

```
//Verify the signatures attached to a signed PDF document
```

```
    iTextSharp.text.pdf.PdfReader reader = new iTextSharp.text.pdf.PdfReader
("c:\\signedDocument.pdf");

    //signature is an Adobe Acrobat file
    iTextSharp.text.pdf.AcroFields af = reader.AcroFields;

    //every signature has a name
    ArrayList names = af.GetSignatureNames();

    Console.WriteLine("PDF signature verification engine\n\n");

    foreach (string signature in names)
    {
        //signature covering
        Console.WriteLine("Signature covers whole document : " + af.
SignatureCoversWholeDocument(signature).ToString());

        //obtain information about the signature
        iTextSharp.text.pdf.PdfPKCS7 signatureProperties = af.VerifySignature
(signature);

        //sha1, sha256, md5, etc
        Console.WriteLine("Digest algorithm: " + signatureProperties.
GetDigestAlgorithm());

        //certificate used for signing
        Console.WriteLine("Signer: " + signatureProperties.Certificates[0].
SubjectDN.ToString());

        //information form signature
        Console.WriteLine("Location: " + signatureProperties.Location + "
Reason: " + signatureProperties.Reason + " Date:" + signatureProperties.SignDate.
ToString());

        try
        {

            //sig.Certificates[0].CheckValidity();

            //signing certificate
            Console.WriteLine("Certificate is valid: " + signatureProperties.
Certificates[0].IsValidNow.ToString());

            //certificate transformed in X509Certificate2 structure - part of
.NET Framework 2.0. Recommended to be used when you need to use the certificate for
other operations
            System.Security.Cryptography.X509Certificates.X509Certificate2
cert = new System.Security.Cryptography.X509Certificates.X509Certificate2
(signatureProperties.Certificates[0].GetEncoded());
            Console.WriteLine(cert.GetNameInfo(System.Security.Cryptography.
X509Certificates.X509NameType.SimpleName, false));

            //to see the certificate validity method in action - this will
throw an exception
            signatureProperties.Certificates[0].CheckValidity(DateTime.Now.
AddYears(-20));

            Console.WriteLine("Certificate validation: valid");
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }
    }
}
```

```
        try
        {
            //!!! THIS METHOD MAY BE USED ONLY THE SIGNATURE ALGORITHM IS SHA1 !!!
            Console.WriteLine("Document integrity: " + signatureProperties.Verify().ToString());
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
        }

        //generic signature name
        Console.WriteLine("Signature name: " + signature);

        Console.WriteLine("\n\n");
    }
}
```