BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **11**, No 2

Sofia • 2011

An Approach for Automatic Transmission of Authenticated Data over Computer Networks

Nikolay Dokev, Ivan Blagoev

New Bulgarian University, 1618 Sofia
E-mails: n.dokev@nbu.bg blagoev.i@gmail.com

Abstract: *This article presents an approach for automated transmission of authenticated data over computer networks. We discuss some of the more popular approaches widely used to authenticate web site users and their shortcomings. Some web spoofing attacks are also considered that may be used for temporary diversion of information and possibly changing or replacing its contents without the knowledge of the sender. We propose as a solution to this problem the developed specialized system "Signer and sender through HTTP" and describe the technologies used and the various modules and steps comprising the system. Some experimental resuts are shown.*

Keywords: *Network communications, security, digital signatures.*

1. Introduction

There are more and more cases in which it is essential that the data being sent or received is authenticated (unaltered). Although there is a number of widely available tools for information authenticity, they require the involvement and assistance of an operator [3]. The purpose of this article is to present an approach for automated transmission of authenticated data over computer networks. This will increase the reliability of the communication and will reduce the need for an intervention by a person (an operator) which is otherwise a part of the process.

The analysis of existing similar software systems shows that there are few publications about such systems and most of them are dedicated to the particular problem being solved. In three similar publications [1-3] there is discussion of the problems being analyzed which makes it evident that these systems were built for authenticated transmission of specific types of data.

The first article describes a software system designed for transmission of binary data over computer networks which are to be authenticated by digital signing [1]. Since the binary data in question is in a ready for execution stream, the risks that the data can be compromised and become a security threat are completely real. For example, such data are sent between computers when watching online video, listening to music or receiving images. The author has brought up the security issues related to sending binary data and has suggested approaches for solving this problem. A significant obstacle in this task is the fact that the data arrive sequentially in a stream and can be of a large quantity. This leads to difficulties in their authentication before they are transferred completely, because the hash confirming their state should be generated at the last stage before the final validity check of the digital signature.

The second article offers a solution for transferring data to a web service [2]. The data are digitally signed using a XML signature format and are packaged as a SOAP object which is then sent over any communication protocol. After receiving, the data are validated and accepted if they pass the digital signature check. The author's chosen area of investigation is again concerning the problems in exchanging data between applications that communicate over computer networks. A solution is offered to prevent such problems, which relies on XML format for a digitally signed object that can be validated by a given web service.

The third article also deals with the validation of data passing through firewalls [3]. In this case transmission of authenticated data over computer networks is being used. The main issue with this system is that the data is not authenticated while passing through the firewall and cannot be verified, particularly if encryption has been applied in addition to the digital signature. This is because the data are first signed and then encrypted. For this reason the authors have proposed an approach which allows the signing of encrypted data but in a way that makes it possible to verify the digital signature. Thus, after having put the firewall in operation, they are able to solve the problem and allow the encrypted data to reach the user, but only after verification.

The software system proposed in the present paper aims to avoid these and other shortcomings by using a combination of different technologies.

The main difference between the software system proposed and the ones above mentioned is in the purpose set and in the implementation approach. For example, the proposed system relies on a specially designed web service which automatically receives the whole content and performs authentication of the data and their sender. If the data are unaltered, they are passed to the corresponding application and rejected otherwise. In both cases the user is notified. The system developed may be used for transmitting various authenticated information from many clients to one or between two users. On the other hand, it can make use of both licensed digital certificates and unlicensed ones that have been generated by the users themselves.

One of the goals of the present paper is to demonstrate how such a product can be developed, what modules comprise it and what technologies can be used.

To confirm the necessity of such a system, this article analyzes some of the most popular methods for authenticating a sender and the information being sent, as well as their shortcomings.

Some of the more widely used approaches for web site user authentication are given below [3].

- **Transmitting data through the web by user name and password authentication [10].** This is the most common approach, but in its basic form, also the most vulnerable. The user enters his user name and password in the specified form fields, which are then sent as a plain text to the server. Some web sites also use e-mail-based authentication in order to be more confident about the user's identity, but even then we may doubt whether that person's e-mail is not being read by someone else. The web authentication user name and password may also be compromised by interception and eaves dropping on the network traffic. Even if the traffic between the web site and the user is protected by some kind of encryption, this approach still poses a lot of vulnerabilities.
- **Data encryption using a symmetric key algorithm [1-3].** This type of data encryption uses the same key for both encrypting and decrypting the information. Transferring data on the web is actually contrary to this type of information protection, particularly if we want to be able to identify the source. If the key is sent over the public web, using HTTP or another protocol, we cannot be sure how many people besides us and the recipient may possess it. Therefore the application of this approach requires that we combine it with other types of encryption, for example, asymmetric key algorithms which allow the transmission of the symmetric key encoded through asymmetric cryptography. But even in this case we cannot be certain that we have received an unique key known only to us and our correspondent. We also do not know whether his computer has not already become a victim of a hacker who may have stolen that symmetric key.
- **Data encryption using an asymmetric key algorithm [4-6, 16].** This approach to encoding the data is one of the most secure in case we want to keep them hidden from view. Based on what has been said so far and mentioning the term "asymmetric", it should be clear that the technology uses a key pair – a public and a private key. If we own a key pair, we may provide our public key to someone who wants to send us confidential information. After the sender encrypts and sends the message, it is assumed that only we can decrypt and read it again. After the data has been encrypted, it may not be accessed in this form even by the message author who sends it to us. The downside, however, is that even if the information is well secured, there is no guarantee that it has been sent by a specific author. Anyone could have the public key and use it to encrypt and then send us data. Hence this technology may not be considered to give a solution to the information authentication problem.
- **Data channel encryption via SSL or HTTPS [13].** Encrypting the connection when using the web and HTTP is a good solution to enhance security as a whole,which may be combined with other security methods, such as any of the aforementioned. But, as in the other examples so far, there is no guarantee that we

have identified the sender of the information, which automatically brings us back to the fundamental problem.

2. Problem description

Suppose that we have a web-based information system receiving user data and that authentication of the sender is of great importance to us. To make sure we are protected against spoofing by forgery of data, we have chosen to combine some of the most popular methods for HTTP data security, some of which have been listed above. But there is a well-prepared hacker, who is motivated and interested to mislead our information system, so he plans one of the most dangerous web spoofing attacks – the man-in-the-middle attack. With this attack, a convincing but false copy of the entire web site is created. The fake web site looks like the real one, i.e., it contains all pages and links as in the original. The hacker has full control over it, so that all network traffic between the client and the web pages passes through the hacker (Fig. 1).

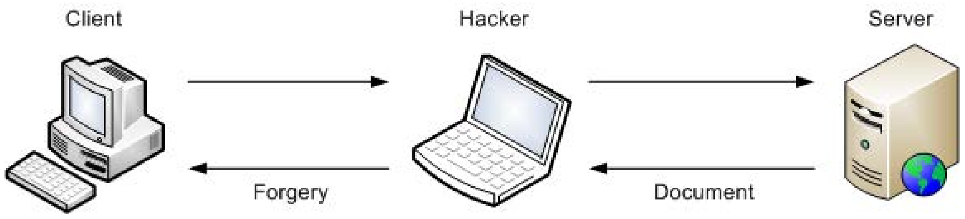


Fig. 1. All network traffic between the client and the web pages passes through the hacker

During a man-in-the-middle attack, the hacker can monitor or modify any data being passed from the client to the server. He can also control the entire traffic in the reverse direction. That means that there is an ample opportunity to modify the data as they are passing through from the client to the server or vice versa. It should be noted that the two most commonly used methods for network intrusion are spying and spoofing. Spying is an observation activity, where the network traffic is monitored passively, while spoofing is an active intervention, because the hacker impersonates a trusted host in order to obtain the necessary information.

In web spoofing, the full content of the pages visited by the client may be recorded. After a form in the HTML page is completed, the browser submits it via HTTP to the server, but as the attacker is situated between them, he may record all the sent data. In addition, he may also record the content of the responses sent from the server to the client. Acting as an intermediary between the client and the server, the hacker could change the content of the message being sent by one party and cause the other party to act against its will. Imagine what might happen with such an attack if, for example, you need to transmit important financial data to a government institution or if you use online banking to submit a payment order! When considering this type of attack, we must keep in mind the fact that the hacker may conduct surveillance even if a relatively secure connection is used by the client. Whether the connection uses SSL or S-HTTP, the hacker may still exploit it, i.e., despite the fact that the client’s browser may display the icon for secure connection between the client and the server, at that moment the client actually transmits data over an insecure connection (Fig. 2).

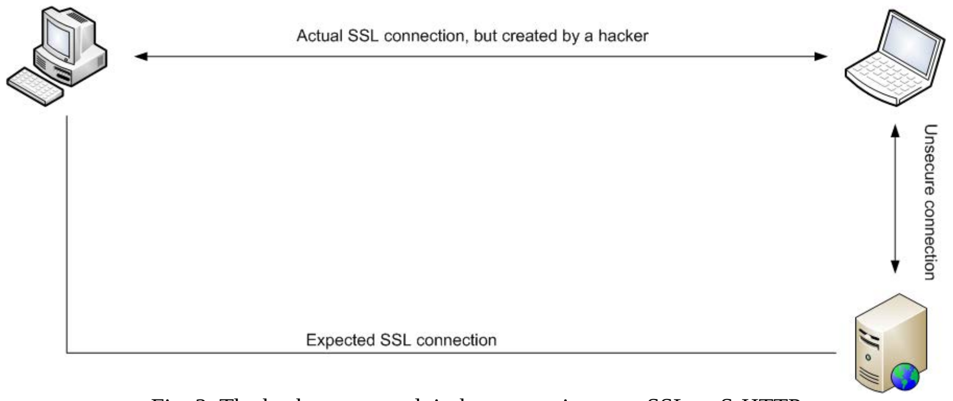


Fig. 2. The hacker may exploit the connection uses SSL or S-HTTP

In this case the client requests the page while a secure connection is available. When attempting to secure the connection (via SSL or S-HTTP), the browser will behave in a completely standard way and the security indicator icon will be turned on. At this point the browser’s behavior is actually misleading because while we have a secure connection to the server, in reality we are connected to a fake web server. To enhance the user’s protection, modern web browsers have built-in functionality to validate automatically the SSL certificate which should be used to encrypt the traffic. However, this is not a very efficient protection, being one-sided, because perhaps a large fraction of the web servers on the Internet do not use certificates correctly issued and signed by an authorized digital certificate provider. To be able to access such a web site, one must force the browser to accept the certificate by adding an exception.

After such an attack has begun, it is very difficult to be stopped. But its launch does require some participation by the client. For this purpose, he must somehow be deceived and redirected to the hacker server. Here are some possible ways of client’s deception for a particular web site:

- Inclusion of a link to a fake web site inside a popular web site.
- When using a mail client with HTML support, an attacker may send a message with a link to his server. Alternatively, the fake web site content may be sent.
- A hacker may spoof a search engine so that it sends users to a portion of the fake web site.
- A hacker may create a control for a browser already adopted by the client which will be able to replace a correct URL with a hacked one.
- If the client is infected with a trojan or a similar virus which can modify the hosts file of the operating system being used. This would ensure the redirection of a correct URL to the hacker machine.

- A hacker may gain access to any of the routing machines serving the client. This may be a router, or a proxy server, which could redirect the client to the hacker machine when he tries to access the correct web address.

3. An approach for automatic transmission of authenticated data over computer networks [7-9, 14-15]

We have developed and proposed a specialized functional system “Signer and sender through HTTP” as a solution to this problem. The implementation uses Microsoft .NET Framework technology and C# programming language. This client-server type of the system implements the necessary functionality to identify the sender of a message received through web and HTTP in a fully automatic way, as well as to verify the authenticity of the received data. It also has the capability to accept and process simultaneously and automatically information received from multiple locations, which is the main advantage of this solution. Analyzing the system, it can be logically divided into two main software modules – client and server. The server module is implemented as web service which plays a key role in the software solution. It accepts the SOAP encapsulated data sent over HTTP. The sender is identified by the certificate present in the digital signature, which is also used to authenticate the message. The process is fully automated and works without the need for an IT operator, which allows the solution to be also used in other systems where the problem of identifying and authenticating the received data is of great importance. The client part of the system is a Windows GUI desktop application intended for the user who is actually in the role of a client in this information system. The user actions have been simplified to a great extent so that the software is more efficient and easy to use. Only the data file that needs to be sent and a certificate to digitally sign it must be selected. Then the client application establishes a connection to the web service so that these two modules can completely perform independent and automatic transmission, verification and storage of the authenticated information, as well as notification of the user about the outcome from the operation.

To implement this, it is necessary to combine various technologies which separately used cannot help us with the task at hand. But combining them together in a certain way to form a software system will lead us to the desired solution.

- **Web via HTTP [10].** This is one of the most common Internet protocols used to exchange data between two computers. Nearly all web services work over this higher level protocol, i.e., HTTP does not function by itself but depends on lower level protocols such as TCP and IP. Because most Internet users mainly use the global network for different web services working over HTTP, which enables them to seek information, share data, view multimedia and communicate with other users, HTTP has become a fundamental protocol for communication in the global information network.
- **Web service [2, 9].** A web-based service which allows using web servers and client-server technology, as a finished product providing the communication between separate software applications. It helps the applications to communicate

over computer networks and combines them in a functioning system. The web service allows sending and receiving serialized objects and different types of data. The main advantages obtained by this approach are that we may leave the communication tasks to proven and powerful software, like the web server thus avoiding the need to build, maintain and develop a specialized interface for this purpose. Another important fact is that it allows interconnection and communication between software applications built upon either similar or completely different platforms or technologies.

- **Transmission of objects through SOAP (Simple Objects Access Protocol) [2, 7-9].** This technology defines a mechanism for exchanging messages in a decentralized distributed environment. Different types of data may be carried through this mechanism, such as binary data or XML wrapped encoded content. There are other types of technology for transmitting such data, for example: IIOP/CORBA, ORPC/DCOM, as well as Java Remote Method Protocol. The big advantage of SOAP is that the protocol is based on a textual instead of a binary representation (using XML) which is not associated with any particular manufacturer and does not lead to a commitment to a particular software or hardware platform. Most modern software technologies for application and software system development contain also built-in methods for processing SOAP objects. However, since this approach only gives us a way to represent data, but not to transmit it, most often the SOAP packaged objects are transmitted to remote computer systems over HTTP protocol.
- **Digitally signing data according to CMS/PKCS7 standard [1, 3-6].** Data encryption according to CMS/PKCS7 allows data to be saved in a format that is almost impossible to be tampered with, unknown to us. However, the means for falsifying information transmitted in such a way are too complicated, so that we may consider this method to be sufficiently reliable. The asymmetric algorithm digital signature technology uses a private and a public key and requires that a hash of the data to be signed must be extracted first, using a reliable hashing algorithm, such as SHA1. Then the hash in question is digitally signed by applying the asymmetric algorithm in combination with the private key of the sample data sender. The result is a digital signature and is saved in a format described by CMS/PKCS7 standard. There are two different ways for this: the digital signature may be saved separately or may be joined to the signed content in an appropriate manner as described by the standard.
- **Digital verification of CMS/PKCS7 encoded data [1, 3-6].** The digital verification of PKCS7 signed data is a process that aims to demonstrate whether the data to which the digital signature belongs, are unchanged. The process is the reverse of the digital signing and CMS/PKCS7 encoding. It begins by reading the format and checking whether all the expected content is present. The correctness of the signed hash is verified using the public key and the asymmetric algorithm. Then the hash for the signed content is generated again and a check is made whether it matches the hash from the verification. Additional checks may be added to this type of verification, for example, checking the digital certificate used for signing. But they are not mandatory and the process can function without them.

Examining the last two technologies for digital signing may lead to a conclusion that they are sufficient for creating the solution being sought. This may be true in some cases, but not for the problem specifically considered by us.

Suppose that we have purchased a digital certificate from an authorized provider of certification services, which offers also hardware means of protecting the private key. In addition, we were provided with a high quality and well working software application for digital signing. At a first glance, it may seem we have everything required to digitally signed data, but actually that is not sufficient to accomplish the set goal. The reason is that the widely available and accessible commercial and non-commercial software products for digital signing do not usually offer an implementation of the function to digitally sign data and automatically send that data to a remote computer by communicating with a service supported by it. Thus we will have to seek another software solution to assist us with sending and receiving the digitally signed data. Let us then imagine that we achieve this through the easiest option, by e-mail. This means that we will need to have a mail server available to relay the e-mail messages. In the absence of such a server, we will need to use alternative techniques, such as a direct connection to the recipient's mail server, in order to send the message as desired. But it is not likely that the provided cryptographic software will have an implementation of this or a similar functionality. Another option is to create a solution that does this, but using shell scripts, which is entirely possible. We can use console commands to take the digitally signed content and send it as an attached file, just like it happens with a mail client. But it is not certain that all networks and all firewalls will allow this type of direct connection, which would terminate the process immediately. After all keep in mind that the system must accept data from all kinds of computer systems, from different computer networks, and from as many users as possible. As a third case, assume that, although unlikely, SMTP servers are available everywhere and rely on an e-mail client that has a built-in function to send digitally signed e-mail messages. But in this situation another problem arises which would be an obstacle to the process. For example, consider working with larger files because e-mail services support attaching files, but with very limited sizes. Finally, we must take into account the fact that the e-mail account, to which we send digitally signed content will need to be checked by an operator. He would need to store and verify the digital signatures and to return a response to the corresponding client again via e-mail, which is quite labor intensive for larger volumes. And that diverts us from our goal because this and other similar solutions that may be devised would require large resources for their continuous operation. They would also require the users that send digitally signed information to be computer experts in order to be able to cope with the techniques used, deal with any problems that may arise and also have technology knowledge. This will lead to restricting the system availability to a specific range of users, who are specialists in this field. However, our goal is to create a solution that can be used by people who are not IT specialists and is not exposed to the risk of tampering with the information that should be received from them. The cost of operating such a system is also an important issue because engaging an individual as an operator who receives and verifies the digitally signed messages would lead to higher operating cost and delays in processing the received and verified data.

Therefore the conclusion can be made that the options offered by generally available technologies and software solutions are quite insufficient. Their inherent disadvantages confirm the need for finding a different solution for automated transmission and authentication of data over computer networks.

The solution proposed is a software system that combines in an original way the different advantages of each of the mentioned technologies in order to find the best possible alternative to meet our needs. It also offers functionality that cannot be found in existing popular software systems that can be used for the given task. Schematic of the functionality of the software system is given on Fig. 3.

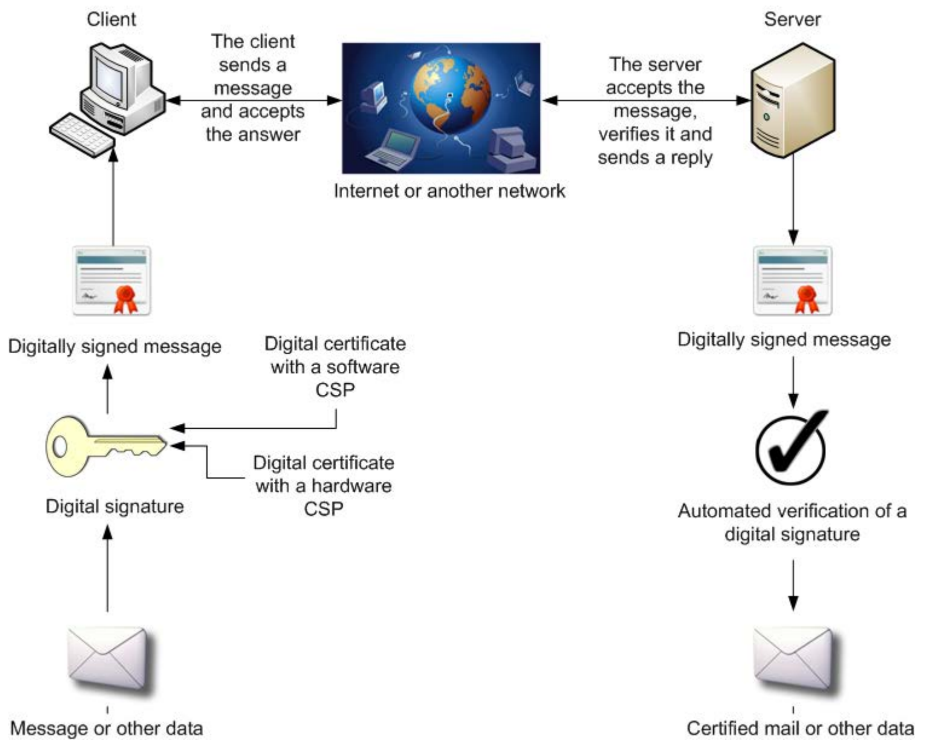

Client
The client sends a message and accepts the answer
Internet or another network
Server
The server accepts the message, verifies it and sends a reply
Digitally signed message
Digital certificate with a software CSP
Digital certificate with a hardware CSP
Digital signature
Message or other data
Digitally signed message
Automated verification of a digital signature
Certified mail or other data

Fig. 3. Schematic of the functionality of the software system

The process under consideration is comprised of the following seven basic steps.

1. The client uses the software, which is a part of the developed system, to pick the desired data file to be sent. Afterwards, using the same program, he selects a digital certificate to be used by the proposed software solution to sign and send the data in a fully automatic way. When choosing a digital certificate, the program allows the option to use any digital certificate and not just those issued by an official PKI provider. Also, it is not a mandatory requirement that the digital certificates must be protected and stored on an external or internal hardware cryptographic module. This makes it possible to satisfy a different range of needs,

where resources are extremely limited but it is still necessary for users to exchange data over computer networks. In this case instead of purchasing hardware and digital certificates, the users may generate the certificates themselves using the tools of the operating systems. Afterwards the public parts of the digital certificates are exchanged so that they can be recognized by the proposed software system. After all, there are situations which lack such extreme importance which could necessitate the acquisition of hardware cryptographic devices and digital certificates but nevertheless most software products offered require the maximum available safeguard methods in question.

2. Starting the process of digital signing and sending, the client program reads the data file and digitally signs it in a CMS/PKCS7 format with an attached digital signature. The digitally signed message is packaged in a ready for transmittal SOAP object. Then a HTTP connection is established to the web service of the other (server) part of the system, which will accept, store and authenticate the data. Upon successful connection, the SOAP-packaged digitally signed content is transmitted.

3. The web service which receives the data over HTTP is public and does not require any additional identification. Accepting the SOAP data sent, it completely automatically validates the format of the information received. Then it decodes the SOAP content and retrieves the CMS/PKCS7 digitally signed data together with the signature.

4. Message validation [5] – in this part of the process the data is verified and their author is identified. Considering this check, we may divide it into two main stages: primary and secondary. The primary check is a low level test which means that it only deals with the format of the received digitally signed data. It certifies that the received data is complete; its format meets the requirements of CMS/PKCS7 and has not been damaged. If it succeeds, we proceed to the secondary check. Its role, through unpacking of the digitally signed data, is to identify their author and then to verify the authenticity of the signed data. If the secondary check is successful, the process may proceed to its next stage.

5. The received data are converted into a file which will be stored by the server in the specified in advance for this purpose location.

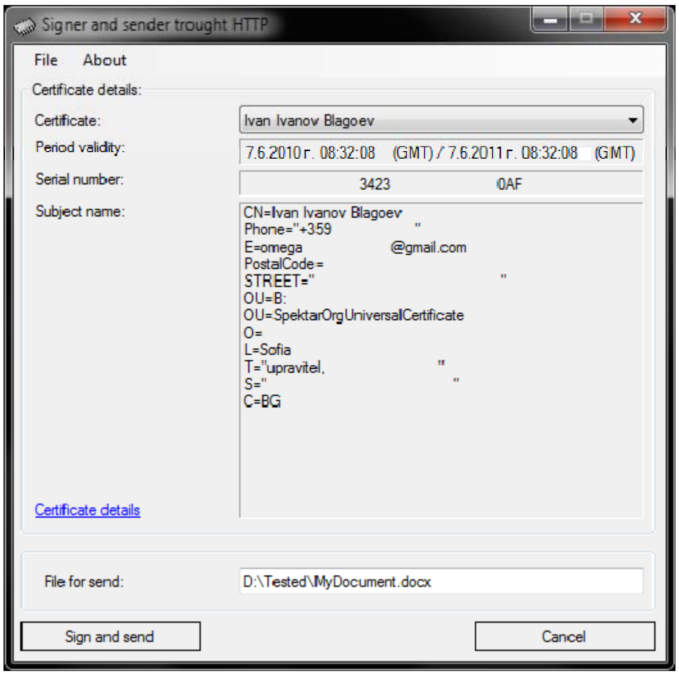
6. Whether the validation was successful and the file was accepted or not, the web service prepares a return response to be sent to the client software. This response actually is the result of the operation so far. It is encoded as a SOAP object and is returned over the already established HTTP connection.

7. The client software accepts the response SOAP object over HTTP; decodes it and displays the result to the user who will know whether the desired information was sent correctly.

4. Experimental results

The described software system was subject to a series of tests which proved its correct operation and efficiency. We used a sample scenario recreating sending and receiving a file over a computer network, which has not been been modified in transit, as well as a file which was modified before being received:

1. Choose a text document named “MyDocument.docx” and choose our personal digital certificate for its signing. Then clicking the “Sign and send” button will perform the actions of digitally signing the content of the file, packaging it in a SOAP object and sending it to the web server over HTTP (Fig. 4).



Certificate details:

Certificate: Ivan Ivanov Blagoev

Period validity: 7.6.2010 г. 08:32:08 (GMT) / 7.6.2011 г. 08:32:08 (GMT)

Serial number: 3423 0AF

Subject name:

```
CN=Ivan Ivanov Blagoev
Phone="+359                "
E=omega              @gmail.com
PostalCode=
STREET="                         "
OU=B:
OU=SpektarOrgUniversalCertificate
O=
L=Sofia
T="upravitel,              "
S="                          "
C=BG
```

File for send: D:\Tested\MyDocument.docx

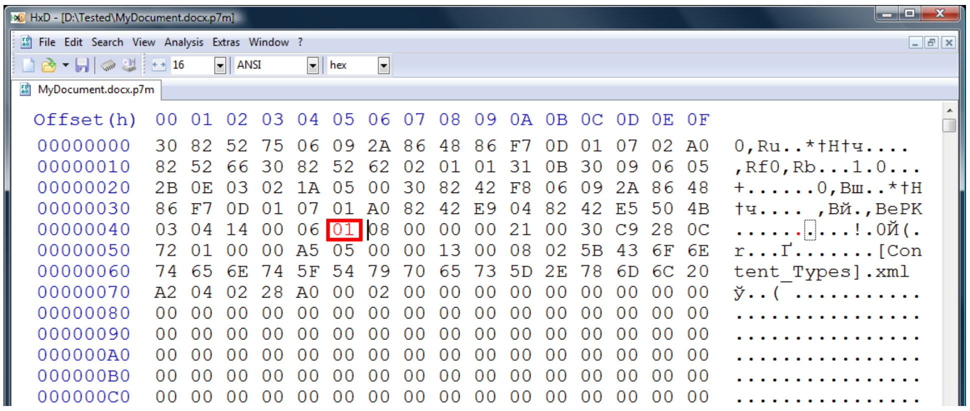
Fig. 4. Digitally signing the content of the file, packaging it in a SOAP object and sending it to the web service over HTTP

2. The acceptance and verification of the file is performed by the web service, which checks the received data for valid format and integrity and if everything is proper, proceeds to authenticate the data with the digital signature. Once the verification process is completed successfully, the client receives over HTTP and decodes the SOAP packaged message “File is received successfully”. Then the file is saved to the specified for this purpose location on the hard drive.

3. In steps 1 and 2 we saw the ideal case of operation of the process but if we did not expect cases where problems occur, there would be no point in looking for such a solution. Hence, we will create a situation in which the data were modified before their transport to the web service.

Using HxD – Hex editor or another similar program, we may view and modify the contents of the file being sent.

In this case only one byte of data has been modified, by changing its value from 00H to 01H at offset 46H (Fig. 5).

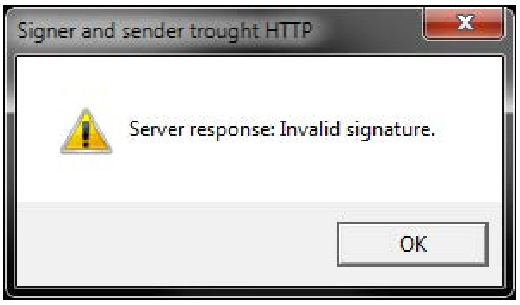

HxD - [D:\Tested\MyDocument.docx.p7m]
File Edit Search View Analysis Extras Window ?
16
ANSI
hex
MyDocument.docx.p7m

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  30 82 52 75 06 09 2A 86 48 86 F7 0D 01 07 02 A0  0,Ru..*†H†ч....
00000010  82 52 66 30 82 52 62 02 01 01 31 0B 30 09 06 05  ,Rf0,Rb...1.0...
00000020  2B 0E 03 02 1A 05 00 30 82 42 F8 06 09 2A 86 48  +......0,Bш..*†H
00000030  86 F7 0D 01 07 01 A0 82 42 E9 04 82 42 E5 50 4B  †ч.... ,Bй.,BePK
00000040  03 04 14 00 06 01 08 00 00 00 21 00 30 C9 28 0C  ..........!.0Й(.
00000050  72 01 00 00 A5 05 00 00 13 00 08 02 5B 43 6F 6E  r...Ґ.......[Con
00000060  74 65 6E 74 5F 54 79 70 65 73 5D 2E 78 6D 6C 20  tent_Types].xml
00000070  A2 04 02 28 A0 00 02 00 00 00 00 00 00 00 00 00  ў..( ...........
00000080  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000B0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Fig. 5. In this case only one byte of data has been modified,
by changing its value from 00H to 01H at offset 46H

Upon validation of this digitally signed message, the result is a failed verification of the digital signature. Then the web service will again send a SOAP packaged response to the client over HTTP, but its content this time will be "Invalid signature" (Fig. 6).



Server response: Invalid signature.

Fig. 6. If the result of validation is a failed verification of the digital signature,
then the web service will send to the client "Invalid signature"

The message shown is displayed by the client program that is designed to establish communication with the web service and is a part of the system.

The system performs also authentication of the sender who has digitally signed the message received by us. This occurs after we have registered with the system in advance the public parts of the digital certificates of the users by whom we expect to receive data. Using the software code that we wrote for this purpose, the web service extracts from the received digitally signed data the certificate, which was used to sign them. Then we find by searching whether it is present in the list of digital certificates from which we should be receiving data. To increase security, we can activate the verification of digital signatures through technologies supported by digital certificate providers, such as CRL and OCSP. But this will limit the system to accepting only data signed using digital certificates purchased from official providers. And there may be cases where these safeguards are not necessary because the information being transmitted through the system is not of such high importance. Therefore this feature has been left as optional. This is yet another advantage of this software solution according to the needs we may have.

5. Discussion and conclusion

The advantages of the solution presented in this article allows an easy way of sending authenticated information from many authors at the same time. The process of sending the data may not be consecutive and the authors may send their data simultaneously from different Internet locations. The information will be received by the proposed system simultaneously and will be validated and stored in fully automatic way. At the same time, a real-time response is sent to each of the data senders about the outcome of the operation. Thus the sender will know immediately whether the data have been accepted and validated successfully or an exception has occurred which has resulted in rejection.

The system allows also some other applications. For example, it may facilitate secure communication between two people. They will be able to send authenticated messages to each other without the need of other utility programs. Another possibility is the system to be easily expanded and upgraded to a different type of system that is designed to meet the specific requirements of a particular organization. The technologies that have been selected and used for this purpose are sufficiently flexible and popular so that the solution given may inter-operate with other types of systems.

References

1. G e n n a r o, R., P. R o h a t g i. How to Sign Digital Streams. – Lecture Notes in Computer Science, Advances in Cryptology – CRYPTO'97, Vol. **1294**, 1997, 180-197. **http://www.springerlink.com/content/855385q3103t7403/**
2. H o n d o, M., N. N a g a r a t n a m, A. N a d a l i n. Securing Web Services (IBM Software Group, One Charles Park, Cambridge, Massachusetts 02142, USA, Issue Date: 2002) – IBM Systems Journal**,** Vol. **41**, 2002, 06 April 2010, No 2, 228-241. **http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5386919**
3. G a m a g e, C., J. L e i w o, Y. Z h e n g. Encrypted Message Authentication by Firewalls. – Lecture Notes in Computer Science, Peninsula Scool of Computing and Information Technology, Monash University, Australia, Vol. **1560**, 1999, No 634, DOI: 10.1007/3-540-49162-7_6), H. Imai, Y. Zheng (Eds.) PKC'99, LNCS 1560, Berlin, Heidelberg, Springer-Verlag,1999, 69-81. **http://www.springerlink.com/content/updpmmnukbytvhvh/**
4. Microsoft – Designing and Managing a Windows Public Key Infrastructure, 2003.
5. H o u s l e y, R., T. P o l k. Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure, 2001.
6. X e n i t e l l i s, S. The Open-Source PKI Book. 2000, Last Update 2009-07-17. **http://ospkibook.sourceforge.net/docs/OSPKI-2.4.7/OSPKI-html/ospki-book.htm**
7. Microsoft – Designing and Developing Windows Applications by Using the Microsoft .NET Framework, 2005.
8. Microsoft – Designing and Developing Enterprise Applications by Using the Microsoft .NET Framework, 2005.
9. Microsoft – Designing and Developing Web-Based Applications, 2005.

10. F i e l d i n g et al. Internet RFC 2616, The Internet Society – RFC 2616 Hypertext Transfer Protocol HTTP/1.1, 2004.
http://www.w3.org/Protocols/rfc2616/rfc2616.html
11. The Internet Society Nystrom & Kaliski – RFC 2986 Certification Request Syntax Specification, 2000.
http://tools.ietf.org/html/rfc2986
12. The Internet Society – RFC 2798 The LDAP inetOrgPerson Object Class, 2000.
http://www.ietf.org/rfc/rfc2798.txt
13. Netscape Communications – Signing Text from JavaScript, 1997.
http://docs.sun.com/source/816-6152-10/
14. Microsoft. Developing Custom Windows Forms Controls with the .NET Framework, 2010.
http://msdn.microsoft.com/en-us/library/6hws6h2t.aspx
15. Microsoft MSDN. System. Security. Cryptography Namespace, 2010.
http://msdn.microsoft.com/en-us/library/system.security.cryptography.aspx
16. The Internet Society – RFC 3447 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1, 2003.
http://tools.ietf.org/html/rfc3447

Application

The appendix presents the more essential parts of the system source code. They are fundamental to solving the problem described and could be used for the implementation of other similar systems.

1. The following source code is a part of the client application and is responsible for reading the transmitted data and their digital signing:

```
byte[] dataForSign = null;

        if (!FileHelper.FileRead(FilePath + FileName, ref dataForSign,
SAHS_Common.Messages.CaptionMsg))
        {
            dataForSign = null;
            FilePath = null;
            FileName = null;
            return;
        }

        X509Certificate2 signerCert = new X509Certificate2();
        if
(!CertUtils.CertStoreHelper.GetCertificateFromStore((int)cmboCertSelect.SelectedValue,
ref signerCert, SAHS_Common.Messages.CaptionMsg))
        {
            signerCert = null;
            return;
        }
        if (!CertUtils.SignatureUtils.Sign(signerCert, ref dataForSign,
SAHS_Common.Messages.CaptionMsg)) return;
        if (signerCert == null) return;

        dataForSign = SAHS_Common.Vars.SendMessage_Wrapper(dataForSign,
FileName);
```

2. The client application establishes a connection via HTTP to the receiving server. Packs the signed content in an XML object and transmits the data, then monitors for a server response confirming the result of the operation:

```
HttpWebRequest httpReq =
(HttpWebRequest)WebRequest.Create(SAHS_Common.Vars.Url_ToSend);
        httpReq.Method = "POST";
        httpReq.ContentType = SAHS_Common.Vars.SendMessage_ContentType;
        httpReq.ContentLength = dataForSign.Length;

        Stream reqStream = null;
        Stream respStream = null;
        HttpWebResponse httpResp = null;
        string result = "";

        try
        {
            // Write the request content
            reqStream = httpReq.GetRequestStream();
            reqStream.Write(dataForSign, 0, dataForSign.Length);

            // Read the response
            httpResp = (HttpWebResponse)httpReq.GetResponse();
            StreamReader reader = new StreamReader(httpResp.GetResponseStream());
            result = reader.ReadToEnd();
        }
        catch (Exception ex)
        {
            MessageBox.Show("File transfer error: " + ex.Message,
SAHS_Common.Messages.CaptionMsg, MessageBoxButtons.OK,
MessageBoxIcon.Warning);
            return;
        }
        finally
        {
            if (reqStream != null) reqStream.Close(); reqStream = null;
            if (respStream != null) respStream.Close(); respStream = null;
            if (httpResp != null) httpResp.Close(); httpResp = null;
            dataForSign = null;
            httpReq = null;
        }

        if (result.IndexOf("<UploadFileResult>Ok</UploadFileResult>") > 0)
        {
            MessageBox.Show("File is received successfully.",
SAHS_Common.Messages.CaptionMsg, MessageBoxButtons.OK,
MessageBoxIcon.Information);
            txtFileUpload.Text = "";
        }
        else if (result.IndexOf("<UploadFileResult>Invalid signature.</UploadFileResult>")
> 0)
        {
            MessageBox.Show("Incorrect digital signature. File is rejected.",
SAHS_Common.Messages.CaptionMsg, MessageBoxButtons.OK,
MessageBoxIcon.Warning);
         }
         else
         {
            MessageBox.Show("File is rejected. Undefined error.",
SAHS_Common.Messages.CaptionMsg, MessageBoxButtons.OK,
MessageBoxIcon.Warning);
         }
```

3. For better system consistency, the methods for retrieving the digital certificates from the certificate store and for retrieving a specific digital certificate are located in a separate library which is a part of the project and the methods:

```
private X509Certificate2[] certsInStore = null;

public bool ReadCertStore(string AppCaptionMsg)
         {
            bool retval = true;
            X509Store storeOpen = null;

            storeOpen = new X509Store(StoreName.My, StoreLocation.CurrentUser);
            certsInStore = null;
            certsCount = 0;

            try
            {
               storeOpen.Open(OpenFlags.ReadOnly);

               // No certificates in store
               if (storeOpen.Certificates.Count == 0)
               {
                  return retval;
               }

               // Certificate counter
               int iCount = 0;

               // Get cert number
               certsInStore = new X509Certificate2[storeOpen.Certificates.Count];
               foreach (X509Certificate2 foundcert in storeOpen.Certificates)
               {
                  certsInStore[iCount] = foundcert;
                  iCount++;
               }
               certsCount = iCount;
            }
            catch (Exception ex)
            {
               certsCount = 0;
               retval = false;
               if (AppCaptionMsg != null) MessageBox.Show("Certificate store reading error:
" + ex.Message, AppCaptionMsg, MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
            }
            finally
            {
                storeOpen.Close();
                storeOpen = null;
            }

            return retval;
        }

public static bool GetCertificateFromStore(int certPositionInStore,
            ref X509Certificate2 x509CertRawData, string AppCaptionMsg)
        {
            bool retval = true;
            x509CertRawData = null;
            X509Store storeOpen = null;
            int iCount = 0;

            storeOpen = new X509Store(StoreName.My, StoreLocation.CurrentUser);

            try
            {
                storeOpen.Open(OpenFlags.ReadOnly);

                // Read certificates
                foreach (X509Certificate2 foundcert in storeOpen.Certificates)
                {
                    if (certPositionInStore == iCount)
                    {
                        x509CertRawData = foundcert;
                        break;
                    }
                    iCount++;
                }
            }
            catch (Exception ex)
            {
                retval = false;
                if (AppCaptionMsg != null) MessageBox.Show("Certificate store reading error:
" + ex.Message, AppCaptionMsg, MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
            finally
            {
                storeOpen.Close();
                storeOpen = null;
            }

            return retval;
        }
```

4. The following web method carries out the receiving, verification checking and saving of the sent digitally signed data, then returns a response to the client about the result of the operation:

```
[WebMethod]
        public string UploadFile(byte[] f, string fileName)
        {
            MemoryStream ms = null;
            FileStream fs = null;
            byte[] retrFileData = null;
            string retval = "Ok";

            try
            {
                ms = new MemoryStream(f);

                retrFileData = ms.ToArray();
                if (CertUtils.SignatureUtils.SignatureVerify(ref retrFileData, null))
                {
                    fs = new
FileStream(System.Web.Hosting.HostingEnvironment.MapPath(@"~/TransientStorage/") +
fileName, FileMode.Create);
                    ms.WriteTo(fs);
                }
                else
                {
                    retval = "Invalid signature.";
                }
            }
            catch (Exception ex)
            {
                retval = ex.Message.ToString();
            }
            finally
            {
                if (ms != null)
                {
                    ms.Close();
                    ms.Dispose();
                }
                if (fs != null)
                {
                    fs.Close();
                    fs.Dispose();
                }
                retrFileData = null;
            }

            return retval;
        }
```