 2nd + col of 1st
result +=
positionCharacterInMatrix[rc1[0].ToString() +
rc2[1].ToString()];
result +=
positionCharacterInMatrix[rc2[0].ToString() +
rc1[1].ToString()];
}
return result;
}
private string RepairWord(string message)
{
string trimmed = message.Replace(" ", "");
string result = "";
for (int i = 0; i < trimmed.Length; i++)
{
result += trimmed[i];
}
}

```

```

if (i < trimmed.Length - 1 && message[i] == message[i + 1])
//check if two consecutive letters are the same
    {
        result += 'x';
    }
}

if (result.Length % 2 != 0)//check if length is even
    {
        result += 'x';
    }
return result;
}

private void FillMatrix(ICollection<char> key, Dictionary<char,
string> characterPositionsInMatrix, Dictionary<string, char>
positionCharacterInMatrix)
    {
char[,] matrix = new char[5, 5];
int keyPosition = 0, charPosition = 0;
List<char> alphabetPF = alphabet.Keys.ToList();
alphabetPF.Remove('j');
for (int i = 0; i < 5; i++)
    {
for (int j = 0; j < 5; j++)
        {
if (charPosition < key.Count)
            {
matrix[i, j] = key[charPosition]; //fill matrix with
key
alphabetPF.Remove(key[charPosition]);
charPosition++;
}
else //key finished...fill with rest of alphabet
            {
matrix[i, j] = alphabetPF[keyPosition];
keyPosition++;
}
string position = i.ToString() + j.ToString();
//store character positions in dictionary to avoid searching
everytime
characterPositionsInMatrix.Add(matrix[i, j],
position);
positionCharacterInMatrix.Add(position, matrix[i,
j]);
}
}
}

private int RepairNegative(int number)
    {
if (number < 0)
        {

```

```

number += 5;
}
return number;
}
}

```

3.8. Vigenere Cipher

Improved Vernami encryption proposed by a military officer named Joseph Mauborgne. Use a key that is as long as the message so that the key does not need to be repeated. The key is used to encrypt and decrypt the single message and then leave it in use. Each new message requires a new key as long as the length of the new message. The schema is invincible produces random output that holds no statistical relation to the original text. Because the encrypted text contains no information about anything that is related to the original text, there is no way to break the code.

Implementing the said algorithm in C # does the following:

```

namespace Algorithmet
{
using System.Collections.Generic;
using System.ComponentModel.Composition;
using System.Data.Common;
public class Vigenere : SecurityAlgorithm
    {
string key;
public Vigenere(string key)
        {
this.key = key;
}
#region Public Methods
public override string Encrypt(string plainText)
        {
return Process(plainText, Mode.Encrypt);
}
public override string Decrypt(string cipherText)
        {
return Process(cipherText, Mode.Decrypt);
}
#endregion
#region Private Methods
private string Process(string message, Mode mode)
        {
key = key.ToString().ToLower().Replace(" ", "");
key = DuplicateKey(message, key);
return Common.Shift(message, key, mode, alphabet);
}
private string DuplicateKey(string message, string key)

```

```

    {
    if (key.Length < message.Length)
        {
    int length = message.Length - key.Length;

    for (int i = 0; i < length; i++)
        {
            key += key[i % key.Length];
        }
    }
    return key;
    }
#endregion
}

```

3.9. Rail Fence

Simplified displacement coding. Plaintext is written as a sequence of diagonals and then read off as a sequence of lines. In this technique, the characters of plain text are written in diagonal form at the beginning. This arrangement forms two rows, which resemble the rail track. That is why it is called Rail Fence or Rail Fence. After both rows are produced, the digit text is read consecutively. In the figure of Rail Fence, the plain text is written down and diagonally on successive rails of a fictional fence. When we reach the bottom rail, we pass upward moving diagonally, once we reach the top rail, the direction is changed again. Thus, message alphabets are written in a zigzag manner. After each alphabet is written, the individual rows are combined to obtain the encoding text.

Implementing the said algorithm in C # does the following:

```

namespace Algorimet
{
using System;
using System.ComponentModel.Composition;
public class RailFence : SecurityAlgorithm
    {
    readonly int key;

    public RailFence(int key)
        {
    this.key = key;
        }
#region Public Methods
public override string Encrypt(string plainText)
    {
    return Process(plainText, Mode.Encrypt);
    }

```

```

public override string Decrypt(string cipherText)
    {
    return Process(cipherText, Mode.Decrypt);
    }
#endregion
#region Private Methods
private string Process(string message, Mode mode)
    {
    int rows = key;
    int columns = (int) Math.Ceiling((double) message.Length / (double) rows);
    char[,] matrix = FillArray(message, rows, columns, mode);
    string result = "";
    foreach (char c in matrix)
        {
            result += c;
        }
    return result;
    }
private char[,] FillArray(string message, int rowsCount, int columnsCount, Mode mode)
    {
    int charPosition = 0;
    int length = 0, width = 0;
    char[,] matrix = new char[rowsCount, columnsCount];
    switch (mode)
        {
    case Mode.Encrypt:
        length = rowsCount;
        width = columnsCount;
        break;
    case Mode.Decrypt:
        matrix = new char[columnsCount, rowsCount];
        width = rowsCount;
        length = columnsCount;
        break;
        }
    for (int i = 0; i < width; i++)
        {
    for (int j = 0; j < length; j++)
        {
    if (charPosition < message.Length)
        {
            matrix[j, i] = message[charPosition];
        }
    else
        {
            matrix[j, i] = '*';
        }
        charPosition++;
        }
    }
    }

```



```

    }
    } return matrix;
}
}

```

4. Comparison of Algorithms and Conclusions

We will first compare the term cryptography, what it is in a broader sense, comparing it with the term encryption. Cryptography and Encryption have some differences which we will mention below:

- Cryptography is the study of concepts such as encryption, decryption, used to provide secure communication while Encryption is the process of encoding a message with an algorithm.
- Cryptography can be considered a field of study, which incorporates many techniques and technologies, while Encryption is more of a mathematical nature and algorithms.
- Cryptography, being a field of study has a broader category and range of techniques, where encryption is one such technique, while Encryption is one of the aspects of Cryptography that can codify the communication process efficiently.
- Cryptography has a symmetric and asymmetric version, with a concept of a shared rather than a shared key, while Encryption follows the same approach with some specific terms such as cipher text, plaintext, and cipher.
- Cryptography involves working with algorithms with basic cryptographic properties while Encryption is one of the subcategories of Cryptography that uses mathematical algorithms called digits.
- The areas of cryptography include computer programming, algorithm, mathematics, information theory, transmission technology while Encryption is more digitized in nature since the modern age.
- Cryptography includes two major components called Encryption and Decryption while Encryption is a process of storing information to prevent unauthorized and illegal use.

4.1. Comparison of Mono-Alphabetic and Poly Alphabetic Algorithms

Mon A mono-alphabetic digit is one where each symbol in the input (known as 'plaintext ') is mapped to an output fixed symbol (referred to as the predicted figure). Poly alphabetic encoding is any substitution-based cipher using multiple substitution

characters. In the mono-alphabetic figure, after a key is selected each alphabetical character of the plaintext is marked with a unique alphabetic character of the cipher text. On the other hand, in the poly alphabetic code, any alphabetical character of the plaintext can be listed in the alphabetical letters " m " of a cipher text. In mono alphabetic cipher, the relationship between a plaintext character and characters in cipher text is one-to-one, while in Poly alphabetic the relationship between a plaintext character and characters in cipher text is one-to-many.

4.2. Creaser's algorithm

A mono-alphabetic cipher is where each plaintext letter is replaced by another letter to form cipher text. It is a simpler form of replacement figure scheme. This cryptosystem is commonly called Shift Cipher. The concept is to replace each letter of the alphabet with another letter that has been "moved" with a fixed number between 0 and 25. For this type of scheme, both the sender and the receiver agree on a "secret shift number" for the alphabet relocation. This number that is between 0 and 25 becomes the encryption key. Caesar Cipher is not a secure cryptosystem because there are only 26 possible keys to try. An attacker can perform an exhaustive search of keystrokes with limited computing resources available.

4.3. Play fair Cipher

In the Play fair cipher, a master table is first created. The main table is a 5 x 5 alphabet grid table that acts as the key to plaintext encryption. Each of the 25 letters of the alphabet must be unique and one alphabet (usually J) is removed from the table as we only need 25 alphabets instead of 26. If simple writing contains J, then it is replaced by I. The sender and receiver decide on a separate key, and in a main table, the first characters (going left to right) place the key, excluding the same letters. The rest of the table will be filled with the remaining letters of the alphabet, in natural order. It is also a replacement cipher and is difficult to break compared to simple replacement cipher. Cryptanalysis is also possible in the Play fair cipher; however, it would be possible to replace 625 possible letter combinations (25x25 characters) instead of 26 different characters. Play The Play fair cipher was mainly used to protect important but not critical secrets, as it is fast to use and requires no special equipment.

4.4. Vigenere Cipher

Vigenere Cipher was created by taking Caesar's standard cipher to reduce the effectiveness of cryptanalysis in the cipher text and to make a cryptosystem stronger. Certainly this is significantly safer than an ordinary Caesar figure. In history, it was regularly used to protect sensitive political and military information. It was referred to as the unbreakable figure because of the difficulty it posed to cryptanalysis.

Vigenere encryption becomes a cryptosystem of perfect secrecy, called One-Time-Pad. One-Time-Pad has several features:

- The word length is the same as the plaintext length.
- The keyword is a series of randomly generated alphabets.
- The keyword is used only once.

Let's say we encrypt the name "point" with One-Time-Pad. It is a 5 letter text. To break the meticulous figure with brute force, you have to try all the keys options and calculate for $(26 \times 26 \times 26 \times 26 \times 26) = 26^5 = 11881376$ times. This is for a message with 5 characters or characters. Thus, for a longer message the computation increases exponentially for each additional letter. This makes it logically impossible to break the cipher text with brute force.

5. Conclusion

Cryptography encompasses various techniques and technologies including algorithms, mathematics, information theory, transmission, encryption, and more. Information security issues are critical for individuals, institutions and companies worldwide. Nowadays it is difficult to consider a computer system completely secure without the encryption technology. Cryptographic figures and algorithms are basic mechanisms for protecting data and communications. In this paper we have analyzed cryptographic algorithms, their cryptographic features, and created an application where we can encrypt and decrypt various texts. Then we showed each algorithm from an example and how that algorithm works, and finally we compared some of the algorithms in detail. Cryptographic technologies are advancing: new attack techniques, designs and implementations of widely studied algorithms. So it is important to analyze their structure, efficiency and scalability metrics depending on what we need. New coding paradigms that are significant in new systems and technologies and have access to computational theory. There is a constant contradiction between efficiency and computing power. For this reason, we need to solve new difficult problems to provide better cryptographic schemes.

References

1. Albahari, J., Albahari, B. (2012). *C# 7.0 in a Nutshell*: ISBN: 1491987650 ISBN13: 9781491987650 ASIN: 1491987650.
2. Schneier B. (2006), *Applied Cryptography*, Second Edition: Protocols, Algorithms, and Source Code in C (cloth) ISBN: 0471128457, Publication 1996.

3. Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanston. (1997). *Handbook of Applied Cryptography*, ISBN: 0-8493-8523-7-1996.
4. Denning, Dorothy E, (1999). *Information Warfare and Security*, Addison-Wesley, ISBN 0-201-43303-6. *Introduction to Cyber Information warfare*.
5. Stallings, W. (2006), *Cryptography and Network Security Principles and Practices*, Fourth Edition. Pub. November 16, 2005 Print ISBN-10, 0-13-187316-4 Print ISBN-13 978-0-13-187316-2.
6. Kelsey, J., Schneier, B., and Hall, C. (1998), "Cryptanalytic Attacks on Pseudo-random Number Generators." *Proceedings, Fast Software Encryption*. <http://www.schneier.com/paper-prngs.htm>
7. Oppliger, R., and Rytz, R. (2005), "Does Trusted Computing Remedy Computer Security Problems?" *IEEE, Security and Privacy*, March/April 2005.
8. Biham, E., and Shamir, A. (2000) "Power Analysis of the Key Scheduling of the AES Candidates" *Proceedings, Second AES Candidate Conference*, 24 October 2000.