```vb
Imports System.Xml

Public Class SecureCard

    Private isDecrypted As Boolean = False
    Private isEncrypted As Boolean = False
    Private _cardHolder As String
    Private _cardNumber As String
    Private _issueDate As String
    Private _expiryDate As String
    Private _issueNumber As String
    Private _cardType As String
    Private _encryptedData As String
    Private _xmlCardData As XmlDocument

    Private Sub New()
        ' private default constructor
    End Sub

    Public Sub New(ByVal newEncryptedData As String)
        ' constructor for use with encrypted data
        _encryptedData = newEncryptedData
        DecryptData()
    End Sub

    Public Sub New(ByVal newCardHolder As String, ByVal newCardNumber As String, _
                   ByVal newIssueDate As String, ByVal newExpiryDate As String, _
                   ByVal newIssueNumber As String, ByVal newCardType As String)
        ' constructor for use with decrypted data
        _cardHolder = newCardHolder
        _cardNumber = newCardNumber
        _issueDate = newIssueDate
        _expiryDate = newExpiryDate
        _issueNumber = newIssueNumber
        _cardType = newCardType
        EncryptData()
    End Sub

    Private Sub CreateXml()
        ' encode card details as XML document
        _xmlCardData = New XmlDocument()
        Dim documentRoot As XmlElement = _
            _xmlCardData.CreateElement("CardDetails")
        Dim child As XmlElement

        child = _xmlCardData.CreateElement("CardHolder")
        child.InnerXml = _cardHolder
        documentRoot.AppendChild(child)

        child = _xmlCardData.CreateElement("CardNumber")
        child.InnerXml = _cardNumber
        documentRoot.AppendChild(child)

        child = _xmlCardData.CreateElement("IssueDate")
        child.InnerXml = _issueDate
        documentRoot.AppendChild(child)

        child = _xmlCardData.CreateElement("ExpiryDate")
        child.InnerXml = _expiryDate
        documentRoot.AppendChild(child)

        child = _xmlCardData.CreateElement("IssueNumber")
        child.InnerXml = _issueNumber
        documentRoot.AppendChild(child)
```

```
    child = _xmlCardData.CreateElement("CardType")
    child.InnerXml = _cardType
    documentRoot.AppendChild(child)

    _xmlCardData.AppendChild(documentRoot)
End Sub

Private Sub ExtractXml()
    ' get card details out of XML document
    _cardHolder = _
       _xmlCardData.GetElementsByTagName("CardHolder").Item(0).InnerXml
    _cardNumber = _
       _xmlCardData.GetElementsByTagName("CardNumber").Item(0).InnerXml
    _issueDate = _
       _xmlCardData.GetElementsByTagName("IssueDate").Item(0).InnerXml
    _expiryDate = _
       _xmlCardData.GetElementsByTagName("ExpiryDate").Item(0).InnerXml
    _issueNumber = _
       _xmlCardData.GetElementsByTagName("IssueNumber").Item(0).InnerXml
    _cardType = _
       _xmlCardData.GetElementsByTagName("CardType").Item(0).InnerXml
End Sub

Private Sub EncryptData()
    Try
        ' stuff data into XML doc
        CreateXml()

        ' encrypt data
        _encryptedData = EmailNotification.StringEncryptor.encryptStrong(_xmlCardData.OuterXml)

        ' set encrypted flag
        isEncrypted = True
    Catch
        Throw New SecureCardException("Unable to encrypt data.")
    End Try
End Sub

Private Sub DecryptData()
    Try
        ' decrypt data
        _xmlCardData = New XmlDocument()
        _xmlCardData.InnerXml = EmailNotification.StringEncryptor.decryptStrong(_encryptedData)

        ' extract data from XML
        ExtractXml()

        ' set decrypted flag
        isDecrypted = True
    Catch
        Throw New SecureCardException("Unable to decrypt data.")
    End Try
End Sub

Public ReadOnly Property CardHolder() As String
    Get
        If isDecrypted Then
            Return _cardHolder
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property
```

```
Public ReadOnly Property CardNumber() As String
    Get
        If isDecrypted Then
            Return _cardNumber
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property

Public ReadOnly Property CardNumberX() As String
    Get
        If isDecrypted Then
            Return "XXXX-XXXX-XXXX-" & _cardNumber.Substring(_cardNumber.Length - 4, 4)
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property

Public ReadOnly Property IssueDate() As String
    Get
        If isDecrypted Then
            Return _issueDate
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property

Public ReadOnly Property ExpiryDate() As String
    Get
        If isDecrypted Then
            Return _expiryDate
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property

Public ReadOnly Property IssueNumber() As String
    Get
        If isDecrypted Then
            Return _issueNumber
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property

Public ReadOnly Property CardType() As String
    Get
        If isDecrypted Then
            Return _cardType
        Else
            Throw New SecureCardException("Data not decrypted.")
        End If
    End Get
End Property

Public ReadOnly Property EncryptedData() As String
    Get
        If isEncrypted Then
```

```
                Return _encryptedData
            Else
                Throw New SecureCardException("Data not encrypted.")
            End If
        End Get
    End Property
End Class
```