عنوان: **رمزنگاری فایل‌های PDF با استفاده از کلید عمومی توسط iTextSharp**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۸/۲۱ ۱۳:۴۴:۰۰
آدرس: www.dotnettips.info
گروه‌ها: iTextSharp

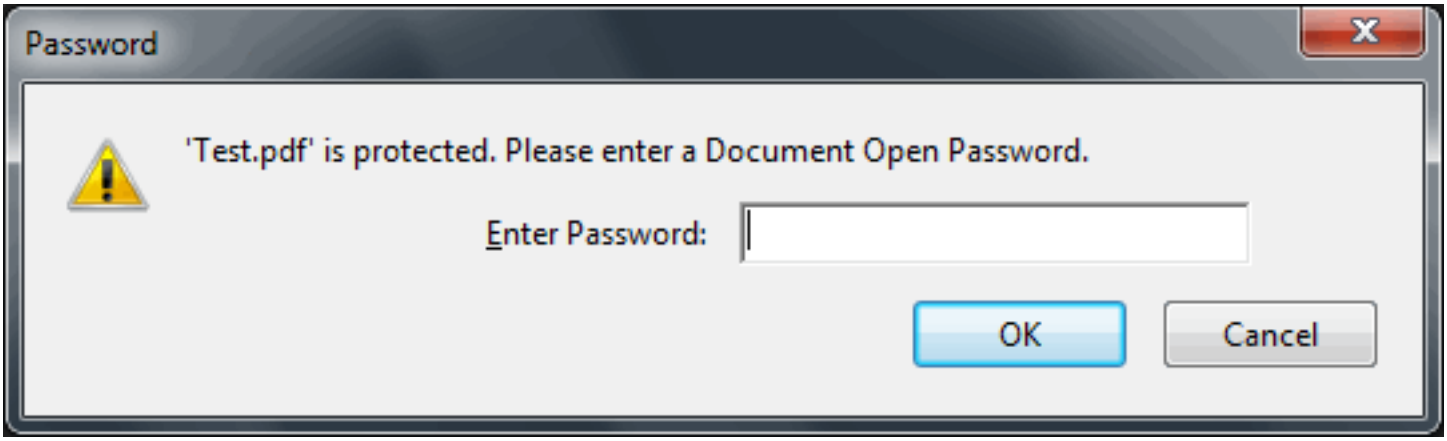
دو نوع رمزنگاری را می‌توان توسط iTextSharp به PDF تولیدی و یا موجود، اعمال کرد:
الف) رمزنگاری با استفاده از کلمه عبور
ب) رمزنگاری توسط کلید عمومی

الف) رمزنگاری با استفاده از کلمه عبور

در اینجا امکان تنظیم read password و edit password به کمک متد SetEncryption شیء pdfWrite وجود دارد. همچنین می‌توان مشخص کرد که مثلا آیا کاربر می‌تواند فایل PDF را چاپ کند یا خیر (PdfWriter.ALLOW_PRINTING).
ذکر read password اختیاری است؛ اما جهت اعمال permissions حتما نیاز است تا edit password ذکر گردد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;
using System.Text;

namespace EncryptPublicKey
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));

                var readPassword = Encoding.UTF8.GetBytes("123");//it can be null.
                var editPassword = Encoding.UTF8.GetBytes("456");
                int permissions = PdfWriter.ALLOW_PRINTING | PdfWriter.ALLOW_COPY;
                pdfWriter.SetEncryption(readPassword, editPassword, permissions,
PdfWriter.STRENGTH128BITS);

                pdfDoc.Open();

                pdfDoc.Add(new Phrase("tst 0"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("tst 1"));
            }

            Process.Start("TestEnc.pdf");
        }
    }
}
```

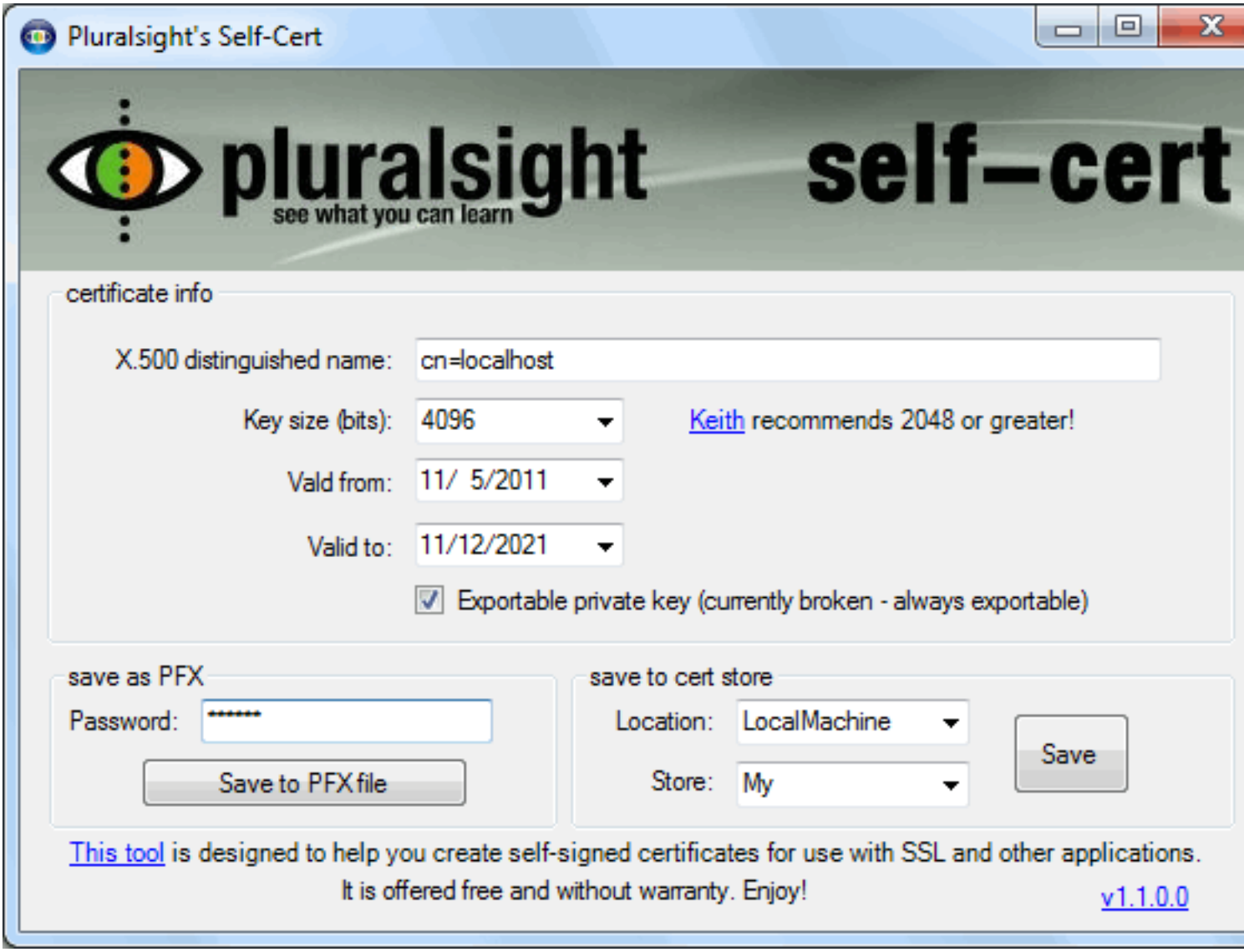
اگر read password ذکر شود، کاربران برای مشاهده محتویات فایل نیاز خواهند داشت تا کلمه‌ی عبور مرتبط را وارد نمایند:



این روش آنچنان امنیتی ندارد. هستند برنامه‌هایی که این نوع فایل‌ها را «آنی» به نمونه‌ی غیر رمزنگاری شده تبدیل می‌کنند (حتی نیازی هم ندارند که از شما کلمه‌ی عبوری را سؤال کنند). بنابراین اگر کاربران شما آنچنان حرفه‌ای نیستند، این روش خوب است؛ در غیراینصورت از آن صرفنظر کنید.

ب) رمزنگاری توسط کلید عمومی
این روش نسبت به حالت الف بسیار پیشرفته‌تر بوده و امنیت قابل توجهی هم دارد و «نیستند» برنامه‌هایی که بتوانند این فایل‌ها را بدون داشتن اطلاعات کافی، به سادگی رمزگشایی کنند.

برای شروع به کار با public key encryption نیاز است یک فایل PFX یا Personal Information Exchange داشته باشیم. یا می‌توان این نوع فایل‌ها را از CA's یا Certificate Authorities خرید، که بسیار هم نیکو یا اینکه می‌توان فعلا برای آزمایش، نمونه‌ی self signed این‌ها را هم تهیه کرد. مثلا با استفاده از این برنامه .



در ادامه نیاز خواهیم داشت تا اطلاعات این فایل PFX را جهت استفاده توسط iTextSharp استخراج کنیم. کلاس‌های زیر اینکار را انجام می‌دهند و نهایتا کلیدهای عمومی و خصوصی ذخیره شده در فایل PFX را بازگشت خواهند داد:

```
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.X509;

namespace EncryptPublicKey
{
    /// <summary>
    /// A Personal Information Exchange File Info
    /// </summary>
    public class PfxData
    {
        /// <summary>
        /// Represents an X509 certificate
        /// </summary>
        public X509Certificate[] X509PrivateKeys { set; get; }

        /// <summary>
        /// Certificate's public key
        /// </summary>
        public ICipherParameters PublicKey { set; get; }
    }
}
```

```
using System;
using System.IO;
using Org.BouncyCastle.Crypto;
using Org.BouncyCastle.Pkcs;
using Org.BouncyCastle.X509;

namespace EncryptPublicKey
{
    /// <summary>
    /// A Personal Information Exchange File Reader
    /// </summary>
    public class PfxReader
    {
        X509Certificate[] _chain;
        AsymmetricKeyParameter _asymmetricKeyParameter;

        /// <summary>
        /// Reads A Personal Information Exchange File.
        /// </summary>
        /// <param name="pfxPath">Certificate file's path</param>
        /// <param name="pfxPassword">Certificate file's password</param>
        public PfxData ReadCertificate(string pfxPath, string pfxPassword)
        {
            using (var stream = new FileStream(pfxPath, FileMode.Open, FileAccess.Read))
            {
                var pkcs12Store = new Pkcs12Store(stream, pfxPassword.ToCharArray());
                var alias = findThePublicKey(pkcs12Store);
                _asymmetricKeyParameter = pkcs12Store.GetKey(alias).Key;
                constructChain(pkcs12Store, alias);
                return new PfxData { X509PrivateKeys = _chain, PublicKey = _asymmetricKeyParameter };
            }
        }

        private void constructChain(Pkcs12Store pkcs12Store, string alias)
        {
            var certificateChains = pkcs12Store.GetCertificateChain(alias);
            _chain = new X509Certificate[certificateChains.Length];

            for (int k = 0; k < certificateChains.Length; ++k)
                _chain[k] = certificateChains[k].Certificate;
        }

        private static string findThePublicKey(Pkcs12Store pkcs12Store)
        {
            string alias = string.Empty;
            foreach (string entry in pkcs12Store.Aliases)
            {
                if (pkcs12Store.IsKeyEntry(entry) && pkcs12Store.GetKey(entry).Key.IsPrivate)
                {
                    alias = entry;
                    break;
                }
            }

            if (string.IsNullOrEmpty(alias))
                throw new NullReferenceException("Provided certificate is invalid.");

            return alias;
        }
    }
}
```

اکنون رمزنگاری فایل PDF تولیدی توسط کلید عمومی، به سادگی چند سطر کد زیر خواهد بود:

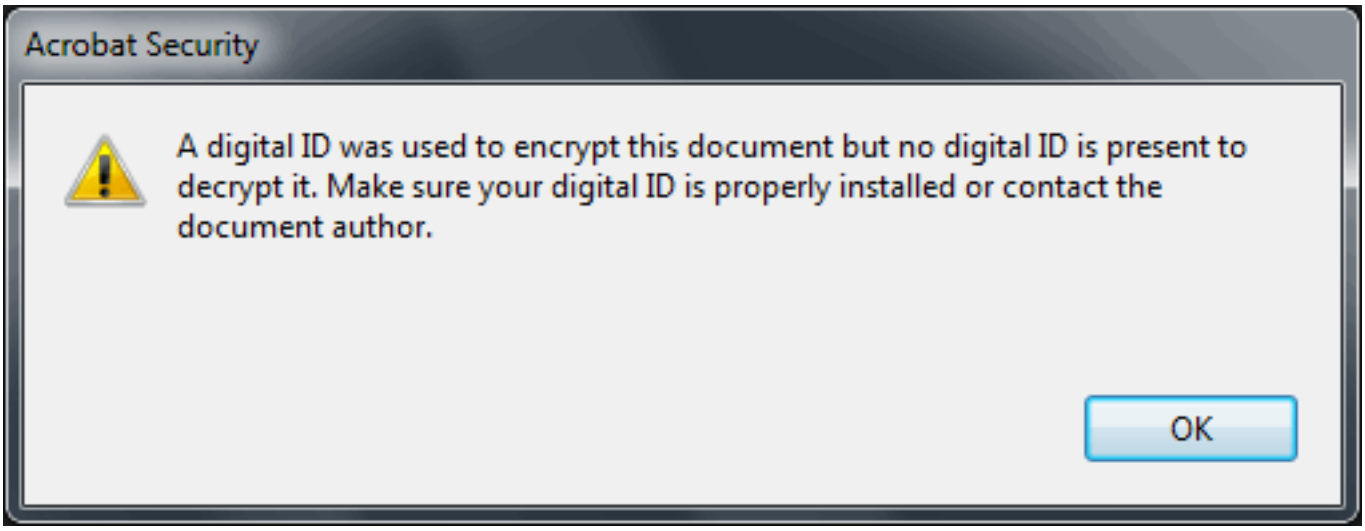
```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace EncryptPublicKey
```

```
{
    class Program
    {
        static void Main(string[] args)
        {
            using (var pdfDoc = new Document(PageSize.A4))
            {
                var pdfWriter = PdfWriter.GetInstance(pdfDoc, new FileStream("Test.pdf",
FileMode.Create));

                var certs = new PfxReader().ReadCertificate(@"D:\path\cert.pfx", "123");
                pdfWriter.SetEncryption(
                          certs: certs.X509PrivateKeys,
                          permissions: new int[] { PdfWriter.ALLOW_PRINTING, PdfWriter.ALLOW_COPY },
                          encryptionType: PdfWriter.ENCRYPTION_AES_128);

                pdfDoc.Open();

                pdfDoc.Add(new Phrase("tst 0"));
                pdfDoc.NewPage();
                pdfDoc.Add(new Phrase("tst 1"));
            }

            Process.Start("Test.pdf");
        }
    }
}
```

پیش از فراخوانی متد Open باید تنظیمات رمزنگاری مشخص شوند. در اینجا ابتدا فایل PFX خوانده شده و کلیدهای عمومی و خصوصی آن استخراج می‌شوند. سپس به متد SetEncryption جهت استفاده نهایی ارسال خواهند شد.

نحوه استفاده از این نوع فایل‌های رمزنگاری شده:

اگر سعی در گشودن این فایل رمزنگاری شده نمائیم با خطای زیر مواجه خواهیم شد:



کاربران برای اینکه بتوانند این فایل‌های PDF را بار کنند نیاز است تا فایل PFX شما را در سیستم خود نصب کنند. ویندوز فایل‌های PFX را می‌شناسد و نصب آن‌ها با دوبار کلیک بر روی فایل و چندبار کلیک بر روی دکمه‌ی Next و وارد کردن کلمه عبور آن، به پایان می‌رسد.

سؤال: آیا می‌توان فایل‌های PDF موجود را هم به همین روش رمزنگاری کرد؟

بله. iTextSharp علاوه بر PdfWriter دارای PdfReader نیز می‌باشد:

```
using System.Diagnostics;
using System.IO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace EncryptPublicKey
{
    class Program
    {
        static void Main(string[] args)
        {
            PdfReader reader = new PdfReader("TestDec.pdf");
            using (var stamper = new PdfStamper(reader, new FileStream("TestEnc.pdf",
FileMode.Create)))
            {
                var certs = new PfxReader().ReadCertificate(@"D:\path\cert.pfx", "123");
                stamper.SetEncryption(
                         certs: certs.X509PrivateKeys,
                         permissions: new int[] { PdfWriter.ALLOW_PRINTING, PdfWriter.ALLOW_COPY },
                         encryptionType: PdfWriter.ENCRYPTION_AES_128);
                stamper.Close();
            }

            Process.Start("TestEnc.pdf");
        }
    }
}
```

سؤال: آیا می‌توان نصب کلید عمومی را خودکار کرد؟

سورس برنامه SelfCert که معرفی شد، در دسترس است. این برنامه قابلیت انجام نصب خودکار مجوزها را دارد.

نظرات خوانندگان

نویسنده: مجتبی
تاریخ: ۱۳۹۰/۰۸/۲۱ ۱۵:۳۵:۲۹

خیلی عالی بود . من همیشه برام سؤال بود که این کدها چگونه ساخته میشدند . در ضمن من این کدهای pfx رو در فایلهای کتابهای " اداره کل چاپ و کتب درسی " دیده بودم . فایلهای pdf اونها برای خواندن احتیاج به نصب pfx داشتند.

نویسنده: Mojtaba Shagi
تاریخ: ۱۳۹۰/۰۸/۲۲ ۰۸:۳۴:۵۶

خیلی ممنون از مطالب خوب و مفید شما

نویسنده: K Liberal
تاریخ: ۱۳۹۰/۱۰/۰۸ ۱۴:۲۹:۰۵

سلام.دستت بابت این اطلاعات درد نکنه.
راستش من یه سری کتاب درسی دانلود کردم میخوام ادیتش کنم.چون از برنامه نویسی چیز زیادی سرم نمیشه از توضیحات بالا زیاد چیزی نفهمیدم حتی نمیدونم تو چه محیطی این دستوراتو باید نوشت.ولی واقعا به اون فایل های پی دی اف نیاز دارم اگه میشه 1 توضیحی بدین که باید چیکار کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۱۰/۰۸ ۱۷:۰۵:۳۸

ادیت کردن نیاز به رمزگشایی دارد؛ مطلب ارسالی من در اینجا «رمزنگاری» است.

برای رمزگشایی برنامه پر است در اینترنت مثلا: (^)

نویسنده: سینا
تاریخ: ۱۳۹۲/۱۱/۰۲ ۸:۳۷

با تشکر - ایا فایلی که با استفاده از PFX رمز شده در تبلت و مک بوک غیر ویندوز باز میشه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۲ ۹:۵

PFX منحصر به ویندوز نیست. اگر برنامه‌ها قادر به پردازش رمزنگاری‌های از نوع کلیدهای عمومی باشند، قادر به خواندن آن نیز خواهند بود. عموما برنامه‌های ساده تا این حد انواع و اقسام استانداردها را رعایت نمی‌کنند. ولی در کل امکان « حذف محدودیت‌های فایل‌های PDF توسط iTextSharp » وجود دارد تا فایل رمزنگاری شده‌ی توسط کلیدهای عمومی را بتوان تبدیل به فایل‌های غیر رمزنگاری شده کرد. پس از رفع محدودیت، برنامه‌های ضعیف PDF خوان هم قادر به گشودن آن‌ها خواهند بود.