

A New Security Paradigm of File Sharing

Seifedine Kadry
Faculty of General Education
American University of the Middle East

Abstract

Windows Right Management Services protects RMS-enabled files or applications from unauthorized users. However, the offered security on the whole file prevents other trusted recipients with minor privileges to access it. The sender is obliged to send each time to these recipients another file that resembles to the former but does not contain sensitive information which is considered as time wasting especially when the number of recipients is increased.

This paper designs and implements a new security layer that extends the WRMS security provided on a certain file, in a way that the file still keeps its security towards unintended or unauthorized recipients, but can be sent only once to trusted recipients having different privileges in such a way that each recipient will only see the data that grants access to.

Keywords: WRMS, File Sharing, Security, XML, XrML, .NET.

1. Introduction

Windows Right Management Services proves its ability in protecting RMS-enabled information from unauthorized use no matter where this information goes inside or outside the organization [1]. This solution comes subsequent to many perimeter-based methods such as Firewalls, Access Control Lists, encryption, and authentication technologies. Although these methods could protect the information while in transit, but couldn't provide any layer of security when the information is exposed to the recipient. WRMS was the first technology that could introduce this layer of security whenever the file is received because the permission to the file is stored with the file itself. It can prevent the recipient to copy, modify, print, forward the RMS-based file via email to non-intended or perhaps malicious recipients, or to access it after a certain period of time. However, WRMS also prevents other trusted recipients with minor privileges to access a certain file since the offered security is implemented on the whole document [2]. Thus, the sender is obliged each time to send to these recipients another file similar to the former but does not contain confidential data. This is considered as time consuming especially when we are talking about a big organization that has numerous branches perhaps in

different countries and each branch consists of many departments. This limitation of WRMS motivated us to think about a solution to the addressed problem.

The objective of this paper is to extend the functionality of WRMS by adding a new security layer to Microsoft Excel worksheets, in such a way that the file still keeps its WRMS features and its security towards unintended recipients, but can be sent only once to trusted recipients having different privileges. To allow to this file to be sent to different recipients, the provided solution will apply a security on a portion of the document, this portion contains the sensitive data. As a final result, each recipient will only see the data that grants access to.

This paper is divided into the following sections:

In section two, we will discuss why we need this solution. And we will explain the importance of WRMS and how it works. The proposed solution that we adopt in this paper and how it is implemented, are clearly detailed in section three. Section four will discuss the conclusion, and the advantages of this solution in addition to future works in this domain.

2. Problematic

For a better understand the goal of this paper, we must look at the described real life scenario:

At the end of the business year, the Financial Director of "GlobalCom" Company prepares the annual bonuses file. The company has three departments: IT, Logistics and Quality Assurance. Each department has a manager and a group of employees that reports to this manager. Every department's manager receives from the financial director the bonuses file and approves it. The confidentiality plays an important role here since it is part of the company policy. For instance, the IT manager is not allowed to view the Logistics and Quality bonuses. The same scenario is repeated for the Logistics and Quality department, taking into consideration that all departments are sharing the same file.

How to achieve this? The financial director has a program which will allow him to encrypt any data from the Excel

file. In our case, the bonus column value for all employees in all departments will be the encrypted data.

In other words, the role of the sender is to encrypt whatever data from the excel sheet and sends it to each department's manager, and to apply WRMS to this file to assure that the file will not be edited or forwarded. The receiver's role is to view only its authorized data.

In absence of our solution: the final director will create three files to be sent for each department's manager and applies WRMS on each of these files without getting the benefit to send the common data. Each of the received RMS-enabled file will contain the bonus value to each department.

In the presence of our solution: The financial director can get the benefit of the file-reusability and will send a single shared RMS-enabled file to each department's manager. This file contains encrypted bonuses column for all employees in all departments.

Upon receiving the file by each department's manager, only the relevant data of each department will be decrypted, and the rest of the data that is relevant to other department's manager will remain encrypted because this solution allows the decryption of selective information based on the privileges of each department's manager.

After presenting both cases before and after applying the solution, it is logical to adopt this study that addresses this problem. It is obvious that this solution is time and resource effective (file reusability) and it deals with confidential data of the company in a secure way up to the latest technology techniques.

2.1 What is WRMS?

Windows Right Management Services is information protection technology that works with RMS-enabled applications to protect digital information from unauthorized use—both online and offline, inside and outside of the firewall.

RMS increases the security strategy of an organization by providing protection of the information through usage rights and conditions, which remain with the information, no matter where it goes. In other words, these permissions are assigned to an authorized recipient after the information is accessed [3].

2.2 WRMS Workflow

In a reference to [3], Windows RMS which includes both server and client components provides the following capabilities:

- **Creating rights-protected files and containers:**
Users who are trusted entities in an RMS system

could apply usage rights and conditions to digital information using RMS-enabled applications or browsers.

They can easily create and manage protected files using applications that people use every day -for example: computer-aided design (CAD) applications or Microsoft Office 2007 Editions- that incorporate Windows RMS technology features. Using common task management procedures within a familiar on-screen environment, organizations could assign usage rights and conditions to digital information such as an e-mail message or document.

In addition, RMS-enabled applications provide users with the option of applying authorized rights policy templates such as "Company Confidential."

- **Licensing and distributing rights-protected information:** The XrML-based [4] certificates issued by an RMS system (right account certificate) identify trusted entities that can publish or view rights-protected information. Users who are trusted entities in an RMS system can assign usage rights and conditions to information they want to protect through an RMS-enabled application. These usage policies specify who can use the information and what they can do with it.

The RMS system validates transparently the trusted entities and issues the publishing licenses that contain the specified usage rights and conditions for the information. The information is encrypted using the electronic keys from the RMS-enabled application (for example: Microsoft Word) and the XrML-based certificates of the trusted entities. After the information is encrypted or locked, only the trusted entities specified in the publishing licenses can unlock and use that information.

Users could then distribute the rights-protected information to other users in their organization via e-mail, internal servers, or external sites to enable trusted external partners to access the information.

- **Acquiring licenses to decrypt rights-protected information and enforcing usage policies:**

Trusted entities recipients who are named by information author, can open or view rights - protected information by using trusted computers having WRMS client software

installed along with RMS-enabled applications or browsers.

In a process that is transparent to the recipient, the RMS server, which has the public key that was used to encrypt the information, validates the recipient and then issues a user license that contains the usage rights and conditions that were specified in the publishing license. The information is decrypted using the electronic keys from the end-user license and the XrML-based certificates of the trusted entities. The usage rights and conditions are then enforced by the RMS-enabled application. The usage rights are persistent and enforced everywhere the information goes.

Referring to [3], for a better understanding of the RMS workflow within an organization, see Figure 1:

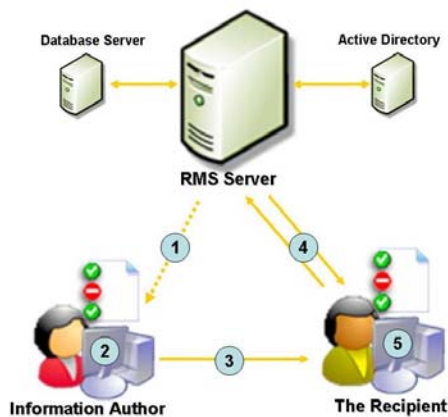


Fig1: RMS Workflow

1. Authors receive a client licenser certificate from the RMS server the first time they rights-protect information. (This is a one-time step that enables offline publishing of rights-protected information in the future.)
2. An author creates a file and defines a set of rights and rules. The RMS-enabled application in conjunction with the Windows Rights Management client software creates a “**publishing license**” and encrypts the file.
3. The author can distribute the file in any preferred manner.

4. The recipient clicks the file to open it. The RMS-enabled application calls to the RMS server, which validates the user and issues a user license.

The application renders the file and enforces the rights defined in the use license.

3. The Proposed Solution

3.1 Design

Since WRMS works in Microsoft-based environment, it was logical to use VB.NET [5] in our solution as an object oriented programming language in order to get some benefits of .NET Framework class libraries.

The proposed solution should be tested first in VB.NET Windows Form Application and then it should be applied in Excel application where there is a need to be created in a separate module.

This security layer provides us the opportunity to encrypt a simple node or an entire column through the .NET and XML technology.

The implementation of this solution in VB.NET in the Encryption process includes: Converting Excel to XML file, Encrypt / Decrypt XML Nodes, and converting the encrypted XML to Excel file (Figure 2). The conversion from an Excel to XML file helps us to select the specific nodes to be encrypted in later step.

XML was used because it is the most popular technology for structuring data; therefore XML-based encryption is the natural way to handle complex requirements for security in data interchange applications [8].

In the decryption process, the result is revealed: The description of the Microsoft Windows logged-on user is compared with the privileges of each user in the Excel input file. Thus, each recipient can see the information that grants access; the other part of the data will remain encrypted. The decrypted XML file is also converted to decrypted Excel file.

Finally, after testing the proposed solution in VB.NET Windows Form Application, we can apply it in Excel file supporting macros and we must get the same result.

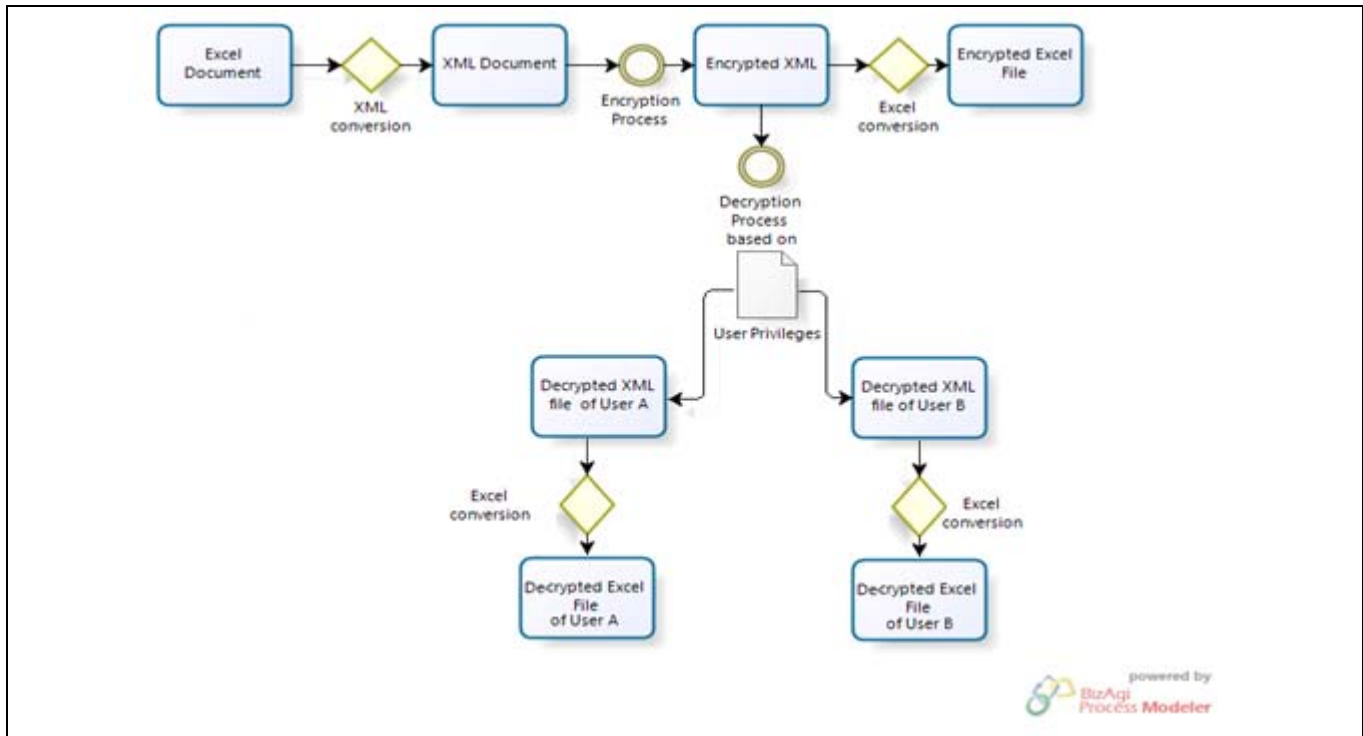


Fig2: Workflow of the proposed solution

3.2 Implementation

This code in the proposed solution is divided into two main parts:

A. Windows Form Application: “Excel Security” in VB.NET

▪ Convert Excel File to XML:

ADO.NET: stands for (ActiveX Data Object), is special set of .net framework allows the user to work with different type of databases such as Access, SQL server and Oracle. It provides two ways to work with data in a database: connected mode and disconnected mode [5]. In order to convert Excel file to XML file, we will use the ADO.NET in disconnected mode.

➤ The ConvertExcelToXML()method will perform the following Actions:

1. Creates a connection to the Excel file using *OleDbConnection* object.
2. Opens a connection with the property settings specified.

3. Selecting the data from the Excel file using *DataAdapter* object.
4. *TableMapping*.
5. Fills this *DataTable* with the imported data.
6. Set the property “Table” with the first *DataTable* contained in the *DataSet* “DtSet”.
7. Exports this *Dataset* to an XML file.
8. Reads the *DataColumns* and adds them the *ArrayList* “collist”.
9. Closes the connection to the data source.

To create a connection to the Excel file:

Referring to [6, 7], we have three ways to manipulate an Excel file. It can be done either by using Microsoft Office Component, Microsoft Jet Engine, and Access Database Engine. As per Microsoft recommendation, it is not advisable to use Office components on the server. Since Microsoft Jet Engine is only used to open a connection to Excel 2003 worksheets, but doesn’t support connectivity to Excel 2007 worksheets. So, the connection will be done using Microsoft.ACE.OLEDB.12.0 engine.

Before using Access Database Engine we must download the 2007 Office system driver from:

<http://www.microsoft.com/downloads/details.aspx?Family>

ID=7554F536-8C28-4598-9B72-
EF94E038C891&displaylang=en

To connect to an Excel file, we can use an OLEDB object that treats Excel as a database. The required information can be easily fetched by using SQL queries.

▪ **Encrypt Node(s) in XML file:**

Since we need to encrypt only a portion of the file, XML Encryption is the best choice if the application requires a combination of secure and insecure communication (which means that some of the data will be securely exchanged and the rest will be exchanged as is)[8].

➤ The SelectNodesToEncrypt() method will perform the following actions:

1. Gets the "EmployeeID" of the nodes to be encrypted.
2. Loops through the selected items and creates an EncryptedXML object "exml".
3. Selects the XML element(s) needed to be encrypted.
4. Encrypts the element(s) using the key generated "sharedkey" by the encryption algorithm object.
5. Creates an encrypted data object "ed" and specifies its properties.
6. Creates a cipherData element and sets its value to the encrypted XML element "EncryptedElem".
7. Replaces the plaintext XML element "Elem" with the encrypted data object "ed".
8. Saves the encrypted data to a file "EncryptedData.xml"

▪ **Encrypt Entire Column in XML file:**

➤ The SelectColumnToEncrypt() method will perform the following actions:

This method is similar to the SelectNodesToEncrypt() method but instead, it takes the entire column to be encrypted. Therefore, an "XMLNodeList" object will be created to hold all the elements inside the plaintext XML file:

To select an ordered list "xmllist" of XML nodes, we should create an object of XmlNodeList() and use the function SelectNodes() which takes an argument as X-Path expression. Referring to [9], X-path is a syntax for defining parts of an XML document.

▪ **Convert Encrypted XML file to Excel:**

➤ The ConvertXMLtoExcel() will perform the following actions:

1. Creates a new Excel Application, workbook "exbook" containing one Datasheet "exsheet".
2. Creates a new dynamic DataSet which has a DataTable that itself contains DataRows and DataColumns.
3. Loads the encrypted XML file specified by the user.
4. Creates DataColumns, sets their Data types and adds them to the DataTable.
5. Creates XmlNodeList object "empList" that contains all "TableMapping" nodes specified by the user.
6. The DataRows of the XMLElements in "empList" are filled with data available in "EncryptedData.xml" file
7. Sets the value of the property "DataTableENC" to the first table of the dataSet "ds".
8. Fills the Datasheet's cells "exsheet" with the DataTable's headers and rows.
9. Saves the new workbook Excel file and closes it, and quits the Excel Application.

▪ **Decrypt the XML file:**

➤ The DecryptXML() method will perform the following actions:

1. Loads the encrypted XML document specified by the user.
2. Creates an XmlNodeList 'regList' holding all encrypted nodes in the encrypted XML document.

3. *Loops through this list and searches inside all “<Employee>” tags if they contain the current description of the logged on user.*
4. *When this description is found , it retrieves the encrypted XML Element(s), Creates an encrypted data object ”ed2” and loads the encrypted element “c.firstChild” into the encrypted data object.*
5. *Creates an encrypted XML object “xml2”.*
6. *Decrypts the encrypted element”ed2” using the shared key*
7. *Replaces the encrypted element “c.firstChild” with the plain-text XML element “decryptedElem”.*
8. *Saves the decrypted data to an external file (optional).*

▪ **Post XML file in Web Browser:**

- The PostXMLFile() method will perform the following actions:

It declares an instance of URI ”uri” that takes the file name as an argument and sets the WebBrowser’s URL’s value to this instance.

▪ **Write the shared key:**

- The WriteSharedKey() method will perform the following actions:

1. *Creates an instance ”sharedkey” of the encryption algorithm provider*
2. *Converts the generated key to Base-64 String and write it to an external file specified by the user.*

1. Create an instance of the encryption algorithm provider chosen:
here, TripleDES will be the algorithm used to encrypt the XML data.

The .NET framework provides us with multiple types of encryption algorithms we can use such as: Triple DES, AES 128, AES 192, AES 256, RSA (Rijndael algorithm) and X509CertificateEx.

2. Using the instance created “sharedkey”, we can get the value of the secret key for this algorithm used, and convert it from 8 bit-Array to Base-64 String and write it to an external file specified by the user:

▪ **Read the shared key:**

- The ReadSharedKey() method will perform the following actions:

1. *Reads the stored generated key using StreamReader class.*
2. *Converts the read key from base 64 digit to an equivalent 8-bit integer array.*
3. *Sets the secret’s key value for the TDES algorithm to the convert 8-bit read key.*

1. To read the stored shared key, we should create an instance “reader” of the StreamReader class:
2. After reading the key stream until the end, we can convert it from base 64-bit to an equivalent 8-bit integer array and saves it into the data() array
3. Set the secret’s key value “sharedkey” to the converted 8-bit array and close the reader:

▪ **Load the XML file:**

- The LoadXmlFile()method will perform the following actions:

It sets the value of the property to the new “XMLDocument()”, and then it loads the XML Document from the user specified location. If an error occurred during loading this file, an error message will appear.

▪ **Get Current User Information:**

- The GetCurrentUserInfo()function will perform the following actions:

1. *Searches for all ‘win32’ accounts available in this local machine and saves them in ‘ColCSItems’ object.*
 2. *Gets the current logged-on user.*
 3. *Gets the user Description for this logged-on user.*
1. This function will search for all win32 user accounts [10] available in this local PC including the built-in accounts and saves them in “ColCSItems” object [11]. ConnectionOptions.Impersonation property sets COM impersonation level to be used for operations in this connection. In our case,

“root\cimv2” is the namespace used and “WIN32” is the provider used.

“impersonationLevel=impersonate” is used when the provider is a trusted application or service. It eliminates the need for the provider to perform client identity and access checks for the requested operations [12].

2. It also gets the current logged-on user. Since the name of the logged-on user will be returned as ‘computername\username’ shape and we are only interested in the username itself, we should split it and get the second part of it and save it in a string array called “name”. This string array will hold both parts on the split logged-on user [10].
3. Loops through all the win-32 accounts retrieved, when a match is found between the win-32 user and the logged-on user: we can get the description of this logged-on user and return it. Because we are only interested in the description of the logged-on user, the other cases are skipped.

Now, after we have presented the “**SecurityLayer**” class, we must build it and convert it to “**dll**” file to be used as reference in both “**Excel Security**” form application and “**emp.xlsm**” Excel file Visual Basic Editor.

The following items are required in order to create this form:

- Browse Button called “btn_Browse”
- Convert encrypted XML to Excel Button called “Btn_ConvertEncryptedXMLtoExcel”
- Convert decrypted XML to Excel Button called “Btn_ConvertDecryptedXMLtoExcel”
- Decrypted XML Button called “Btn_DecryptXML”
- Instance of TabControl to contain 6 tab pages called “TabControl1”
- Textbox to write the path of the excel file called “txt_Path”
- CheckBox to select all nodes in the column called “chk_all”
- ComboBox to select which column should be encrypted called “cb_colnames”

- Instance of OpenFileDialog control called “OpenFileDialog1”
 - Three instances of DataGridView control called “DataGridView1”, “DataGridView2”, and “DataGridView3” to display the plaintext Excel file, the encrypted Excel file and the decrypted files respectively.
 - Instances of WebBrowser control to display the content of encrypted XML, Decrypted XML and plaintext XML called “webBrowser1”, “webBrowser2” and “webBrowser3” respectively.
- We can set the *text* property of the form to “**New Security Layer To Excel**”
- We could set the properties of both “txtPlainText”, “txtEncrypted” and “TxtDecryptedXML” to the following:
- *ReadOnly*: **True**, to prevent the user to change the text’s content.
 - *MultiLine*: **True**, to display more than one line of the text.
 - *Scrollbars*: **Both**, Both the vertical and the horizontal scrollbars are displayed when the text contains multiline.
- We can set an initial directory to “OpenFileDialog1” control: *InitialDirectory*: **c:** so, the first directory will open when the user clicks on the browse button is C:.
- We set the **AutoSizeColumnMode** property to **AllCells** of the “DataGridView1”, “DataGridView2” and “DataGridView3” to determine the auto size mode for the visible columns.
- We must set the **DropDownStyle** of the “cb_colnames” ComboBox to **DropDownList** to oblige the user to select an Item from the list.
- To add the 6 tab members to the “TabControl1”, we need to click on the *tabPages: (Collection)...* property and add 6 members TabPage1 ... to TabPage6, and set their texts to: “**XML Data**”, “**XML Plaintext**”, and “**Encrypted XML**”, “**Encrypted Excel File**”, “**Decrypted XML**” and “**Decrypted Excel File**” respectively.

➤ Finally, we should add a column to the “DataGridView1” to allow the user to select a node to be encrypted. This is done by clicking on the property *Columns: (Collection)...* and add a column.

The added column has “SelectEmployee” as a Name, “DataGridViewCheckBoxColumn” as a type, and “Select” as a Header text. After the “SelectEmployee” column is added to the DataGridView1, the user is able to browse the Excel file:

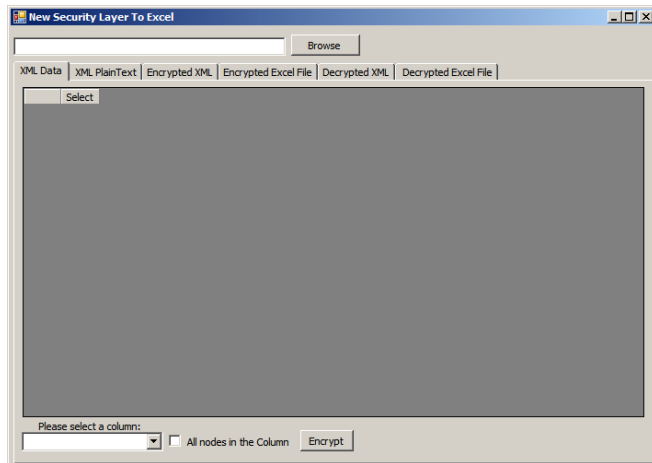


Fig3: Final form shape

➤ The Btn_Browse_Click() method will perform the following actions:

1. Filters the Excel file Extension choice
2. Saves the selected file in textbox control
3. Sets the values of properties in “SecurityLayer” class & convert the Excel file to XML
4. Sets the DataGridView’s DataSource to the property “dataTable”
5. Fills the ComboBox “cb_colnames” with the items in “columnList” property
6. Posts the converted xml file on the “xml plaintext” tab and sets this DataGridView to “read-only”.

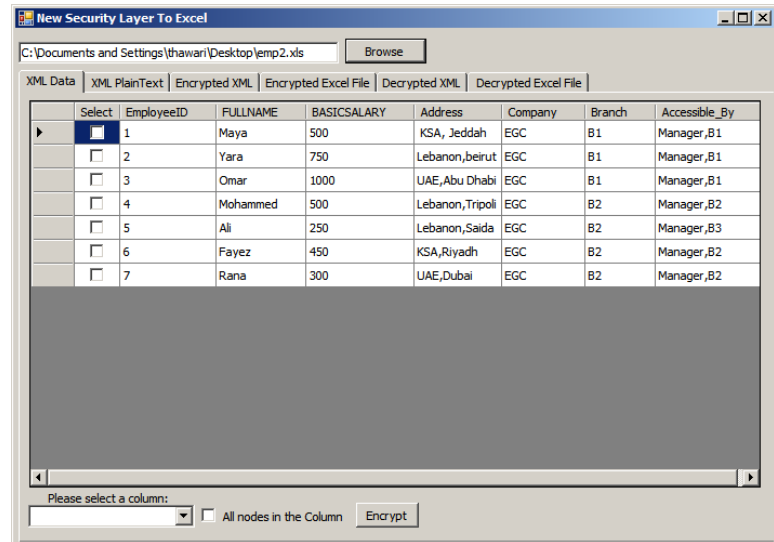


Fig4: Displaying the Excel data in a DataGridView

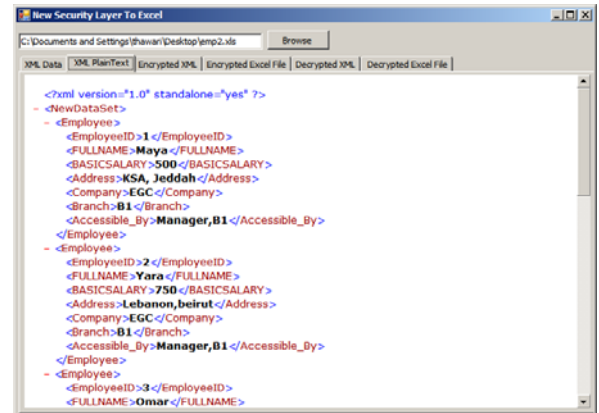


Fig5: Displaying the exported XML file.

▪ **Set the DataGridView to ‘Read-only’:**

➤ The SetGridColumn() method:

Since the first column of the DataGridView “SelectEmployee” should be editable to allow the user to select node(s) to encrypt, The SetGridColumn() sub will loop through each column of this DataGridView, except the first column, and make their values “read-only” in order to prevent the user from changing the DataGridView Cells:

- **Selecting all nodes in a column in one click:**

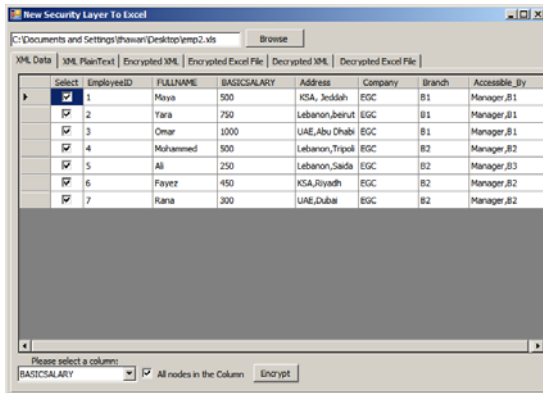


Fig 6: Encryption of all nodes in “BASIC SALARY” by clicking “chk_all” checkbox

- The chk_all_CheckedChanged() method:

The following method assigns the value of the checkboxes in the DataGridView1 to the value of “chk_all” checkbox. If it was selected, then these checkboxes will be selected too; similarly if “chk_all” was reset, these checkboxes will be reset too. This sub also makes the value of the DataGridView1 read-only after the “chk_all” checkbox was checked, to prevent the user to un-check any cell in the DataGridView1.

Encrypt node(s) or entire column in XML file:

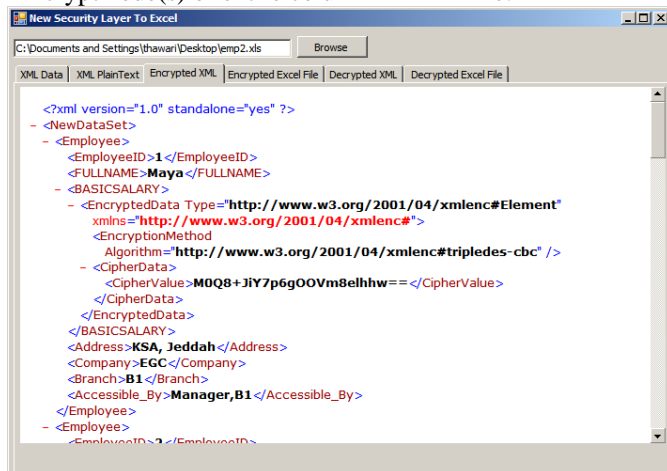


Fig 7: Encryption of “<BASIC SALARY>” XML node

- The Btn_encrypt_click() method will perform the following actions:

1. Loads the XML document posted in “XML Plaintext” tab.
2. Sets the values of some properties and writes the shared key to an external file.

3. User Validation
4. Checks whether “chk_all” checkbox was selected, and calls either *SelectColumnToEncrypt()* and Posts the encrypted xml file.
5. If not, it calls *SelectNodesToEncrypt()* & also posts the encrypted xml file in “web Browser”.

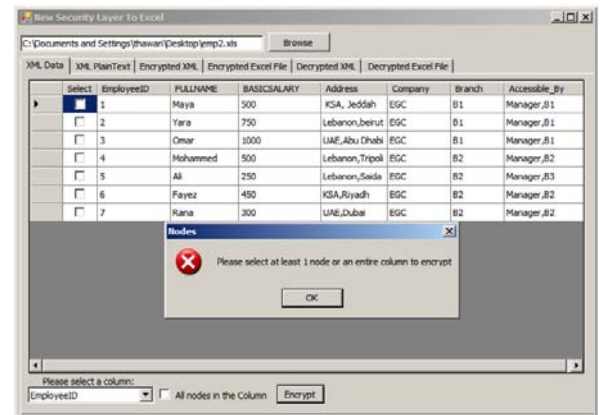


Fig 8: No node or an entire column was selected to be encrypted

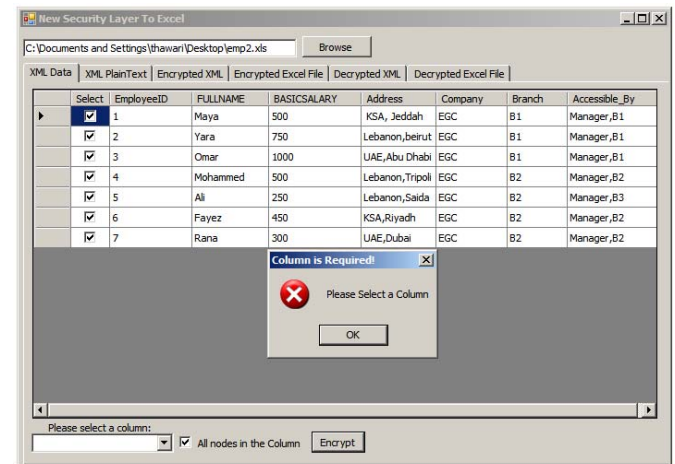


Fig 9: The Column from the ComboBox wasn't selected

1. Check whether “chk_all” checkbox was selected. If so, call *SelectColumnToEncrypt()* method and post the encrypted file in “Encrypted XML” tab:
2. If “chk_all” checkbox was not selected, loop through the “ID” of the selected nodes and save them in a string separated by “,” to call *SelectNodesToEncrypt()* method. the encrypted file is also posted under “Encrypted XML” tab in a web Browser:

▪ **Decrypt the encrypted XML file:**

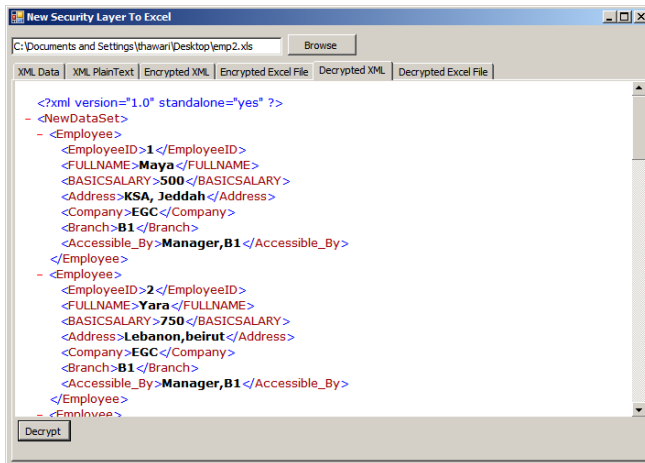



Fig 10: Decryption of the XML file for the 'Manager,B1'

- The Btn DecryptXML Click() method will perform the following actions:

The values of some properties in "securityLayer" class should be set to call the decryptXML() method. the decrypted xml file will be posted on the "Web Browser".

 **Note:** Only the information that the logged-on user has access to it will be decrypted; the other information will remain encrypted. Here, the logged-on user has access to view the information of EmployeeID: 1, 2, and 3; but doesn't have enough privileges to view the information related to EmployeeID: 4,5,6,7.

▪ **Convert the encrypted XML to Excel file:**

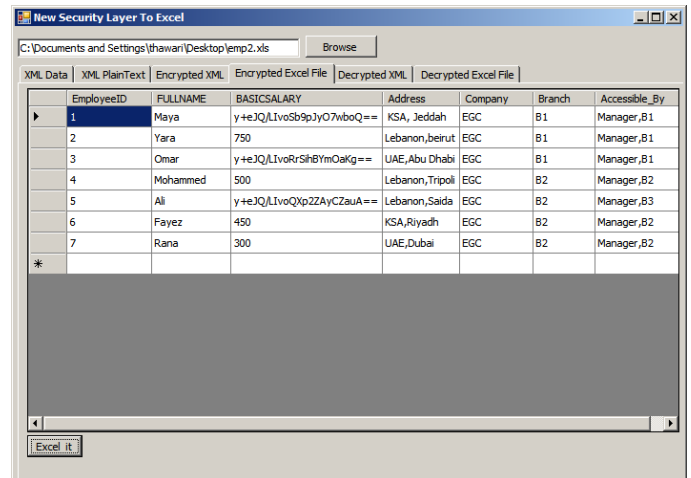


Fig 11: Encryption of the some nodes in "BASICSALARY" column

- The Btn ConvertEncrypted XMLtoExcel Click() method will perform the following actions:

The values of both InputXMLFile() and OutPutExcelFile() properties will be set in order to call the ConvertXMLtoExcel() method. The result will be posted on DataGridView2

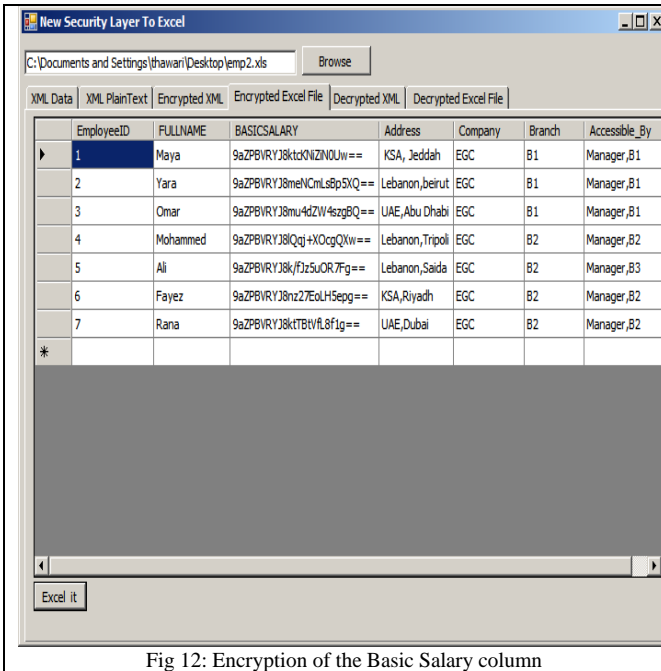


Fig 12: Encryption of the Basic Salary column

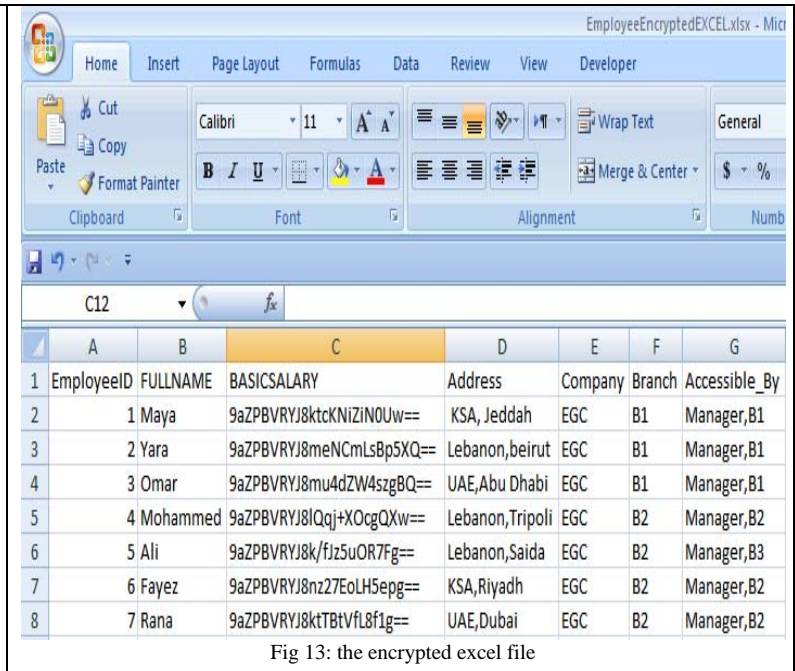


Fig 13: the encrypted excel file

▪ **Convert the Decrypted XML file to Excel:**

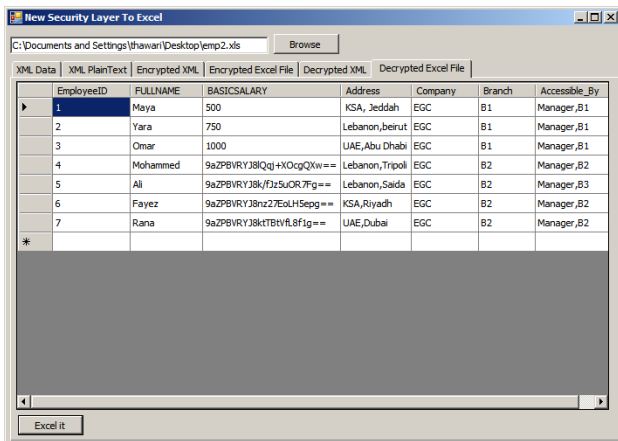


Fig 14: Displaying the "EmployeeDecryptedExcel.xlsx" file

➤ The Btn_ConvertDecrypted_XMLtoExcel_Click() method will perform the following actions:

The values of both `InputXMLFile()` and `OutPutExcelFile()` properties will be set in order to call the `ConvertXMLtoExcel()` method. The result will be posted on `DataGridView3`

B. Excel file supporting Macros: "emp.xlsm" in Visual Basic for Application.

Create Command Buttons and set their properties:

1. Go to Developer tab>Insert>ActiveX Control>Command Buttons
2. Drag 5 command buttons and set their names respectively: "ConvertXMLtoExcel", "EncryptSelectedNodes", "EncryptAll", "Create_Encrypted_Excel", "Decrypt" and "Create_Decrypted_Excel".
3. Set their captions respectively to: "Convert this Excel to XML", "Encrypt Selected Nodes", "Encrypt All 'BASICSALARY' Column", "Create Encrypted Excel", "Decrypt", and "Create Decrypted Excel".

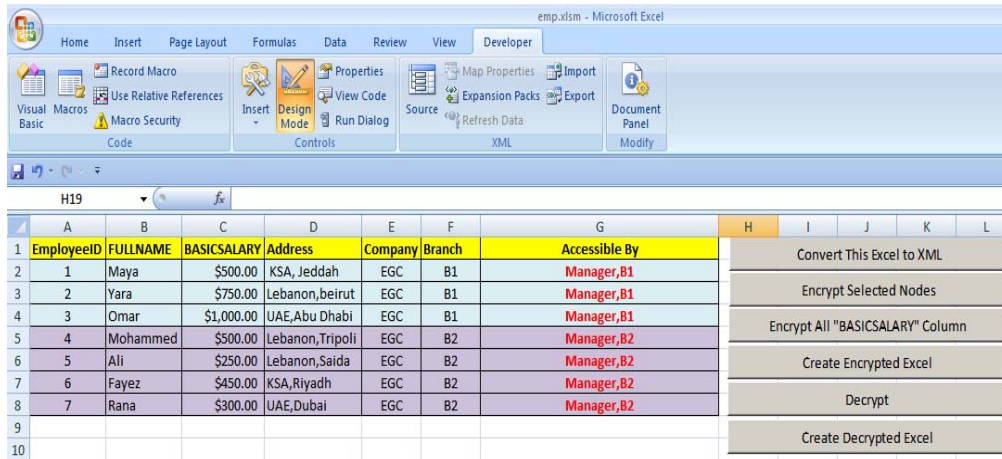


Fig 15: emp.xlsm excel file

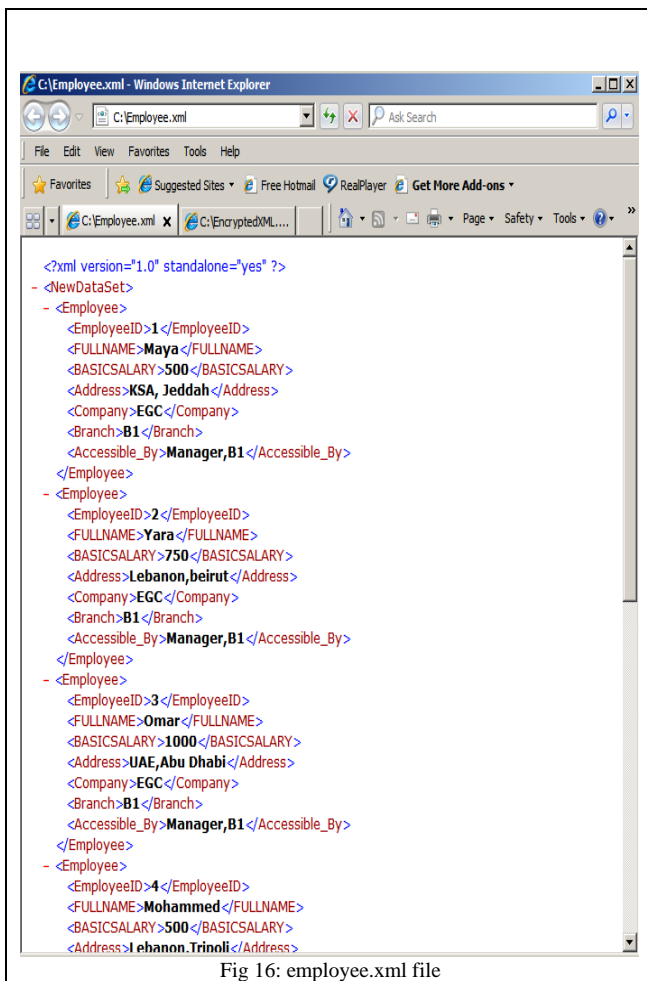


Fig 16: employee.xml file

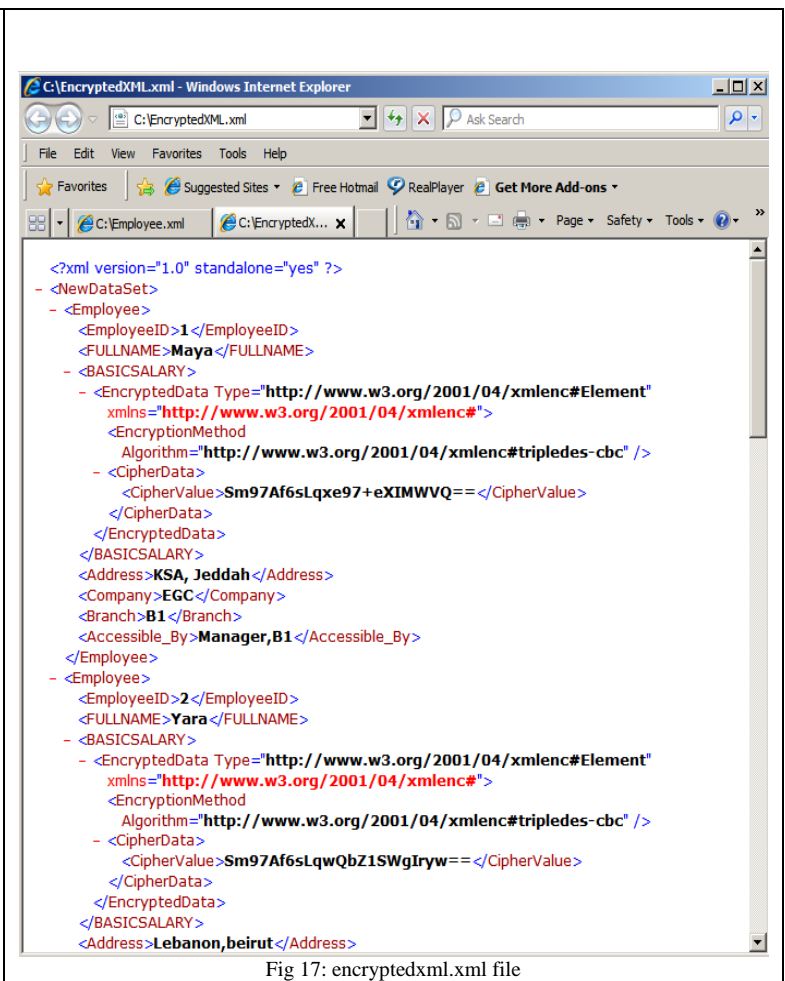


Fig 17: encryptedxml.xml file

	A	B	C	D	E	F	G
1	EmployeeID	FULLNAME	BASICSALARY	Address	Company	Branch	Accessible_By
2		1 Maya	Sm97Af6sLqxe97+eXIMVWVQ==	KSA, Jeddah	EGC	B1	Manager,B1
3		2 Yara	Sm97Af6sLqwQbZ1SWgIryw==	Lebanon,beirut	EGC	B1	Manager,B1
4		3 Omar	Sm97Af6sLqy7ewuI34NCEg==	UAE,Abu Dhabi	EGC	B1	Manager,B1
5		4 Mohammed	Sm97Af6sLqzRiHubDK59pQ==	Lebanon,Tripoli	EGC	B2	Manager,B2
6		5 Ali	Sm97Af6sLqz5Oj1UdHKTfQ==	Lebanon,Saida	EGC	B2	Manager,B2
7		6 Fayez	Sm97Af6sLqyqh9EfiIk24Q==	KSA,Riyadh	EGC	B2	Manager,B2
8		7 Rana	Sm97Af6sLqxmlmjy8WtOsw==	UAE,Dubai	EGC	B2	Manager,B2

Fig 18: "EncryptedExcel.xlsx" file.

```

<?xml version="1.0" standalone="yes" ?>
<NewDataSet>
  <Employee>
    <EmployeeID>1</EmployeeID>
    <FULLNAME>Maya</FULLNAME>
    <BASICSALARY>500</BASICSALARY>
    <Address>KSA, Jeddah</Address>
    <Company>EGC</Company>
    <Branch>B1</Branch>
    <Accessible_By>Manager,B1</Accessible_By>
  </Employee>
  <Employee>
    <EmployeeID>3</EmployeeID>
    <FULLNAME>Omar</FULLNAME>
    <BASICSALARY>1000</BASICSALARY>
    <Address>UAE, Abu Dhabi</Address>
    <Company>EGC</Company>
    <Branch>B1</Branch>
    <Accessible_By>Manager,B1</Accessible_By>
  </Employee>
  <Employee>
    <EmployeeID>4</EmployeeID>
    <FULLNAME>Mohammed</FULLNAME>
    <BASICSALARY>
      <EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      </EncryptedData>
    </BASICSALARY>
  </Employee>
  <Employee>
    <EmployeeID>4</EmployeeID>
    <FULLNAME>Mohammed</FULLNAME>
    <BASICSALARY>
      <EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      </EncryptedData>
    </BASICSALARY>
  </Employee>
  <Employee>
    <EmployeeID>5</EmployeeID>
    <FULLNAME>Ali</FULLNAME>
    <BASICSALARY>
      <EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      </EncryptedData>
    </BASICSALARY>
  </Employee>
  <Employee>
    <EmployeeID>6</EmployeeID>
    <FULLNAME>Fayez</FULLNAME>
    <BASICSALARY>
      <EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      </EncryptedData>
    </BASICSALARY>
  </Employee>
  <Employee>
    <EmployeeID>7</EmployeeID>
    <FULLNAME>Rana</FULLNAME>
    <BASICSALARY>
      <EncryptedData
        Type="http://www.w3.org/2001/04/xmlenc#Element"
        xmlns="http://www.w3.org/2001/04/xmlenc#"
        EncryptionMethod
          Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      </EncryptedData>
    </BASICSALARY>
  </Employee>
</NewDataSet>
    
```

Fig 19: decryptedxml.xml file

	A	B	C	D	E	F	G
1	EmployeeID	FULLNAME	BASICSALARY	Address	Company	Branch	Accessible_By
2		1 Maya	500	KSA, Jeddah	EGC	B1	Manager,B1
3		2 Yara	750	Lebanon,beirut	EGC	B1	Manager,B1
4		3 Omar	1000	UAE, Abu Dhabi	EGC	B1	Manager,B1
5		4 Mohammed	Sm97Af6sLqzRiHubDK59pQ==	Lebanon,Tripoli	EGC	B2	Manager,B2
6		5 Ali	Sm97Af6sLqz5Oj1UdHKTfQ==	Lebanon,Saida	EGC	B2	Manager,B2
7		6 Fayez	Sm97Af6sLqyqh9EfiIk24Q==	KSA,Riyadh	EGC	B2	Manager,B2
8		7 Rana	Sm97Af6sLqxmlmjy8WtOsw==	UAE,Dubai	EGC	B2	Manager,B2

Fig 20: decryptedexcel.xlsx file

Setting privileges to Windows logged-on users:
 Logged-on users have different privileges between each others. To give a certain user a permission to view a portion of data while other logged-on user is given other portion, we have to add a meaningful description to them. This description may present their positions in the Company. This description is used in the decryption process and leads to different results. In

Adding Description to windows logged-on Users:

our case, we have two logged-on users: "hawari" and "Rouwa". "hawari" is a Manager of the first branch of the EGC company and must grant access only to information relevant to EmployeeID: 1,2, and 3 while "Rouwa" is a Manager of the second branch of this company and must grant access only to information relevant to EmployeeID: 4,5,6 and 7.

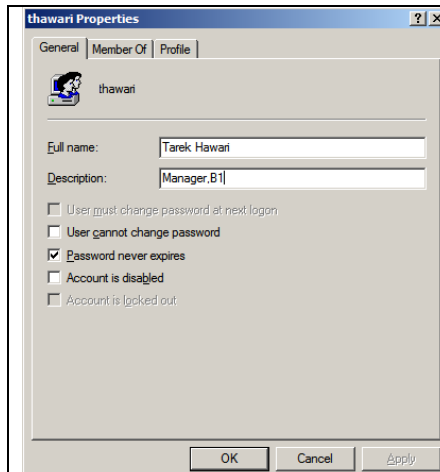


Fig21: Adding Description to user 'thawari'

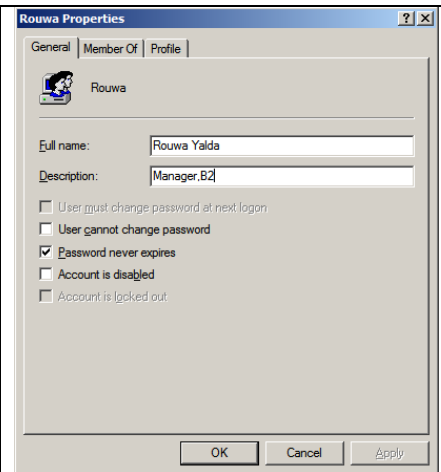


Fig22: Adding Description to user 'Rouwa'

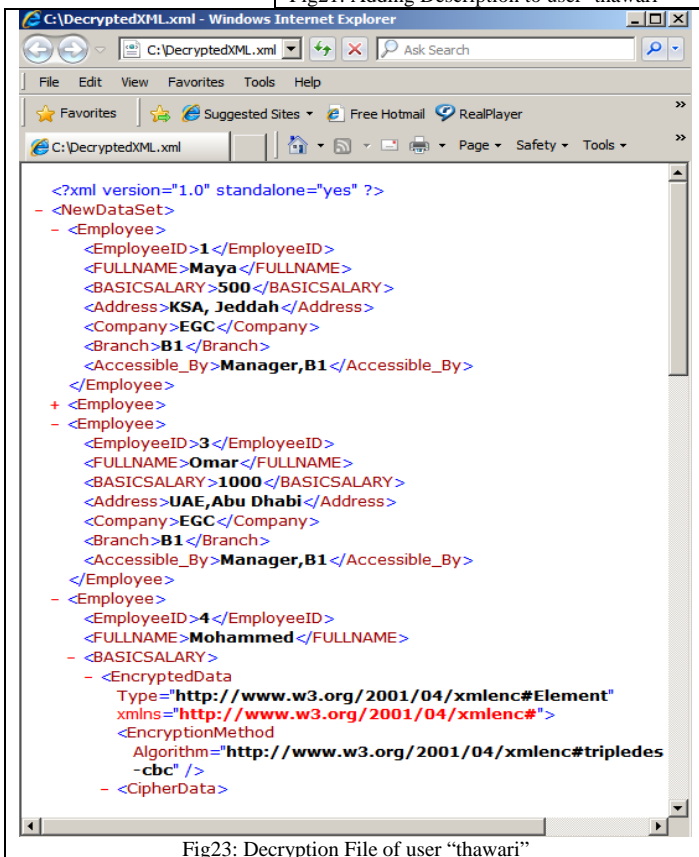


Fig23: Decryption File of user "thawari"

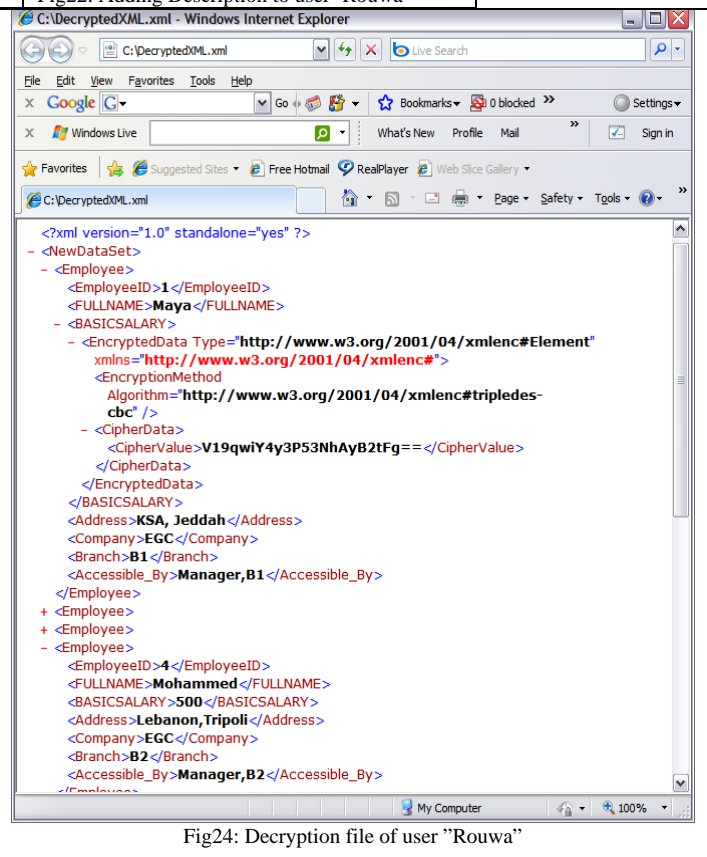


Fig24: Decryption file of user "Rouwa"

Getting User Information using WMI:

In order to get the description of the current user in the local PC, we must know some information concerning Windows Management Instrumentation.

WMI: Windows Management Instrumentation

Referring to [13], WMI is the instrumentation and plumbing through which almost all—Windows resources can be accessed, configured, managed, and monitored.

There are three steps common to any WMI script used in the script to retrieve information about a WMI managed resources: connecting to WMI service, retrieving a WMI managed resource and displaying properties of WMI managed resource. Establishing a connection to the Windows Management Service on a local or remote computer is done by calling VBScript's **GetObject** function and passing **GetObject** the name of the WMI Scripting Library's moniker, which is "winmgmts:" followed by the name of the target computer.

WMI Architecture

The WMI architecture consists of three primary layers: Managed resources, WMI infrastructure and Consumers.

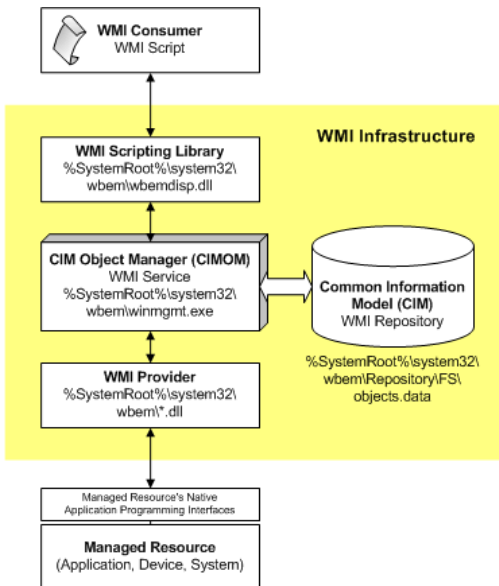


Fig25: WMI Infrastructure

Managed Resources

A managed resource is any logical or physical component, which is exposed and manageable by using WMI. Windows resources that can be managed using WMI include the computer system, disks, peripheral devices, event logs, files, folders, file systems etc... A WMI managed resource communicates with WMI through a provider.

WMI Infrastructure

The middle layer is the WMI infrastructure. WMI consists of three primary components: the Common Information Model Object Manager (CIMOM), the Common Information Model (CIM) repository, and providers. Together, the three WMI components provide the infrastructure through which configuration and management data is defined, exposed, accessed, and retrieved. A fourth component, even if small, but absolutely essential to scripting is the WMI scripting library.

CIM Repository

WMI is based on the idea that configuration and management information from different sources can be uniformly represented with a schema. The CIM is the schema, also called the object repository or class store that models the managed environment and defines every piece of data exposed by WMI. The schema is based on the DMTF Common Information Model standard.

Much like Active Directory's schema is built on the concept of a class, the CIM consists of *classes*. A class is a blueprint for a WMI manageable resource.

Classes are grouped into *namespaces*, which are logical groups of classes representing a specific area of management. For example, the root\cimv2 namespace includes most of the classes that represent resources commonly associated with a computer and operating system.

WMI Providers

WMI providers act as an intermediary between WMI and a managed resource. Providers request information from, and send instructions to WMI managed resources on behalf of consumer applications and scripts. For example, **WIN32 provider** provides information about the computer, disks, peripheral devices, files, folders, file systems, networking components, operating system, printers, processes, security, services, shares, SAM users and groups, and more.

4. Conclusions

To sum up, the idea of this paper has born to address a certain limitation in Windows Right Management Services which is the non-reusability of a file per different trusted recipients. Since WRMS provides a security on the whole document, it is not possible to share a part of the data in an RMS-enabled file with other trusted recipients having different privileges. Therefore, the sender is obliged to send to those recipients' different files that do not contain confidential data, and applies WRMS on each file.

This paper has introduced a new layer for securing sensitive data in Excel worksheet. It gives the flexibility to the sender to send only one copy of the file to different trusted recipients having different privileges and permissions. The provided solution will apply a security on a portion of the document which contains the confidential and sensitive data. As a final result, it does a selective access to the data: each receiver can only see data that grants access to.

If this solution is accompanied with WRMS, the sender can get extra benefits such as: the Excel file can be kept secure toward unauthorized recipients in addition it can be ensured that the received data will not be changed.

The advantages of this solution are:

- Efficiency in memory usage, time saving and file reusability in a proper secure way that fits many recipients having different privileges and protects the file from unintended or malicious recipients.
- The provided solution can be considered as scalable since the tremendous number of users does not affect on how the solution works.
- Since this solution was implemented as a separate module (DLL file), any update or improvement in

the code became easier to apply since it doesn't affect other parts of the program [14].

- In addition, the separate module makes the solution user-friendly, since any basic programmer can get all its benefits simply by sending few parameters (such as name of certain files) without the need to master programming languages.
- Furthermore, this solution also provides a large compatibility and interoperability with different types of applications such as (Component Object Model) and different programming languages (such as C, Pascal, or standard call).
- This solution is not relying on WRMS; it is capable of working independently.

In future works, this security layer should be applicable in all Microsoft Office applications (word documents, PowerPoint presentations, Microsoft Access ...) and perhaps portable document file (.pdf) files. Furthermore, if the solution was extended to a web service, any application that supports web services can use it (such as: Java application, PHP application ...).

References

- [1] Deb Shinder, How the Windows Rights Management Service can Enhance the Security of your Documents. Published: Sep 23, 2003 Updated: Apr 06, 2005. http://www.windowsecurity.com/articles/Windows_Rights_Management_Service_Documents.html
- [2] Tony Bradley, NETWORK SECURITY TACTICS, Information protection: Using Windows Rights Management Services to secure data. Published: Aug 01, 2008: <http://searchsecurity.techtarget.com/tip/0,289483,sid14gci1287738,00.html>
- [3] Technical Overview of Windows Rights Management Services for Windows Server 2003: www.winbr.com.br/downloads/RMSTechOverview
- [4] XrML, W3C.
- [5] Visual Basic .NET for first time programmers Workbook, Question Edition, Document Version 1.1 Copyright 2004 LearnVisualStudio.Net pages: 131, 132, and 138. <http://www.learnvisualstudio.net/>
- [6] Jaspal Singh, Excel Connectivity in VB.NET published: Aug 18, 2005: http://www.codeproject.com/KB/vb/Excel_Connectivity.aspx
- [7] Import from excel 2007 into dataset problem: <http://forums.asp.net/p/1093352/1644797.aspx#164479>
- [8] Bilal Siddiqui, Exploring XML Encryption Part 1, *demonstrating the secure exchange of structured data*. Published: Mar 01, 2002: <http://www.ibm.com/developerworks/xml/library/x-encrypt/>
- [9] XPath Tutorial: <http://www.w3schools.com/XPath/default.asp>
- [10] Win32_userAccount Class: [http://msdn.microsoft.com/en-us/library/aa394507\(VS.85\).aspx#properties](http://msdn.microsoft.com/en-us/library/aa394507(VS.85).aspx#properties)
- [11] Vicky Desjardins, Script to get local user, description, last logon, Group Membership for dummies'. Published: Feb 15, 2007: <http://www.visualbasicscript.com/m43246.aspx>
- [12] Impersonation of Client: <http://msdn.microsoft.com/enus/library/system.management.connectionoptions.impersonation.aspx>
- [13] Windows Management Instrumentation: <http://msdn.microsoft.com/en-us/library/ms974579.aspx>
- [14] Advantages of Dynamic Linking: [http://msdn.microsoft.com/en-us/library/ms681938\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms681938(VS.85).aspx)