BEST: International Journal of Management, Information Technology and Engineering (BEST: IJMITE)
ISSN (P): 2348-0513, ISSN (E): 2454-471X,
Vol. 4, Issue 4, Apr 2016, 49-58

DIGITALLY SIGNING A WEB PAGE

SANJEET SINGH

R&D Department, Syscom Corporation Ltd**,** Uttar Pradesh, India

ABSTRACT

This paper talks about

- Why Digitally Signing a Web Page?
- Features of Digitally Signing a Web Page?
- How to use?

KEYWORDS: Digital, Digitally Signature, Signature, Web Page

INTRODUCTION

Hello Readers,

First of all, I would say this paper is just for ASP.net developers.

Digitally Signing a Web page as the name implies signing a webpage using digital certificate for maintaining the authenticity of the page. The most common use of a digital certificate is to verify that a user sending a message is who he or she claims to be, and to provide the receiver with the means to encode a reply. We can either use self-signed certificate or the certificate issued by Certificate authority (CA). The CA issues an encrypted digital certificate containing the applicant's public key and a variety of other identification information. The CA makes its own public key readily available through print publicity or perhaps on the internet. Both self-signed and CA signed certificates provide encryption for data in motion. A CA-signed certificate also provide authentication – a level of assurance that the site is what it reports to be, and not an imposter website.

Features of Digitally Signing a Web Page

- Maintains the authenticity of a web page. It means page is genuine not a fake.
- Reduces the effort by automating the signing process through digital signature instead of pen and paper. If your organization is big and there is lot of paper work, you can digitize the whole process saving pen, paper, time and effort.

How to Use

Step 1: Here I am using Self-signed certificate for signing the webpage. So first step will be to create the Self-signed certificate. Open IIS Manager by going to start and type inetmgr to open IIS. Choose server certificate and click create self-signed certificate in the action pane.

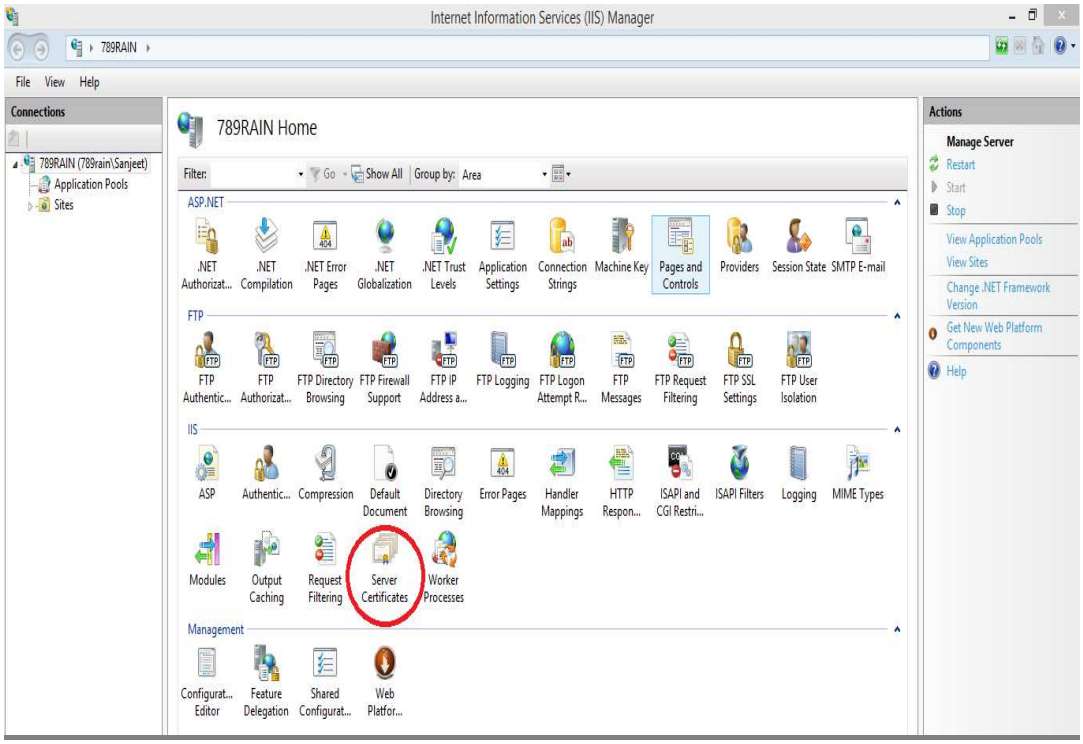

Figure 1: IIS Manager

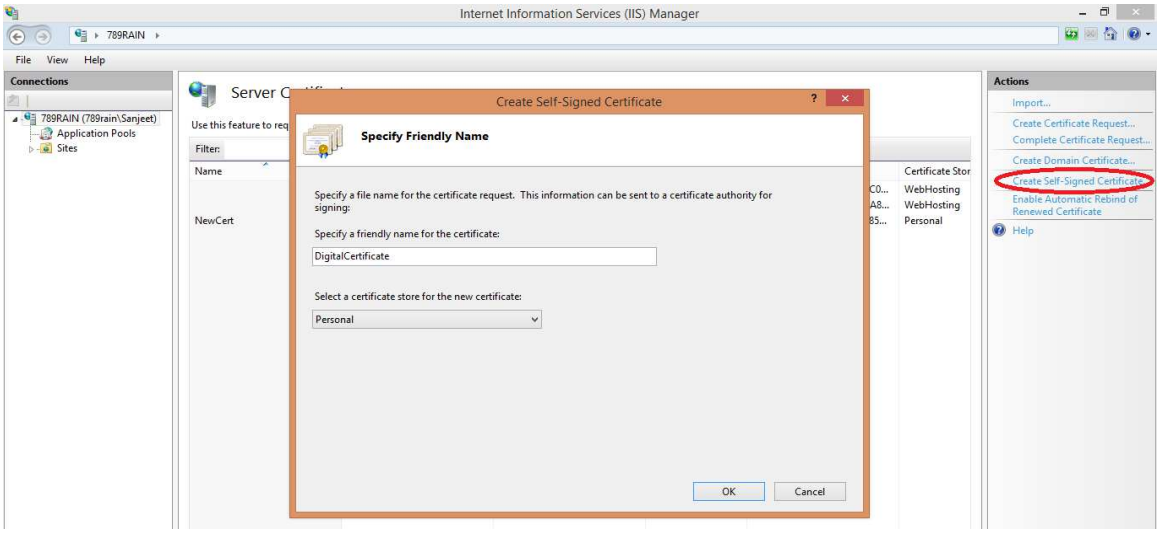

Figure 2: Create Self-Signed Certificate

Step 2: Export the certificate, double click on your certificate, go to details and choose copy to file. Export the private keys, otherwise we are going to face key not found error.

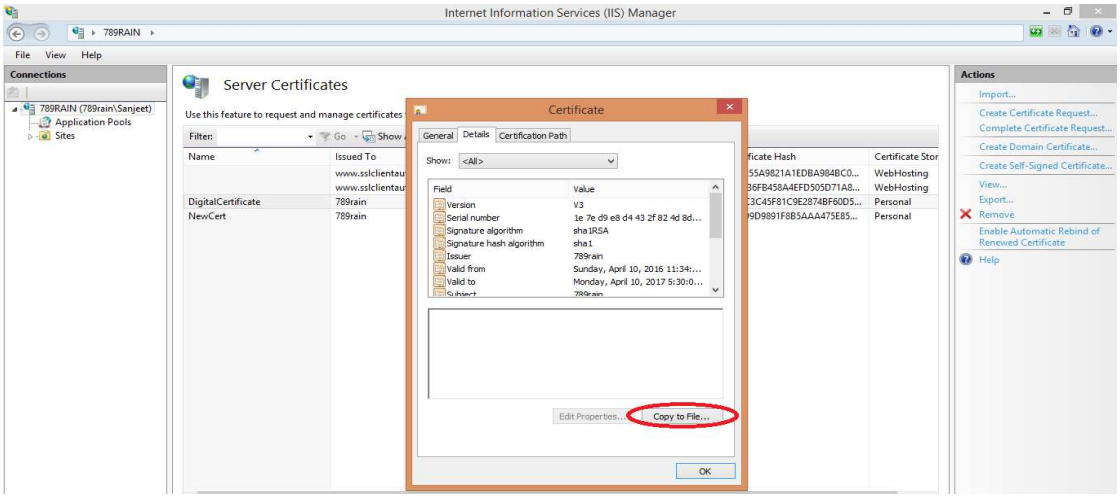

Figure 3: Save Certificate

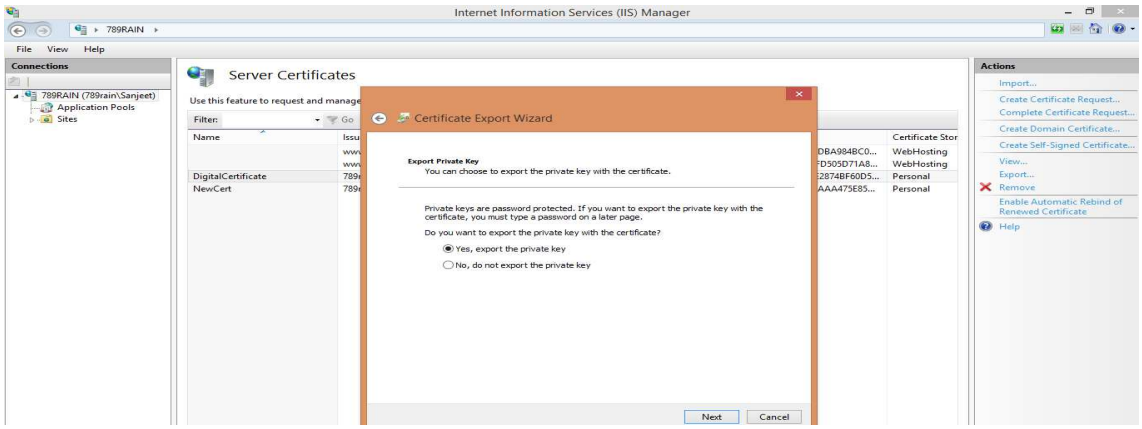

Figure 4: Export Private Keys

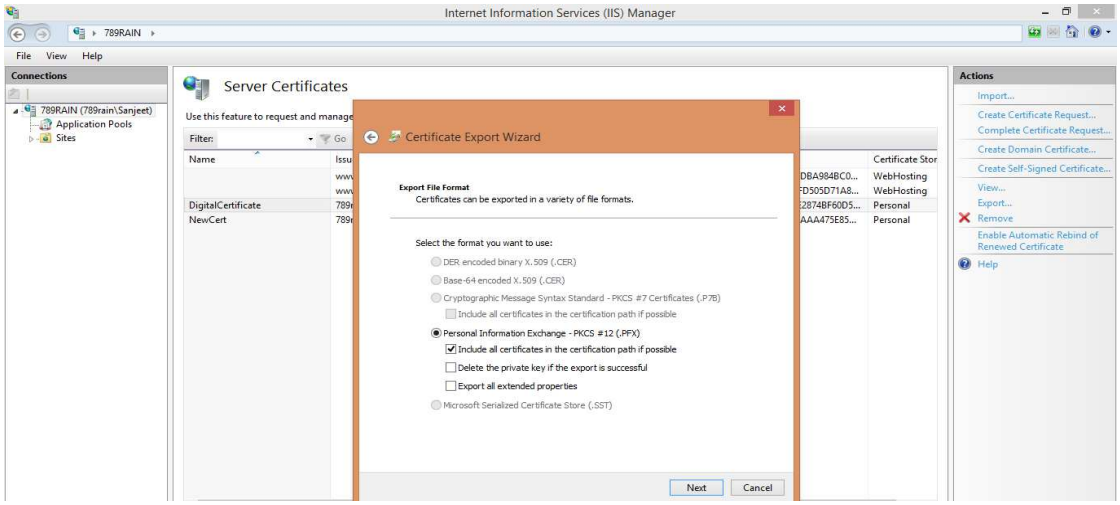

Figure 5: Choose Personal Information Exchange

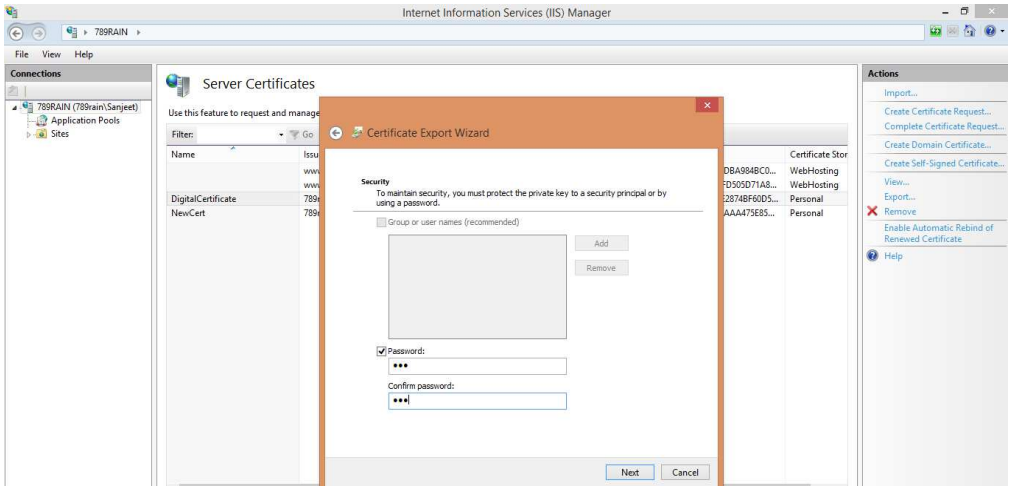

Figure 6: Set Password of Your Certificate

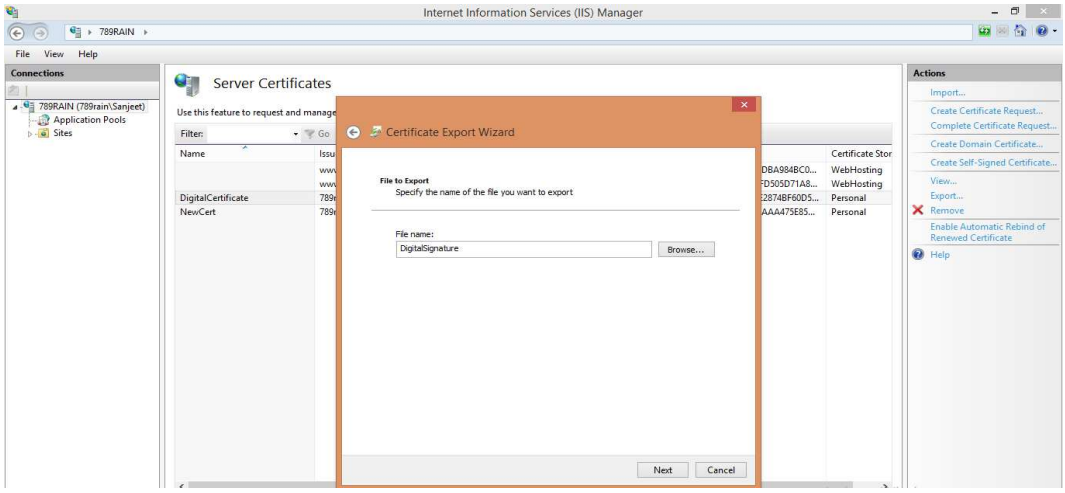

Figure 7: Give Name to Your PFX File

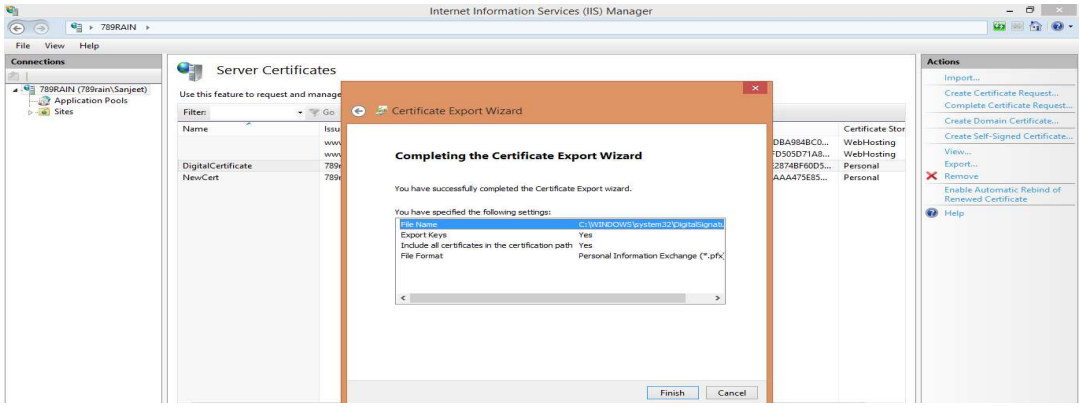

Figure 8: Complete the Certificate Wizard

Step 3: Place your certificate in the certificate store. Open Microsoft management console by going to start and type mmc.

Go to file and choose add/remove snap in option. Choose Certificates add and click ok. Choose the location where you want to save the certificate

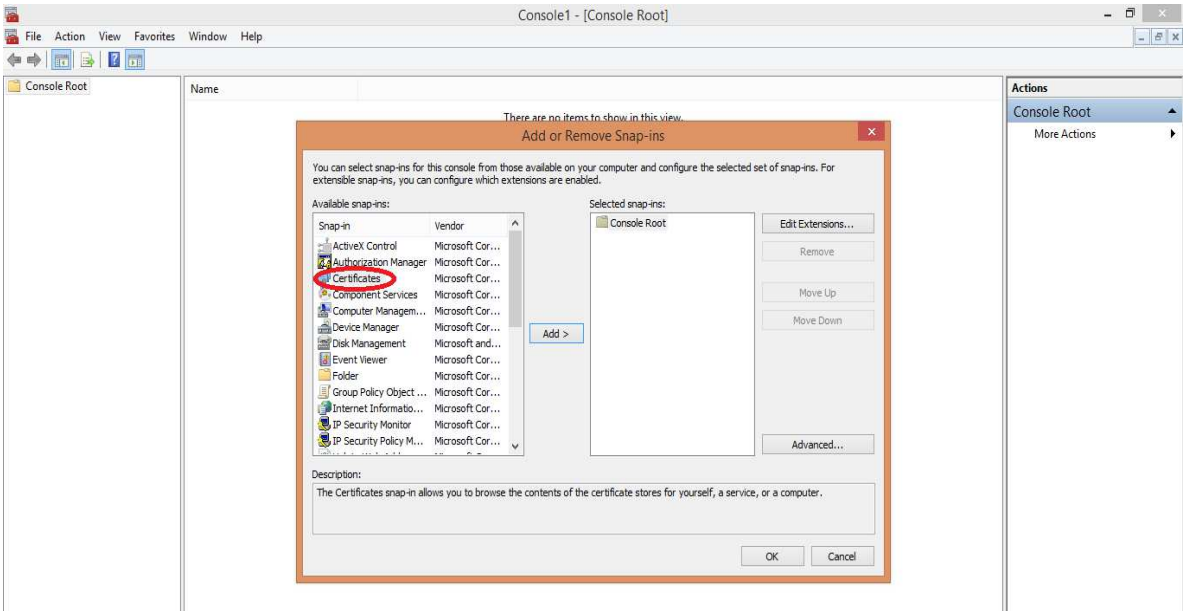

Figure 9: Microsoft Management Console

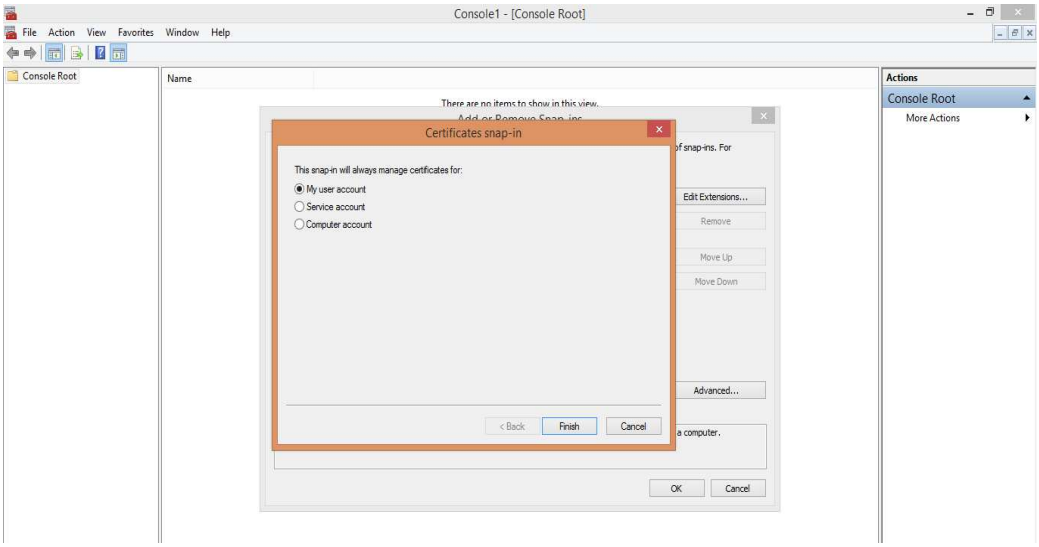

Figure 10

Step 4: Go to Personal folder, right click to import the certificate in the certificate store. Choose the certificate which you want to import.

Figure 11: Import the Certificate

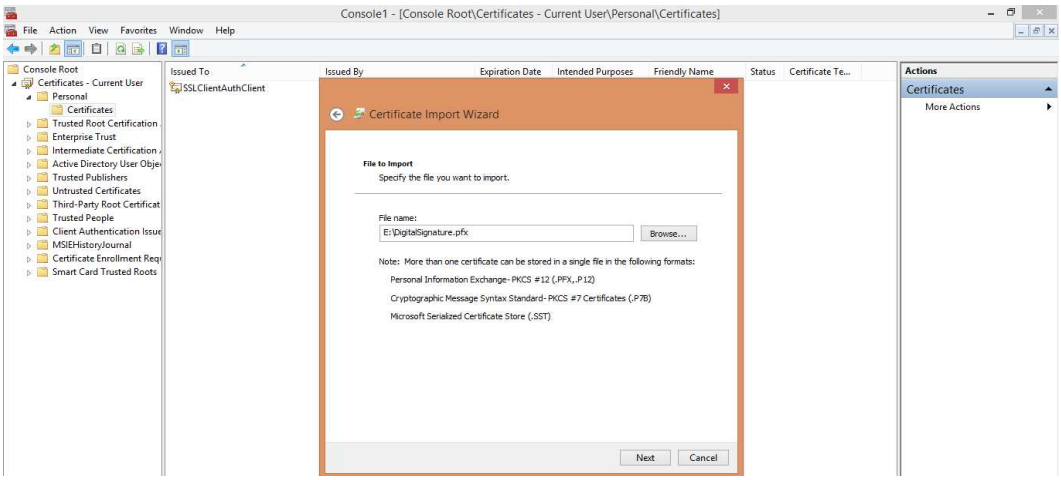

Figure 12: Choose the Certificate

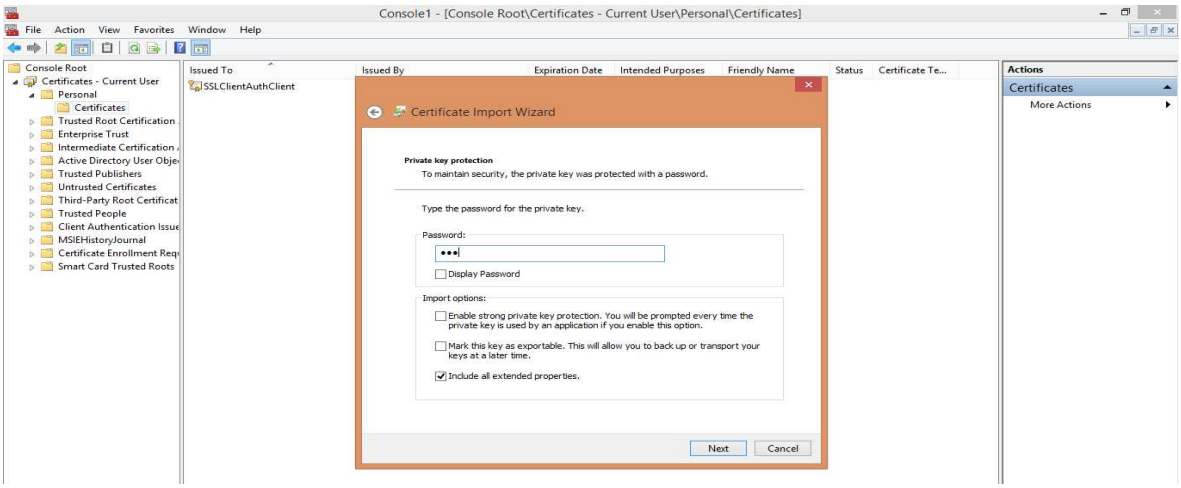

Figure 13: Password of the Certificate

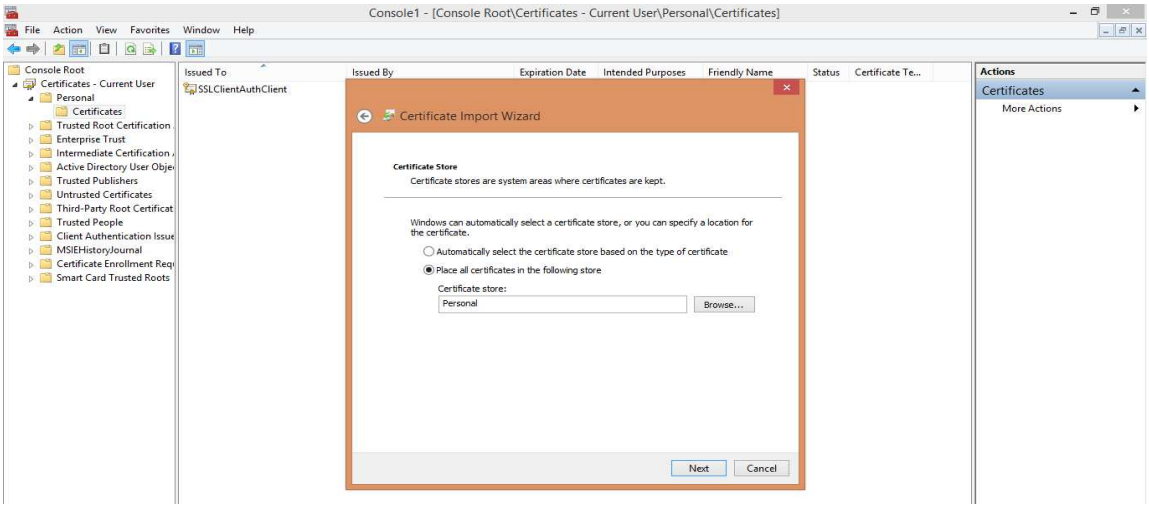

Figure 14: Place the Certificate in the Store

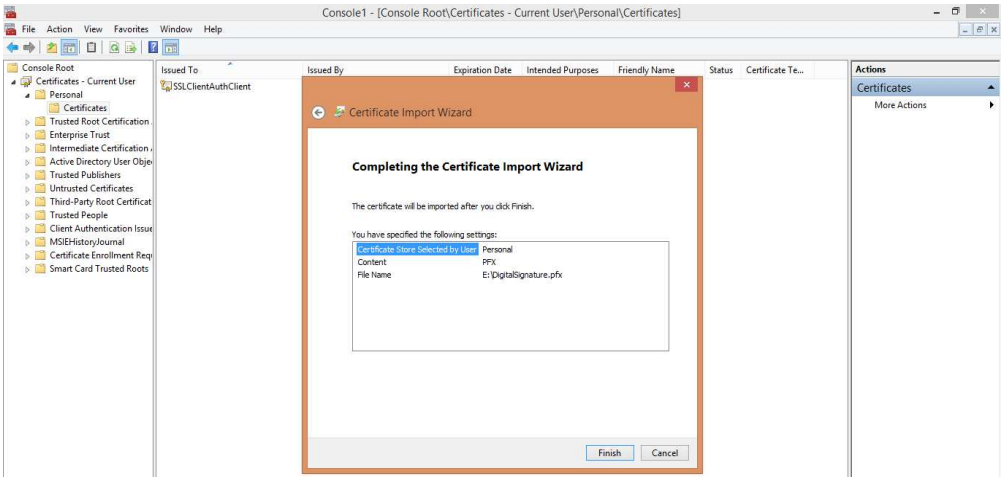

Figure 15: Complete the Import Process

Step 5: Export the certificate, which we are going to verify the digital signature. Follow the same steps as discussed above.

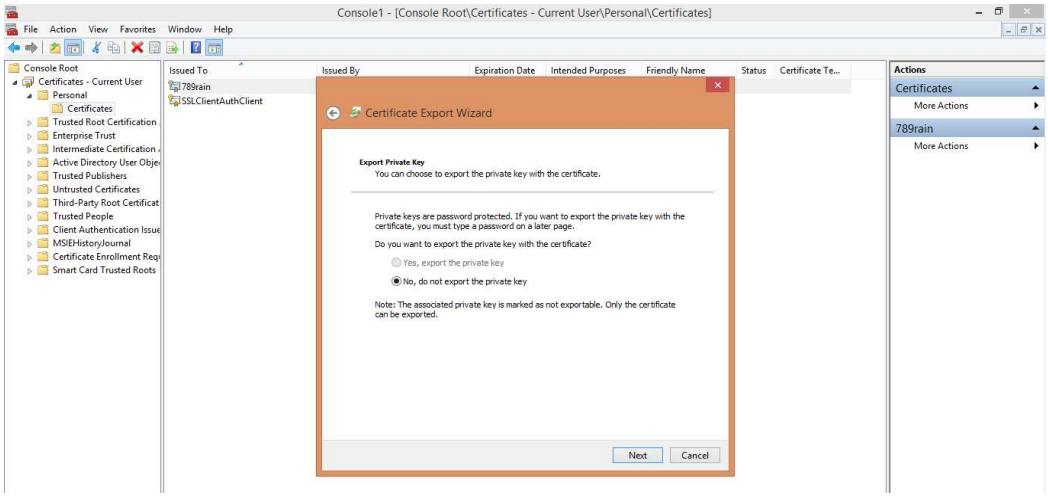

Figure 16: Export the Certificate

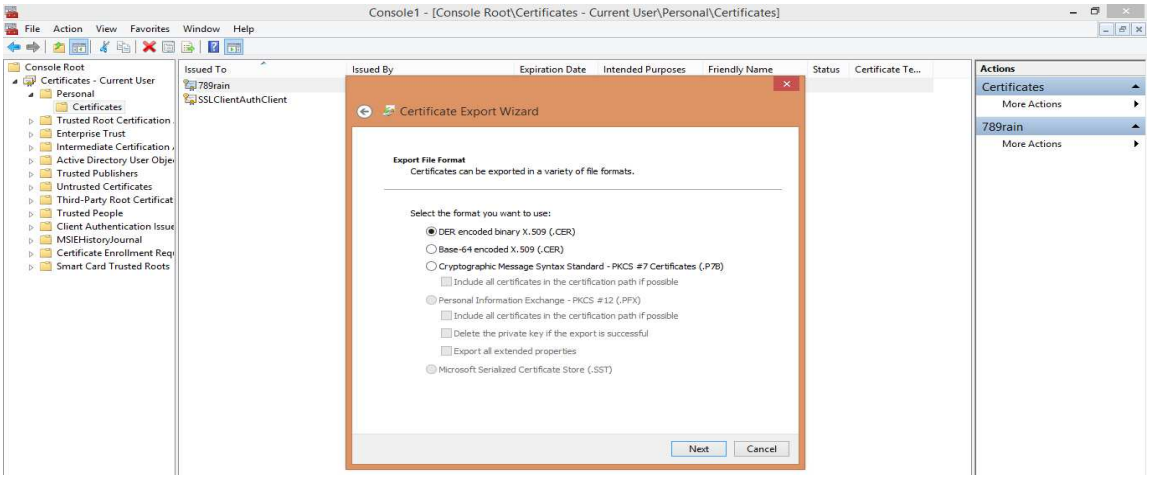

Figure 17: Export the Certificate

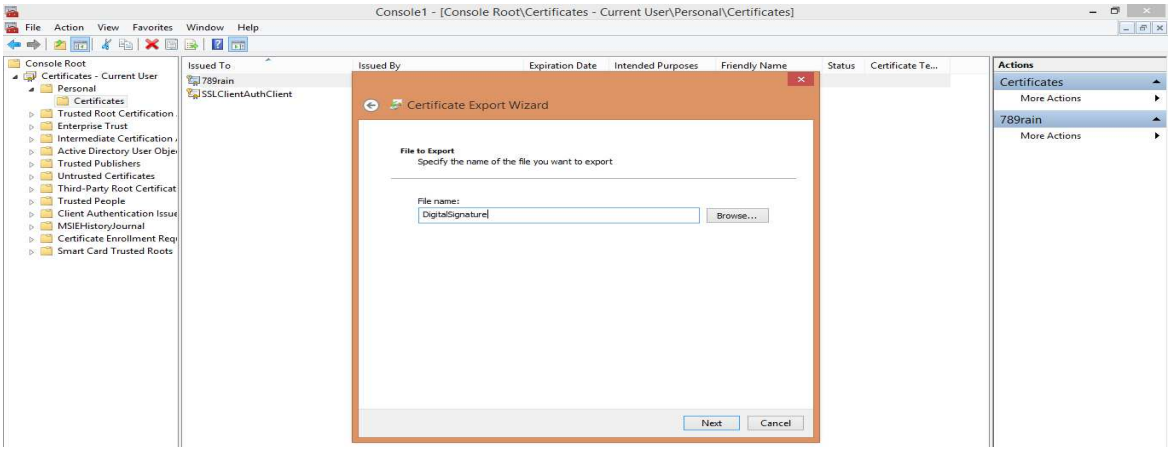

Figure 18: Name Your Certificate and Complete the Process

Step 6: Now certificate is created and placed in the certificate store, now is the time to use in our program.

In aspx file: In designing page we are placing the code to show, whether document is digitally signed or not.

```
<div>
        <asp:Label ID="lblHeading" runat="server" Text="Digital
signature"></asp:Label><br />
        <br />
        <asp:Panel runat="server" BorderWidth="1" BorderColor="Black" Width="250">
        <asp:Label runat="server" ID="lblSigned" ></asp:Label><br />
        <asp:Label ID="lbltext" runat="server"></asp:Label>
        </asp:Panel>
        </div>
```

Figure 19

In aspx.cs file: in code file, we are signing and verifying the signature.

```
protected void Page_Load(object sender, EventArgs e)
{
    try
    {
        // Sign text
        byte[] signature = Sign("DigitalSign", "CN=789rain");

        //Verify Text
        if (Verify("DigitalSign", signature, @"E:\search\Digital Signature\DigitalSignature.cer"))
        {
            lblSigned.Text = "Digitally Signed By :";
            lbltext.Text = "Sanjeet";
        }
        else
        {
            lbltext.Text = "ERROR: Signature not valid!";
        }
    }
    catch (Exception ex)
    {
        lbltext.Text = "EXCEPTION: " + ex.Message;
    }
}

static byte[] Sign(string text, string certSubject)
{
    // Access Personal (MY) certificate store of current user
    X509Store my = new X509Store(StoreName.My, StoreLocation.CurrentUser);
    my.Open(OpenFlags.ReadOnly);


    // Find the certificate we'll use to sign
    RSACryptoServiceProvider csp = null;
    foreach (X509Certificate2 cert in my.Certificates)
    {
        if (cert.Subject.Contains(certSubject))
        {
            //We found it.
            // Get its associated CSP and private key
            csp = (RSACryptoServiceProvider)cert.PrivateKey;
        }
    }
    if (csp == null)
    {
        throw new Exception("No valid cert was found");
```

Figure 20

```
// Hash the data
    SHA1Managed sha1 = new SHA1Managed();
    UnicodeEncoding encoding = new UnicodeEncoding();
    byte[] data = encoding.GetBytes(text);
    byte[] hash = sha1.ComputeHash(data);
    // Sign the hash
    return csp.SignHash(hash, CryptoConfig.MapNameToOID("SHA1"));
}
protected bool Verify(string text, byte[] signature, string certPath)
{
    // Load the certificate we'll use to verify the signature from a file
    X509Certificate2 cert = new X509Certificate2(certPath);
    RSACryptoServiceProvider csp = (RSACryptoServiceProvider)cert.PublicKey.Key;



    // Hash the data
    SHA1Managed sha1 = new SHA1Managed();
    UnicodeEncoding encoding = new UnicodeEncoding();
    byte[] data = encoding.GetBytes(text);
    byte[] hash = sha1.ComputeHash(data);



    // Verify the signature with the hash
    return csp.VerifyHash(hash, CryptoConfig.MapNameToOID("SHA1"), signature);
}
```

Figure 21

Step 7: If certificate is valid and signed text is verified than signature signed by Sanjeet will be shown otherwise Invalid certificate will be shown

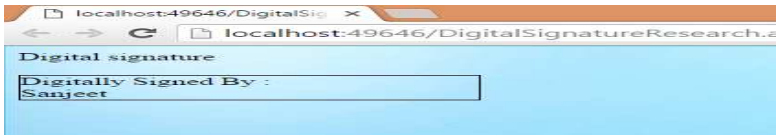

Figure 22: Valid Certificate Message

CONCLUSIONS

This study shows that digital signature is best for maintaining the authenticity of the page.

Reduces the efforts and save lot of time if paper work is large in your firm you can automate the whole process.

More user friendly, user will be provided with the interface they just needed a click of button to sign a document.

At last I want to say that it worth implementing in your organization because it provides more security.

ACKNOWLEDGEMENTS

This work was supported by Syscom Corporation Ltd.

REFERENCES

1 https://www.wikipedia.org/digitalcertificate