Application of digital signature for verification of the examination results

Lucie Pivnickova, Viliam Dolinay, Vladimir Vasek, Roman Jasek

***Abstract*—** This paper presents the application of the digital signature into the medical software applications, which work with the private data of the examination patients. Digital signature is the law recognized alternative to physical signature, intended for use in an electronic environment. At present, in most states which legalized a digital signature is used the signature in conjunction with the standard X. 509, which defines the format of certificates, organization and conduct of certification authorities. Certification Authority provides the trusted connection of people and public key to use for digital signature. At first the paper will explain the basic concepts associated with the digital signature. Then prepared tool to create digitally signed document will be introduced. Finally, the use of this tool will be demonstrated on the example and pointed possibility of its use in practical situations.

***Keywords*—** evaluation of patients, digital signature, standard X. 509, iText library, certificates, public and private key

I. Introduction

A digital signature or digital signature scheme is a mathematical scheme for demonstrating the authenticity of a digital message or document. A valid digital signature gives a recipient reason to believe that the message was created by a known sender such that they cannot deny sending it (authentication and non-repudiation) and that the message was not altered in transit (integrity). Digital signatures are commonly used for software distribution, financial transactions, and in other cases where it is important to detect forgery or tampering. [13]

This study deals with the application of the digital signature to medical software applications, which work with the private data of the examination patients. In current practice, it is necessity to keep result of patient examination in the approved state. Such state may be achieved by printing all important result on the paper and liable person confirm this document with his/her hand signature. This method is easy to perform, however it is now outdated and in many ways surpassed. Archiving of such records requires a lot of space and the possible search is laborious and slow. Also, environmental considerations should be taken into account.

All these disadvantages lead to efforts to promote the use of electronic signatures in medical practice.

The main idea of presented solution is that a large part of the measurement results is obtained in electronic form, such as photographs, X-ray images, audiometry and tympanometry curves, etc. and these data are then printed on paper. However it may as well be printed to electronic output, e.g. PDF document. Such outputs can be then accompanied by an electronic signature and stored in any database or at least in a suitable directory structure on computer hard disk.

Electronic signature system is not easy to add to software, or expect that all software used in medical offices will soon implement this feature or will be replaced by one with this functionality. This would be naive assumptions and therefore was more appropriate to prepare the tools that will help integrate electronic signature feature into existing systems.

There will be explained the prepared application that use open source iText library to create digitally signed document and the possibility of its use in practical situations.

II. Digital signatures

Digital signatures are another use of asymmetric encryption algorithms and their associated public and private keys. Digital signatures allow to sign a message in order to enable detection of changes to the message contents, to ensure that the message was legitimately sent by the expected party, and to prevent the sender from denying that he or she sent the message, known as nonrepudiation. To digitally sign a message, the sender would generate a hash of the message, and then use his private key to encrypt the hash, thus generating a digital signature. The sender would then send the digital signature along with the message, usually by appending it to the message itself.

When the message arrives at the receiving end, the receiver would use the sender's public key to decrypt the digital signature, thus restoring the original hash of the message. The receiver can then verify the integrity of the message by hashing the message again and comparing the two hashes. Although this may sound like a considerable amount of work to verify the integrity of the message, it is often done by a software application of some kind and the process typically is largely invisible to the end user. [2, 6]

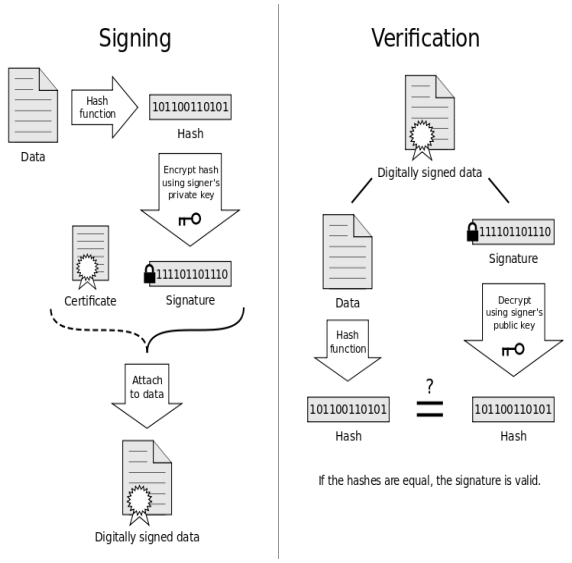

Fig. 1 Digital Signature Scheme [9]

For the purposes of the present act: [4]

a) electronic signature shall mean data in electronic form which are attached to or logically associated with a data message and which serve as a method of unequivocal authentication of a signatory in relation to a data message

b) advanced digital signature shall mean an digital signature which meets the following requirements:
   1. it is uniquely linked to the signatory
   2. it is capable of identifying the signatory in relation to a data message
   3. it has been created and attached to a data message using means that the signatory can maintain under his sole control
   4. it is linked to the data message to which it relates in such a manner that any subsequent change of the data is detectable

c) data message shall mean electronic data that can be transmitted by means of electronic communication and stored on storage media used to process and transmit data in electronic form

d) signatory shall mean a natural person who holds a signature creation device and acts either on his own behalf or on behalf of another natural or legal person

The condition that the signature was seen as warranted is the ability of keeping the signatory to create a digital signature under its exclusive control. By these means we understand software or equipment that is used to create a digital signature.

A digital signature should fulfill certain functions. There are:

- Identification
- Authentication
- Integrity
- Non-repudiation

A. *Identification*

Identification is simply an assertion of who we are. This may include who we claim to be as a person, who a system claims to be over the network, who the originating party of an e-mail claims to be, or similar transactions. It is important to note that the process of identification does not extend beyond this claim and does not involve any sort of verification or validation of the identity that we claim. That part of the process is referred to as authentication and is a separate transaction. [2, 7]

B. *Authentication*

Authentication is, in an information security sense, the set of methods we use to establish a claim of identity as being true. It is important to note that authentication only establishes whether the claim of identity that has been made is correct. Authentication does not infer or imply anything about what the party being authenticated is allowed to do; this is a separate task known as authorization. The important thing to understand for now is that authentication needs to take place first. [2, 7]

C. *Integrity*

Integrity refers to the ability to prevent our data from being changed in an un-authorized or undesirable manner. This could mean the unauthorized change or deletion of our data or portions of our data, or it could mean an authorized, but undesirable, change or deletion of our data. To maintain integrity, we not only need to have the means to prevent unauthorized changes to our data but also need the ability to reverse authorized changes that need to be undone. [2, 7]

D. *Non-repudiation*

Nonrepudiation refers to a situation in which sufficient evidence exists as to prevent an individual from successfully denying that he or she has made a statement, or taken an action. In information security settings, this can be accomplished in a variety of ways. We may be able to produce proof of the activity directly from system or network logs, or recover such proof through the use of digital forensic examination of the system or devices involved. We may also be able to establish nonrepudiation through the use of encryption technologies, more specifically through the use of hash functions that can be used to digitally sign a communication or a file. An example of this might be a system that digitally signs every e-mail that is sent from it, thus rendering useless any denial that might take place regarding the sending of the message in question. [2, 7]

III. CERTIFICATES

In addition to hashes and digital signatures, we have another construct by which we can scale up the use of message signing, in the form of digital certificates, commonly known as certificates. Certificates are created to link a public key to a

particular individual and are often used as a form of electronic identification for that particular person. A certificate is typically formed by taking the public key and identifying information, such as a name and address, and having them signed by a certificate authority (CA). A CA is a trusted entity that handles digital certificates.

The advantage of having a certificate is that it allows us to verify that a public key is truly associated with a particular individual. In the case of the digital signature we discussed in the preceding section, it might be possible that someone had falsified the keys being used to sign the message and that the keys did not actually belong to the original sender. If we have a digital certificate for the sender, we can easily check with the CA to ensure that the public key for the sender is legitimate.

A CA is only a small part of the infrastructure that can be put in place to handle certificates on a large scale. This infrastructure is known as a public key infrastructure (PKI). A PKI is generally composed of two main components, although some organizations may separate some functions out into more than just these. In a PKI, we often find the CAs that issue and verify certificates and the registration authorities (RAs) that verify the identity of the individual associated with the certificate.

In PKI, we also deal with the concept of certificate revocation, in the case where a certificate reaches its expiration date, the certificate is compromised, or another reason arises in which we need to ensure that the certificate can no longer be used. In this case, we will likely see the certificate added to a certificate revocation list (CRL). The CRL is a generally public list that holds all the revoked certificates for a certain period of time, depending on the organization in question. [2]

A. List of certification service providers issuing qualified certificates, qualified system certificates and qualified time stamps

The Ministry of Interior publish a review a certification service providers and their qualified services by the law pursuant to Section 9 (2) letter (e) of the Act No. 227/2000 Coll. [4]

Ser. No.	Certification Service Providers	Qualified Services
1.	**První certifikační autorita, a. s.**, Identification Number 26 43 93 95, Podvinný mlýn 2178/6, PSČ 190 00 Prague 9	Issue qualified certificates. Issue qualified system certificates. Issue qualified time stamps.
2.	**Česká pošta, s. p.**, Identification Number 47 11 49 83, Olšanská 38/9, PSČ 225 99 Prague 3	Issue qualified certificates. Issue qualified system certificates.
3.	**eIdentity a. s.**, Identification Number 27 11 24 89, Vinohradská 184/2396, PSČ 130 00 Prague 3	Issue qualified certificates. Issue qualified system certificates.

Fig. 2 Certification Service Providers [4]

IV. SYMMETRIC VERSUS ASYMMETRIC CRYPTOGRAPHY

When we look at the use of symmetric key cryptography versus asymmetric key cryptography, we do not have a situation in which one is necessarily better overall than the other for all situations. Instead, each has a set of strengths and weaknesses when used in a given situation. In many cases, symmetric key cryptography is much faster than asymmetric, but symmetric cryptography brings with it the issue of key exchange and so on. We will discuss each type of algorithm and a few specific examples of each type in this section.

A. Symmetric Cryptography

Symmetric key cryptography, also known as private key cryptography, utilizes a single key for both encryption of the plaintext and decryption of the ciphertext. The key itself must be shared between the sender and the receiver, and this process, known as key exchange, constitutes an entire subtopic of cryptography. We will discuss key exchange at greater length later in this chapter. The symmetric in symmetric key cryptography is a reference to the use of a single key. One of the chief weaknesses of symmetric key cryptography lies in the use of one key. If the key is exposed beyond the sender and the receiver, it is possible for an attacker who has managed to intercept it to decrypt the message or, worse yet, to decrypt the message, alter it, then encrypt it once more and pass it on to the receiver in place of the original message. Since such issues are present, symmetric key cryptography provides only confidentiality, and not integrity, as we would not be aware that the message in our example had been altered. [1, 2, 6]

B. Asymmetric Cryptography

Although symmetric key cryptography makes use of only one key, asymmetric key cryptography, also known as public key cryptography, utilizes two keys: a public key and a private key. The public key is used to encrypt data sent from the sender to the receiver and is shared with everyone. We see public keys included in e-mail signatures, posted on servers that exist specifically to host public keys, posted on Web pages, and displayed in a number of other ways. Private keys are used to decrypt data that arrives at the receiving end and are very carefully guarded by the receiver. Complex mathematical operations are used to create the private and public keys. These operations are, at present, difficult enough that the means do not exist to reverse the private key from the public key. [1, 2, 6]

RSA

The RSA algorithm is used for both public key encryption and digital signatures. It is the most widely used public key encryption algorithm. The basis of the security of the RSA algorithm is that it is mathematically infeasible to factor sufficiently large integers. The RSA algorithm is believed to be secure if its keys have a length of at least 1024-bits. [12]

Key Generation Algorithm: [12]

1. Choose two very large random prime integers:

p and q

2. Compute n and φ(n):
   n = pq and φ(n) = (p-1)(q-1)

3. Choose an integer e, 1 < e < φ(n) such that:
   gcd(e, φ(n)) = 1(where gcd means greatest common denominator)

4. Compute d, 1 < d < φ(n) such that:
   ed ≡ 1 (mod φ(n))

Where:

- the public key is (n, e) and the private key is (n, d)
- the values of p, q and φ(n) are private
- e is the public or encryption exponent
- d is the private or decryption exponent

Encryption:
The ciphertext C is found by the equation 'C = Me mod n' where M is the original message.

Decryption:
The message M can be found form the ciphertext C by the equation 'M = Cd mod n'.

A simple example:
This is an extremely simple example and would not be secure using primes so small, normally the primes p and q would be much larger.

1. Select the prime integers q=11, q=3.
2. n=pq=33; φ(n)=(p-1)(q-1)=20
3. Choose e=3
   - Check gcd(3,20)=1
4. Compute d=7
   - (3)d ≡ 1 (mod 20)

Therefore the public key is (n, e) = (33, 3) and the private key is (n, d) = (33, 7).

Now say we wanted to encrypt the message M=7

- C = Me mod n
- C = 73 mod 33
- C = 343 mod 33
- C = 13

So now the ciphertext C has been found. The decryption of C is performed as follows.

- M' = Cd mod n
- M' = 137 mod 33
- M' = 62,748,517 mod 33
- M' = 7

As you can see after the message has been encrypted and decrypted the final message M' is the same as the original message M. A more practical way to use the algorithm is to convert the message to hexadecimal and perform the encryption and decryption steps on each octet individually.

Digital signing:
In order to sign a message the sender does the following: [12]

1. Produces a hash value of the message
2. Uses his/her private key (n, d) to compute the signature
3. S = Md mod n
4. Sends the signature S to the recipient

Signature verification:
The recipient does the following in order to verify the message: [7]

1. Uses the senders public key (n, e) to compute the hash value
2. V = Se mod n
3. Extracts the hash value from the message
4. If both hash values are identical then the signature is valid

C. Hash Function

Hash functions represent a third cryptography type alongside symmetric and asymmetric cryptography, what we might call keyless cryptography. Hash functions, also referred to as message digests, do not use a key, but instead create a largely unique and fixed-length hash value, commonly referred to as a hash, based on the original message, something along the same lines as a fingerprint. Hashes cannot be used to discover the contents of the original message, or any of its other characteristics, but can be used to determine whether the message has changed. In this way, hashes provide confidentiality, but not integrity. Hashes are very useful when distributing files or sending communications, as the hash can be sent with the message so that the receiver can verify its integrity. The receiver simply hashes the message again using the same algorithm, then compares the two hashes. If the hashes match, the message has not changed. If they do not match, the message has been altered. Although it is theoretically possible to engineer a matching hash for two different sets of data, called a collision, this is a very difficult task indeed, and generally requires that the hashing algorithm be broken in order to accomplish. Some algorithms, such as Message-Digest algorithm 5 (MD5), have been attacked in this fashion, although producing a collision is still nontrivial. When such cases occur, the compromised algorithm usually falls out of common use. Hashing algorithms such as SHA-2 and the soon-to-arrive SHA-3 have replaced MD5 in cases where stringent hash security is required. Many other hash algorithms exist and are used in a variety of situations, such as MD2, MD4, and RACE. [2]

V. TRUST

Trust is a key element in an electronic communication. Verify the identity of the other party is utilized entity that has the resources. There are several ways to achieve this.
PGP is the first system that uses an electronic signature on the level of communication between individuals via e-mail. It is

not accepted by the authorities because it lacks a central authority, which could be monitored.
The second system is the standard X.509 certificates, which is based on the hierarchical structure of CAs. This system is used in the Czech Republic for the certification of electronic signatures.tt

A. PGP

Pretty Good Privacy (PGP) is a data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication. PGP is often used for signing, encrypting and decrypting texts, e-mails, files, directories and whole disk partitions to increase the security of e-mail communications. It was created by Phil Zimmermann in 1991.

PGP uses a variation of the public key system. In this system, each user has a publicly known encryption key and a private key known only to that user. You encrypt a message you send to someone else using their public key. When they receive it, they decrypt it using their private key. Since encrypting an entire message can be time-consuming, PGP uses a faster encryption algorithm to encrypt the message and then uses the public key to encrypt the shorter key that was used to encrypt the entire message. Both the encrypted message and the short key are sent to the receiver who first uses the receiver's private key to decrypt the short key and then uses that key to decrypt the message. [8]

For sending digital signatures, PGP uses an efficient algorithm that generates a hash (or mathematical summary) from the user's name and other signature information. This hash code is then encrypted with the sender's private key. The receiver uses the sender's public key to decrypt the hash code. If it matches the hash code sent as the digital signature for the message, then the receiver is sure that the message has arrived securely from the stated sender. PGP's RSA version uses the MD5 algorithm to generate the hash code. PGP's Diffie-Hellman version uses the SHA-1 algorithm to generate the hash code. [8]

The structure of the PGP is as follows:

- PGP version
- Information on key holder
- Key certified
- Signature of the holder of the key
- Preferred encryption algorithms
- Validity
- Signatures

B. X.509

Standard ITU-T X.509 is an internationally valid recommendation, which describes the use of a form of PKI (Public Key Infrastructure). PKI is a set of hardware, software, people, policies and methods used to unambiguously assign the public key of a particular entity in the use of electronic signatures.

At present it is used for the third version. Unlike PGP X.509 system is centralized. It uses the so-called tree of trust. On its top is the main certification authority. That is called a root certification authority (Root CA) and it is seen as trustworthy. The root certification authority must comply strictest conditions. The function of the root CA is mainly act as a CA for other providers of certification services. By root CA certifies another CA that will lower the confidence to use almost the same as the root CA. You can then certify more. This creates a tree in which the user can work out up to authority, he considered to be credible. The path from the endpoint certificate to the trusted location is called a certification path. The path is shorter, the better the system works in practice. Each CA publishes its certification policy. It is a document in which the CA specifies the procedure for verifying the identity of the certificate, the demands placed on it, and under what circumstances, issues certificates and how to take care of security. CA should have a security policy, a plan for crisis management and recovery plan. These documents greatly affect the local legislation. The standard is the requirement for regular processing and publishing CRLs. His leadership is also one of the functions of CA. However, it may not perform by itself, but it can delegate to another institution.

Structure of a certificate:

The structure foreseen by the standards is expressed in a formal language, namely Abstract Syntax Notation One.
The structure of an X.509 v3 digital certificate is as follows: [11]

- Certificate
  - Version
  - Serial Number
  - Algorithm ID
  - Issuer
  - Validity
    - Not Before
    - Not After
  - Subject
  - Subject Public Key Info
    - Public Key Algorithm
    - Subject Public Key
  - Issuer Unique Identifier (optional)
  - Subject Unique Identifier (optional)
  - Extensions (optional)
- Certificate Signature Algorithm
- Certificate Signature

VI. Application of the Digital Signature

This section is focused on practical use of the digital signature in own programs. The iTextSharp open source library was used to develop tool to build PDF document and to complement the electronic signature into it. Created application was named pdfsign. This section also shows steps to generate own certificate for testing purposes.

A. Generate certificates

The Certificate Creation tool was utilized for testing the functionality of the application. The Certificate Creation tool generates X.509 certificates for testing purposes only. It creates a public and private key pair for digital signatures and stores it in a certificate file. This tool also associates the key

pair with a specified publisher's name and creates an X.509 certificate that binds a user-specified name to the public part of the key pair.

Makecert.exe includes basic and extended options. Basic options are those most commonly used to create a certificate. Extended options provide more flexibility.

BASIC OPTIONS:

Option	Description
-n *x509name*	Specifies the subject's certificate name. This name must conform to the X.500 standard. The simplest method is to specify the name in double quotes, preceded by CN=; for example, "CN=*myName*".
-pe	Marks the generated private key as exportable. This allows the private key to be included in the certificate.
-sr *location*	Specifies the subject's certificate store location. *Location* can be either currentuser (the default), or localmachine.
-ss *store*	Specifies the subject's certificate store name that stores the output certificate.

Fig. 3 Basic options [5]

EXTENDED OPTIONS:

Option	Description
-b ***mm/dd/yyyy***	Specifies the start of the validity period. Defaults to the certificate's creation date.
-e **mm/dd/yyyy**	Specifies the end of the validity period. Defaults to 12/31/2039 11:59:59 GMT.
-r	Creates a self-signed certificate.
-sv *pvkFile*	Specifies the subject's .pvk private key file. The file is created if none exists.

Fig. 4 Extended options [5]

Command line:
The following commands (Fig. 5) generate a private key and a certificate which will be saved into the resulted file Pivnickova.pfx.

```
"makecert.exe -r -pe -n "CN=Lucie Pivnickova"
-b 01/01/2012 -e 01/01/2099 -ss
DevelopCertStore -sv Pivnickova.pvk
Pivnickova.cer"

"pvk2pfx.exe -pvk Pivnickova.pvk -spc
Pivnickova.cer -pfx Pivnickova.pfx -po
myPassword"
```

Fig. 5 Command to generate experimental certificate

B. Created application

In this work was created a simple console application, named pdfsign.exe, to create a PDF (Portable Document Format), which is then digitally signed. PDF document was chosen as the standard with which most users are accustomed to work and is basically the closest to conventional printing on paper. For creating and signing of the document are provided several command line parameters. See example below. The command line parameters can be divided into three groups:

1. parameters relating to information on signing
2. data to build PDF file from
3. additional information (PDF file metadata, such as author and title of the document)

Help string to show how to format command line:

```
public const string commandLine =
        "\"pdfFile\" ...output pdf file\n" +
        "\"certFile\" ...cert. pfx file\n" +
        "\"certPassword\" ...cert. password\n" +
        "-i:20 \"path\" ...imagePath:page
        percentage(optional)\n" +
        "-t \"text\" ...text to write into pdf file\n" +
        "Signature info\n" +
        "-sR \"text\" ...reason\n" +
        "-sC \"text\" ...contact\n" +
        "-sL \"text\" ...location\n" +
        "-sI \"path\" ...signature image\n" +
        "-sV ...signature visibility\n" +
         "PDF metadata\n" +
        "-mA \"text\" ...author\n" +
        "-mT \"text\" ...title\n" +
        "-mS \"text\" ...subject\n" +
        "-mK \"text\" ...keywords\n" +
        "-mC \"text\" ...creator\n" +
        "-mP \"text\" ...producer";
```

Command line:
The following command (Fig. 6) creates PDF document named sample.pdf and sign it by use Pivnickova.pfx certificate.

```
pdfsign "d:\Sample.pdf" "d:\Pivnickova.pfx"
cerKlic2 -t "VERIFICATION OF THE EXAMINATION
RESULTS" -t " " -t " " -i:80
"d:\IMGSample.png" -sC pivnickova@fai.utb.cz -
mA "Lucie Pivnickova" -mT Sample -sV -sR
"Exam" -sL Holesov
```

Fig. 6 Command to sign PDF document

Where:

- Sample.pdf - output PDF file signed
- Pivnickova.pfx - is the file that is encapsulated private key and certificate
- cerKlic2 - the password, which is protected pfx file
- i: 80 - percentage displayed image SampleExam.jpg
- sC pivnickova@fai.utb.cz – contact
- mT Sample - title
- sV - visibility signature

- sR "Exam" - reason
- sL Holesov – location

The .NET framework does not contain any native way to work with PDF files. So, it is necessary to rely on one of the many third party components that are available. This work used free DLL library iTextSharp. iTextSharp is a C# port of a Java library written to support the creation and manipulation of PDF documents.

The following code shows part of the created pdfsign application appending signature into the PDF document.

Source code:

```
using iTextSharp.text.xml.xmp;

//Signing document
PDFSigner pdfs = new
PDFSigner(pIn.pdfFile+"tmp", pIn.pdfFile,
myCert, MyMD);
pdfs.Sign(pIn.Reason, pIn.Contact,
pIn.Location, pIn.Image, pIn.visible);

/// this is the most important class
/// it uses iTextSharp library to sign a PDF
document
class PDFSigner
{
	private string inputPDF = "";
	private string outputPDF = "";
	private Cert myCert;
	private MetaData metadata;

	public PDFSigner(string input, string
output, Cert cert, MetaData md)
	{
		this.inputPDF = input;
		this.outputPDF = output;
		this.myCert = cert;
		this.metadata = md;
	}

	public void Sign(string SigReason,
	string SigContact, string SigLocation,
	string SigImage, bool visible)
	{
		PdfReader reader = new
		PdfReader(this.inputPDF);
		PdfStamper st =
		PdfStamper.CreateSignature(reade
		r, new
		FileStream(this.outputPDF,
		FileMode.Create,
		FileAccess.Write), '\0', null,
		true);

		st.MoreInfo =
	this.metadata.getMetaData();
		st.XmpMetadata =
	this.metadata.getStreamedMetaData();
		PdfSignatureAppearance sap =
	st.SignatureAppearance;

		sap.SetCrypto(this.myCert.Akp,
		this.myCert.Chain, null,
		PdfSignatureAppearance.WINCER_SI
		GNED);
		sap.Reason = SigReason;
		sap.Contact = SigContact;
		sap.Location = SigLocation;

		if( SigImage.Length > 0 )
			sap.Image =
		Image.GetInstance(new
		Uri(SigImage));
		if (visible)
		{
			sap.SetVisibleSignature(ne
			w
			iTextSharp.text.Rectangle(
			500, 1, 100, 50), 1,
			null);
		}

		st.Close();
	}
}
```

C. *Use of the console application*

Prepared application is primarily intended to be called from third-party software.

At present, PdfSign was integrated into the ENT system Fowler and expanded its functionality to electronically sign the results of audiometry and tympanometry exams. Signed result shows Fig. 7.

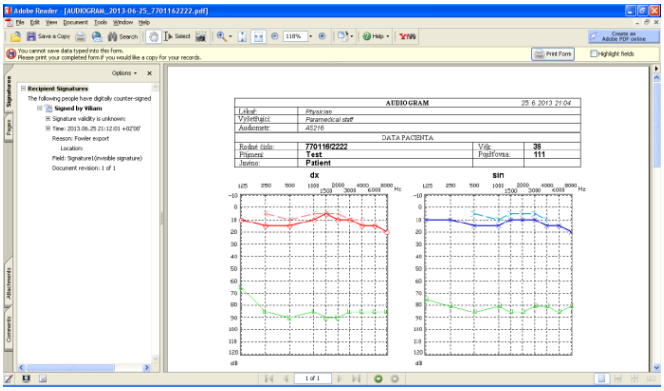

Fig. 7 Signed output of ENT system

VII. Conclusion

The aim of this work was to apply a digital signature to documents that contain resulting data from various programs, focusing on medical software. The practical part of this work was to create a simple console application based on C # library iTextSharp to create a PDF file built from the image output of

the other applications and digitally signed the resulting file. This paper only shows basic concepts how the resulting output can be converted into signed PDF file. This paper at the beginning also focused on description of the theoretical aspects of the digital signature such as used principle of encryption.

VIII. Acknowledgment

The project "Verification of the examination results" is supported by Internal Grant Agency of Tomas Bata University, IGA/FAI/2013/043. The work was also supported by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

References

[1] ALVAREZ, Rafael, Francisco-Miguel MARTINEZ, Jose-Francisco VICENT, and Antonio ZAMORA. A Matricial Public Key Cryptosystem with Digital Signature. WSEAS TRANSACTIONS on MATHEMATICS: WSEAS Press, 2008, p. 10. ISSN 1109-2769.

[2] ANDRESS, Jason. *The basics of information security: understanding the fundamentals of InfoSec in theory and practice*. Boston: Singers Press, ©2011, xviii, 171 p. ISBN 978-159-7496-537

[3] DOLINAY, Viliam, Lucie PIVNICKOVA and Vladimir VASEK. Methods for evaluating and improving audiometric examinations - ORL system Fowler. Proceedings of the 13th International Carpathian Control Conference. 2012, no. 19626, p. 115-118. ISBN: 978-1-4577-1867-0. DOI: 10.1109/CarpathianCC.2012.6228626

[4] MINISTERSTVO VNITRA ČESKÉ REPUBLIKY. *Electronic Signature* [online]. ©2010 [cit. 2012-11-08]. Available from: http://www.mvcr.cz/docDetail.aspx?docid=45011&docType=ART&chnum=2

[5] MSDN. *Certificate Creation Tool*. Certificate Creation Tool [online]. 2012 [cit. 2013-01-14]. Available from: http://msdn.microsoft.com/en-us/library/bfsktky3(v=vs.80).aspx

[6] NAKAKUNI Masanori, Eisuke ITO, Yoshiaki KASAHARA, Sozo INOUE, Hiroshi DOZONO. Construction and Use Examples of Private Electronic Notary Service in Educational Institutions. WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION: WSEAS Press, 2008, p. 11. ISSN 1790-1979.

[7] STOLFO, Salvatore J, Steven M. BELLOVIN, Shlomo HERSHKOP, Angelos D. KEROMYTIS, Sara SINCLAIR a Sean W. SMITH. COLUMBIA UNIVERSITY AND DARTMOUTH COLLEGE. *Insider Attack and Cyber Security: Beyond the Hacker*. New York: Springer, ©2008, xi, 222 p. Advances in information security, 39. ISBN 03-877-7322-3.

[8] SEARCHSECURITY. *What is Pretty Good Privacy (PGP)*. What is Pretty Good Privacy (PGP) [online]. 2005 [cit. 2013-01-14]. Available from: http://searchsecurity.techtarget.com/definition/Pretty-Good-Privacy

[9] STACKOVERFLOW. *Java - Checking serial code correctness*. Java - Checking serial code correctness [online]. 2009 [cit. 2013-01-14]. Available from: http://stackoverflow.com/questions/1832640/checking-serial-code-correctness

[10] TUBA, Milan, Nadezda STANAREVIC. Relation between Successfulness of Birthday Attack on Digital Signature and Hash Function Irregularity. WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS: WSEAS Press, 2010, p. 10. ISSN 1790-0832.

[11] WIKIPEDIA - THE FREE ENCYCLOPEDIA. *X.509*. X.509 [online]. 2013 [cit. 2013-01-14]. Available from: http://en.wikipedia.org/wiki/X.509

[12] WRIGHT, Dan. *The RSA Algorithm*. The RSA Algorithm [online]. 2007 [cit. 2013-01-13]. Available from: http://imps.mcmaster.ca/courses/SE-4C03-07/wiki/wrighd/rsa_alg.html

[13] WIKIPEDIA - THE FREE ENCYCLOPEDIA. *Digital signature*. Digital signature [online]. 2013 [cit. 2013-01-14]. Available from: http://en.wikipedia.org/wiki/Digital_signature

[14] PIVNICKOVA, L., V. DOLINAY and V. VASEK. Verification of the examination results by digital signature. 4th International Conference on Bioscience and Bioinformatics (ICBB '13) In: Chania, Crete Island, Greece: WSEAS Press, 2013, p. 5. ISBN 978-960-474-326-1.