Secure Internet Phone*

Update: 2005/07/15

Patrick Horster, Martin Schaffer, Peter Schartner, and Dieter Sommer

University of Klagenfurt, Austria
Computer Science · System Security
www.syssec.at

Abstract. This paper describes the design and implementation of a secure internet phone utilizing a smartcard-based public key infrastructure. First, available hardware and software based systems are discussed followed by a description of the architecture of our system. More detailed information on the public key infrastructure including the certification authority, the client and the public key server are to come in chapters thereafter. The article finishes with a concise description of the audio subsystem.

1 Introduction

The curriculum of the studies in "Angewandte Informatik" (Applied Computer Science) requires a small project to be carried out in order to gain hands-on experience of how software development works. As in real life and in order to experience a true project feeling, the project should be done by at least two people. Because we started specializing in security, we decided to carry out our project in this field. We found the first lack of security in Internet communication systems while we were playing games that use voice communication over networks. After analyzing several other communication systems, the idea of the "Secure Internet Phone" (SIP) was born. The aims our SIP-design set out to achieve were as follows:

- authenticity of the involved users (calling and receiving party),
- confidentiality and integrity of the call data and protocol data,
- required bandwidth $\leq$ 8 kByte/s,
- low speech delay, and
- use of smart cards to store vital security information.

2 Overview of existing systems

2.1 Hardware Systems

Various companies such as AT&T or Motorola offer hardware solutions for secure telephony which are unfortunately quite expensive. In general only poor information about security mechanisms can be found. Trust is placed almost entirely in

* Originally published in the Proceedings of IFIP TC6/TC11 CMS'01.

either the manufacturer or in the designer (usually the National Security Agency (NSA)). Predominantly two standards are used, namely the Secure Telephone Unit Type III (STU-III) [16, 11] and the Secure Terminal Equipment (STE) [15]. Both of them use hardware security tokens (Crypto Ignition Key (CIK) [16] or Fortezza Plus KRYPTON Card [14] respectively) to enhance security. The CIK is a storage device for key data and not capable of performing cryptographic operations; in contrast, the Fortezza Plus KRYPTON Card contains the whole cryptographic unit. STU-III devices provide a back door for the NSA [17]; moreover, the non-government and export versions are weakened. All Fortezza Cards contain the well known Capstone Chip [2] implementing Key Escrow [5].

Summing up, it can be said that these hardware phone devices are not suitable for private use because of the high prices, and questionable for commercial use because of the NSA's back door.

2.2 Software Systems

Secure software telephony systems are programs running on common operating systems such as Windows or UNIX and standard hardware providing secure voice communication over telephone lines or the Internet. Only a fraction of the Internet phone programs support cryptographic mechanisms such as encryption, authentication or use of a public key infrastructure. The following three systems are commonly referenced on security related pages on the Web:

– Speek Freely uses the Web of Trust of the PGP-program to negotiate a session key. The available encryption algorithms are DES, Blowfish and IDEA.
– The Nautilus Secure Phone offers Diffie-Hellman key agreement with a subsequent verbal comparison of the hash value of the agreed session key in order to prevent the system from being compromised by the ”man in the middle” attack. Three algorithms, namely Triple-DES, Blowfish and IDEA are supported for voice encryption. A drawback of this system is the lack of full duplex audio capability.
– PGPfone - a widespread system - uses the same mechanism for key agreement as the Nautilus Secure Phone. The offered encryption algorithms are Triple-DES, CAST and Blowfish. Unlike the Nautilus Secure Phone it does support full duplex audio connections.

The products using Diffie-Hellman do not offer authentication; Speek Freely on the other hand does so by using PGP's Web of Trust [8] concept. None of the above systems support the use of security tokens such as smartcards; moreover, no public key infrastructure (PKI) with a trusted party is used.

3 Public Key Infrastructure

To achieve mutual authenticity of caller and callee we designed a public key infrastructure based on hardware security tokens (smartcards) and public key certificates. The core component of the PKI is the certification authority (CA)

whose task it is to register the public key server (S) and the users by generating their keys and certifying them. We used RSA [10] with 1024 bit moduli for public key encryption and digital signatures. In order to prevent the digital signature mechanism from existential forgery there was a need for a cryptographically secure hash function - the RIPEMD-160 [3] algorithm seemed to be appropriate. In the implementation of our system we used the ASECrypto Cryptographic Library [1] which provides software implementations of the algorithms and the interface to smartcards. For symmetric encryption we integrated IBM's reference implementation of the AES finalist MARS [7]. However, the source code can easily be replaced by Rijndael - the winner of the AES competition.

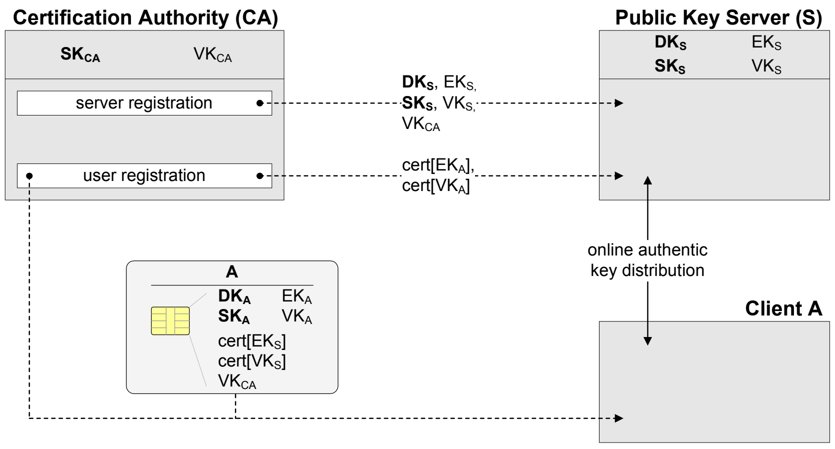


Fig. 1. Public key infrastructure

The distribution of the public keys in our system is done by a public key server. That means users do not get certificates except that of the public key server, but authentic public keys delivered by the server. The reason why we chose this key distribution mechanism is because due to its centrality it enables the easy and instant revocation of certificates. However, certificates could be made available to users by a public directory which would render the Public Key Server obsolete, i.e. it would be degraded to a simple address directory server.

3.1 Certification Authority

The Certification Authority is the center of trust and must be run offline in a highly secure environment. When initializing the SIP infrastructure the public key server is the first instance to be registered at the CA. In this process the Server receives two key pairs, namely (EK_S, DK_S), (SK_S, VK_S) and VK_{CA}.

In the registration process of a user A the two key pairs (EK_A, DK_A) and (SK_A, VK_A) are generated and stored together with VK_CA, cert[EK_S], cert[VK_S] and id_A on A's smartcard. Further, the certificates containing the user's public keys (cert[EK_A] and cert[VK_A]) are transferred to the public key server. Thus, each registered user is able to verify public key certificates with VK_{CA} and mutual authentication becomes possible.

We designed simple certificates shown in figure 6 (a). Our certificates are identified by a unique id. The most important components are of course the user name, user id and the public key blob because the task of certificates is to bind a key to a user. This is achieved through the signature of a hash of the certificate data. Via the element "key type" we can identify whether the contained key is the user's encryption or signature verification key. The public key is embedded as a so called key blob used by the ASECrypto Cryptographic Library [1]. The issuing date and the validity is necessary to prevent usage of outdated certificates.

EK_X	RSA 1024 bit key of X for encryption (public)
DK_X	RSA 1024 bit key of X for decryption (private)
VK_X	RSA 1024 bit key of X for signature verification (public)
SK_X	RSA 1024 bit key of X for signature generation (private)
E(m, EK_X)	encryption of message m with EK_X
D(m, **DK_X**)	decryption of message m with **DK_X**
V(sm, VK_X)	signature verification of sm = (m, S(m, **SK_X**))
S(m, **SK_X**)	signature generation of hash of m with **SK_X**
cert[EK_X]	public key certificate of X for EK_X signed by the CA with SK_{CA}
cert[VK_X]	public key certificate of X for VK_X signed by the CA with SK_{CA}
id_X	unique identifier of X
$sk_{X/Y}$	symmetric session key between X and Y for encryption and decryption
$mk_{X/Y}$	symmetric session key between X and Y for MAC generation and verification
req(X)	request for public keys EK_X and VK_X and $addr_X$
$addr_X$	network address of X

Fig. 2. Acronyms used throughout the article

3.2 Client

After the user has been registered at the certification authority he/she can use any phone client of the system by using the received smartcard. When starting the client program an authentication process between A and the public key server takes place in which $sk_{A/S}$, $mk_{A/S}$ and the respective initialization vectors are established. These keys and initialization vectors are used for protecting subsequent protocol communication between client A and the Public Key Server which we will refer to as secure protocol connection (SPC) because protocols can be run encrypted and integrity protected (see section 5). A client uses an established SPC for getting the network addresses and public keys of the interlocutor when placing a phone call or being called.

3.3 Public Key Server

The public key server always has to remain online to allow registered users to place phone calls. After a couple of user registrations the newly created user certificates are transferred to the Public Key Server (offline transmission) which stores them on its hard disk. When a client authenticates to the public key server, its public keys will be extracted from the certificates into main memory. Furthermore, network addresses and other user specific information will be kept there. Such data are necessary for authentication processes between two clients and can be transferred to clients over the SPC on request.

4 Authentication

First of all the authentication method we used in our system should be described in order to understand the authentication process between two instances (client-server and client-client) as detailed below. We decided to implement a slightly modified X.509 [6] protocol because it provides strong entity authentication and is based on our public key infrastructure. The X.509 standard defines two-pass and three-pass protocols. Using the two-pass protocol would have required synchronized clocks because time-stamps are used for replay detection. This can be avoided by using the three-pass protocol which makes use of nonces to guarantee the freshness of the protocol messages. In the SIP-design the three-pass protocol seemed to be more appropriate (see Figure 3) since it does not rely on synchronized clocks.

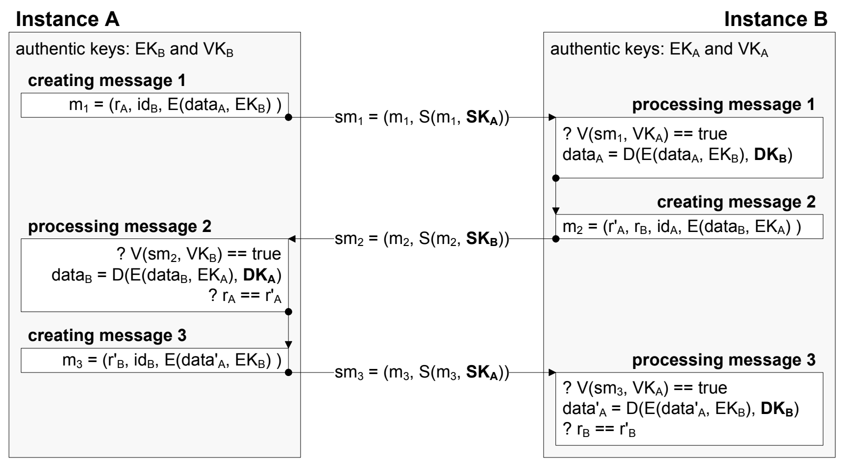

Instance A
authentic keys: EK_B and VK_B
creating message 1
$m_1 = (r_A, id_B, E(data_A, EK_B))$
processing message 2
$? V(sm_2, VK_B) == true$
$data_B = D(E(data_B, EK_A), \mathbf{DK_A})$
$? r_A == r'_A$
creating message 3
$m_3 = (r'_B, id_B, E(data'_A, EK_B))$
$sm_1 = (m_1, S(m_1, \mathbf{SK_A}))$
$sm_2 = (m_2, S(m_2, \mathbf{SK_B}))$
$sm_3 = (m_3, S(m_3, \mathbf{SK_A}))$
Instance B
authentic keys: EK_A and VK_A
processing message 1
$? V(sm_1, VK_A) == true$
$data_A = D(E(data_A, EK_B), \mathbf{DK_B})$
creating message 2
$m_2 = (r'_A, r_B, id_A, E(data_B, EK_A))$
processing message 3
$? V(sm_3, VK_A) == true$
$data'_A = D(E(data'_A, EK_B), \mathbf{DK_B})$
$? r_B == r'_B$

Fig. 3. X.509 three-pass authentication

Before the authentication protocol can be run, the public keys of the respective other party must be made available. An instance A has to start the protocol by transmitting message m1 signed with SK_A. This message contains a nonce

r_A, the user id of B (id_B) and an encrypted portion of data which will be described in the next paragraph. After having received sm_1, instance B verifies the signature of m_1 and in case of success, the contained encrypted data can be decrypted using DK_B. In order to authenticate to A, B has to embed the received nonce r_A into m_2 as r'_A. Moreover, B has to include its own nonce r_B, so that A can authenticate to B in the last step of the protocol. The processing of m_2 is analogous to that of m1. However, A has to check, whether the received r'_A is equal to the sent r_A. If equality holds, B has authenticated to A. The generation of m_3 is similar to m_2, with one difference: only the received nonce r_B will be sent as r'_B. After having checked the validity of the signature of m_3 and the equality of r_B and r'_B, A has authenticated to B. Thus, the protocol has finished successfully.

The encrypted data of m_1 and m_2 contain partial session keys and initialization vectors. These keys will be combined to the final symmetric session key $sk_{A/B}$ on both instances. In comparison, the encrypted data of m_3 contains the MAC session key $mk_{A/B}$ and it's initialization vector. The structure of an X.509 message is detailed in figure 6 (b). Header 1 distinguishes between X.509 and SPC messages on the server side.

4.1 Authentication between client and public key server

Here, client A and the public key server need the public keys of the respective other party as required in the general description of the authentication protocol. The client extracts EK_S and VK_S from the certificates stored on it's smartcard after having checked the validity of the certificates with VK_{CA}. In order to verify the signature of m_1 sent by A, the public key server has to extract A's public keys from the certificates stored on hard disk. Thus, the authentication protocol can proceed as described above.

Furthermore, the public key server stores the network address of A to make it available for other clients. Some additional data, like user id and name, will be kept in main memory, too. After having finished the authentication process, client A and the Server share a symmetric session key $sk_{A/S}$, a MAC key $mk_{A/S}$ and the related initialisation vectors. These data units are the fundamental elements of the SPC that has been established between A and the public key server within the framework of the authentication.

4.2 Authentication between two clients

The authentication between two clients is an integral part in the process of establishing a phone call. The requirements of authentication between two clients is the fact that caller and callee have both authenticated to the public key server (gone online). In the SIP infrastructure every caller has to know the callee's user id which acts as the "phone number". The distribution of phone numbers could be performed by public web directories, which are not currently designed or implemented. Caller A now requests the public keys EK_B and VK_B and network address $addr_B$ of callee B (see message $req(B)$ in figure 5). After having received

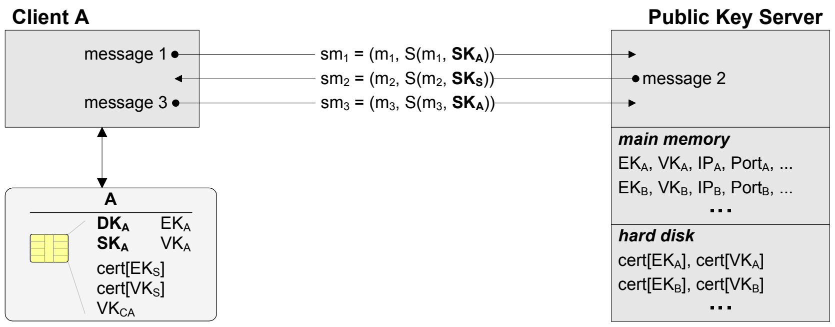


Fig. 4. Authentication between client and public key server

the requested data via the SPC, A initiates the X.509 three-pass protocol by sending sm_1 to B. However, B neither can verify the signature of m_1, nor can it decrypt the data embedded in m_1, because B does not have A's public keys. So, B has to request the public keys EK_A and VK_A of A over the SPC between the public key server and itself. After the verification of m_1's signature the protocol can proceed as described in the general description of the authentication.

Once authentication has finished the two clients possess the session key $sk_{A/B}$ and an initialization vector for the CBC mode of operation of the MARS cipher for audio encryption. Furthermore, A and B hold a MAC key $mk_{A/B}$ and the related initialization vector which are used for integrity protection of the audio data. Now the audio data can be transmitted in a secure, integrity-protected and authentic manner. More details about the audio subsystem can be found in section 6.

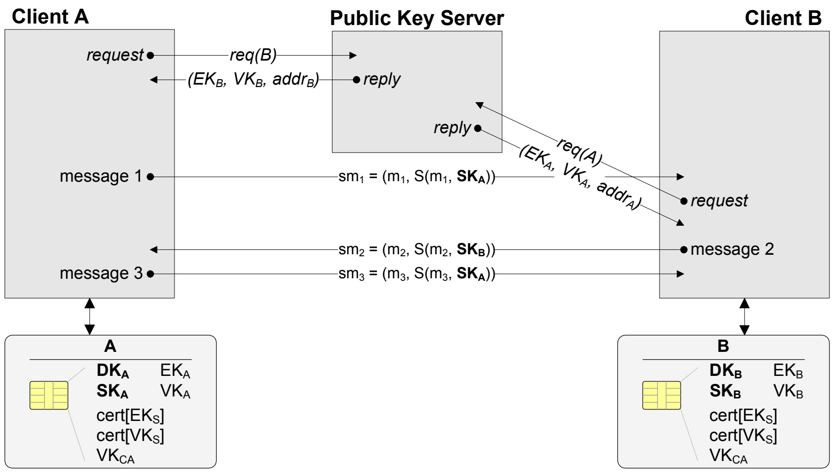


Fig. 5. Authentication between two clients

5 Secure Protocol Connection

The SPC can be considered as an encrypting, integrity-protecting and authenticity-guaranteeing protocol which is used for all subsequent client-server communications. Whenever a client goes online, a new SPC between this client and the public key server is established by exchanging the keys and initialization vectors needed within the authentication.

The communication over the SPC is realized by the use of SPC messages (cf. figure 6 (c)). Header 1 is used to distinguish respectively between an SPC message and an X.509 message on the server side. In the case of an SPC message, header 2 identifies the type of protocol being run over SPC (cf. figure 7).

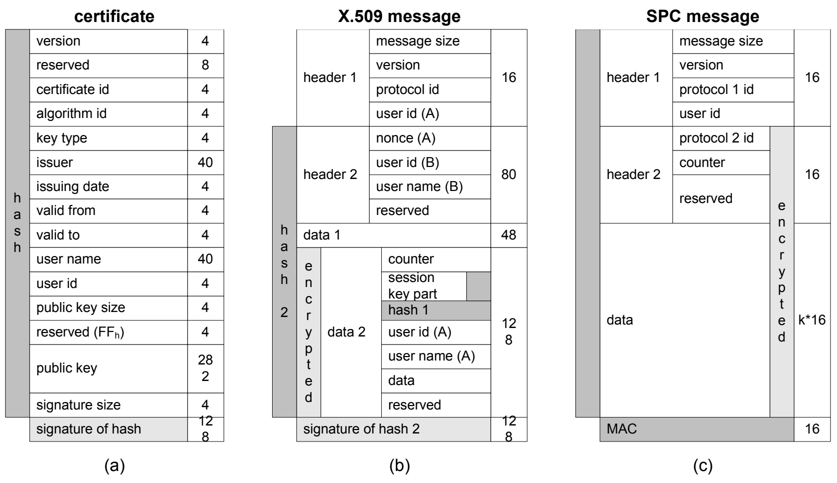


Fig. 6. (a) certificate, (b) X.509 message, (c) SPC message

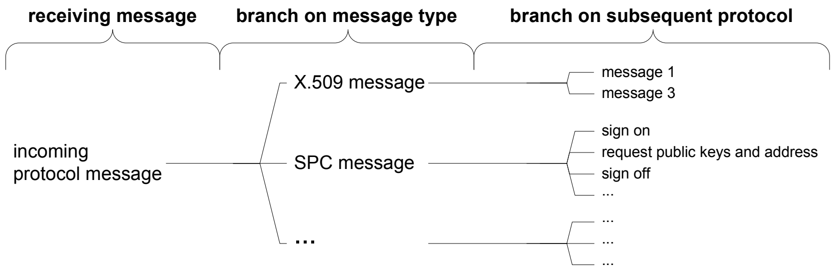
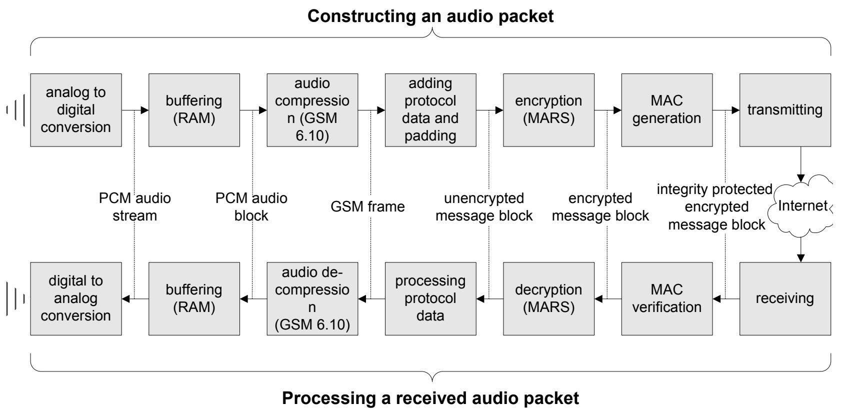


Fig. 7. Protocol handling of the public key server

6 Audio Processing Subsystem

The symmetric keys ($sk_{A/B}$ and $mk_{A/B}$) that have been established during the authentication process between A and B are now used for the secure audio communication. The callee B initializes its audio subsystem and starts audio recording. When the first audio block has been recorded, it is processed and transmitted to A. On receiving the first packet, A initializes its audio subsystem and starts audio recording and playback. B initializes the audio playback after having received the first packet from A. In this full duplex stream of audio packets, protocol data can be embedded, which at present is only used for terminating the phone connection.



Constructing an audio packet
analog to digital conversion
buffering (RAM)
audio compression (GSM 6.10)
adding protocol data and padding
encryption (MARS)
MAC generation
transmitting
PCM audio stream
PCM audio block
GSM frame
unencrypted message block
encrypted message block
integrity protected encrypted message block
Internet
digital to analog conversion
buffering (RAM)
audio de-compression (GSM 6.10)
processing protocol data
decryption (MARS)
MAC verification
receiving
Processing a received audio packet

Fig. 8. The audio subsystem

Every data packet sent over the network must be processed as shown in the upper part of figure 8. Digitized audio data, sampled at 11025 Hz, is compressed block-by-block (160 16 bit samples per block) by the GSM 6.10 algorithm [4] to 33 byte blocks. After attaching protocol data and padding, the message is encrypted using MARS (key: $sk_{A/B}$) in CBC mode. Note that compression is done before encryption for two reasons: first, encrypted audio data could not be compressed any further and second, security is increased by encrypting data containing very little redundancy. In the next step a MAC, also produced by MARS in CBC mode (key: $mk_{A/B}$), is generated and added. Now the packet can be passed to the network.

Every received audio packet is processed as shown in the lower part of figure 8. The steps are the respective "inverse" steps from construction of an audio packet.

7 Conclusion

Considering the aims for the SIP stated in section 1, it can be seen that all of them have been realized:

– Authenticity is achieved through the public key infrastructure.
– Confidentiality and integrity of the call data and protocol data is guaranteed by the use of symmetric encryption and MAC generation.

– GSM 6.10 audio compression cuts the data rate down to a sub-ISDN bandwidth.
– The use of small audio blocks and the full exploitation of Windows' audio capabilities minimize speech delay.
– The use of smartcards further enhances security of the public key infrastructure by storing vital keys in a tamper-proof manner.

References

1. FAST Software Security GmbH & Co. KG. ASECrypto Cryptographic Library for 32-bit Windows. Application Programmers Guide and API-Reference, Version 1.3, Mrz 1998, Germering, 1998.
2. D.E. Denning. Description of Key Escrow Systems (http://www.cosc.georgetown.edu/ denning/crypto/Appendix.html).
3. H. Dobbertin, A. Bosselaers, B. Preneel. RIPEMD-160 - A strengthened version of RIPEMD, st Software Encryption - Cambridge Workshop, Md. 1039, Springer-Verlag, Berlin, S.71-82, 1996.
4. Audio compression algorithm GSM-6.10 source code. (http://kbs.cs.tu-berlin.de/ jutta/toast.html)
5. Escrowed Encryption Standard. FIPS PUB 185, 1994.
6. A. Menezes, P. van Oorschot, S. Vanstone. Handbook of Applied Cryptography, CRC Press, 1999.
7. C. Burwick et al. MARS - a candidate cipher for AES, IBM Corporation, 1999.
8. Network Associates, Inc. An Introduction to Cryptography, 1999.
9. Nautilus Secure Phone (http://www.lila.com/nautilus/).
10. R.L. Rivest, A. Shamir, L.A. Adleman. A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, Vol.21, Nr.2, S.120-126, 1978.
11. B. Schneier. Applied Cryptography, John Wiley & Sons, 1996.
12. Microsoft Corporation. Microsoft Platform SDK, Juli 2000 Edition, Redmond, 2000.
13. Speek Freely (http://www.fourmilab.ch/netfone/windows/doc/pgp.html).
14. Spyrus Inc. Fortezza Crypto Card - Technical Specification (http://www.spyrus.com/content/products/legacy/fortezza/).
15. https://infosec.navy.mil/PRODUCTS/SECUREVOICE/stu3.html
16. Dept. of Defense Security Institute. STU-III Handbook for Industry, 1997.
17. P.R. Zimmermann. PGPfone - Owner's Manual, 1996.