УВАЖАЕМЫЕ СТУДЕНТЫ! Изучите теоретические сведения к лабораторной работе, выполните практическое задание, дайте ответы на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: r.bigangel@gmail.com до 20.03.2023.

Требования к отчету:

Отчет предоставляется преподавателю в электронном варианте и должен содержать:

- **название работы, постановку цели, вывод;**
- **ответы на контрольные вопросы, указанные преподавателем.**

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)111-37-59, (Viber, WhatsApp), vk.com: https://vk.com/daykini

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лабораторная работа № 36

Тема: «Шифрование информации с использованием стандарта RSA»

Цель: получение навыков работы с пространством CryptoAPI для шифрования информации; разработка Web - приложения для шифрования информации с использование м стандарта RSA.

Теоретическая часть

Стандарт шифрования RSA относится к ассиметричному методу шифрования. Безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших (100-200 разрядов или даже больше) простых чисел. Предполагается, что восстановление открытого текста по шифротексту и открытому ключу эквивалентно разложению на множители двух больших чисел.

Для генерации двух ключей используются два больших случайных простых числа p и q. Для максимальной безопасности выбирайте p и q равной длины. Рассчитывается произведение:

n=p*q

Затем случайным образом выбирается ключ шифрования e, такой что e и(p-1)(q-1) являются взаимно простыми числами. Наконец расширенный алгоритм Евклида используется для вычисления ключа дешифрирования d, такого что

Другими словами

d и n также взаимно простые числа. Числа e и n - это открытый ключ, а число d - закрытый. Два простых числа p и q больше не нужны. Они должны быть отброшены, но не должны быть раскрыты.

Для шифрования сообщения m оно сначала разбивается на цифровые блоки, меньшие n (для двоичных данных выбирается самая большая степень числа 2, меньшая n). То есть, если p и q - 100-разрядные простые числа, то n будет содержать около 200 разрядов, и каждый блок сообщения mi должен быть около 200 разрядов в длину. (Если нужно зашифровать фиксированное число блоков, их можно дополнить несколькими нулями слева, чтобы гарантировать, что блоки всегда будут меньше n. Зашифрованное сообщение c будет состоять из блоков ci той же самой длины. Формула шифрования выглядит так :

e*d=1mod(p-1)(q-1)

Другими словами

$d=e^{-1}mod(p-1)(q-1)$

d и n также взаимно простые числа. Числа e и n - это открытый ключ, а число d - закрытый. Два простых числа p и q больше не нужны. Они должны быть отброшены, но не должны быть раскрыты.

Для шифрования сообщения m оно сначала разбивается на цифровые блоки, меньшие n (для двоичных данных выбирается самая большая степень числа 2, меньшая n). То есть, если p и q - 100-разрядные простые числа, то n

будет содержать около 200 разрядов, и каждый блок сообщения mi должен быть около 200 разрядов в длину. (Если нужно зашифровать фиксированное число блоков, их можно дополнить несколькими нулями слева, чтобы гарантировать, что блоки всегда будут меньше n. Зашифрованное сообщение c будет состоять из блоков ci той же самой длины. Формула шифрования выглядит так

$$c_i = m_i^e \bmod n$$

Для расшифровки сообщения возьмите каждый зашифрованный блок ci и вычислите

$$m_i = c_i^d \bmod n$$

Шифрование RSA

Открытый ключ:

n произведение двух простых чисел p и q (p и q должны храниться в секрете) e число, взаимно простое с (p-1)(q-1)

Закрытый ключ:

d e^{-1} mod ((p-1)(q-1))

Шифрование:

c = m^e mod n

Дешифрирование:

m = c^d mod n

Для выполнения нашей лабораторной работы, как и в предыдущем случае мы воспользуемся созданием Web-приложения.

Среда .NET Framework содержит два класса, реализующих асимметричные алгоритмы: **RSACryptoServiceProvider** и **DSACryptoServiceProvider**. Оба метода пригодны для подписи и верификации данных, т. е. для создания цифровых подписей. Подобно симметричным системам, асимметричные имеют базовый класс, от которого наследуются объекты, реализующие конечные криптометоды – это AsymmetricAlgorithm.

Как было отмечено, асимметричные алгоритмы очень медленны, и потому пригодны лишь для небольших объёмов информации. Так, алгоритм RSA может обработать 43 байта.

Практическая часть

Для выполнения работы, как и в предыдущей лабораторной работе мы создадим новое Web - приложение.

Для создания интерфейса мы можем:

1. оставаться в окне редактора кода и формировать интерфейс с использование стандартных html-тэгов

2. перейти в режим конструктора и наполнить форму необходимыми компонентами.

При использовании второго метода заполнения необходимо перейти в режим конструктора и перетащить на форму компоненты Button и 3 текстовых поля. Отредактируйте свойства этих компонентов в соответствии со следующим кодом (для просмотра редактора перейдем в соответствующее окно кода нажав кнопку Sourse):

```
<%@ Page Language="vb" AutoEventWireup="false" Inherits="RSA.WebForm1" CodeFile="default.aspx.vb" %>
<HTML>
    <HEAD>
        <title>RSA</title>
    </HEAD>
    <body>
        <form id="Form1" method="post" runat="server">
            <b>Clear text:</b><br>
            <textarea id="txt" runat="server"></textarea><br>

            <b>Encrypted text:</b><br>
            <textarea id="txtEnc" runat="server" rows=4></textarea><br>

            <b>Decrypted text:</b><br>
            <textarea id="txtDec" runat="server"></textarea><br>

            <hr>
            <asp:Button ID="btnEnc" Runat="server" Text="Encrypt and Decrypt" />
        </form>
    </body>
</HTML>
```

Форма будет выглядеть следующим образом:

Реализация стандарта RSA

Все основные действия происходят в обработчике события нажатия кнопки Encrypt and Descrypt, поэтому необходимо вызвать этот обработчик, в нашем случае это default.aspx.vb, и прописать туда соответствующий участок кода. Для начала надо подключить компоненты шифрования и созданную нами Web - форму следующим кодом:

```
Imports System.Security.Cryptography
Imports System.Text

Namespace RSA

Partial Class WebForm1
    Inherits System.Web.UI.Page
```

Код реализации асимметричного метода будет намного короче, чем код симметричного. Одной из причин столь малых габаритов является то, что нет нужды инициализировать множество объектов, связывая их друг с другом. В этом примере используется один-единственный класс – RSACryptoServiceProvider.

В процедуре обработки события нажатия кнопки btnEnc сначала создаётся экземпляр класса RSACryptoServiceProvider, при этом автоматически генерируется пара из открытого и секретного ключей

```
Dim enc As New UnicodeEncoding

Private Sub btnEnc_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnEnc.Click
    ' Генерируем пару из открытого и секретного ключей
    Dim RSAcsp As New RSACryptoServiceProvider
```

Далее для шифрования и дешифрования используется функция RSACrypt. Сравните вызовы этой функции для обеих операций:

```
    ' шифруем
    txtEnc.Value = RSACrypt(enc.GetBytes(txt.Value), _
        RSAcsp.ExportParameters(False), KindOfAction.RSAEncrypt)
    ' дешифруем
    txtDec.Value = RSACrypt(enc.GetBytes(txtEnc.Value), _
        RSAcsp.ExportParameters(True), KindOfAction.RSADecrypt)
End Sub
```

Сама функция RSACrypt выглядит следующим образом:

```
Private Function RSACrypt(ByVal DataToEncrypt() As Byte, ByVal RSAKey As RSAParameters, ByVal Action As KindOfAction) As String
    Dim RSA As New RSACryptoServiceProvider

    RSA.ImportParameters(RSAKey)
    If Action = KindOfAction.RSAEncrypt Then
        Return enc.GetString(RSA.Encrypt(DataToEncrypt, False))
    Else
        Return enc.GetString(RSA.Decrypt(DataToEncrypt, False))
    End If
End Function
```

При шифровании в параметр includePrivateParameters метода ExportParameters передается значение False , а при дешифровании - значение True . Этот параметр отвечает за информацию, которая должна быть экспортирована. Таким образом, если стоит значение False, то экспортируются данные только об открытом ключе, а если стоит True, то об открытом и о секретном.

В функции **RSACrypt** создаётся ещё один экземпляр объекта RSACryptoServiceProvider (переменная RSA), но уже не для генерации

ключей, а для выполнения криптоопераций с использованием ключей, переданных классом-генератором (**RSAcsp**). Итак, для выполнения криптоопераций с помощью объекта RSACryptoServiceProvider необходимы 2 его экземпляра: один для генерации ключей, а другой для выполнения криптоопераций.

Для шифрования и дешифрования вызываются методы **Encrypt** и **Decrypt** объекта RSACryptoServiceProvider.

Первый параметр этого метода принимает массив байтов для шифрования, а второй активизирует или деактивизирует Дополнение Оптимального Асимметричного Шифрования (OAEP - Optimal Asymmetric Encryption Padding). Этот механизм работает только на системах Windows XP и старше, т. е. XP и более новые операционные системы могут работать как со значением параметра False, так и со значением True, а все остальные поддерживают только False.

Для выбора типа операции описывается следующим кодом:

```
        Private Enum KindOfAction As Integer
            RSAEncrypt = 0
            RSADecrypt = 1
        End Enum
End Class

End Namespace
```

Теперь запустите созданное приложение и попробуйте что-нибудь зашифровать:

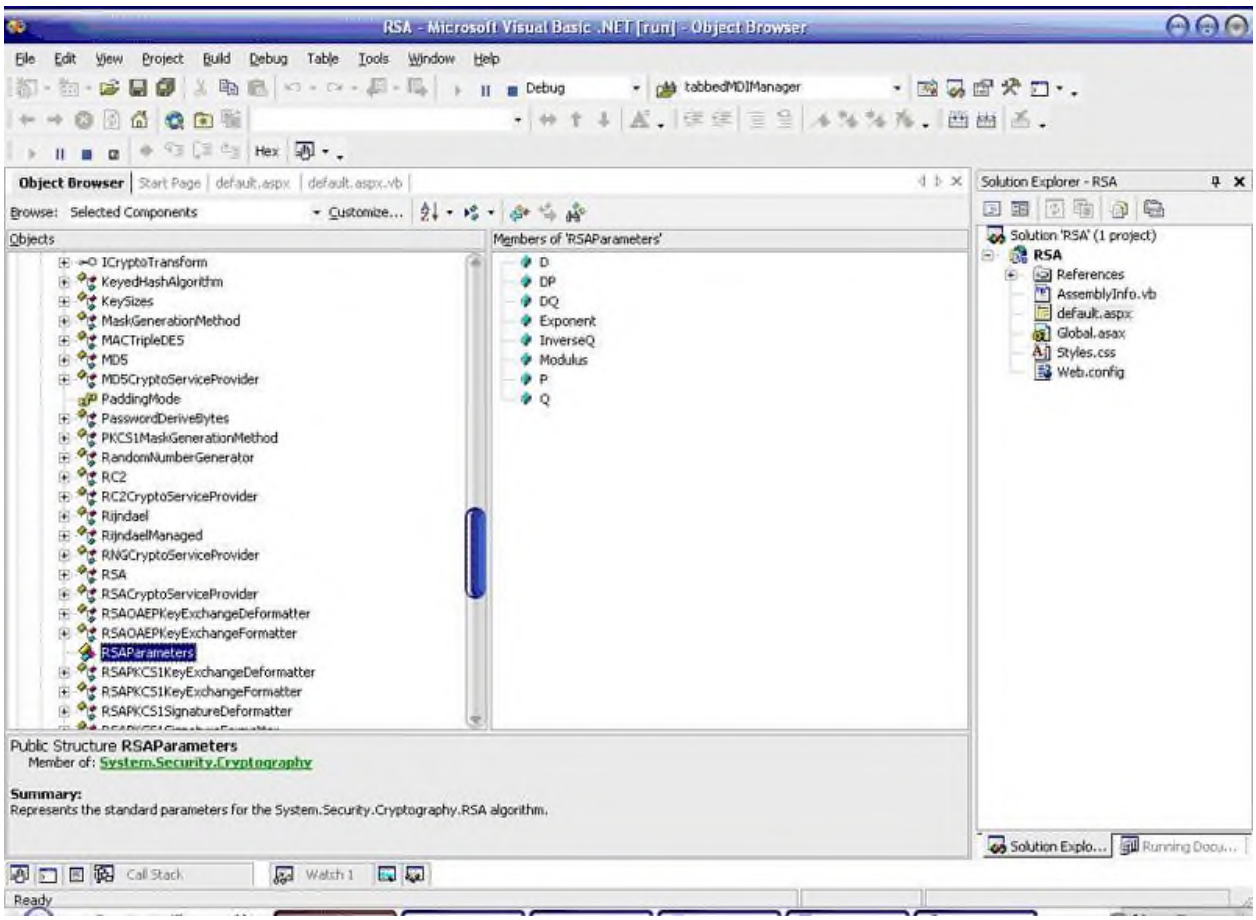
Возможен вариант, когда приложение будет правильно работать и давать результат не с первого раза. Это возникает в результате объявления 2-х ключей. Раньше в этом не было необходимости, поскольку мы только шифровали, но не дешифровали данные. Даже если мы сами не прописываем провайдеру криптоалгоритма, что нужно использовать такой-то ключ, он генерирует их самостоятельно как при шифровании, так и при

дешифровании. В результате могут возникнуть разногласия при их генерации.

Сохраните созданное вами приложение.

Информация о ключах алгоритма RSA хранится в структуре RSAParameters. Эта структура содержит в себе несколько параметров, имена которых никоим образом не связаны с открытым и секретным ключами. Эту структуру можно увидеть на рисунке.

Определить, какие переменные к чему относятся, довольно легко.



Структура RSAParameters.

Включить в отчет

1.Тема лабораторной работы

2. Цель лабораторной работы

3. Созданные вами Web формы и их назначение

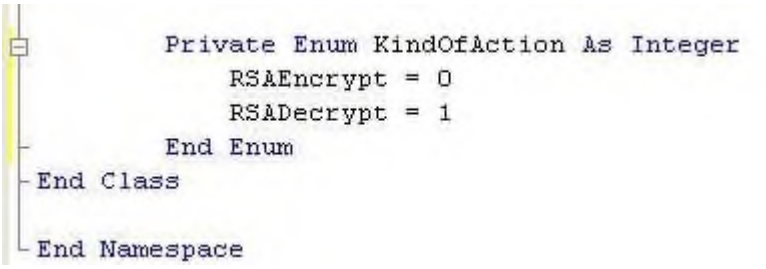
4. Программа шифрования информации (в электронном или письменном виде)

Контрольные вопросы:

1. К какому методу шифрования относится криптостандарт RSA?

2. Укажите структуру инициализации криптопровайдера RSA.

3. Какое действие предполагает следующий участок кода:



```
        Private Enum KindOfAction As Integer
            RSAEncrypt = 0
            RSADecrypt = 1
        End Enum
End Class

End Namespace
```

4. Объясните причину того, что приложении может не с первого раза сработать корректно.