Задание по технологиям защиты информации

На тему: Электронная цифровая подпись

Горшков Ян

**Задание:**

Построить графическую программу, эмулирующую работы цифровой подписи. Программа должна подписывать файл, и проверять его на целостность данных и принадлежность к владельцу.

**Теоретические сведения:**

Электронная цифровая подпись (ЭЦП) — это реквизит электронного документа, полученный в результате криптографического преобразования информации с использованием закрытого ключа подписи и позволяющий проверить отсутствие искажения информации в электронном документе с момента формирования подписи, принадлежность подписи владельцу сертификата ключа подписи, а в случае успешной проверки подтвердить факт подписания электронного документа. Принцип работы ЭЦП изображен на рисунке 1.



Рисунок 1 – Принцип работы ЭЦП

При помощи определенной хеш-функции вычисляется хеш-значение файла. Для шифрования хеш-значения используется закрытый ключ. После шифрования, данная информация добавляется к файлу. Таким образом происходит процесс подписи документа.

Для проверки подписанных данных происходит следующий процесс:

Из файла читается зашифрованное хеш-значение, и дешифруется открытым ключом. Также, для самого файла высчитывается хеш-значение при помощи определенной хеш-функции. Если эти два значения будут равны, то документ считается подтверждённым.

**Разработанная программа:**

Программа разрабатывалась на языке C# при помощи Windows Forms. Интерфейс программы изображен на рисунке 2.

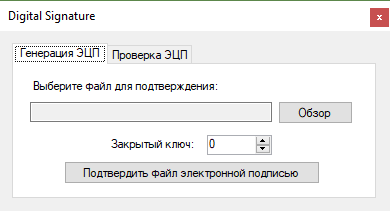


Рисунок 2 – Интерфейс программы

Суть программы заключается в следующем: Для генерации ЭЦП, пользователь выбирает необходимый файл, указывает закрытый ключ, и подтверждает генерацию подписи. Подпись представляет собой зашифрованной хеш-значение, полученное методом ComputeHash() класса MD5. Данное значение шифруется при помощи шифра Цезаря. Зашифрованное хеш-значение помещается в имя выбранного файла, и он помещается в папку с исполнительным файлом программы. После этого, пользователю будет выведено диалоговое окно с открытым ключом.

Для проверки файла необходимо выбрать нужный файл, в имени которого будет присутствовать зашифрованное хеш-значение, указать открытый ключ, и нажать на кнопку “Проверить”. В этом случае, при помощи метода ComputeHash() будет найдено хеш-значение файла, и зашифровано при помощи открытого ключа. Данное значение будет сравнено с именем файла и при их совпадении будет выведено соотвутствующее сообщение. Окно программы изображено на рисунке 3.

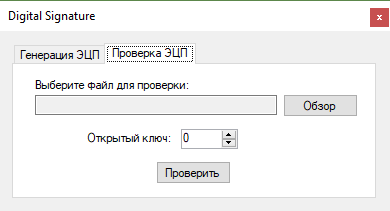


Рисунок 3 – Проверка ЭЦП

**Вывод:**

Данная практическая работа ставила перед собой цель – изучение электронной цифровой подписи, и программно реализовать её используя шифр Цезаря. ЭЦП позволяет подписать владельцу документ, и проверить целостность данного документа и его владельца используя открытый ключ. В то время, как закрытый ключ известен лишь одному владельцу.

**Листинг:**

using System;

using System.IO;

using System.Security.Cryptography;

using System.Windows.Forms;

using Microsoft.VisualBasic;

namespace DigitalSignature

{

public partial class Form1 : Form

{

private string temp;

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

MD5 sha = MD5.Create();

// byte[] b = new byte[3] { 5, 3, 2 };

using (Stream str2 = new StreamReader("Readme.txt").BaseStream)

{

byte[] b1 = sha.ComputeHash(str2);

string str = "";

MessageBox.Show(b1.Length.ToString());

foreach (var item in b1)

{

str += Conversion.Hex(item)+"\n";

}

MessageBox.Show(str);

}

}

private string Coder(string str,int key)

{

string result = String.Empty;

foreach (char item in str)

{

if (!Char.IsLetter(item))

{

result += item;

continue;

}

if (Char.IsUpper(item))

result += Char.ConvertFromUtf32(65 + (int)(item - 65 + key % 26) % 26);

}

return result;

}

private void button\_Browse\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

if (open.ShowDialog() == DialogResult.OK)

{

textBox\_Path.Text = open.FileName;

}

}

private void Button\_Accept\_Click(object sender, EventArgs e)

{

try

{

string[] filename = textBox\_Path.Text.Split('.');

MD5 md5 = MD5.Create();

string str = "";

using (Stream stream = new StreamReader(textBox\_Path.Text).BaseStream)

{

byte[] bytes = md5.ComputeHash(stream);

foreach (var item in bytes)

str += Conversion.Hex(item);

}

File.Copy(textBox\_Path.Text,Coder(str,(int)numericUpDown\_CloseKey.Value)+'.'+filename[filename.Length-1]);

MessageBox.Show($"Файл был успешно подтвержден!\nОткрытый ключ= {26-numericUpDown\_CloseKey.Value%26}");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void button\_Browse2\_Click(object sender, EventArgs e)

{

OpenFileDialog open = new OpenFileDialog();

if (open.ShowDialog() == DialogResult.OK)

{

textBox\_Path2.Text = open.FileName;

temp=open.SafeFileName.Split('.')[0];

}

}

private void button\_Check\_Click(object sender, EventArgs e)

{

try

{

MD5 md5 = MD5.Create();

string str = "";

using (Stream stream = new StreamReader(textBox\_Path2.Text).BaseStream)

{

byte[] bytes = md5.ComputeHash(stream);

foreach (var item in bytes)

str += Conversion.Hex(item);

}

if(str== Coder(temp, (int)numericUpDown\_OpenKey.Value))

MessageBox.Show("Файл проверен!");

else

MessageBox.Show("Файл не проверен!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}