Protecting Microsoft Azure Blob Storage with Microsoft Azure AD Rights Management in cloud services and web applications

Summary

Microsoft Azure AD Rights Management service introduces a whole new set of scenarios, enabling on-premises and cloud services to protect any type of data when storing or sharing it to provide state-of-the-art security against unauthorized access or usage by end-users.

In some of these scenarios, services might want to protect data they store in Microsoft Azure Storage and sharing them with users.

In other scenarios, web applications might want to create a confidential document, protect it, and then grant access to a specific user with the option to download and view it on every PC or mobile device.

This document tries to enlighten these scenarios and to provide a how-to guide and some sample code.

Resources

- Download and Install Active Directory Rights Management Service Client 2.1: http://www.microsoft.com/en-us/download/details.aspx?id=38396
- AD RMS SDK 2.1 Interop Library: http://code.msdn.microsoft.com/windowsdesktop/AD-RMS-SDK-20-Interop-eb3fbce7
- How to enable your service application to work with Microsoft Azure AD Rights Management: http://msdn.microsoft.com/en-us/library/dn133057(v=vs.85).aspx
- Manage Microsoft Azure AD using Windows PowerShell: http://technet.microsoft.com/en-us/library/jj151815.aspx
- Microsoft Azure AD Rights Management Administration http://www.microsoft.com/en-us/download/details.aspx?id=30339

- How to use the Microsoft Azure Blob Storage Service in .NET: http://www.windowsazure.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs-20/

Prerequisites

The scenarios described in this document require the following from your service:

- **Supported Operation System:** Windows 7, Windows 8, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012.
- **Programming Language:** C/C++ *or* C# (.NET Framework 3.0 and above).
- **For Microsoft Azure development:** Azure Cloud Services (web roles and worker roles) and Azure Virtual Machines (with a supported OS) are supported *(Azure Web Sites and Azure Media Services are currently not supported).*
- **Azure RMS license:** If your organization is already using Office 365, Azure RMS is available for you (with E3, E4, A3, A4 plans) - you can read more about it here.
  If you want to try this for yourself, by using a Trial Office 365 subscription, sign up here.

Scenarios

Some organizations would like to give their users a way to continue to work and consume data in their usual behaviors even when data leaves the organization premises to the cloud and to devices. Microsoft Azure AD RMS allows enterprises to persistently protect their data using identity-based policy, so the organization's data gets protected before it leaves its premises, to provide their users the same behavior they are used to, but with the encryption and security required. These scenarios are sometimes called CEG (Content Encryption Gates).

AD RMS SDK 2.1 for .NET, which works with Microsoft Azure AD Rights Management, enables the key scenarios below.

- **Scenario 1: a cloud service, which processes and stores confidential information, can encrypt any type of data before storing it in Microsoft Azure Blob Storage Service using Microsoft Azure AD Rights Management protection.**
  In this scenario, a developer has a service, such as an Azure web role or an Azure worker role, hosted in Microsoft Azure. The service needs to store a document (or any kind of files) with confidential information (which can be in any size and in any form of unstructured data). This document can be shared or accessed by only a specific set of users, from anywhere in the world via HTTP or HTTPS.
  Thus, before the service uploads the data to Microsoft Azure Blob Storage Service, it creates a policy and applies it on the data, encrypts it, and only then stores the document in Azure Storage. This document can be later read and be used (decrypted) by the users in the policy, or by allowed services for further processing.

Microsoft provides AD RMS SDK 2.1 (for .NET services) or AD RMS SDK 3.0 (for mobile clients, e.g. iOS, Android, WinPhone) to use and decrypt these blobs.

For a code sample that demonstrate this scenario, see Protecting Microsoft Azure Blob Storage using Microsoft Azure AD RMS.

- **Scenario 2: a web application, which contains confidential information, can protect exported data or reports before a user downloads them to his device**
  there is a service or a web application running on an on-premises Windows server or on a Windows virtual machine in Microsoft Azure. The service allows users to export and download confidential reports that may be shared and accessed by only specific users with certain permissions or usage rights (e.g. view only, no print, no edit, etc), from any device (mobile or PC) anywhere in the world.
  Thus, when a user wants to download a confidential document from the web site, the web application will create a RMS policy based on the user's profile and identity, then it will encrypt the file, and will store it in Microsoft Azure Blob Storage. The user can then download the file from any device anywhere in the world using HTTP/S.

  For a code sample that demonstrate this scenario, see Applying Protection in Web Applications using Microsoft Azure AD RMS and Microsoft Azure Blob Storage.

Features

AD RMS SDK 2.1 and Microsoft Azure AD Rights Management offer the following features for cloud applications and services:

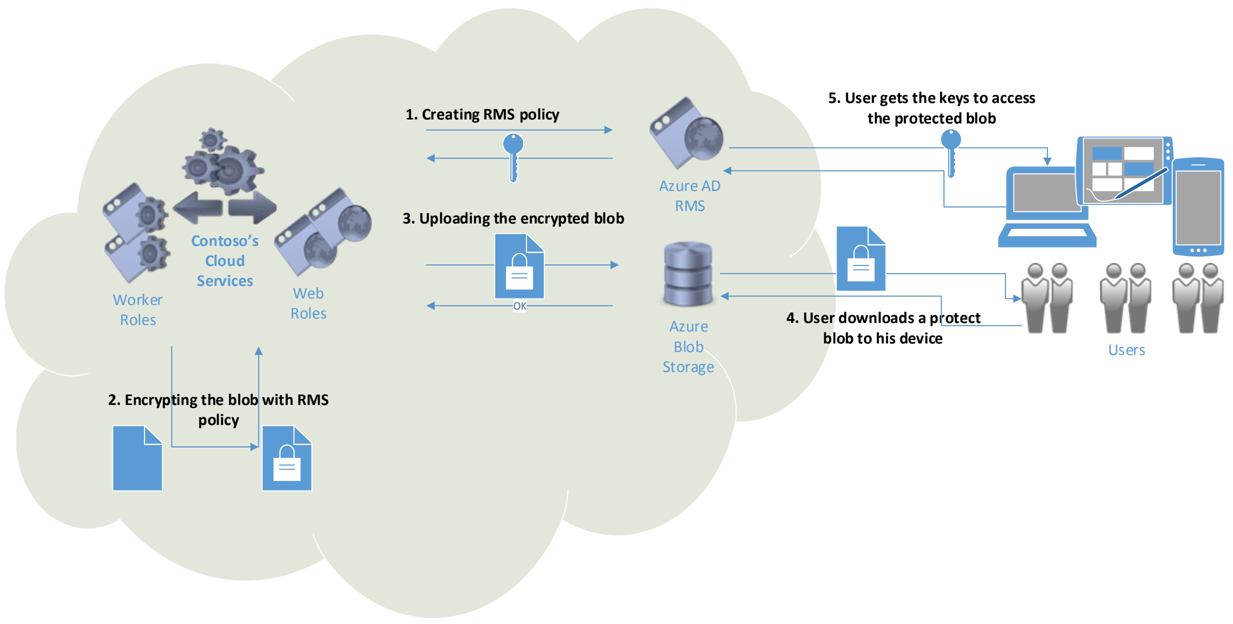
- **Encrypt any type of information:** Microsoft Azure AD Rights Management can protect and encrypt every file type, including documents, media, binary objects (BLOBS), cryptographic keys, and more.
  With the new generic protected file format (PFILE) your service can protect any file type. The protected file format contains the encrypted information, the usage policy (e.g. rights, users, expiration date) and other metadata fields.

-

- **Apply custom usage policies:** Microsoft Azure AD Rights Management allows you to choose which users will be able to access the files, which rights will they have (e.g. view, edit, print, copy), when will their access expire, and more. When a user wants to download confidential information, your service can create a RMS policy based on the user's policy, apply this policy and encrypt the file (sometimes known as "Content Encryption Gates"). Your users will be able to consume the protected content on any device (e.g. Windows, Mac, Android, iOS, WinPhone, etc) from anywhere in the world using HTTP/S.

High-Level Architecture

The following diagram describes how one of Contoso's cloud services uses Azure AD Rights Management to protect data and store it in Azure Blob Storage, which can be later accessed by end-users.



1. Creating RMS policy
5. User gets the keys to access the protected blob
Azure AD RMS
Contoso's Cloud Services
Worker Roles
Web Roles
3. Uploading the encrypted blob
OK
Azure Blob Storage
4. User downloads a protect blob to his device
Users
2. Encrypting the blob with RMS policy

1. **Creating RMS policy –** Contoso's cloud service creates RMS policy using Azure AD RMS. The RMS policy contains the content encryption key.
2. **Encrypting the blob with RMS policy –** Contoso's cloud service encrypts the blob with RMS policy before uploading it.
3. **Uploading the encrypted blob –** Contoso's cloud service uploads the encrypted blob to Azure Blob Storage and sends the blob's URL to the end-user.
4. **User downloads a protected blob to his device –** the end-user downloads the protected blob from Azure Blob Storage, which contains the RMS policy and the encrypted content
5. **User gets the keys to access the protected blob –** the end-user acquires the key to decrypt the content from Azure AD RMS with his usage rights.

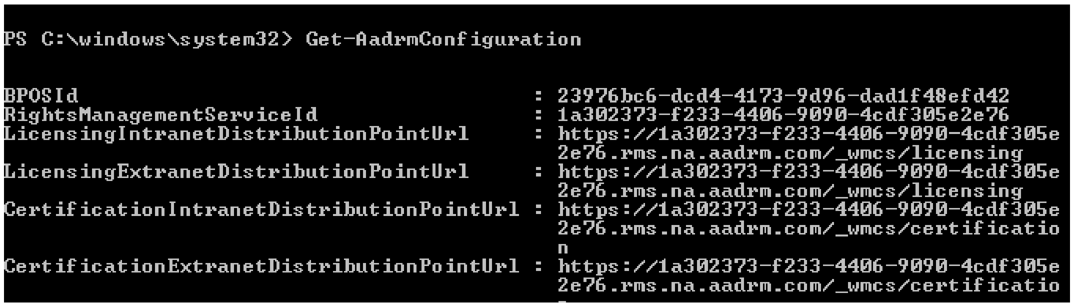
How to enable Microsoft Azure AD Rights Management in existing on-premises or cloud service

If you already have an on-premises or cloud service, follow these steps to enable your service or cloud app to use Azure AD Rights Management to protect data:

To use Microsoft Azure Rights Management service, you need to enable the service and create a service principal credential for authentication.

1. If your organization does not have Azure RMS subscription, sign up here to try it for yourself, by using a Trial Office 365 subscription.

2. You must activate Rights Management before you can begin to use the information rights management (IRM) features (read more here). If Azure Rights Management is not enabled in your organization, follow these instructions to **enable the Azure Rights Management service**:
    a. Download and install the RMS module for PowerShell from here.
    b. Run PowerShell as an Administrator.
    c. Import RMS module using the following CmdLet:
    `Import-Module AADRM`
    d. Connect to the service with your administrator credentials:
    `Connect-AadrmService –Verbose`
    e. Enable RMS in your organization:
    `Enable-AADRM`
    f. Get your tenant ID by running:
    `Get-AadrmConfiguration`



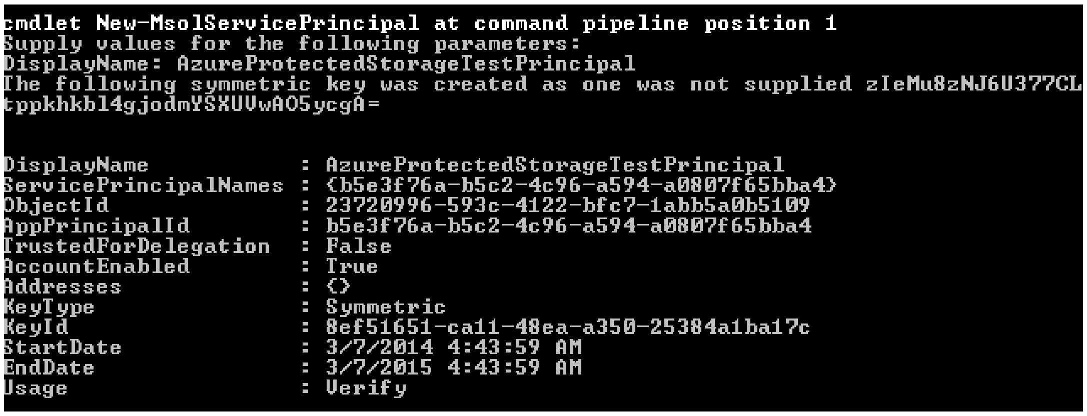
```
PS C:\windows\system32> Get-AadrmConfiguration

BPOSId                                   : 23976bc6-dcd4-4173-9d96-dad1f48efd42
RightsManagementServiceId                : 1a302373-f233-4406-9090-4cdf305e2e76
LicensingIntranetDistributionPointUrl    : https://1a302373-f233-4406-9090-4cdf305e
                                           2e76.rms.na.aadrm.com/_wmcs/licensing
LicensingExtranetDistributionPointUrl    : https://1a302373-f233-4406-9090-4cdf305e
                                           2e76.rms.na.aadrm.com/_wmcs/licensing
CertificationIntranetDistributionPointUrl : https://1a302373-f233-4406-9090-4cdf305e
                                           2e76.rms.na.aadrm.com/_wmcs/certificatio
                                           n
CertificationExtranetDistributionPointUrl : https://1a302373-f233-4406-9090-4cdf305e
                                           2e76.rms.na.aadrm.com/_wmcs/certificatio
```

    Note: your tenant ID is the GUID value of BPOSId, keep it somewhere to use it again.
    g. Disconnect from the service:

`Disconnect-AadrmService`Note: services need to use *service principals (*also known as *service identities)*, which are a type of credentials configured globally for access control, and allow your service or to authenticate directly with Microsoft Azure AD and to protect information using

Microsoft Azure AD Rights Management service.

3. To **create a service principal**, which will be used by your service for interacting with Azure RMS:
   a. Install the Microsoft Azure AD Module for Windows PowerShell from here.
   b. Run PowerShell as an Administrator.
   c. Import Microsoft Azure AD module using the following CmdLet:
      `Import-Module MSOnline`
   d. Connect to your online service with your administrator credentials:
      `Connect-MsolService`
   e. Create a new service principal by running:
      `New-MsolServicePrincipal`
      After you provide your desired display name, the service principal will be created.
      Note: copy the new symmetric key and save it somewhere safe, as you will not be able to get it again.



```
cmdlet New-MsolServicePrincipal at command pipeline position 1
Supply values for the following parameters:
DisplayName: AzureProtectedStorageTestPrincipal
The following symmetric key was created as one was not supplied zIeMu8zNJ6U377CL
tppkhkbl4gjodmYSXUVwAO5ycgA=

DisplayName           : AzureProtectedStorageTestPrincipal
ServicePrincipalNames : {b5e3f76a-b5c2-4c96-a594-a0807f65bba4}
ObjectId              : 23720996-593c-4122-bfc7-1abb5a0b5109
AppPrincipalId        : b5e3f76a-b5c2-4c96-a594-a0807f65bba4
TrustedForDelegation  : False
AccountEnabled        : True
Addresses             : {}
KeyType               : Symmetric
KeyId                 : 8ef51651-ca11-48ea-a350-25384a1ba17c
StartDate             : 3/7/2014 4:43:59 AM
EndDate               : 3/7/2015 4:43:59 AM
Usage                 : Verify
```

   f. Save the symmetric key, application principal id, and tenant id as you will use them soon, in this example that would be:
      Symmetric key: *zIeMu8zNJ6U377CLtppkhkbl4gjodmYSXUVwAO5ycgA=*
      AppPrincipalId: *b5e3f76a-b5c2-4c96-a594-a0807f65bba4*
      Tenant Id (BPOSId): *23976bc6-dcd4-4173-9d96-dad1f48efd42*

4. **Download Active Directory Rights Management Service Client 2.1** from here:
   http://www.microsoft.com/en-us/download/details.aspx?id=38396
   and **install** it on the machines where your service runs to enable RMS functionality.

   Note: if you have several instances that running your service please make sure to install AD RMS client 2.1 on each of them.

5. If you need to protect other files types than Office and PDF files, you need to **add the following registry key**:

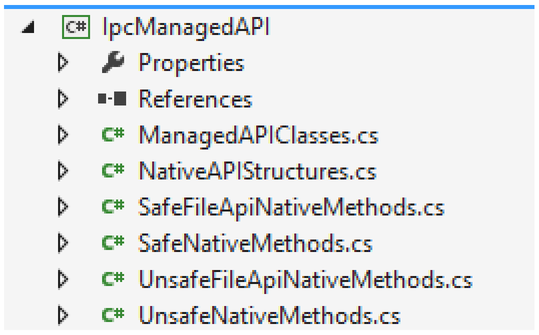
```
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSIPC\FileProtection\*]
"Encryption"="Pfile"
```

Or if your service is 32-bit running on 64-bit machine add the following key:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\MSIPC\FileProtection]
"Encryption"="Pfile"
```

Note: for more information about File API configuration click here.

6. **Download AD RMS SDK 2.1 Interop Library** from here: http://code.msdn.microsoft.com/windowsdesktop/AD-RMS-SDK-20-Interop-eb3fbce7 extract the .zip file, and add IpcManagedAPI project into your Visual Studio solution.



```
IpcManagedAPI
    Properties
    References
    ManagedAPIClasses.cs
    NativeAPIStructures.cs
    SafeFileApiNativeMethods.cs
    SafeNativeMethods.cs
    UnsafeFileApiNativeMethods.cs
    UnsafeNativeMethods.cs
```

7. **In your service project, add a reference to IpcManagedAPI:** Right-click on your project in the Solution Explorer, then choose Add->Reference... and choose IpcManagedAPI project and click OK.

8. To enable RMS in your project, your service first need to initialize RMS and create a symmetric key credential with your service principal.
If you write a role-based class (web Role or worker Role), a good place to init can be the overridden OnStart() function:

```
public override bool OnStart()

{
        SafeNativeMethods.IpcInitialize();
        SafeNativeMethods.IpcSetAPIMode(APIMode.Server);

        // Create the credentials for the service principal using a symmetric key
        SymmetricKeyCredential symmetricKeyCred = new SymmetricKeyCredential();
        symmetricKeyCred.AppPrincipalId = “b5e3f76a-b5c2-4c96-a594-a0807f65bba4”;
        symmetricKeyCred.Base64Key = "zIeMu8zNJ6U377CLtppkhkbl4gjodmYSXUVwAO5ycgA=";
        symmetricKeyCred.BposTenantId = "23976bc6-dcd4-4173-9d96-dad1f48efd42";

        // TODO: save the credential object reference so you can use it when calling RMS
        client methods

        return base.OnStart();
}
```

9. That's it, your service is now set and ready to protect and unprotect data using Microsoft Azure AD Rights Management service.

The following code samples show how to protect or unprotect data in Microsoft Azure Blob Storage using Azure RMS (you can learn more on how to use the Microsoft Azure Blob Storage Service in .NET here: http://www.windowsazure.com/en-us/documentation/articles/storage-dotnet-how-to-use-blobs-20/)

Protect a file using a template and upload it to Azure Blob Storage

The following code demonstrates how to get the list of RMS templates using the symmetric key of the service principal as authentication method, then to protect and encrypt the file, and to upload it to Microsoft Azure Blob Storage:

```
        /// <summary>
        /// Protect a file using a template and upload it to Azure Blob Storage
        /// </summary>
        /// <param name="filePath">input file path</param>
        public void ProtectWithTemplateAndUploadBlob(string filePath)
        {
            // Get the available templates for this tenant
            Collection<TemplateInfo> templates =
SafeNativeMethods.IpcGetTemplateList(null, 0,
                true, false, true, null, null, this.symmetricKeyCredential);

            // Encrypt the file using the first template in the collection
            TemplateInfo selectedTemplateInfo = templates.ElementAt(1);
            string encryptedFilePath = SafeFileApiNativeMethods.IpcfEncryptFile(filePath,
selectedTemplateInfo.TemplateId,
                SafeFileApiNativeMethods.EncryptFlags.IPCF_EF_FLAG_KEY_NO_PERSIST, true,
false, true, null, this.symmetricKeyCredential);

            // Upload the encrypted file to the Azure Blob Storage
            UploadBlobToContainer(encryptedFilePath);

            // Delete the encrypted file
            File.Delete(encryptedFilePath);
        }


        public void UploadBlobToContainer(string filePath)
        {
            // Retrieve storage account from connection string
            CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
                CloudConfigurationManager.GetSetting("StorageConnectionString"));
            // Create the blob client
            CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

            // Retrieve a reference to a container
            CloudBlobContainer container =
blobClient.GetContainerReference("mycontainer");

            // Retrieve reference to a blob
            CloudBlockBlob blockBlob =
container.GetBlockBlobReference(Path.GetFileName(filePath));

            // Create or overwrite the "myblob" blob with contents from a local file.
            using (var fileStream = System.IO.File.OpenRead(filePath))
            {
                blockBlob.UploadFromStream(fileStream);
            }
        }
```

Protect a file using an ad-hoc policy and upload it to Azure Blob Storage

The following code demonstrates how to create a custom RMS policy with read and print rights, with granted access for johndoe@domain.com using the symmetric key of the service principal as authentication method, then to protect and encrypt the file, and to upload it to Microsoft Azure Blob Storage:

```
        /// <summary>
        /// Protect a file using an ad-hoc policy and upload it to Azure Blob Storage
        /// </summary>
        /// <param name="filePath"></param>
        public void ProtectWithAdHocPolicyAndUploadBlob(string filePath)
        {
            // Create an ad-hoc policy for one email address with 'read' and 'print'
rights
            Collection<string> rights = new Collection<string>();
            rights.Add("READ");
            rights.Add("PRINT");

            Collection<UserRights> userRights = new Collection<UserRights>();
            userRights.Add(new UserRights(UserIdType.Email, "johndoe@domain.com",
rights));

            string templateName = "Confidential - Read and Print only";
            string templateDescription = "Confidential information shared with John Doe
only - Read and Print allowed";
            string issuerDisplayName = "Contoso web page";

            // Get the available issuers of rights policy templates.
            // The available issuers is a list of RMS servers that this user has already
contacted.
            Collection<TemplateIssuer> templateIssuers =
SafeNativeMethods.IpcGetTemplateIssuerList(null,
                true, false, false, true, null, this.symmetricKeyCredential);
            // Create the policy and associate the chosen user rights with it
            SafeInformationProtectionLicenseHandle handle = 
SafeNativeMethods.IpcCreateLicenseFromScratch(templateIssuers.ElementAt(0));
            SafeNativeMethods.IpcSetLicenseUserRightsList(handle, userRights);
            SafeNativeMethods.IpcSetLicenseDescriptor(handle, new TemplateInfo(null, 
CultureInfo.CurrentCulture,
                templateName, templateDescription, issuerDisplayName, false));

            // Encrypt the file using the ad-hoc policy
            string encryptedFilePath = SafeFileApiNativeMethods.IpcfEncryptFile(filePath, 
handle,
                SafeFileApiNativeMethods.EncryptFlags.IPCF_EF_FLAG_KEY_NO_PERSIST, true, 
                false, true, null, this.symmetricKeyCredential);

            // Upload the encrypted file to the Azure Blob Storage
            UploadBlobToContainer(encryptedFilePath);

            // Delete the encrypted file
            File.Delete(encryptedFilePath);
        }
```

Download a RMS protected blob and unprotect it

The following code demonstrates how to download a protected file from Microsoft Azure Blob Storage, then unprotect and decrypt the file using the symmetric key of the service principal as authentication method, so the file can be read and used by the service:

```
        /// <summary>
        /// Download a RMS protected blob and unprotects it
        /// </summary>
        /// <param name="blobName">blob name</param>
        /// <param name="downloadedBlobPath">the unprotected downloaded blob path</param>
        public void DownloadAndUnprotectBlob(string blobName, string downloadedBlobPath)
        {
            // Download the protected blob
            string downloadedEncryptedBlobPath = downloadedBlobPath + ".tmp";
            DownloadBlobFromContainer(blobName, downloadedEncryptedBlobPath);

            // Unprotect the blob
            string decryptedFilePath = 
SafeFileApiNativeMethods.IpcfDecryptFile(downloadedEncryptedBlobPath,
                SafeFileApiNativeMethods.DecryptFlags.IPCF_DF_FLAG_DEFAULT, true, false, 
true, null, this.symmetricKeyCredential);

            // Delete the downloaded protected blob, and move the unprotected blob to the 
required path
            File.Delete(downloadedEncryptedBlobPath);
            File.Move(decryptedFilePath, downloadedBlobPath);
        }



       public void DownloadBlobFromContainer(string blobName, string downloadedBlobPath)
        {
            // Retrieve storage account from connection string
            CloudStorageAccount storageAccount = CloudStorageAccount.Parse(
                CloudConfigurationManager.GetSetting("StorageConnectionString"));

            // Create the blob client
            CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();

            // Retrieve a reference to a container
            CloudBlobContainer container =
blobClient.GetContainerReference("mycontainer");

            // Retrieve reference to a blob by its name
            CloudBlockBlob blockBlob = container.GetBlockBlobReference(blobName);

            // Save blob contents to a file.
            using (var fileStream = System.IO.File.OpenWrite(downloadedBlobPath))
            {
                blockBlob.DownloadToStream(fileStream);
            }
        }
```

Conclusion

Microsoft Azure AD Rights Management is a simple and a powerful service that allows your applications and services to protect your organization data before it goes to the cloud. By implementing the steps in this tutorial you quickly enabled the ability to protect or unprotect files in your service and store or share them in the cloud.

See also

- RMS for IT Professionals:
  Find videos, step-by-step guides, and presentations to help you manage the systems that protect your data, provide compliance reports, teach users how to securely share information, and keep everything always up and running.
  http://technet.microsoft.com/en-us/dn175751.aspx

- RMS for Developers:
  Find getting-started guides, code samples, and videos to add data protection to your application. Also, access straightforward libraries for encryption and security on all platforms: Windows computers, Windows tablets, Windows phones, Macintosh, iOS, Android, and web.
  http://technet.microsoft.com/en-us/dn175488