Реализация алгоритмов управления электронной подписью

С.И. Носков, А.П. Медведев

Иркутский государственный университет путей сообщения

Аннотация: Использование электронной подписи в последнее время приобрело самые широкие масштабы и стало неотъемлемой частью большинства бизнес-процессов. Предлагаемый поставщиком средств криптографии инструментарий управления электронной подписью не всегда способен удовлетворить все запросы организаций. В данной работе рассмотрен подход, направленный на решение большинства задач управления электронной подписью. Суть метода состоит в комбинированном использовании как библиотек разработчика средств криптографии, так и возможностей узкоспециализированных библиотек для работы с криптографией и документами.
Ключевые слова: программное обеспечение, управление электронной подписью, штамп, визуализация электронной подписи, защита информации.

Использование электронной подписи стало неотъемлемой частью многих процессов, протекающих в рамках цифровой трансформации экономики. Благодаря тому, что законодательно закреплена юридическая значимость подписанных квалифицированной электронной подписью документов, использование возможностей электронной подписи приобрело самые различные формы, такие, как, например её использование в автоматизированных системах, а число выданных сертификатов ключа проверки электронных подписей неуклонно растет. Так, авторы статьи [1] наглядно показывают всю значимость применения электронных подписей в цифровой экономике. Не стало преградой для этого процесса и резкое сокращение числа удостоверяющих центров в связи с изменениями в федеральном законе от 06.04.2011 № 63-ФЗ, вступившими в силу 1 июля 2021 года.

При рассмотрении вопросов, связанных с использованием криптографических алгоритмов зачастую опираются на два фактора, а именно, на их стойкость и скорость. Так, автор работы [2] показывает подобный математический анализ стойкости алгоритма ГОСТ 28147-89, авторы статьи [3] приводят его подробный анализ на предмет слабых блоков,

а в работе [4] наглядно показан анализ скорости работы блочного алгоритма шифрования AES.

Существует множество прикладных задач, в которых гарантия подлинности и целостности документов имеет решающее значение. В этих случаях типичное решение полагается на использовании цифровой подписи, которая, в свою очередь, основана на использовании инфраструктуры открытых ключей (PKI). Вместе с тем, тот функционал, который предлагает производитель средств криптографической защиты информации далеко не всегда может удовлетворить всем требованиям заказчика. Так, авторы работы [5] рассматривают возможность вовсе отойти от PKI, предлагая к рассмотрению новый тип электронной подписи, основанный на использовании социальных сетей, а в статье [6] был показан подход к защите программного обеспечения от незаконного копирования на основе использования алгоритма с двумя ключами.

Проблеме применимости электронной подписи в разрезе функционала посвящено достаточное количество работ, а само применение электронной подписи давно вышло за рамки простого подписания файлов «оффлайн». Так, например, авторы работы [7] рассматривают эту задачу в контексте вопросов безопасности, статья [8] подчеркивает важность функционала использования электронных подписей в сфере электронной коммерции, а автор статьи [9] еще более 40 лет назад придавал значимость вопросам использования электронной подписи в электронной почте.

К сожалению, тот инструментарий управления электронной подписью, который предлагает поставщик средств криптографической защиты информации (в частности, разработчик систем криптопровайдера) далеко не всегда способен удовлетворить всем запросам организаций. Среди типовых подобных задач: пакетное подписание файлов, постановка визуального штампа наличия электронной подписи и подписание pdf-документа

«внутри». Определенную сложность здесь представляет также и ГОСТ-алгоритм, не позволяющий использовать общие подходы к управлению электронными подписями ввиду несовместимости с большинством библиотек и решениями на базе open-source.

Однако, выход из сложившейся ситуации существует и заключается он в комбинировании возможностей криптопровайдера и узкоспециализированных библиотек.

В качестве примера реализации таких возможностей был выбран язык программирования C#. Все примеры приведены для криптопровайдера КриптоПро CSP и файлов формата *.pdf. В качестве библиотек использованы:

- КриптоПро Sharpei – библиотека, позволяющая использовать средство криптографической защиты информации КриптоПро CSP на платформе Microsoft .Net Framework;
- Bouncy Castle – библиотека open-source, представляющая большие возможности управления сертификатом;
- iTextSharp – библиотека, позволяющая оперировать документами *.pdf.

Рассмотрим один из вариантов использования подобного подхода – подписание документа pdf «внутри». В данном случае подпись помещается непосредственно внутрь файла *.pdf и будет видна пользователю только при открытии файла. Сам процесс подписания состоит из нескольких этапов.

Этап подготовки сертификата

Находим секретный ключ по серийному номеру сертификата в хранилище Личные:

```
X509Store store = new X509Store("My", StoreLocation.CurrentUser);
store.Open(OpenFlags.OpenExistingOnly | OpenFlags.ReadOnly);
```

```
X509Certificate2Collection found = store.Certificates.Find(X509FindType.FindBySerialNumber, certificate_dn, true);
X509Certificate2 certificate = found[0];
```

Создаем экземпляр класса Gost3410CryptoServiceProvider из библиотеки КриптоПро Sharpei:

```
Gost3410CryptoServiceProvider cert_key = certificate.PrivateKey as Gost3410CryptoServiceProvider;
```

Создаем новый объект и копируем параметры из исходного контекста сертификата:

```
var cspParameters = new CspParameters();
cspParameters.KeyContainerName=cert_key.CspKeyContainerInfo.KeyContainerName;
cspParameters.ProviderType = cert_key.CspKeyContainerInfo.ProviderType;
cspParameters.ProviderName = cert_key.CspKeyContainerInfo.ProviderName;
cspParameters.Flags = cert_key.CspKeyContainerInfo.MachineKeyStore?(CspProviderFlags.UseExistingKey | CspProviderFlags.UseMachineKeyStore):(CspProviderFlags.UseExistingKey);
cspParameters.KeyPassword = new SecureString();
foreach (var c in "1")
  {
     cspParameters.KeyPassword.AppendChar(c);
  }
```

Далее воспользуемся классом X509Certificate2, который содержит все необходимые методы управления сертификатом:

```
certificate = new X509Certificate2(certificate.RawData);
certificate.PrivateKey = new Gost3410CryptoServiceProvider(cspParameters);
```

*Открываем документ *.pdf:*

```
PdfReader reader = new PdfReader(path + "//" + item);
```

Далее воспользуемся классом PdfStamper библиотеки iTextSharp, позволяющий добавить в документ любой дополнительный контент:

```
PdfStamper st = PdfStamper.CreateSignature(reader, new FileStream(Directory.GetCurrentDirectory() + "//Signed//" + item + "_signed.pdf", FileMode.Create, FileAccess.Write), '\0');
```

Класс PdfSignatureAppearance библиотеки iTextSharp предлагает удобные методы для работы с подписью:

```
PdfSignatureAppearance sap = st.SignatureAppearance;
```

Создаем объект класса X509CertificateParser для извлечения полей и атрибутов сертификата:

```
X509CertificateParser parser = new X509CertificateParser();
Org.BouncyCastle.X509.X509Certificate[] chain = new Org.BouncyCastle.X509.X509Certificate[]
parser.ReadCertificate(certificate.RawData)};
sap.Certificate = parser.ReadCertificate(certificate.RawData);
sap.Reason = "";
sap.Location = "";
sap.Acro6Layers = true;
sap.SignDate = DateTime.Now;
```

Выбираем подходящий тип фильтра:

```
PdfName filterName = new PdfName("CryptoPro PDF");
```

Создаем подпись:

```
PdfSignature signature = new PdfSignature(filterName, PdfName.ADBE_PKCS7_DETACHED);
```

Заполняем данными соответствующие поля:

```
signature.Date = new PdfDate(sap.SignDate);
signature.Name = certificate_dn;
signature.Reason = sap.Reason ?? "No reason";
```

```
signature.Location = sap.Location ?? "No location";
sap.CryptoDictionary = signature;
int intCSize = 6000;
```

Создаем таблицу с хэшами:

```
Dictionary<PdfName, int> hashtable = new Dictionary<PdfName, int>();
hashtable.Add(PdfName.CONTENTS, intCSize * 2 + 2);
sap.PreClose(hashtable);
Stream s = sap.GetRangeStream();
MemoryStream ss = new MemoryStream();
int read = 0;
byte[] buff = new byte[8192];
while ((read = s.Read(buff, 0, 8192)) > 0)
{ss.Write(buff, 0, read);}
```

Вычисляем подпись:

```
System.Security.Cryptography.Pkcs.ContentInfo contentInfo = new
System.Security.Cryptography.Pkcs.ContentInfo(ss.ToArray());
SignedCms signedCms = new SignedCms(contentInfo, true);
CmsSigner cmsSigner = new CmsSigner(certificate);
signedCms.ComputeSignature(cmsSigner, false);
byte[] pk = signedCms.Encode();
```

Этап внедрения подписи в документ

Помещаем подпись в документ, используя класс PdfDictionary библиотеки iTextSharp:

```
byte[] outc = new byte[intCSize];
PdfDictionary dic2 = new PdfDictionary();
Array.Copy(pk, 0, outc, 0, pk.Length);
dic2.Put(PdfName.CONTENTS, new PdfString(outc).SetHexWriting(true));
sap.Close(dic2);
```

Алгоритм проверки электронной подписи

Подготовительный этап:

```
byte[] dataFileRawBytes = System.IO.File.ReadAllBytes(dialog.FileName);
using (MemoryStream fileStream = new MemoryStream(dataFileRawBytes))
using (PdfReader pdfReader = new PdfReader(fileStream))
```

Для работы с полями документа pdf используем класс AcroFields библиотеки iTextSharp:

```
AcroFields acroFields = pdfReader.AcroFields;
```

Получаем названия контейнеров, содержащих подписи:

```
List<string> signatureNames = acroFields.GetSignatureNames();
```

Если контейнеры отсутствуют, следовательно, и сам файл не подписан:

```
if (signatureNames.Count==0)
  {
  MessageBox.Show("Файл не подписан!","Подпись отсутствует",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
  }
foreach (string signatureName in signatureNames)
  {
```

Далее выделяем подпись из контейнера:

```
PdfDictionary singleSignature =
acroFields.GetSignatureDictionary(signatureName);
PdfString asString1 = singleSignature.GetAsString(PdfName.CONTENTS);
byte[] signatureBytes = asString1.GetOriginalBytes();
RandomAccessFileOrArray safeFile = pdfReader.SafeFile;
PdfArray asArray = singleSignature.GetAsArray(PdfName.BYTERANGE);
using (Stream stream = new iTextSharp.text.io.RASInputStream(
new iTextSharp.text.io.RandomAccessSourceFactory().CreateRanged(
safeFile.CreateSourceView(), asArray.AsLongArray())))
```

```
{using (MemoryStream ms = new MemoryStream((int)stream.Length))
{CopyStream(stream, ms);
byte[] data = ms.GetBuffer();
ContentInfo contentInfo = new ContentInfo(data);
```

Для проверки электронной подписи отлично подойдет класс SignedCms:

```
SignedCms signedCms = new SignedCms(contentInfo, true);
signedCms.Decode(signatureBytes);
bool checkResult;
```

После получения подписи проверяем её:

```
try{
   signedCms.CheckSignature(true);
   checkResult = true;
   }
catch (Exception)
  {checkResult = false;}
```

Алгоритм постановки штампа

Подготовительный этап:

```
using (Stream inputPdfStream = new FileStream(path + "//" + item, FileMode.Open, FileAccess.Read, FileShare.Read))
using (Stream inputImageStream = new FileStream(filePath, FileMode.Open, FileAccess.Read, FileShare.Read))
using (Stream outputPdfStream = new FileStream(Directory.GetCurrentDirectory() + "//Stamped//" + item, FileMode.Create, FileAccess.Write, FileShare.None))
var reader4stamp = new iTextSharp.text.pdf.PdfReader(inputPdfStream);
```

Определяем количество страниц:

```
int pdfpages = reader4stamp.NumberOfPages;
stranica = "1-" + pdfpages.ToString();
```

```
iTextSharp.text.Image image = iTextSharp.text.Image.GetInstance(inputImageStream);
var stamper = new iTextSharp.text.pdf.PdfStamper(reader4stamp, outputPdfStream);
```

Непосредственное добавление штампа на страницы:

```
for (int i = 1; i < reader4stamp.NumberOfPages + 1; i++)
   { var pdfContentByte = stamper.GetOverContent(i);
   image.Alignment = iTextSharp.text.Element.HEADER;
   pdfContentByte.AddImage(image);}
```

Приведенные алгоритмы могут быть использованы как при разработке отдельного самостоятельного продукта, так и входить в состав компонентов уже существующих автоматизированных систем. На основе указанных подходов и алгоритмов был разработан программный продукт, позволяющий решать большинство вышеуказанных задач [10].

Таким образом, в работе подробно рассмотрены программные методы управления и визуализации электронной подписи, работы с контейнерами и полями сертификата на примере встраиваемой подписи в документы формата *.pdf. Существуют и другие задачи в рамках указанной темы, такие, как, например, управление шифрованием и обновление списков аннулированных сертификатов. Именно этим вопросам будут посвящены будущие работы.

Литература

1. Ruzic F. Electronic Signature: The Core Legislation Category in Digital Economy // Digital Economy: Impacts, Influences and Challenges, 2005, pp. 98-135.

2. Маро Е.А. Алгебраический анализ стойкости криптографических систем защиты информации // Инженерный вестник Дона, 2013, №4. URL: ivdon.ru/ru/magazine/archive/n4y2013/1996.

3. Бабенко Л.К., Ищукова Е.А. Анализ алгоритма ГОСТ 28147-89: поиск слабых блоков // Известия ЮФУ. Технические науки, 2014, С. 129-138.

4. Стариков А.А., Лысенко А.В., Клевцов А.А. Разработка и анализ скорости работы блочного симметричного алгоритма шифрования AES с использованием различных языков программирования // Молодой исследователь Дона, 2022, №4, С. 38-41.

5. Nicolazzo S. A new approach for electronic signature // Proceedings of the 2nd International Conference on Information Systems Security and Privacy ICISSP, 2016, Vol.1, pp. 440-447.

6. Медведев А.П. Алгоритм защиты программного обеспечения от незаконного копирования и воспроизведения // Инженерный вестник Дона, 2024, №5. URL: ivdon.ru/ru/magazine/archive/n5y2024/9185.

7. Egerszegi K., Erdősi P. Problems in the implementation of the electronic signature // Periodica politechnica social and management sciences, 2003, Vol.1, No. 1, pp. 67-82.

8. Velentzas J., GKiriakoulis G., Broni G., Kartalis N., Panou G., Fragulis G. Digital and advanced electronic signature: the security function, especially in electronic commerce // The 4th ETLTC International Conference on ICT Integration in Technical Education (ETLTC2022), 2022, Vol. 139, pp. 1-4.

9. Davies D.W. Applying the RSA Digital Signature to Electronic Mail // Computer, 1983, Vol. 16, pp. 55-62.

10. Медведев А.П. Свидетельство о регистрации №2024660037. Программное обеспечение для выполнения функций по созданию и проверке электронной подписи, шифрованию и расшифрованию файлов, управления сертификатами с использованием СКЗИ «КриптоПро CSP» // 2024. URL: fips.ru/EGD/f6863a38-9793-4263-9b66-3f8ff5f77fb7.

References

1. Ruzic F. Digital Economy: Impacts, Influences and Challenges, 2005, pp. 98-135.

2. Maro E.A. Inzhenernyj vestnik Dona, 2013, №4. URL: ivdon.ru/ru/magazine/archive/n4y2013/1996.

3. Babenko L.K., Ishhukova E.A. Izvestija JuFU. Tehnicheskie nauki, 2014, pp. 129-138.

4. Starikov A.A., Lysenko A.V., Klevcov A.A. Molodoj issledovatel' Dona, 2022, №4, pp. 38-41.

5. Nicolazzo S. Proceedings of the 2nd International Conference on Information Systems Security and Privacy ICISSP, 2016, Vol.1, pp. 440-447.

6. Medvedev A.P. Inzhenernyj vestnik Dona, 2024, №5. URL: ivdon.ru/ru/magazine/archive/n5y2024/9185.

7. Egerszegi K., Erdősi P. Periodica politechnica social and management sciences, 2003, Vol.1, No. 1, pp. 67-82.

8. Velentzas J., GKiriakoulis G., Broni G., Kartalis N., Panou G., Fragulis G. The 4th ETLTC International Conference on ICT Integration in Technical Education (ETLTC2022), 2022, Vol. 139, pp. 1-4.

9. Davies D.W. Computer, 1983, Vol. 16, pp. 55-62.

10. Medvedev A.P. Svidetel'stvo №2024660037 [Certificate No. 2024660037]. Programmnoe obespechenie dlja vypolnenija funkcij po sozdaniju i proverke jelektronnoj podpisi, shifrovaniju i rasshifrovaniju fajlov, upravlenija sertifikatami s ispol'zovaniem SKZI «KriptoPro CSP» [Software for performing functions of electronic signature creation and verification, file encryption and decryption, certificate management using CryptoPro CSP encryption system], 2024. URL: https://fips.ru/EGD/f6863a38-9793-4263-9b66-3f8ff5f77fb7.

Дата поступления: 25.09.2024

Дата публикации: 2.11.2024