

# SSA

**Client**

**User Guide**

Version 1.2.5

Cybernetica

Akadeemia tee 21, EE0026 Tallinn, ESTONIA

<http://www.cyber.ee/>

© 1998 Cybernetics Ltd., all rights reserved.

Cybernetica and the Cybernetica logo are registered trademarks and Privador, Barricade, and SSA are trademarks of Cybernetics Ltd. All other brand, product or company names are trademarks or registered trademarks of their respective owners.

# Contents

Introduction.....	5
Operation .....	7
Technology .....	8
User authentication. Certificates.....	9
Session reuse .....	10
Distinguished names.....	11
Strength of the cryptoalgorithms used .....	13
Model of the SSA system.....	15
Session.....	18
Licencing .....	19
SSA system components .....	21
Graphical user interface .....	21
Command-line interface.....	22
SSA Client operations.....	25
SSA configuration file .....	25
Creating the configuration file.....	26
Opening the configuration file.....	27
Saving the configuration file .....	27
Saving the configuration file under another name .....	28
Closing the configuration file .....	28
Changing the passphrase of the configuration file.....	28
Viewing the contents of the configuration file .....	29
Viewing the object list in the configuration file .....	29
RSA key.....	30
Generating the RSA key .....	30
Deleting the RSA key .....	31
Viewing the RSA keys.....	32

The certificate .....	32
Requesting the certificate .....	33
Adding certificates to the configuration file .....	34
Certificate deletion .....	35
Viewing a certificate.....	36
Viewing the certificate list.....	37
Certification Authority public key .....	37
Adding a Certification Authority public key to the configuration file.....	38
Deleting the Certification Authority public key.....	39
Viewing the Certification Authority public key.....	40
Viewing the list of Certification Authorities' public keys.....	41
Licencing information.....	41
Adding the licencing information to the configuration file .....	42
Deletion of licencing information.....	42
Viewing the licencing information .....	43
Viewing the list of licences.....	44
Session.....	44
Creating a session .....	48
Session modification .....	50
Session viewing .....	51
Session deletion.....	52
Session activation .....	53
Session deactivation .....	55
Session errors viewing.....	55
Session list viewing .....	56
Partners list .....	56
Partner addition .....	57
Partner data modification.....	58
Partner deletion .....	59

Partners list viewing .....	60
SSL proxy support .....	60
SSA Client: a step-by-step guide .....	63
Step 1: Learn the server requirements and acquire the rights to use the SSA .....	63
Step 2: Learn the Certification Authority requirements .....	63
Step 3: Create a new SSA configuration file.....	64
Step 4: Generate the RSA private key.....	64
Step 5: Generate the certification request .....	65
Step 6: Leave the new configuration file temporarily .....	65
Step 7: Certify yourself .....	65
Step 8: Open the previously created configuration file.....	66
Step 9: Add the Certification Authority public key to the SSA configuration file.....	66
Step 10: Add your certificate to the configuration file.....	66
Step 11: Add the licence to the configuration file.....	67
Step 12: Create the session .....	67
Step 13: Save the configuration file.....	67
Step 14: Activate the session .....	67
Step 15: Start the secured application and connect to the server .....	68
Step 16: Work with the application .....	68
Step 17: Deactivate the session.....	68
Step 18: Close the configuration file .....	69
Automating the routine operations .....	71
Configuration file autoload and session activation .....	71
Autostart of the application.....	72
Session automatic deactivation .....	72
SSA autoclose .....	72
SSA security.....	73
Theft of the private key .....	73
Multiuser systems .....	74

Fake servers .....	75
The SSA Client in other computer .....	75
Conclusion .....	76
Installation .....	77
Installing the Windows version .....	77
Installing the Unix version.....	77
Problems .....	79
I lost my configuration file passphrase. What can I do? .....	79
Which cipher is the best one? .....	79
The client cannot connect to the server.....	79
The client configuration is incorrect.....	80
The SSA Client configuration is incorrect.....	80
The SSA Server configuration is incorrect .....	81
Securing the Windows with the SSA .....	83
Securing the SMB network file system.....	83
Securing the MS Exchange.....	84
Command line switches .....	85
SSA Client with graphical interface.....	85
SSA configuration file editor .....	85
SSA Client with command-line interface.....	90
Cybernetica Software End User License Agreement.....	91
SSLeay copyright notice .....	95

## Introduction

The fast development of the computer and telecommunication technologies has brought about an explosive growth of the Internet and a massive use of client/server applications. However, this advance has its dark side – the information security concerns. As the Internet protocol stacks have no embedded safeguards, the data traffic can be easily eavesdropped or modified. The users are often not aware of the threats and transmit their sensitive information with no protection through the Internet.

Dedicated software with necessary safeguards for networked environments is often expensive, unavailable, unfriendly, inadequate or not existing at all.

Therefore, the problem needs to be solved how to protect with minimum costs the existing applications that are quite effective for their intended purposes. One has to consider as well the direct costs (the price of the security solution) as the indirect ones (user training, maintenance etc.).

The solution for the problem is the *Secure Sockets Agent (SSA)*.

The SSA is a system for securing the communication between unprotected or inadequately protected applications. The SSA consists of a pair of proxies, one of which works on the client computer and the other on the server computer. The proxies create an encrypted channel between them with all the traffic between the client and the server being tunneled into this channel.

This Guide explains the functioning, installation and configuring of the SSA software package, providing some examples of using the SSA. It would be useful to study the underlying principles of the SSA, as a profound understanding of these helps in learning to use the SSA more extensively and to assure the secure data communication.

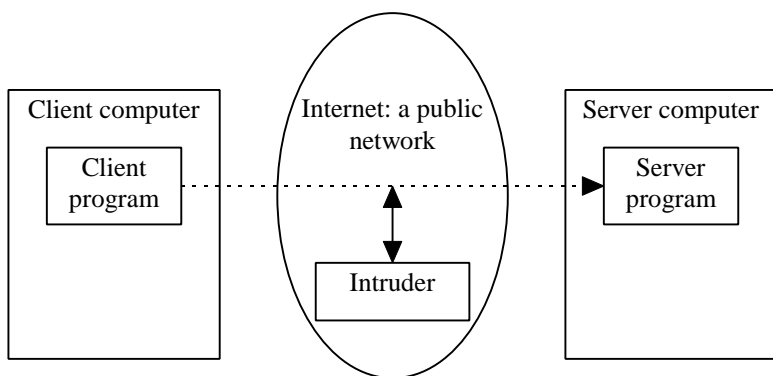




## Operation

SSA is designed to secure the client/server applications based on the TCP/IP protocol. Typically these applications have no protection from unauthorized eavesdropping or data modification.

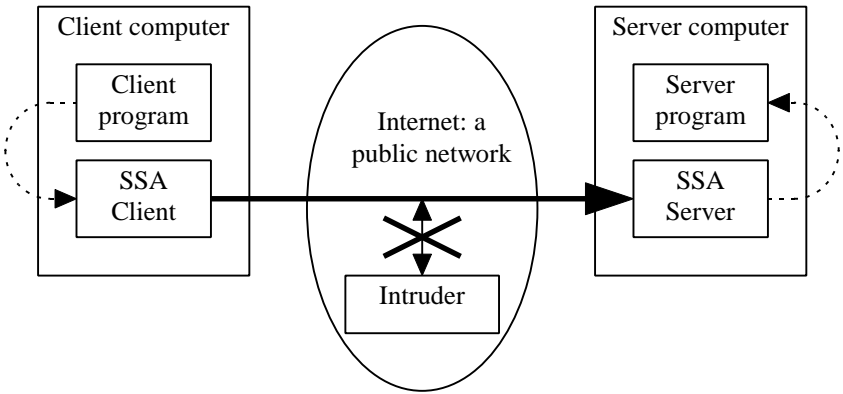
In case of an ordinary client/server application, the client program in the client computer establishes the connection with the server program in the server computer. The traffic through the public network is public, so the intruder can monitor the data exchange between the client and server and/or change the transferred data.



**Figure 1. An unprotected client/server application**

When using the same client/server application with the SSA an extra program is added to the client and server computers. The client starts the SSA Client and the server starts the SSA Server. Now the application to be protected does not directly access the server, but establishes the connection with the SSA Client running in the same client computer. This unprotected connection is made internally in one computer only, so the transferred data never reach the network. When the connection is made, the SSA Client connects to the SSA Server. To secure this connection, the SSL protocol is used providing for data confidentiality and integrity, and for the authentication of the communicating parties. The SSA Server connects to the server program after the secure channel is established and now a secure data exchange can begin between the client and the server.

The method is known as tunneling: in our case SSA tunnels the insecure TCP connection to secure SSL channel (functioning via TCP). As a result, the intruder cannot monitor or change the data flows between the client and the server.



**Figure 2. A client/server application protected with SSA**

As the SSA proxies are self-contained programs, their use does not require any changes in the code of the application: the SSA allows protecting any TCP-based application that uses static ports.

The SSA can also mediate a couple of protocols which do not use any fixed TCP ports; these are the FTP, the Progress database system protocol and SQL\*Net protocol from Oracle. With these protocols, the SSA will listen to the control connection and interpret the commands exchanged between the client and the server, to open a new TCP connection.

## Technology

The SSA is based on the SSL (Secure Sockets Layer) protocol from the Netscape Communications Corporation, worldwide established as de facto standard and massively used for the WWW applications protection. The SSL allows creating a secure communication channel between two applications, assuring confidentiality (by using strong cryptographic algorithms, such as IDEA and 3DES) and integrity (by using message digest algorithms like MD5 and SHA) of the data transferred. The SSL protocol also assures the authenticity of the communicating parties (the client and the server), preventing the unauthorized use of the server resources and preventing the creation of fake servers to get the information from the client. For authentication of the communicating parties the SSL protocol uses digital signatures based on the asymmetric cryptographic algorithms (RSA) and

certificates issued by a trusted third party. The SSA sets no limitations to the lengths of the cryptographic keys.

The SSA supports the SSL versions 2 and 3. The older version contains several significant errors that have been corrected in the new version. The SSLv3 protocol has repeatedly been analyzed and found to be essentially correct. The SSL newer version also allows using the ephemeral Diffie-Hellman algorithm for key exchange, ensuring the perfect forward security.

By default, the client and server are authenticated when using the SSA. The client authentication can be switched off; in this case, a separate configuration file has not to be sent to each client: a single common file is sufficient, containing the session data and the Certification Authority public key for authenticating the SSA Server certificate.

### *User authentication. Certificates*

The SSL uses asymmetric cryptographic algorithms for communication partner authentication.

In asymmetric cryptography, different keys are used for encryption and decryption. Only its owner, i.e. to the person who generated the keypair knows the private key of a keypair. The other key is public, i.e. everybody may know it. The keys are matched and complement each other: a ciphertext created with a public key can only be decrypted with the matching private key and vice versa – a ciphertext created with a private key can only be decrypted with the corresponding public key. It is not possible to derive the private key from a known public key.

These unique properties allow using asymmetric cryptoalgorithms for the authentication of the communication parties. Prior to this the parties must exchange their public keys in such a way that both parties are sure that only the other party has the private key corresponding to the valid public key.

With the SSL protocol, the client generates a random number when establishing the communication session, encodes it with the server public key and transmits the ciphertext to the remote party. The right server only, being a sole owner of the necessary private key, can decrypt the ciphertext. The number generated by the client will later have an important role in the SSL protocol, so the session cannot be continued without the previous action being successful. In case of any fake server not having the right private key the session will be immediately closed. The client authentication is carried out in a similar way.

The public key distribution problem has to be solved in this authentication scheme. The task may seem a simple one, but it is not. A public key is simply a large integer telling nothing about how it has been generated or who owns the matching private key. One public key is no better or worse of the other; the public key becomes a special and important one only with knowledge about the identity of the owner of the matching private key. The link between a private key and its owner is not obvious, one needs some additional information showing that the key  $K$  belongs to the person  $P$ . Moreover, this information must be trustworthy, it must be obtained and kept in a way preventing the use of any false keys.

For a system with two or three users the solution is simple: to get the public key, one can contact the owner personally. Only then can he or she be sure about the ownership of the key. Unfortunately, this scheme is not scalable; it will not work with more users and/or larger geographical area. We need a system allowing without personal contacts to verify a key belonging to a totally strange user or to a person in the other end of the world.

To solve the problem, a third party will be involved with the responsibility to issue certificates proving the ownership of the public keys. The certificate containing user's name and his public key is signed with the digital signature of the third party. This third party is described as the Certification Authority, with the procedure being called certification.

As the certificates are carrying the digital signature of the Certification Authority, they cannot be forged. One can always check the correctness of data in a certificate, using the public key of the Certification Authority. With this scheme, no personal contacts are needed any more; you have only to get the public key of the Certification Authority and now you can securely communicate with all users certified by this Authority.

Using the certificates provides for the scalability of the systems based on the SSL protocol and saves the end users from the necessity of making any critical decisions.

More about certification and cryptography can be found in the Internet, e.g.: <http://www.cyber.ee/>.

## Session reuse

Some protocols make a lot of short-time connections (HTTP in the first place) and may slow down when secured with the SSA. The problem can be solved by reuse of the sessions.

With session reuse, the SSA Client and Server carry out the full authentication procedure only by establishing the first connection. For next connections the private keys will be used which were exchanged by establishing the first connection. Making a new connection with session reuse will be a magnitude faster compared to the option with full authentication.

To secure the reuse, each session must have a certain limited lifetime. When this time expires, the Client and the Server must carry out a new full authentication procedure. At present, the lifetime limit is set to 5 minutes.

Session reuse can be enabled and disabled from Client side as well as from Server side. Sessions will only be reused when enabled from both sides.

### *Distinguished names*

The SSL protocol uses certificates complying to the X.509 standard, where the owner of the certificate and the issuer are identified with so-called distinguished names. A distinguished name is a hierarchical globally unique name defined as a string of relative distinguished names consisting of attribute/value pairs.

The uniqueness of distinguished names is assured by using a hierarchical naming scheme. For example, the four-level name

`/C=UK/O=IOC/OU=IT/CN=John Johnson`

tells us, that its owner works in the United Kingdom (C=UK), in the organization IOC (O=IOC), in the IT department (OU=IT) and his real name is John Johnson (CN=John Johnson). A lot of organizations worldwide can use the acronym IOC (e.g. the International Olympic Committee), but this one is located in the United Kingdom. Thousands of John Johnsons could be found in the world, but only one of them is working in this department of this organization.

The following attributes can be used to create the distinguished names:

<b>Attributes related to the geographic location</b>		
<b>C</b>	Country code	ISO 3166 standard country code. Examples: EE for Estonia, DE for Germany etc.
<b>S</b>	State, province, county, etc.	Examples: Ohio, Sussex, Dalarna
<b>L</b>	Location (city, village etc.)	Examples: Vienna, Bordeaux, Brighton
<b>Attributes related to the organization</b>		
<b>O</b>	Organization name	Examples: Fishfood Ltd., UCLA, Smith & Sons
<b>OU</b>	Department name	Examples: IT, Finances, R&D
<b>Other attributes</b>		
<b>CN</b>	Common name	Name under which the person is known. Can contain titles, abbreviations etc.  Examples: Ann McClaren, Ph. D., Mr. Stanley Higgins
<b>Email</b>	E-mail address	Example: arne@mail.ee

**Figure 3. Attributes in the distinguished names**

For the Certification Authority of an organization the following name format can be used:

`/C=XX/O=Company name/OU=CA`

For server names the following format can be used:

`/C=XX/O=Company name/CN=Name of the service provided by the server`

For user names the following format can be used:

`/C=XX/O=Company name/CN=First name Last name`

## *Strength of the cryptoalgorithms used*

**DES** (with 56-bit key) is the oldest public cryptoalgorithm, which has successfully survived all kinds of attack attempts. There is no effective method known to break DES faster than with brute-force attack (i.e. exhaustive search of the keyspace). DES is considered a little insecure nowadays only due to the fast development of the computer technology in the last decade, making brute-force attacks effective. Therefore, the **3DES** has to be preferred for more rigid security requirements; it is threefold slower, but gives a significantly higher security level. Its formal key length is  $3 \times 56$  bits, making the effective length to be at least 80-90 bits.

**IDEA** (with 128-bit key) can be considered to be secure, as no effective breaking methods are known. Brute-force attacks of IDEA or 3DES will not be successful in the near future.

**RSA** is a thoroughly explored public-key cryptosystem. A great number of top researchers is studying the security of the RSA; for the present, the tools for breaking it are still rather theoretic and can only be effective for short (400-500 bits) keys. RSA with 768-bit or 1024-bit keys can be considered as a completely secure and practically unbreakable system.

The hash functions **MD5** (with 128-bit output) and **SHA** (with 160-bit output) have been explored rather well. By now, some success has only been achieved in breaking of MD5, but existing attack methods are still merely theoretic.

**RC2** and **RC4** are variable key-length ciphers developed by the RSA Data Security. RC2 is a block cipher and RC4 is a stream cipher. These algorithms are proprietary and not very thoroughly analyzed. Therefore, some public and well-explored algorithm like DES or IDEA should be preferred.

The **Ephemeral Diffie-Hellman (EDH) algorithm** is used for the session key exchange. With EDH, the Perfect Forward Security (PFS) property will be ensured. RSA is only used for authentication of the Diffie-Hellman algorithm. It means that even if an attacker succeeds to get the RSA private key, he can decrypt no old sessions. With EDH, compromising the private key will not have any retroactive impact.

Security of the ephemeral Diffie-Hellman key exchange algorithm depends on parameters used. The parameters are public and have not to be concealed. Generating the parameters and checking their suitability is a very complicated and important procedure. The SSA uses a set of parameters defined by the RFC 2412 standard (see RFC 2412 standard "The OAKLEY Key Determination Protocol", appendix E.2 "The Well-Known Groups 2: A 1024 bit prime"). The

long integer in this set is derived from the decimals of pi and its primality has been carefully verified. This number has some favourable properties increasing the effectiveness and security of its use.

The following table lists all the combinations of cryptographic algorithms supported by the SSA. Numbers in parentheses show the length of the key (bits). It is recommended to use the algorithms from the beginning of the list. Weak algorithms with no export restriction can definitely not be recommended.

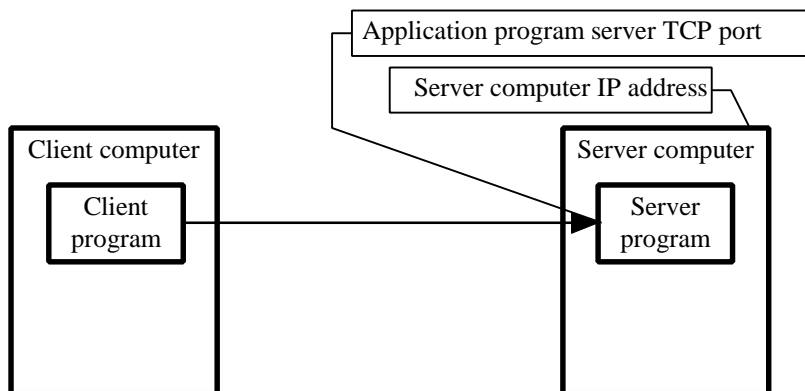
Cipher description	SSLv3	Key exchange	Encryption	MAC	Export
EDH-RSA-DES-CBC3-SHA	✓	EDH	3DES(168)	SHA1	
IDEA-CBC-SHA	✓	RSA	IDEA(128)	SHA1	
DES-CBC3-SHA	✓	RSA	3DES(168)	SHA1	
RC4-SHA	✓	RSA	RC4(128)	SHA1	
RC4-MD5	✓	RSA	RC4(128)	MD5	
EDH-RSA-DES-CBC-SHA	✓	EDH	DES(56)	SHA1	
DES-CBC-SHA	✓	RSA	DES(56)	SHA1	
EXP-EDH-RSA-DES-CBC	✓	EDH(512)	DES(40)	SHA1	✓
EXP-DES-CBC-SHA	✓	RSA(512)	DES(40)	SHA1	✓
EXP-RC4-MD5	✓	RSA(512)	RC4(40)	MD5	✓
EXP-RC2-CBC-MD5	✓	RSA(512)	RC2(40)	MD5	✓
IDEA-CBC-MD5		RSA	IDEA(128)	MD5	
DES-CBC3-MD5		RSA	3DES(168)	MD5	
RC4-MD5		RSA	RC4(128)	MD5	
RC2-CBC-MD5		RSA	RC2(128)	MD5	
RC4-64-MD5		RSA	RC4(64)	MD5	
DES-CBC-MD5		RSA	DES(56)	MD5	
EXP-RC4-MD5		RSA(512)	RC4(40)	MD5	✓
EXP-RC2-CBC-MD5		RSA(512)	RC2(40)	MD5	✓

**Figure 4. Supported ciphers**



## Model of the SSA system

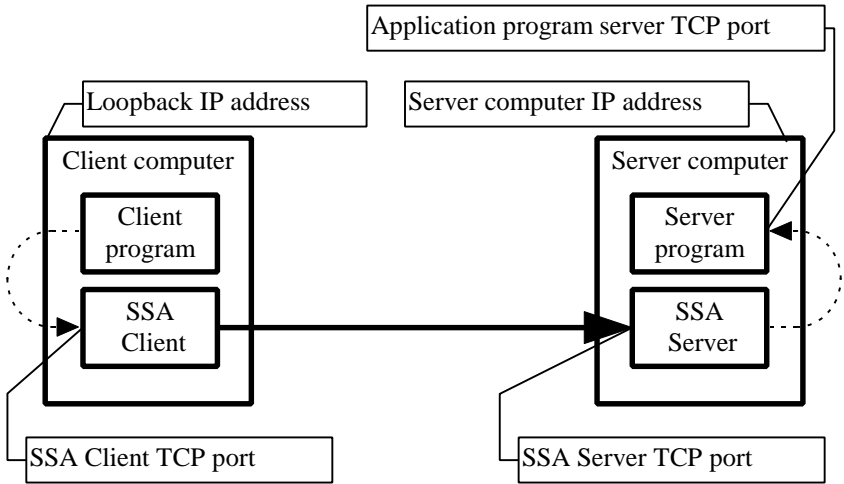
Let's take a look at a non-secured client/server application first, where the client accesses the server directly. Only one program in this system needs to be configured; it is the client program and to configure it, one must know the IP address of the server computer and the TCP port number of the server.



**Figure 5. The model of a non-secured client/server system**

With the SSA, the system will be more complex. Now, three programs need to be configured: the client program, the SSA Client, and the SSA Server. The user manages the first two of them; the server administrator manages the third one.

To configure the client program, one still needs two parameters: the client computer loopback IP address, which is almost always 127.0.0.1 and the TCP port number of the SSA Client, which can be chosen by the user himself.



**Figure 6. The model of the client/server system secured with the SSA**

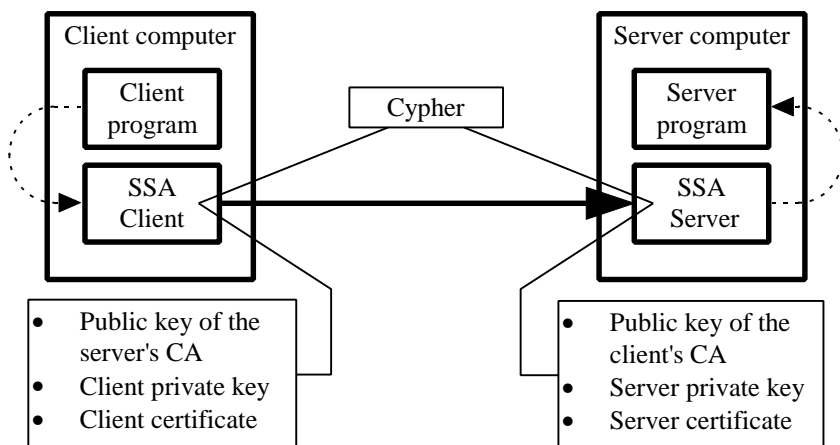
To configure the SSA Client, more data are needed. One must know the SSA Client TCP port, the server computer IP address and the SSA Server TCP port. The client's port number can be chosen by the user, the server computer IP address is, as a rule, set and known from the times when the non-secured application was used; the SSA Server port number is assigned by the server administrator.

To configure the SSA Server, the server administrator must know the SSA server TCP port number, the server computer IP address, and the application program server's TCP port number. The SSA Server port number can be chosen by the administrator; the server computer IP address is, as a rule, fixed and known, and the same holds for the application program server TCP port as well.

In addition to the network addresses and port numbers, the security attributes have to be defined. First of all, the owner of the server has to choose the (-ies) whose services he will use. He has various options: for example, he can use a public certification authority whose security policy is acceptable for the server owner and for the users. In case of so-called "closed applications" having a limited user community, it would be better for the server owner to create his own certification authorities for his clients and servers. In any case, the SSA Client has to be provided with the public key from the certification authority which certified the SSA Server and vice versa – the SSA Server has to be provided with the public key from the certification authority which certifies the SSA Clients.

The system will be more secure when the Clients and the Server are certified by different certification authorities.

The public key of the certification authority has a form of a self-signed X.509 certificate. Note that this key will be used for the final authentication of the client and the server, therefore the authenticity of this key must be verified through independent channels (e.g. by taking a phone call to the partner etc.) before it can be put in use.



**Figure 7. Security attributes in the SSA system**

The owner of the server also determines the cipher to be used for the creation of the secure channel.

Finally, the client and the server shall generate a RSA keypair for them and certificate them in the chosen Certification Authority.

The table below makes a summary of the parameters needed for the SSA system configuration, grouped by the points of application:

Parameter	Source	Point of use
Loopback IP address	Fixed. 127.0.0.1	Application program client
SSA Client TCP port	User-defined	Application program client, SSA Client
Client private key	User-generated	SSA Client
Client certificate	Certification Authority	SSA Client
Server computer IP address	Server administrator	SSA Client, server
SSA Server TCP port	Server administrator	SSA Client, server
Public key of the client's CA	Client's Certification Authority	SSA Server
Public key of the server's CA	Server's Certification Authority	SSA Client
Cryptoalgorithm	Server administrator	SSA Client, Server
Application program server TCP port	Fixed or variable	SSA Server
Server private key	Server administrator	SSA Server
Server certificate	Certification Authority	SSA Server

**Figure 8. The SSA system parameters**

## Session

The notion of session is central in the whole SSA system. The session is a set of parameters describing the secure communication channel created with the SSA. All the parameters together explained in the previous subdivision describe one session.

The session parameters are twofold: SSA Client parameters and SSA Server parameters. We talk about Client sessions and Server sessions correspondingly.

The Client session consists of SSA Client TCP port, server computer IP address, SSA server TCP port, client private key, client certificate, public key of the server's CA, and the cipher.

The Server session consists of SSA Server TCP port, server computer IP address, application program server TCP port, server private key, server certificate, public key of the client's CA, and the cipher.

Session parameters are stored in the SSA configuration file. To prevent modification and copying, the file is encrypted using symmetric cryptoalgorithm with the key derived from the passphrase entered by the user.

## Licencing

The SSA software is being freely distributed, but to use it legally, the customer has to sign an agreement with the Cybernetica giving him the right to use the SSA software under conditions stipulated in the agreement. By signing the agreement, the customer will also get from the Cybernetica the licencing information file(s) containing data about the licensee, the licenced product, and the conditions of its use.

```
License owner:      AS Wonderbar
License number:     1234567
Product:            SSA Server
License type:       Commercial
Number of licenses: 1
Number of clients:  unlimited
```

### Figure 9. The data stored in the licencing information file

This file is secured with Cybernetica's digital signature, so the data in the file cannot be modified. To use the licencing information file, it has to be loaded into the configuration file of the corresponding SSA program. During the loading, the digital signature of the file and the pertinence of the licence for the given program will be checked. Each SSA program requires a pertinent licence: the SSA Server refuses to use a SSA Client licence and vice versa.

The licence will also be checked by the SSA Client and Server before the session activation. If the licence is missing, invalid, or impertinent, a corresponding warning will be recorded in the log file, but the procedure will proceed. If the licence is OK, the data from the licence will be stored in the log file.

Two types of licences exist for the SSA Server:

- the licence for using the SSA Server to secure a public server (*SSA Public Server License*). This licence does not allow the SSA Server to authenticate the clients.
- the licence for using the SSA Server to secure applications having a limited users community. This licence allows the SSA Server to authenticate the clients. The licence gives rights for servicing a preset number of clients. If the preset limit is violated, the SSA Server will log a corresponding warning, but will proceed.

## **SSA system components**

The SSA system is composed from several programs: the SSA Client, the SSA Service Manager, the SSA Server Service, and the SSA User Manager.

The SSA Client runs in the computer of the secured service user and constitutes the client end of the secure tunnel. The session parameters needed to create the secure tunnel are stored in the client configuration file that the user gets from the server administrator or creates himself. A version of the SSA Client with command-line interface is available for all supported platforms. Additionally, a version with graphical user interface is available for the Windows 95 and Windows NT environments. The Client programs come with two different functionalities, as full-fledged version and a light version. The full-featured Client allows the user to create and modify configuration files, to generate keys and certification requests, and to secure the application programs. The Lite Client allows using for application program protection the configuration files prepared by the server administrator using the SSA User Manager.

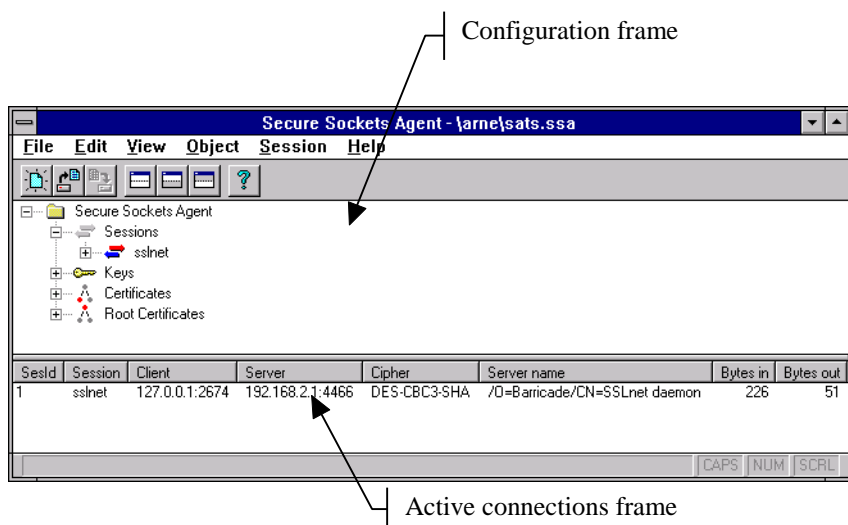
The SSA Service Manager runs in the computer of the server administrator and allows managing the SSA Server Service programs. The SSA Service Manager runs in the Windows NT environment only and can only be used to manage the SSA server services run under the Windows NT. The services can be run in the administrator's computer or some other Windows NT computer; so, several server computers can be managed from one workplace.

The SSA Server Service runs in the application program server computer and constitutes the server end of the secure tunnel. The session parameters needed to create the secure tunnel are stored in the server configuration file which the server administrator creates using the SSA Service Manager. The SSA Server Service can be run in the Windows NT or Unix environment.

The SSA User Manager allows the server administrator to create configuration files for the SSA Lite Client and/or to issue certificates. The SSA User Manager runs in the Windows NT environment.


### ***Graphical user interface***

Graphical user interface is available for the SSA Client Windows version and the Service Manager. The graphical user interface gives a significantly more convenient and clear way to configure and use the SSA.



**Figure 10. The graphical user interface layout**

The main window of the SSA is divided to two parts: the upper frame shows the content of the active configuration file, while the lower one shows information relating to the active connections. To make more space for one of the frames, the user can hide the other one. The configuration frame is equipped with a context-sensitive pop-up menu; a right-click on the wanted item of the configuration tree brings it on the screen top. Information about the window location and appearance will be stored in a register, so its previous shape will be restored on the next start of the program.

The guidelines concerning the graphical interface are marked with .

## Command-line interface

The SSA Client Unix version consists of two programs with command-line interfaces. The SSA Client (`ssaclient`) runs in the client computer and constitutes the client end of the secure tunnel. The session parameters needed for creating the secure channel are stored in the client configuration file created by the user by means of the SSA configuration file editor (`ssa`). The command-line interfaced client programs are also available for the Windows platform (`ssaclien.exe` and `cssa.exe`).



The command 'ssa -h' gives a full overview of the configuration file editor commands.

Secure Socket Agent configuration file editor version 1.2.5  
 Copyright (c) 1998 Cybernetica. All rights reserved  
 For more information refer <http://www.cyber.ee/>

```
=====
ssa [-f <conffile>] -y -- replace password
ssa [-f <conffile>] -key (-list|-delete|-gen|-Req) ...
ssa [-f <conffile>] (-listall|-viewall) ...
ssa [-f <conffile>] (-cert|-root|-session|-license)
                        (-list|-delete|-insert|-view) ...
ssa [-f <conffile>] -partner (-insert|-delete|-list) ...
```

Optional parameters:

```
=====
-N <session name>
-I <certificate file>
-O <certificate request file>
-K <keysize> (for -key -gen only)
-S <randomfile> (for -key -gen only)
-D <distinguished name of the subject> (for -key -Req only)
-L <local listen port> (for -session -insert only)
-H <server host> (for -session -insert only)
-P <server port> (for -session -insert only)
-C <cipher> (for -session -insert only)
-A <session key> (for -session -insert only)
-B <session certificate> (for -session -insert only)
-X <session root certificate> (for -session -insert only)
-M serve multiple connections (for -session -insert only)
-E <command to execute> (for -session -insert only)
-o <server login type> (for -session -insert only)
-e cache sessions (for -session -insert only)
-u <inactivity timeout> (for -session -insert only)
-J <partner name> (for -partner only)
-U <mapped name> (for -partner -insert only)
-b <partner IP address> (for -partner -insert only)
-q <partner netmask> (for -partner -insert only)
-m -- minimize on activation (for -session -insert only)
-a -- autoclose on deactivation (for -session -insert only)
-T <accept timeout> (for -session -insert only)
-j -- no TCP delay (for -session -insert only)
-x -- no client authentication (for -session -insert only)
-Z <ssl mode> (for -session -insert only)
-G <bindaddr> (for -session -insert only)
-Q <licensename> (for -session -insert only)
```


**Figure 11. The SSA configuration file editor command-line interface commands summary**

The command 'ssaclient -h' gives a full overview of the SSA client commands.

```
Secure Socket Agent client forwarder version 1.2.5
Copyright (c) 1998 Cybernetica. All rights reserved
For more information refer http://www.cyber.ee/
=====
ssaclient [-p <password>] [-d] -f <conffile> <session name>
-d -- daemon mode
```

```
In order to use HTTPS proxy set the environment variables:
SSA_PROXY -- name or IP address of the proxy
SSA_PROXYPORT -- TCP port of the HTTPS proxy
SSA_DIRECT -- list of the directly accessible servers
               (separator is ;)
```

### Figure 12. The SSA client command-line interface commands summary

The guidelines for the command-line interface are marked with .

## SSA Client operations

This chapter describes the full set of operations with the SSA Client. Examples are given for the graphical as well as the command-line interface.

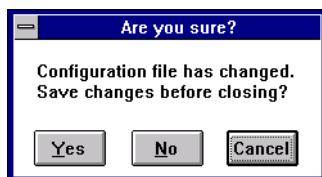
### **SSA configuration file**

The data needed for creating the secure channel are stored in the SSA configuration file. To achieve confidentiality and integrity, the file is provided with check codes and encrypted. These safeguards assure the secrecy of the server private key (preventing unauthorized access to the server and creation of a fake server), the authenticity of the Certification Authority public key (needed for partner authentication), and the integrity of session parameters (preventing unintended use of weak keys). The key for the file encryption is derived from the passphrase entered by the administrator. It should be noted that the effectiveness of the protection directly depends on the passphrase length. For example: about 6 bits of key information can be derived from a passphrase consisting of capitals, small letters and digits. So, the passphrase should be made up of at least 11 characters to give adequate protection. The SSA does not allow using passwords shorter than 8 characters. The passphrases should be changed regularly.

The operations with the configuration file strongly depend on the interface type of the configuration program. The command-line interface can only carry out one operation at a time with the file. With this version, one cannot talk about separate operations for opening or closing the file, as the program always would open a configuration file, perform the operation and, where appropriate, close the file.

The version with the graphical user interface allows carrying out unlimited sequence of operations with the file. Separate open and close operations are available in this case. On opening, the configuration file will be read into the memory where it can be freely manipulated. After performing the necessary operations the file can be stored again (under the same or a new name)


When a file is opened containing any unsaved changes and you want to close the file without previously saving it, the SSA displays a corresponding warning message:



**Figure 13. Warning about unsaved changes**

Now you can choose to save the file, to close it without saving, or to cancel the operation.


### *Creating the configuration file*

 In the *File* menu select *New...* The dialog box for entering a new passphrase appears.




Choose a passphrase and enter it in the dialog box. Click *OK*. A new empty configuration file is created and displayed in the configuration frame of the SSA main window.

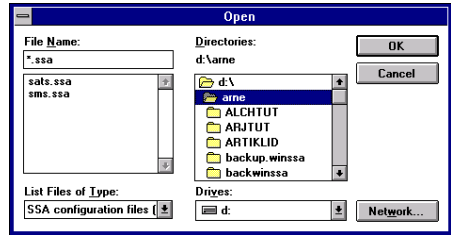


 There is no explicit command for creation of a new configuration file. If the user carries out a valid operation with a non-existing configuration file, the program will ask the passphrase for the new file and the new file will be created. For example, the command `'ssa -f new.ssa -key -generate -N new'` creates the configuration file `'new.ssa'` containing the newly generated key named `'new'`.

## Opening the configuration file

 In the *File* menu select *Open...* The dialog box for opening the configuration file appears.

Note: Depending on the operating system, the appearance of the configuration file selection dialog box can be a little different.




Choose the configuration file you need and click *OK*. The dialog box for entering the passphrase appears.





Enter the passphrase and click *OK*. The loaded configuration file is displayed in the configuration frame of the SSA main window.




 There is no explicit command for opening the configuration file. By default the SSA uses the `‘.ssarc’` file in the user's home directory (defined by environment variable `HOME`). Other files can be selected using the `‘-f’` switch. For example: the command `‘ssa -f other.ssa -session -list’` lists the sessions described in the `‘other.ssa’` configuration file.

## Saving the configuration file

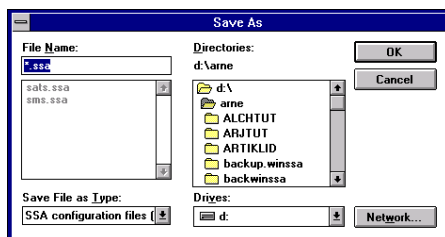
 In the *File* menu select *Save*. The configuration will be saved in the current file which name is shown in the title bar of the window.

 There is no explicit command for saving the configuration file. The SSA saves the file when it has been changed by an operation.


## Saving the configuration file under another name

 In the *File* menu select *Save as...* The dialog box for saving the configuration file appears.


Note: Depending on the operating system, the appearance of the configuration file saving dialog box can be a little different.




Enter a name for the configuration file and click *OK*.


 There is no explicit command for renaming the configuration file. The file can be renamed using the operating system options.

## Closing the configuration file

 In the *File* menu select *Close*. The configuration file will be closed and all sensitive data will be removed from the memory.


 There is no explicit command for closing the configuration file. On completion the SSA automatically closes the configuration file and removes sensitive data from the memory.

## Changing the passphrase of the configuration file

 In the *File* menu select *Change password...* The dialog box for changing the passphrase appears.




Choose a new passphrase and enter it into the dialog box. Click *OK*.

 The passphrase can be changed using the `'-y'` command. The current passphrase will be asked first, and then the new one.

```
kiku:~/ssa>ssa -f file.ssa -y
Enter password for "file.ssa": *****
Enter new password for "file.ssa": *****
kiku:~/ssa>
```

**Figure 14.** Changing the passphrase of the SSA configuration file

## Viewing the contents of the configuration file

 To display the contents of the configuration file use the '-viewall' command.

```
kiku:~/ssa>ssa -f file.ssa -viewall
Enter password for "file.ssa":
Keys:
-----
alpha

User Certificates:
-----
alpha
=====
Certificate:
  Data:
    Version: 1 (0x0)
    Serial Number: 0 (0x0)
    Signature Algorithm: md5WithRSAEncryption

.
.
.

Command:
Map login name :      ftp


Session status:      ok
=====

Session partners:
=====

kiku:~/ssa>
```

**Figure 15. Viewing the contents of the configuration file**

## Viewing the object list in the configuration file

 To view the object list in the configuration file use the '-listall' command.

```
kiku:~/ssa>ssa -f file.ssa -listall
Enter password for "file.ssa":
Keys:
-----
alpha

User Certificates:
-----
alpha

Root Certificates:
-----
serveca

Licenses:
-----
123456

Sessions:
-----
server


kiku:~/ssa>
```

**Figure 16. Viewing the object list in the configuration file**

### ***RSA key***

The RSA algorithm is used for client and server authentication and for session key exchange. Therefore, the security of the whole channel depends on the length of the RSA key. The complexity of breaking the RSA key grows exponentially, i.e. every additional bit doubles the complexity. By now (after many years of cooperative work), attempts to break the 400-bit RSA key have come to success. So, a shorter than 768-bit RSA key is not recommendable any more.

### ***Generating the RSA key***

 In the *New* submenu of the *Object* menu select *Key...* The dialog box for generating a new key appears.






Enter a name and length for the new key. The recommendable length is 768 bit as minimum. Press *OK*. The SSA begins to generate the key. The process can last rather long, depending on the length of the key and the performance of the computer.



The new key adds to the *Keys* section of the configuration frame.




 A new key can be generated with the '-key -gen' command. The command has three parameters:

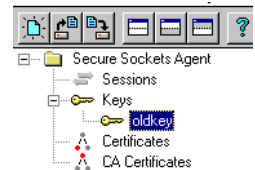
- -N <keyname> – a mandatory parameter defining the name for the key in the configuration file.
- -K <keysize> – an optional parameter defining the length of the new key in bits; the default length is 512 bits.
- -S <randomfile> – an optional parameter pointing the file whose contents will be hashed to the random number generator seed before the private key generating.

```
kiku:~/ssa>ssa -f file.ssa -key -gen -N newkey -K 1024
Enter password for "file.ssa":
kiku:~/ssa>
```

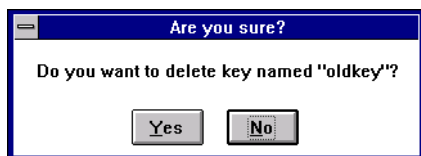
**Figure 17. Generating a new key**

### *Deleting the RSA key*

 In the *Keys* section of the configuration frame select the key to be deleted.




In the *Object* menu select *Delete...*. The key deletion confirmation dialog box appears. Click *Yes*.



The selected key will be deleted from the *Keys* section of the configuration frame.




 A key can be deleted with the command `-key -delete`. This command has a single parameter:

- `-N <keyname>` – defines in the configuration file the name of the key to be deleted.

```
kiku:~/ssa>ssa -f file.ssa -key -delete -N oldkey
Enter password for "file.ssa":
kiku:~/ssa>
```

**Figure 18. Key deletion**

## Viewing the RSA keys

 The command `-key -list` lists the RSA keys.

```
kiku:~/ssa>ssa -f file.ssa -key -list
Enter password for file.ssa":
key1
key2
kiku:~/ssa>
```

**Figure 19. Viewing the keys**

## The certificate


A certificate is a data structure confirming the connection between a public key and a subject. Certificates are secured with the Certification Authority digital signature, so the data contents of the certificate can be verified with the public key of the Certification Authority, which issued the certificate. As the certificates

are protected from forging and as they contain public information only, they can be distributed through public communication channels.

In the SSA both client and server have their certificates which they get from a Certification Authority. To get a certificate, the user has to proceed with the following steps:

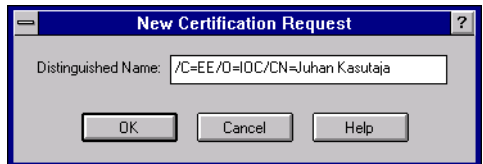
- Creating a certification request – the user creates a certification request containing his public key and proposed distinguished name; the details of the distinguished name have to be agreed with the representative of the Certification Authority prior to the creation of the request.
- Certification – the user takes the created certification request to the Certification Authority and (after the necessary formalities) gets the certificate.
- Adding the certificate to the configuration file.

### Requesting the certificate

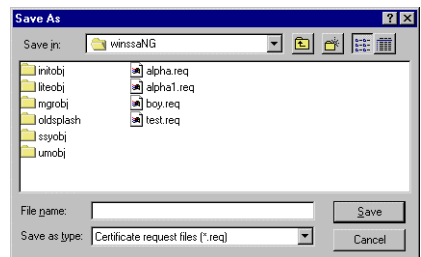
 In the configuration frame *Keys* section select the key you want to certify.




In the *New* submenu of the *Object* menu select *Certificate request...* Enter the proposed distinguished name and click *OK*.



The dialog for storing the certification request will appear. Specify a file for storing the certification request and click *Save*.




 A certification request can be created with the ‘-key -Req’ command. This command has three parameters:

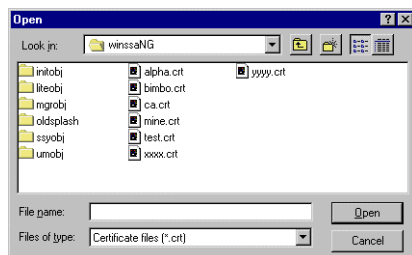
- -N <keyname> – identifies in the configuration file the name of the key whose certification is requested.
- -O <filename> – identifies the file for saving the certification request.
- -D <distinguished name> – defines the distinguished name of the user (if the name includes spaces or special characters, it must be in single or double quotes).

```
kiku:~/ssa>ssa -f file.ssa -key -Req -N newkey -O cert.req
-D '/C=EE/O=IOC/CN=Juhan Kasutaja'
Enter password for "file.ssa":
kiku:~/ssa>
```

**Figure 20. Creating the certification request**

## Adding certificates to the configuration file

 In the *New* submenu of the *Object* menu select *Certificate...* The dialog for selection of a new certificate file will appear. Select a certificate file and click *Open*.




The dialog for naming the certificate will appear. Type the certificate name and click *OK*.



The new certificate adds to the configuration frame *Certificates* section.




 A certificate can be added with the ‘-cert -insert’ command. This command has two parameters:

- `-N <certname>` – identifies the name of the new certificate in the configuration file.
- `-I <filename>` – identifies the file from which the certificate will be read.

```
kiku:~/ssa>ssa -f file.ssa -cert -insert -N newcert -I cert.crt
Enter password for "file.ssa":
kiku:~/ssa>
```

**Figure 21. Adding a certificate to the configuration file**

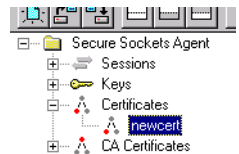
### Certificate deletion


 In the configuration frame *Certificates* section select the certificate to be deleted.



In the *Object* menu select *Delete...* The certificate deletion confirmation dialog box appears. Click *Yes*.

The selected certificate will be deleted from the configuration frame *Certificates* section.




 A certificate can be deleted with the '`-cert -delete`' command. This command has a single parameter:

- `-N <certname>` – identifies in the configuration file the name of the certificate to be deleted.

```
kiku:~/ssa>ssa -f file.ssa -cert -delete -N oldcert
Enter password for "file.ssa":
kiku:~/ssa>
```

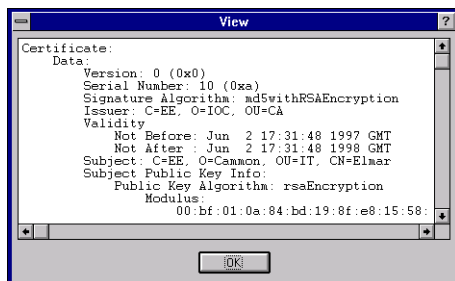
**Figure 22. Deleting a certificate**


## Viewing a certificate

 In the configuration frame *Certificates* section select the certificate you want to view.



In the *Object* menu select *View...*  
The dialog box shows the content of the selected certificate. To close the dialog box click *OK*.



 The '`-cert -view`' command allows to view a certificate. This command has a single parameter:

- `-N <certname>` – identifies in the configuration file the name of the certificate to be shown.

```

kiku:~/ssa>ssa -f file.ssa -cert -view -N oldcert
Enter password for "file.ssa":
Certificate:
  Data:
    Version: 0 (0x0)
    Serial Number: 7 (0x7)
    Signature Algorithm: md5withRSAEncryption
    Issuer: C=EE, O=IOC, OU=IT
    Validity
      Not Before: Feb 27 13:57:10 1997 GMT
      Not After : Feb 27 13:57:10 1998 GMT
    Subject: C=EE, O=IOC, CN=Klient
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
        Modulus:
          00:93:87:67:c9:29:a0:58:1f:70:8b:9a:56:a0:d4:
          31:49:ee:27:0a:3b:ca:1c:72:db:7a:59:ce:ae:82:
          a9:3f:16:73:31:14:13:e4:a7:4a:cc:65:fd:dc:5e:
          94:80:65:32:25:61:17:20:07:f2:22:13:78:5b:cb:
          54:51:66:03:87:22:f6:34:bf:47:f5:b1:f5:d8:83:
          ee:60:d9:09:84:ce:c9:7f:76:12:7f:04:76:11:71:
          42:8e:14:16:66:c7:8e:58:67:b0:36:d1:2d:0f:0c:
          5a:3e:5e:e8:a5:01:ee:b5:88:c6:dc:54:06:cf:32:
          4b:21:30:07:b9:e2:47:f8:65
        Exponent: 65537 (0x10001)
      Signature Algorithm: md5withRSAEncryption
        16:51:4f:76:dd:a6:71:c6:02:12:26:26:f6:a5:5c:0c:6d:c5:
        ee:76:40:46:95:e2:8b:b9:44:ce:76:65:65:06:6b:57:3a:ff:
        a4:29:87:74:ab:b4:0d:76:3f:8b:32:67:6e:95:1f:c0:f6:6b:
        b6:00:59:db:91:0c:fd:20:31:21
kiku:~/ssa>

```

**Figure 23. Viewing a certificate**

### *Viewing the certificate list*

 The ‘-cert -list’ command allows to view the certificate list.

```

kiku:~/ssa>ssa -f file.ssa -cert -list
Enter password for "file.ssa":
cert1
cert2
kiku:~/ssa>

```

**Figure 24. Viewing the certificate list**

## ***Certification Authority public key***


With the public key of the Certification Authority the client and server check the authenticity of each other's public keys. Therefore, it is essential to have the right

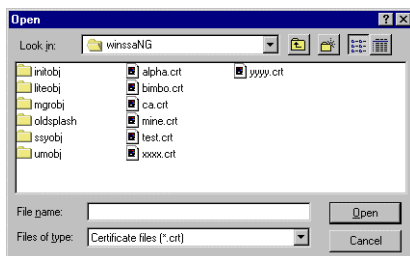
and valid public key of the Certification Authority. Otherwise the partner authentication and all the protection with the SSA based on it will not be reliable.

The best way to acquire the Certification Authority public key is to get it directly from the Certification Authority. The keys acquired through the network can in no case be trusted without additional external-to-system check. As the Certification Authority public key is typically distributed in the form of the certificate, one may have a wrong impression of the data in the certificate being seemingly protected, as they are in case of client's certificate. Actually, a great difference exists between the certificates of the user and the Certification Authority: the Certification Authority certificate is a self-signed one, not allowing to automatically check the authenticity of the data in it.

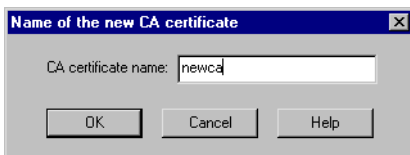
To sum up: before adding a Certification Authority public key to the configuration file one should be sure about authenticity of the key.

### *Adding a Certification Authority public key to the configuration file*

 In the *New* submenu of the *Object* menu select *CA certificate...* The dialog for selecting a new certificate file will appear. Select a certificate file and click *Open*.




The dialog for entering the name of the certification authority public key will appear. Type the name and click *OK*.



The new public key adds to the configuration frame *CA Certificates* section.



 A Certification Authority public key can be added with the '`-root -insert`' command. This command has two parameters:




- `-N <rootname>` – identifies the new Certification Authority public key name in the configuration file.
- `-I <filename>` – identifies the file from which the Certification Authority public key will be read.

```
kiku:~/ssa>ssa -f file.ssa -root -insert -N ca -I cacert.crt
Enter password for "file.ssa":
kiku:~/ssa>
```

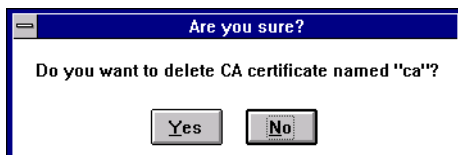
**Figure 25. Adding a Certification Authority public key to the configuration file**

### *Deleting the Certification Authority public key*

 In the configuration frame *CA Certificates* section select the Certification Authority public key to be deleted.




In the *Object* menu select *Delete...* The Certification Authority public key deletion confirmation dialog box appears. Click *Yes*.



The selected Certification Authority public key will be deleted from the configuration frame *CA Certificates* section.




 A Certification Authority public key can be deleted with the '`-root -delete`' command. This command has a single parameter:

- `-N <rootname>` – identifies in the configuration file the name of the public key to be deleted.

```
kiku:~/ssa>ssa -f file.ssa -root -delete -N ca
Enter password for "file.ssa":
kiku:~/ssa>
```

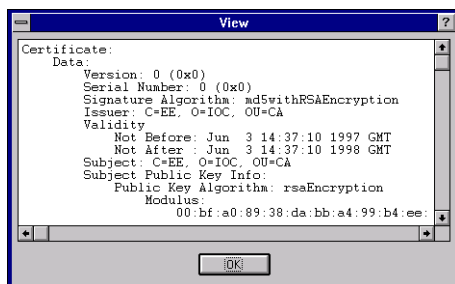
**Figure 26. Deleting a Certification Authority public key**

## Viewing the Certification Authority public key


 In the *CA Certificates* section of the configuration frame select the Certification Authority public key for viewing.



In the *Object* menu select *View...*  
The certificate dialog box shows the content of the selected public key.



Click *OK* to close the dialog box.

 The '-root -view' command allows to view the Certification Authority public key. This command has a single parameter:

- -N <rootname> – identifies in the configuration file the name of the Certification Authority public key to be shown.


```

kiku:~/ssa>ssa -f file.ssa -root -view -N ca
Enter password for "file.ssa":
Certificate:
  Data:
    Version: 0 (0x0)
    Serial Number: 0 (0x0)
    Signature Algorithm: md5withRSAEncryption
    Issuer: C=EE, O=IOC, OU=IT
    Validity
      Not Before: Feb 27 13:57:10 1997 GMT
      Not After : Feb 27 13:57:10 1998 GMT
    Subject: C=EE, O=IOC, OU=IT
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
        Modulus:
          54:51:66:03:87:22:f6:34:bf:47:f5:b1:f5:d8:83:
          ee:60:d9:09:84:ce:c9:7f:76:12:7f:04:76:11:71:
          42:8e:14:16:66:c7:8e:58:67:b0:36:d1:2d:0f:0c:
          5a:3e:5e:e8:a5:01:ee:b5:88:c6:dc:54:06:cf:32:
          4b:21:30:07:b9:e2:47:f8:65
        Exponent: 65537 (0x10001)
      Signature Algorithm: md5withRSAEncryption
        16:51:4f:76:dd:a6:71:c6:02:12:26:26:f6:a5:5c:0c:6d:c5:
        ee:76:40:46:95:e2:8b:b9:44:ce:76:65:65:06:6b:57:3a:ff:
        a4:29:87:74:ab:b4:0d:76:3f:8b:32:67:6e:95:1f:c0:f6:6b:
        b6:00:59:db:91:0c:fd:20:31:21
kiku:~/ssa>

```

**Figure 27. Viewing the Certification Authority public key**

### *Viewing the list of Certification Authorities' public keys*

 The '-root -list' command allows to view the list of Certification Authorities' public keys.

```

kiku:~/ssa>ssa -f file.ssa -root -list
Enter password for "file.ssa":
root1
root2
kiku:~/ssa>

```


**Figure 28. Viewing the list of Certification Authorities' public keys**

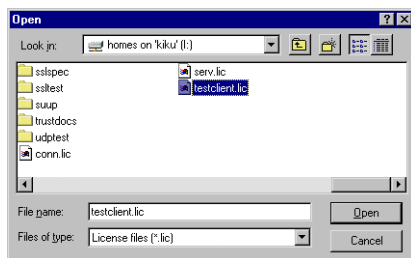
## ***Licensing information***

The licensing information file contains data about the licensee, the licensed product, and the conditions of its use. The customer gets the file from the Cybernetica after signing the agreement giving him the rights to use the SSA. This file is secured with Cybernetica's digital signature, so the data in the file

cannot be modified. To use the licensing information file, it has to be loaded into the configuration file of the corresponding SSA program. During the loading, the digital signature of the file and the pertinence of the license for the given program will be checked. Each SSA program requires a pertinent license: the SSA Server refuses to use a SSA Client license and vice versa.


## Adding the licensing information to the configuration file

 In the *New* submenu of the *Object* menu select *License...* The dialog for selecting the licencing information file will appear. Select the file you want and click *Open*.



In the *Licenses* section of the configuration frame the new license will be included.




 To add the licensing information use the '`-license -insert`' command. The command has the single parameter:

- `-I <filename>` – specifies the file from which the license will be read.

```
kiku:~/ssa>ssa -f file.ssa -license -insert -I license.lic
Enter password for "file.ssa":
kiku:~/ssa>
```

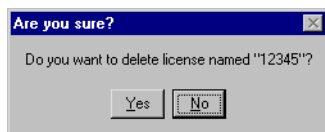
**Figure 29. Adding the licensing information to the configuration file**

## Deletion of licensing information

 In the *Licenses* section of the configuration frame select the license to be deleted.




In the *Object* menu select *Delete...* The dialog appears asking confirmation to the license deletion. Click *Yes*.



In the *Licenses* section of the configuration frame the selected license will be deleted.




 To delete a license use the ‘-license -delete’ command. The command has the single parameter:

- -N <licensename> – specifies the name of the license to be deleted in the configuration file.

```
kiku:~/ssa>ssa -f file.ssa -license -delete -N 12345
Enter password for "file.ssa":
kiku:~/ssa>
```

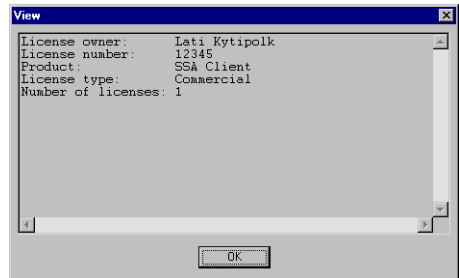
### Figure 30. Deletion of licensing information

#### Viewing the licensing information


 In the *Licenses* section of the configuration frame select the license you want to view.



In the *Object* menu select *View...*  
The data of the selected license will appear on the screen.



To close the dialog click *OK*.


 To view a licence use the ‘-license -view’ command. The command has the single parameter:

- -N <licensename> – specifies for viewing the name of the license in the configuration file.

```
kiku:~/ssa>ssa -f file.ssa -license -view -N 456
Enter password for "file.ssa":
License owner:      serveriomaniak
License number:     123
Product:            SSA Server
License type:       Commercial
Number of licenses: 1
Number of clients:  unlimited
kiku:~/ssa>
```

**Figure 31. Viewing the license information**

### *Viewing the list of licenses*

 To view the list of licenses, use the ‘-license -list’ command.

```
kiku:~/ssa>ssa -f file.ssa -license -list
Enter password for "file.ssa":
123456
kiku:~/ssa>
```

**Figure 32. Viewing the list of licenses**

## **Session**

A session is a set of parameters describing a secure channel.

Parameter	Comment
Session name	Session name in the configuration file
Listen on	The network interface address where the SSA Client waits for the connection from the application client. By default, the Client will wait the connections on the loopback only (127.0.0.1). If you want to use the SSA Client as a gateway for the other computers, set 0.0.0.0 for the network interface address.
Local port	TCP port in the client computer where the SSA Client waits for connection from the application client. Can be chosen freely by the user.
Server host	The host name or IP address of the SSA server computer. The server administrator gives this parameter.

Server port	TCP port in the server computer where the SSA Server waits for connection from the SSA Client. The server administrator gives this parameter.
License	License, under which the SSA Client runs in mediating the given session. The license information must be previously acquired by the system administrator and loaded to the configuration file. Only a license stored in the configuration file can be selected.
Client key	The key has to be generated previously by the user himself. Only the keys available in the configuration file can be selected.
Client certificate	The certificate has to be acquired previously by the user and stored to the configuration file. Only the certificates available in the configuration file can be selected.

Public key of the server's Certification Authority	The CA certificate has to be acquired previously by the user and stored to the configuration file. Only the CA certificates available in the configuration file can be selected.
Cipher	The cipher used to secure the channel. The parameter has to be defined by the server administrator.
The protocol used	Specifies the protocol for creating the channel: SSLv2 or SSLv3. The SSLv3 protocol is recommendable. The server administrator sets this parameter.
Application protocol	Specifies the application protocol used in the TCP connection transferred by the SSA. Significant only where the SSA has to interpret the communication between the client and server for correct mediating of the connection. The SSA recognizes five application protocols at the moment: FTP, HTTP, Progress, MS SQL and SQL*Net.
Command to execute	The SSA graphical client can after the establishing a session automatically start the secured application. For example, when the SSA is used to secure the telnet, the SSA Client listens the TCP port 3333, and the command is 'telnet localhost 3333', then by establishing the session the telnet client will be automatically started and will make a connection to the SSA client without user intervention. This option simplifies the execution of routine procedures.
Secondary timeout	For transferring complicated protocols using multiple TCP connections (e.g. FTP), the SSA has to create temporary sockets. This timeout defines the period during which the corresponding socket will wait for the data connection from the client or server before abandoning the transfer of the connection. The value is in seconds. If the timeout is set to zero, the socket will wait for the connection infinitely. A reasonable timeout would be two or three minutes.




Minimize/Restore	The SSA Client minimizes itself after establishing a session. This extra convenience removes the often unnecessary SSA main window from the screen.
Close on deactivation	SSA Client exits after the deactivation of the session. For example, when the SSA is used to secure the Telnet and this option is enabled, the SSA will automatically exit after the disconnection of the telnet connection.
Cache sessions	Enables/disables session reuse. The reuse will be enabled, when both the client and server select this option.
TCP no delay	To more effectively use the network, the TCP protocol stack tries to send IP packets of optimal length to the network. If a couple of bytes only are written to the open socket, the operating system kernel will send these data after a 0.1-0.2 second delay rather than immediately. If no data add in this period, the waiting data will be sent as a small IP packet. If more data were written in the period, all the data will be assembled in a single large IP packet. Some applications (especially the Progress database system) will run very slow due to this delay. The given option allows switching off the delay, so the data written to the socket will be sent to the network immediately.
No client authentication	Allows abandoning the client authentication. Can be used only, when this option is set on the server side also. Where the client will not be authenticated, the client can have no key or certificate.

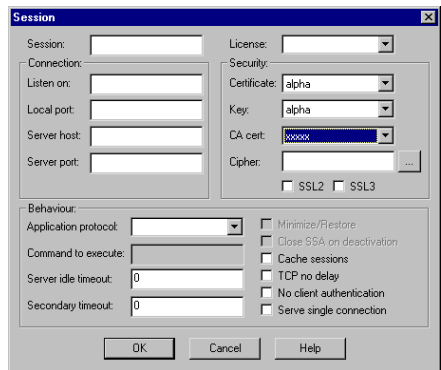
Serve single connection	The SSA Client can be run in two modes. In the first mode the Client serves one connection only and deactivates. This mode is suitable for securing e.g. Telnet, POP3, or other applications using one TCP connection at a time. In the second mode the Client serves multiple TCP connections without deactivation. This mode is suitable for securing e.g. WWW, SQL Server, or other applications using multiple TCP connections at a time.
-------------------------	---

**Figure 33. Session parameters of the Client**

## Creating a session

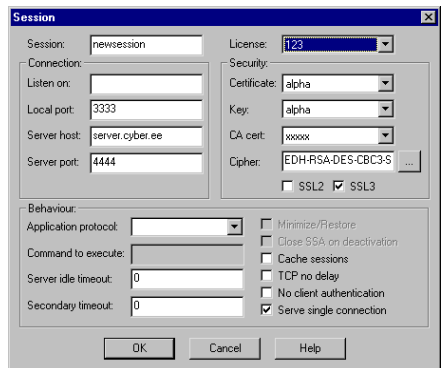
 In the *New* submenu of the *Object* menu select *Session...* The new session creation dialog box appears. If a list box is empty, it means that some important action prior to the creation of the session has not been carried out (the key has not been generated, the certificates are not loaded etc.).

Enter all necessary data and click *OK*.



The image shows the 'Session' dialog box with the following fields and options:

- Session:** (empty text field)
- License:** (empty dropdown menu)
- Security:**
  - Certificate:** (empty dropdown menu)
  - Key:** (empty dropdown menu)
  - CA cert:** (empty dropdown menu)
  - Cipher:** (empty text field)
- Behaviour:**
  - Application protocol:** (empty dropdown menu)
  - Command to execute:** (empty text field)
  - Server idle timeout:** (0)
  - Secondary timeout:** (0)
  - ☐ Minimize/Restore
  - ☐ Close SSA on deactivation
  - ☐ Cache sessions
  - ☐ TCP no delay
  - ☐ No client authentication
  - ☐ Serve single connection
- Buttons:** OK, Cancel, Help




The image shows the 'Session' dialog box with the following fields and options filled out:

- Session:** newsession
- License:** 123
- Security:**
  - Certificate:** alpha
  - Key:** alpha
  - CA cert:** xxxxxx
  - Cipher:** EDH-RSA-DES-CBC3-S
- Behaviour:**
  - Application protocol:** (empty dropdown menu)
  - Command to execute:** (empty text field)
  - Server idle timeout:** 0
  - Secondary timeout:** 0
  - ☐ Minimize/Restore
  - ☐ Close SSA on deactivation
  - ☐ Cache sessions
  - ☐ TCP no delay
  - ☐ No client authentication
  - ☒ Serve single connection
- Buttons:** OK, Cancel, Help

The new session adds to the configuration frame *Sessions* section.



 A session can be added with the ‘-session -insert’ command. This command has the following parameters:


- -N <sesname> – mandatory, defines the name of the new session in the configuration file.
- -L <port> – mandatory, the SSA Client port.
- -H <serverhost> – mandatory, the SSA Server host name or IP address.
- -P <port> – mandatory, the SSA Server port in the server host.
- -A <keyname> – mandatory, the name of client private key in the configuration file. The key must be generated prior to the operation!
- -B <certname> – mandatory, the name of client certificate in the configuration file. The certificate must be loaded to the configuration file prior to the operation!
- -X <rootname> – mandatory, the name of Certification Authority certificate in the configuration file. The certificate must be loaded to the configuration file prior to the operation!
- -M – optional, defines the single-/multiconnection servicing mode. Single connections are serviced by default.
- -E <command> – optional, defines the command to execute on session activation. (If the command contains spaces or special characters, it must be in double or single quotes).
- -m – optional, defines iconization of the SSA Client on session activation.
- -a – optional, defines the exit of the SSA Client on session deactivation.
- -e – optional, session reuse enable. Disabled by default.
- -T <accept timeout> – optional, defines secondary connection timeout. By default, the waiting time is infinite

- -j – optional, enables immediate TCP connections
- -x – optional, switches client authentication off
- -Z <ssl mode> – optional, specifies the SSL protocol. Options: 0 = SSLv2; 1 = SSLv2 and SSLv3; 2 = SSLv3; by default: SSLv2
- -G <bindaddr> – optional, specifies the interface for inbound connections to be transferred by the client. By default, inbound connections will be accepted only from loopback interface.
- -Q <licensename> – optional, licence name in the configuration file. The licence must be stored in the configuration file prior to giving the command!

```
kiku:~/ssa>ssa -f file.ssa -session -insert -N newsession -L 3333  
-H server.ioc.ee -P 4444 -A key -B cert -X ca -M  
Enter password for "file.ssa":  
kiku:~/ssa>
```

**Figure 34. Adding a session to the configuration file**

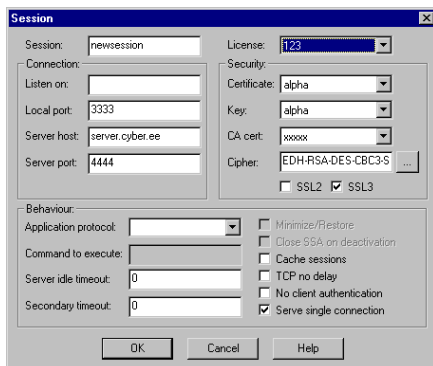
## Session modification


 In the configuration frame *Sessions* section select the session you want to modify.




In the *Object* menu select *Edit...*. The session modification dialog box appears. After necessary modifications click *OK* to save the session data.

Tip! The session can be saved under a new name, leaving the old session unchanged. In this way it is easy to copy sessions in the same configuration file.



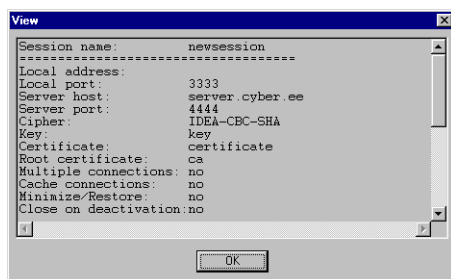
 There is no command for modifying a session. To change session data you have to add the session once more.


## Session viewing

 In the configuration frame *Sessions* section select the session you want to view.



In the *Object* menu select *View...*  
The session dialog box shows the data of the selected session. Press *OK* to close the dialog box.



 The session can be viewed with the '-session -view' command. This command has a single parameter:

- -N <sesname> – defines in the configuration file the name of the session to be viewed.

```

kiku:~/ssa>ssa -f file.ssa -session -view -N newsession
Enter password for "file.ssa":
Session name:      newsession
=====
Local address:
Local port:        3333
Server host:       server.ioc.ee
Server port:       4444
Cipher:            RC4-SHA
Key:               key
Certificate:        cert
Root certificate:   ca
Cache connections: yes
Multiple connections: yes
Minimize/Restore:  no
Close on deactivation: no
Client authentication: yes
SSL version:       SSL3
TCP no delay:      no
Inactivity timeout: 30
Accept timeout:    infinity
Command:           <none>
Map login name:    no
License:           12345

Session status:    ok
=====


Session partners:
=====

kiku:~/ssa>

```

**Figure 35. Viewing a session**

## Session deletion

 In the configuration frame *Sessions* section select the session you want to delete.




In the *Object* menu select *Delete...* The session deletion confirmation dialog box appears. Click *Yes*.



The selected session will be deleted in the configuration frame *Sessions* section.




 A session can be deleted with the '-session -delete' command. This command has a single parameter:

- -N <sesname> – defines in the configuration file the name of the session to be deleted.

```
kiku:~/ssa>ssa -f file.ssa -session -delete -N ca
Enter password for "file.ssa":
kiku:~/ssa>
```

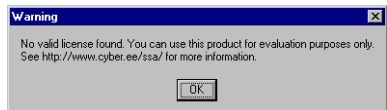
**Figure 36. Deleting a session**

## Session activation

 In the configuration frame *Sessions* section select the session to be activated. In the *Session* menu select *Activate...*



If the session description did not specify the licence to be used by the SSA Client, or if the specified licence is invalid or impertinent, a corresponding warning will appear on the screen. To activate the session click *OK*. Use of the SSA Client without a licence is allowed during the program testing only. For regular use a licence must be acquired.



The selected session activates. Now you can start the application client and direct it to the SSA Client. The SSA can also start the application itself, if the corresponding command is attached to the session to be activated.



**[C>]** To activate a session, start the 'ssaclient' SSA Client service program. Type the configuration file name and the session name to the command line. The '-d' key added to the command line makes the SSA Client service program proceeding in the background after asking the passphrase of the configuration file, releasing the terminal and logging all the messages through the *syslogd*.

```
kiku:~/ssa>ssaclient -f file.ssa -d newsession
Enter password for "file.ssa":
kiku:~/ssa>
```

### Figure 37. Starting the SSA Client as a demon

When started without the '-d' key, the program does not release the terminal and continues to run in foreground, logging all the messages on the screen.

```
kiku:~/ssa>ssaclient -f file.ssa newsession
Enter password for "file.ssa":
INF SSA Client Connection License number 456,
    issued to "AS Wonderbar".
INF CONNECT session: newsession, client: 192.168.2.16:2186,
    server: 192.168.2.2:21, type: FTP client.
    dname: /C=EE/O=xxxxxxx/CN=server,
    cipher: EDH-RSA-DES-CBC3-SHA
```

### Figure 38. Starting the SSA Client in the foreground




If the session description did not specify the licence to be used by the SSA Client, or if the specified licence is invalid or impertinent, a corresponding warning will appear on the screen. Use of the SSA Client without a licence is allowed during the program testing only. For regular use a licence must be acquired.

```
kiku:~/ssa>ssaclient -f file.ssa newsession
Enter password for "file.ssa":
WAR No valid license found. You can use this product
    for evaluation purposes only. See
    http://www.cyber.ee/ssa/ for more information.
```

**Figure 39. The warning message appearing after the start of an unlicensed SSA Client**


## Session deactivation

 In the configuration frame *Sessions* section select the session to be deactivated.




In the *Session* menu select *Deactivate...* The selected session deactivates.



 There is no command for session deactivation. The SSA Client stops itself when the application closes the connection (in the single connection service mode), or it has to be stopped by pressing *Ctrl-C* (in the multiconnection service mode).

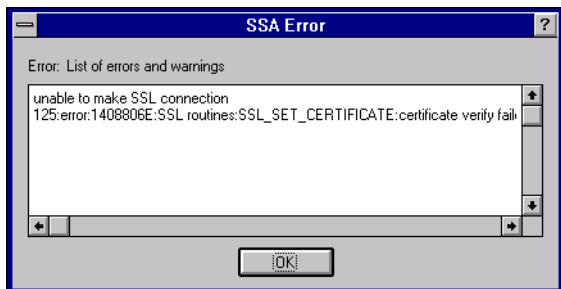
## Session errors viewing


Warnings and error messages can appear by using the SSA; they may not disturb the normal operation, but knowing them can greatly help to solve problems. The session errors viewing command allows to study the list of error messages and warnings relating to the session.

 In the configuration frame *Sessions* section select the session whose error messages you want to view.




A window appears, containing warnings and error messages. To close the window click *OK*.



 There is no command for viewing the session errors. All error messages and warnings are shown directly in the console window.

## Session list viewing

 The session list can be viewed with the ‘-session -list’ command.

```
kiku:~/ssa>ssa -f file.ssa -session -list
Enter password for "file.ssa":
session1
session2
kiku:~/ssa>
```

**Figure 40. Viewing the session list**

## Partners list

A list of partners can be attached to the session. After the secure communication channel creation, but prior to the data exchange both the SSA Client and Server check their partner's distinguished name with this list and make a decision about proceeding with the session. This way, the server can limit the access to the service (by default, any client with valid certificate can connect to the server) and the client can prevent any communication with fake servers).

The metacharacters ‘!’ and ‘\*’ facilitate to create the list. Adding the character ‘\*’ to the end of the name makes the name equal to all names with the same

initial part. The exclamation mark ‘!’ before a name forbids communication with corresponding partner.

The name of the partner is checked according to the following rules:

1. If the list is empty, the connection will be permitted.
2. If the list contains the exact name (without metacharacter ‘\*’) of the partner, the connection will be permitted.
3. If the list contains the exact name (without metacharacter ‘\*’) of the partner, with the preceding exclamation mark ‘!’, the connection will be denied.
4. The longest name will be found containing the metacharacter ‘\*’ and equal to the partner's name; the connection will be permitted or denied depending on presence or absence of the exclamation mark ‘!’ before this name.
5. If the partner's name was not found in the list, the connection will be denied.


#### Partners list:

```
/*
!/C=EE/*
/C=EE/O=IOC/*
!/C=EE/O=IOC/CN=Arne
```


Partner's name	Result
/C=FI/O=XXX/CN=Jurma	Permission
/C=EE/O=XXX/CN=Mati	Denial
/C=EE/O=IOC/CN=Tarvi	Permission
/C=EE/O=IOC/CN=Arne	Denial
/C=EE/O=IOC/CN=Arne Ansper	Permissio

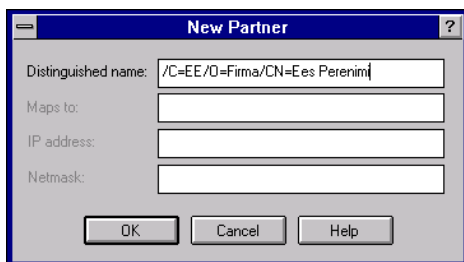
**Figure 41. Example of using the partner list**

### Partner addition

 In the configuration frame *Sessions* section select the session to which you want to add a partner.



 In the *New* submenu of the *Object* menu select *Partner...* The new partner addition dialog box appears. Enter the distinguished name of the partner. The last three fields are for the SSA Server sessions only.




The 'New Partner' dialog box has the following fields and buttons:

- Distinguished name:** /C=EE/O=Firma/CN=Ees Perenim
- Maps to:** (empty field)
- IP address:** (empty field)
- Netmask:** (empty field)
- Buttons:** OK, Cancel, Help

Click *OK*. The new partner adds to the session selected in the configuration frame.




 A partner can be added with the '-partner -insert' command. This command has two parameters:

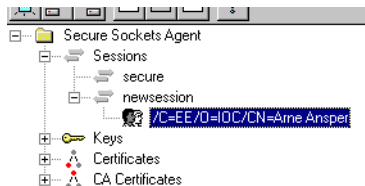
- -N <sesname> – defines the name of the session to which the partner will be added.
- -J <partnername> – defines the name of the partner to be added. (If the name contains spaces or special characters, it must be in double or single quotes).

```
kiku:~/ssa>ssa -f file.ssa -partner -insert -N newsession
-J '/C=EE/O=IOC/CN=Arne Ansper'
Enter password for "file.ssa":
kiku:~/ssa>
```

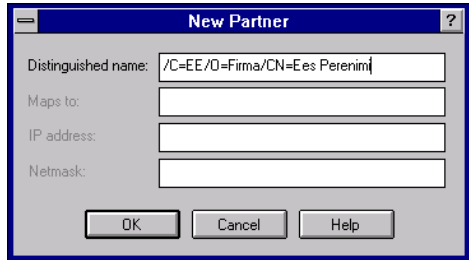
**Figure 42. Adding a partner to the list**


## Partner data modification

 In the configuration frame *Sessions* section select the partner whose data you want to modify.




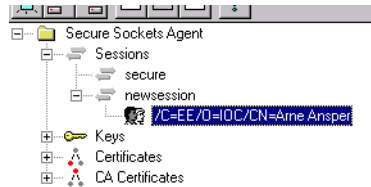
In the *Object* menu select *Edit...*. The partner data modification dialog box appears. After necessary changes click *OK* to save the partner data.



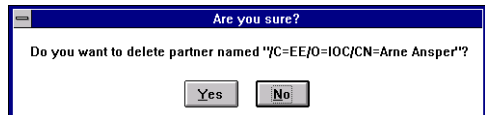
 There is no command for modification of the partner data. To modify the data, the partner has to be added once more.

## Partner deletion

 In the configuration frame *Sessions* section select the partner you want to delete.




In the *Object* menu select *Delete...*. The partner deletion confirmation dialog box appears. Click *Yes*.



The selected partner will be deleted from the configuration frame *Sessions* section.




 A partner can be deleted with the '-partner -delete' command. This command has two parameters:

- -N <sesname> – defines the name of the session where the partner will be deleted.
- -J <partnername> – defines the name of the partner to be deleted (if the name contains spaces or special characters, it must be in double or single quotes).

```
kiku:~/ssa>ssa -f file.ssa -partner -delete -N newsession
-J '/C=EE/O=IOC/CN=Arne Ansper'
Enter password for "file.ssa":
kiku:~/ssa>
```

**Figure 43. Deleting a partner from the list**

### *Partners list viewing*

 The list of partners can be viewed with the ‘-partner -list’ command. This command has a single parameter:

- -N <sesname> – defines the name of the session of which partners list will be shown.

```
kiku:~/ssa>ssa -f file.ssa -partner -list -N newsession
Enter password for "file.ssa":
!/C=EE/*
/C=EE/O=IOC/*
kiku:~/ssa>
```

**Figure 44. Viewing the partners list**


### *SSL proxy support*

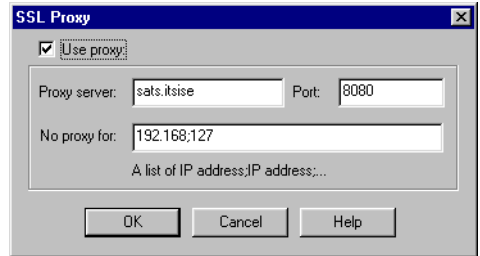
In the original version of the SSA Client, to access the SSA server behind the firewall a special gateway had to be made in the firewall. The new Client includes the SSL proxy support allowing to use existing HTTPS proxies like Squid or Harvest. Configuring the proxy is similar to the procedure for configuring the Netscape or Internet Explorer proxies: the user has to define the name (or IP address) of the proxy, the proxy port and the list of direct-access IP addresses, for which the proxy will not be used. If the IP address last digits are omitted, then the corresponding network can be accessed directly. If you write, for example, 192.168 for the direct-access IP address, then all the 192.168.0.0/255.255.0.0 subnetwork can be accessed directly.

Client versions with graphical user interface store the proxy parameters to the active user data in a register. The versions with command-line interface read the parameters from the environment variables *SSA\_PROXY*, *SSA\_PROXYPORT* and *SSA\_DIRECT*.


The proxy parameters are not related to a specific SSA configuration file. Using a proxy does not create a security risk, as whole the connection between SSA

Client and Server is still secured with the SSL protocol; the proxy has no access to the data flowing through this protocol.

 To configure the proxy, select *SSL proxy...* in the *Tools* menu. The proxy configuration dialog box appears.



Enter the name or address of the proxy, the proxy port, and the list of the IP addresses you want to access directly without using the proxy. Use semicolon (;) to separate the IP addresses. You can get all the parameters from your local system administrator. Click *OK*.

 To configure the proxy, change the environmental variables listed above. The specific commands depend on the operating system and the command interpreter.

```
C:\>set SSA_PROXY=sats.itsise
C:\>set SSA_PROXYPORT=8080
C:\>set SSA_DIRECT=192.168
```

**Figure 45. Configuring the proxy under the Windows**

```
kiku:~>setenv SSA_PROXY sats.itsise
kiku:~>setenv SSA_PROXYPORT 8080
kiku:~>setenv SSA_DIRECT 192.168
```

**Figure 46. Configuring the proxy in tcsh**

```
sh$ SSA_PROXY=sats.itsise
sh$ export SSA_PROXY
sh$ SSA_PROXYPORT=8080
sh$ export SSA_PROXYPORT
sh$ SSA_DIRECT=192.168
sh$ export SSA_DIRECT
```

**Figure 47. Configuring the proxy in sh**

The corresponding lines can also be added to the system configuration files; in this case they will be loaded automatically. In case of the Windows 95 the commands have to be added to the C:\AUTOEXEC.BAT file; with Windows

NT, open the *Control Panel*, select *System* and add the variables to the *User Environment Variables for ...* field; with sh add these commands to the *.profile* file; with tcsh add them to the *.cshrc* file.

**Note:** various HTTPS proxies allow connections to no ports other than 443 and 563. If the SSA Server works with some other port, you have to specially enable the transfer of connections to this port. The details of the procedure depend on the system used: for example, in case of the Barricade firewall you only have to select in the *WWW cache management* menu the *SETUP* submenu and add the required port to the *Additional SSL ports* field. In case of *Squidi* you have to edit the *squid.conf* file and add the required port to the *SSL\_ports* line.

**Tip:** If you do not use the SNEWS protocol (a NNTP protocol secured with SSL), select port 563 for the SSA Server, to make the use of the service easier for the clients.



## SSA Client: a step-by-step guide

The examples in this guide are based on a simple client/server database.

### ***Certification Authority***

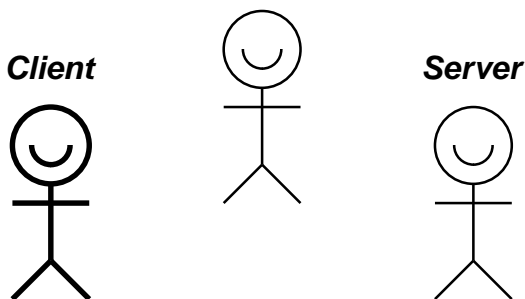


Figure 48. The subjects in the sample system

### ***Step 1: Learn the server requirements and acquire the rights to use the SSA***

In a typical client/server system the server owner dictates to his clients the terms of using the services offered by his server. In the SSA system the server owner decides two essential security attributes: the cipher used, and the Certification Authority for partners' public keys certification. The server administrator also sets purely technical parameters like the name/address of the host offering the service and the SSA Server port.

Before implementing the SSA, the corresponding rights to use it must be acquired. Information concerning the SSA licencing policy can be found from <http://www.cyber.ee/ssa/>.

#### ***Example***

*The server owner determined the cipher to be RC4-MD5, the Certification Authority: xyz, the server host name: server.ee, and the SSA server port: 3333.*

### ***Step 2: Learn the Certification Authority requirements***

Prior to generating the RSA keypair and requesting the certificate learn the certification policy of the Certification Authority selected by the server owner. The certification policy establishes the quality of the certification service offered

and the requirements to the users. If the quality of the service offered by the Certification Authority does not satisfy the customer, he can either refuse to use the service offered by the server or try to persuade the server owner to use some other Certification Authority.

The Certification Authority requirements to the user define the user identification procedure and may set additional restrictions:

- to the user base to be certified (the employees of a company only, the residents of a location etc.)
- to the format of user's distinguished name
- to the length of user's private key
- to the cryptographic techniques of the users
- ...

**Example**

*The XYZ Certification Authority issues certificates to all citizens of Estonia; the customers identify themselves with passports. User's private key must be at least 1024 bits long and user distinguished name must be in the format:  
/C=EE/SP=DistrictName/SettlementName/CN=FirstName LastName  
The data in the distinguished name will be checked against the data in the passport of the certifee.*

### **Step 3: Create a new SSA configuration file**

Create a new SSA configuration file (see “Creating the configuration file”, p. 26). The passphrase for protection of this file must be at least 11 characters long and include numbers, capitals, and small letters; this makes brute-force breaking of the passphrase almost impossible.

### **Step 4: Generate the RSA private key**

Generate the RSA private key (see “Generating the RSA key”, p. 30). You can pick any name for the key; the length of the key must match the length required by the Certification Authority.

**Example**

*The user generated a 1024 bit key named "mykey".*

**Step 5: Generate the certification request**

Generate the certification request containing the user distinguished name and public key (see "Requesting the certificate", p. 33). The distinguished name in the request must match the requirements set by the Certification Authority; if it fails to do so, the Authority can refuse to issue the certificate.

**Example**

*Mr. John User residing in Tallinn should generate the certificate request for the name:*

*/C=EE/L=Tallinn/CN=John User*

**Step 6: Leave the new configuration file temporarily**

As the next step assumes leaving the computer, the newly created configuration file has to be saved (see "Saving the configuration file", p. 27) and closed (see "Closing the configuration file", p. 28). Never leave a computer with the SSA configuration file open: an attacker having access to your computer can covertly get your private key. If a session secured with SSA cannot be closed (because of a time-consuming query etc.), you have to use a password-protected screensaver or some other safeguard.

**Example**

*The user saved the configuration file in the service1.ssa file, using the passphrase "nobody can guess it".*

**Step 7: Certify yourself**

The details of the certification procedure are described in the certification policy and they can greatly vary depending on the quality of the certification service. Nevertheless, direct contact of the certifier with the operator of the Certification Authority is needed to issue a trusted certificate: the operator has to identify the person to be certified and his right to use the proposed distinguished name, as well as to check the authenticity of the public key.

Therefore we assume that to get a certificate, the user personally has to go to the Certification Authority, to submit his certification request (the file created on the previous step; the user may present it on a floppy, for example) and to identify himself with the required documents. The certificate will be issued after the user's data are checked; the certificate could be stored on the same floppy or sent him by e-mail.

During the same visit, it is reasonable to get the public key of the Certification Authority: no authenticity problems can arise with a key given directly by the Authority. This key must certainly be handed over on a floppy, because by transferring it through the network the risk of modification exists.

**Example**

The user stored his certification request in the `user.req` file on a floppy.  
The Certification Authority issued him the certificate and stored it in the `user.crt` file on the floppy.  
The Certification Authority also stored its public key in the `ca.crt` file on the floppy.

### **Step 8: Open the previously created configuration file**

Open the configuration file again (see "Opening the configuration file", p. 27).

**Example**

The user opens the previously saved `service1.ssa` configuration file which passphrase is "nobody can guess it".

### **Step 9: Add the Certification Authority public key to the SSA configuration file**

Add to the configuration file the public key you got from the Certification Authority (see "Adding a Certification Authority public key to the configuration file", p. 38). This is a very serious operation from the system security viewpoint. Prior to adding the key make sure that nobody has changed the key. You can pick any name for the public key.

**Example**

The user adds to the configuration file under the name "root" the Certification Authority public key from the `ca.crt` file on the floppy.

### **Step 10: Add your certificate to the configuration file**

Add to the configuration file the certificate you got from the Certification Authority on the floppy or by e-mail (see "Adding certificates to the configuration file", p. 34). This step is not so serious as the previous one, because the authenticity of a certificate can be checked with the Certification Authority public key. You can pick any name for the certificate.

**Example**

The user adds to the configuration file under the name "cert" the certificate from the user.crt file on the floppy.

**Step 11: Add the license to the configuration file**

Add the license acquired from the Cybernetica to the configuration file (see "Adding the licensing information to the configuration file", p. 42).

**Step 12: Create the session**

Create a new session (see "Creating a session", p. 48). The user can give any name to the session and select the SSA Client port. The server administrator defines the server host name, the SSA Server port, and the cipher. Fill in the certificate, the user private key, and the Certification Authority public key fields, using the object names entered in the SSA configuration file on the previous steps. Select the mode according to the nature of the secured application: for applications using a single TCP connection select "*Serve single connection*", for applications using multiple TCP connections choose "*Serve multiple connections*".

**Example**

The user creates the following session:

Session name: database

SSA Client port: 4444

Server name: server.ee

Server port: 3333

User's certificate name: cert

User's private key name: key

Certification Authority public key name: root

Cipher: RC4-MD5

Mode: Serve multiple connections

License: 123456

**Step 13: Save the configuration file**

Save the configuration file completed (see "Saving the configuration file", p. 27).

**Step 14: Activate the session**

Activate the session you just created (see "Session activation", p. 53). The session icon will be colored, indicating the activated state of the session.

**Example**

The user activates the “database” session.

**Step 15: Start the secured application and connect to the server**

Start the application and direct it to the SSA Client port of the client computer’s loopback SSA Client port. There are no common guidelines for configuring the client, as it depends on specific application. For example, with telnet you can set the host address and port number in command line:

```
telnet 127.0.0.1 port
```

Some applications have a special utility for configuring various server parameters (database applications). In this case, a new profile should be created directing the client to the port selected for the address 127.0.0.1.

A successful connection adds a new line to the SSA Client’s active connections frame:

SesId	Session	Client	Server	Cipher	Server name	Bytes in	Bytes out
1	sslnet	127.0.0.1:1110	192.168.2.1:4466	DES-CBC3-SHA	/O=Barricade/CN=SSLnet daemon	223	70

**Figure 49. Active connections frame**

For every connection the client and server addresses, server distinguished name, the cipher used, and transferred data volume are shown.

**Example**

The user starts secured database application and connects to the port 4444 on the address 127.0.0.1.

**Step 16: Work with the application**

Use the application and check its reliability.

**Step 17: Deactivate the session**

After closing the application deactivate the selected session (see “Session deactivation”, p.55).

*Example*

*The user deactivates the “database” session.*

**Step 18: Close the configuration file**

Close the configuration file (see “Closing the configuration file”, p. 28).





## Automating the routine operations

In everyday use of the SSA, to go through the cycle *start the SSA Client* → *load the configuration file* → *activate session* → *start the application* → *do your work* → *close the application* → *deactivate session* → *close the SSA Client* can be very annoying. Fortunately, the SSA allows automating such kind of routines; skilful use of a secured application is only slightly more inconvenient compared with an unsecured application.

The SSA allows to automate

- loading the configuration file
- session activation
- starting the application
- session deactivation
- closing the SSA

All these options together make the use of a secured application significantly easier: *start the SSA Client* (loads the configuration file automatically) → *do your work* → *close the application* (the SSA Client deactivates the session automatically and terminates).

### **Configuration file autoload and session activation**

Autoloadable configuration file and the name of the session to be activated can be defined with a command line for the SSA Client. To do it, use the switch *-f* with the name of the configuration file and the switch *-s* with session name.

#### *Example*

```
ssa -f \path\config.ssa -s database
```

The command in this example starts the SSA Client, opens the ‘\path\config.ssa’ configuration file, asks the passphrase for decryption of the data in this file and activates the ‘database’ session after the valid passphrase is entered.

You can add shortcuts to the *Start* menu for quick and easy activation of the most frequent sessions.

## **Autostart of the application**

An application name can be attached to the session, to start the application after the session activation. To do it, write the application name with necessary parameters in the *Command to execute* field when creating or modifying the session.

### **Example**

*The SSA is used to secure the telnet. The user has already created the session which transfers the connection made with the client computer loopback port 4444 to the necessary server.*

*He can add the "telnet localhost 4444" command to the Command to execute field; now, on each activation of the session the telnet client will also automatically activate making connection to the right port.*

It will be handy to combine the application autostart option with configuration file autoload and session autoactivation functions.

## **Session automatic deactivation**

Some client/server applications – telnet, for example, and a multitude of mailers – use a single TCP connection per session. The SSA has a special single-connection service mode for securing such applications. After transferring one TCP connection the session deactivates automatically. To activate this mode, the *Serve single connection* option has to be selected in the *Operation mode* field when creating or modifying the session

## **SSA autoclose**

With applications using single TCP connection the SSA automatic closing can be used. After transferring one TCP connection and session deactivation the configuration file will be closed and the program will terminate. To activate this mode, the *Close SSA on deactivation* checkbox has to be checked when creating or modifying the session.

## SSA security

SSA successfully solves the data communication security problem between client and server, but to make the whole system secure, the SSA itself needs protection. In the following chapters some SSA security issues are discussed and solutions are given for security problems.

### ***Theft of the private key***

The private key is needed to create a secure communication channel for user authentication. Theft of this key gives the attacker the capability to covertly operate under the name of the user. Therefore it can be considered to be the most serious attack against the SSA. As a rule, to steal the private key one must have physical access to user's computer. The private key is stored in the SSA configuration file, which is protected with user's passphrase from unauthorized reading. The ways to steal the key are following (with ascending complexity to do it):

1. The user has opened his SSA configuration file and left his computer unsecured. The attacker having physical access to the computer can change the password of the configuration file and save it to a floppy. To prevent this kind of attack close your SSA configuration file before leaving your computer; if it is not possible (a time-consuming query is going on etc.), use a password-based screen lock to protect your workstation.
2. The attacker watches how the user enters his passphrase when he opens the configuration file. Later on, the attacker can copy the configuration file and open it with the stolen passphrase. There is a lot of ways to do such kind of spying: from peeping over the shoulder to dedicated covert cameras and various other spy equipment. As a rule, the attacker has to enter the user's room at least once. To copy the configuration file, he has typically to access user's computer physically, but it can be done through the network, if the computer is poorly configured. General security measures and frequent change of the passphrase (which sets narrower time limits for copying attempts) are the remedy for this attack.
3. The attacker copies the configuration file and tries to guess the passphrase with brute force. To copy the file, he has to have physical access to the user's computer. To guess the passphrase is more difficult when the passphrase is long. Each entered character will add about 6 bits of key information, if the passphrase contains digits, capitals, and small letters. As up to 56 bit keys can

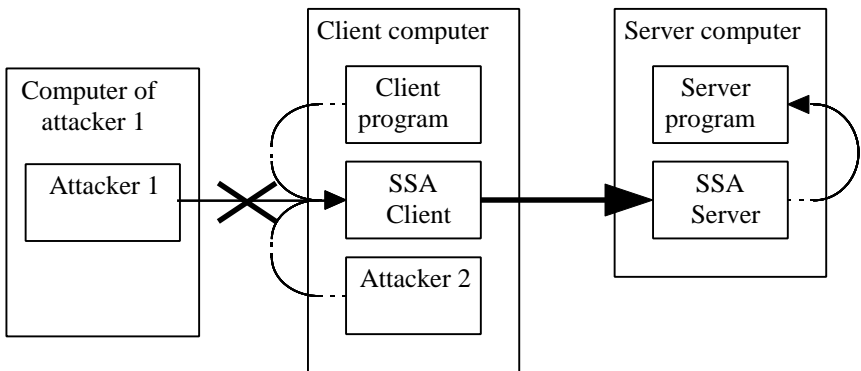
be broken nowadays (with rather great computing effort), a passphrase must be at least 11 characters long. So, using sufficiently long passphrases will defeat this attack.

4. The attacker plants a Trojan in the computer. This is a very dangerous and versatile attack. The rather simple method is to store all user's keystrokes to a file from which the attacker can later read out the passphrase. In a more sophisticated case the attacker himself can connect to the computer and send the collected data covertly. Here the remedies are safeguarding of the whole working environment and regular sanitation.

When a doubt arises about the user's private key being compromised, one must immediately report it to the server owner. The certificate must be cancelled and security measures must be applied to prevent any recurrence of the incident. After that, the user has to generate a new private key for himself and to apply for a new certificate.

## ***Multiusers systems***

The SSA Client and Server authenticate each other during creation of the secure channel. This procedure prevents unauthorized use of the application server and does not allow to create fake servers for swindling any information out of the client. The authentication takes place between the SSA Client and Server, but two connections are still left without protection: between the application client and the SSA Client, as well as between the SSA Server and the application server.



**Figure 50. A possible attack in a multiuser system**

When the user activates a SSA session, the SSA Client will wait for inbound TCP connections and transfer them to the SSA Server. The SSA Client only accepts the connections coming from the same computer through loopback: so the attacks of the kind planned by the Attacker 1 in the figure above will not be successful. Unfortunately, the SSA can do nothing to prevent the attack of the Attacker 2 in the same figure. Therefore, even though the SSA provides protection on the application level, the whole computer must be under control of a single user. There is no problem with Windows workstations: the attacks of the second type can possibly be carried out with Trojans only. With Unix systems the problem is a more complicated one: multiple users can use the same computer simultaneously. A partial solution is to use the single TCP connection servicing mode. After transferring a TCP connection the session will automatically deactivate. To activate this mode, select the *Serve single connection* option to the *Operation mode* field when creating a session.

## **Fake servers**

The SSA Client/Server authentication is carried out in two steps: first, the authenticity of partner's certificate is checked with the Certification Authority public key, then the distinguished name found in the certificate is checked against the partners list. If the list is empty, the communication will be allowed with all the partners having a valid certificate. If the attacker can get his certificate from the same Certification Authority where the Client and the Server are certified, he can set up a fake server. The Client can begin to communicate with the fake server in two cases: first, when its partners list is empty, it will accept communication with any server certified by the right Certification Authority; second, it will take place, when the partners list does define a certain server, but the Certification Authority has issued two certificates with the same name.

For the first case, the solution is to include the right server name to the partners' list. In the second case one has to change his Certification Authority, switching to an Authority whose certification policy rules out such kind of abuse.

## **The SSA Client in other computer**

There may be a situation that the user has not the necessary software (the SSA Client or application client) in his own computer. A great temptation may arise to telnet to some computer where all this is available and to start the SSA Client there.

Never do this! As the passphrase for opening the configuration file will in this case be sent openly over the network, anybody can read it and later use it (see chapter “Theft of the private key”).

You may only start the SSA Client in your own computer and the SSA passphrase may only be entered from the console keyboard. When using the X-Windows, the “*Secure Keyboard*” function must be activated before entering the passphrase.

## **Conclusion**

The SSA is no ultimate solution for all security problems. It is a component helping to build a secure system. To beat the threats described in this chapter, additional IT safeguards, organizational measures and physical measures must be applied. To sum up: the effective use of the SSA presupposes the following:

- a trustworthy Certification Authority
- regular sanitation of workstations to prevent the Trojans
- using long passphrases and changing them frequently
- using a firewall for LAN protection

# Installation

## ***Installing the Windows version***

The Windows version of the SSA Client works in the Window 95, Window NT 3.51 and Windows NT 4.0 environments. You need 2 MB free space on disk to install it. To install the software, start the **setup.exe** program on the first installation diskette.

The Setup program allows choosing the directory for the SSA Client to be installed; the program copies necessary program files and auxiliary files to this directory and creates icons for the installed programs.

If you want to move the SSA Client in some other location on the disk, you only have to copy the program files and help files. The SSA Client needs for its work no additional files: all the program libraries are linked statically, no dynamically linked libraries are used. Due to this feature you can, for example, copy the SSA Client and the configuration file to a floppy and carry with you on your travels. To use the SSA Client in any other computer no additional files have to be installed.

## ***Installing the Unix version***

The Unix version of the SSA Client is shipped as compressed tar archive. To install it unpack it first using `uncompress` or `gunzip` command:

```
uncompress ssa.tar.Z
```

or

```
gunzip ssa.tar.Z
```

Then unpack the tar file using `tar` command:

```
tar xf ssa.tar
```

Copy the SSA Client and configuration file editor to appropriate directory e.g. `/usr/local/bin/`

```
cp ssaclient /usr/local/bin
```

```
cp ssa /usr/local/bin
```





## Problems

### ***I lost my configuration file passphrase. What can I do?***

By creating the SSA system a great care was taken to prevent the possibility of reading the user's configuration file without knowing the passphrase. It is necessary for the protection of user's private key against attacks. No “master passphrase” exists which would allow to read all the configuration files.

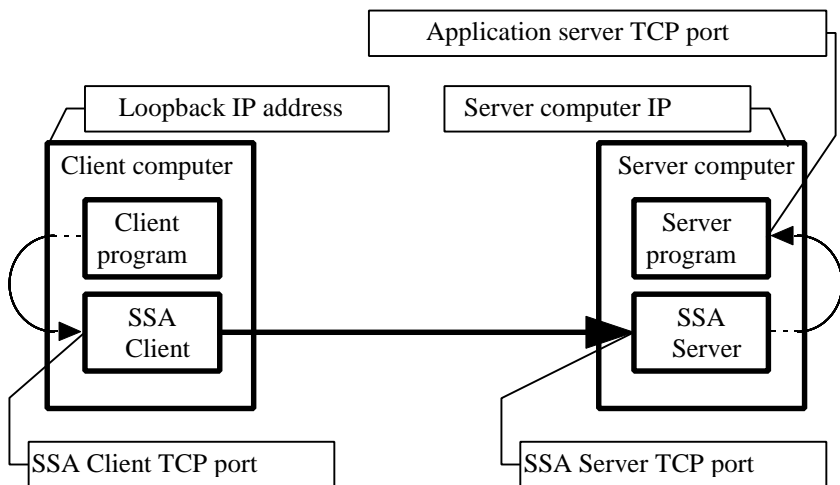
When you lost your passphrase, you have to create the new configuration file, to generate a new private key and to get the new certificate.

### ***Which cipher is the best one?***

There is no such thing as the best cipher. Only general recommendations can be given to select the cipher. For more information about the strength of the cryptographic functions used, see the chapter “Strength of the cryptoalgorithms used”, p.13. The strongest, but the slowest cipher is EDH-RSA-DES-CBC3-SHA, whereas IDEA-CBC-SHA is almost equal in strength, but significantly faster.

### ***The client cannot connect to the server***

There can be a lot of reasons why the application client cannot connect to the server. We will look at them from the beginning, i.e. beginning from the client.



**Figure 51. Model of a system secured with the SSA**

### *The client configuration is incorrect*

The client program must connect to the SSA Client TCP port on the client computer loopback IP address. The loopback IP address is traditionally 127.0.0.1. The port number used is shown in the *Local port* field of the session.

The easiest way to check the client configuration is to observe the active connections frame (in the graphical user interface version) or client log window (in the command-line interface version) and to activate the client program. If the information about the new connection appears in the *active connections* field or in the log, then the client is configured properly.

### *The SSA Client configuration is incorrect*

The SSA Client must connect to the port defined by the server administrator, on the address given by the server administrator. The server address and port are stored in the *Server hostname* and *Server port* fields of the session correspondingly. Check to be sure that these data are entered correctly.

Check the availability of a server on this address; try to telnet to this port:

```
telnet servername portnumber
```

If you get from telnet an error message saying, “*Connect failed*”, or “*Connection refused*”, then the server administrator probably has made a mistake or the SSA server is temporarily down.

Check the consistency of the session security attributes (the key, the certificate, and the Certification Authority public key). To do it, view the session (see chapter “Session viewing”, p. 51) and find the *Session status:* field. If the value of this field is not *OK*, you have made a mistake when selecting the key, the certificate, or the Certification Authority public key.

Make sure you are using the cipher assigned by the Certification Authority administrator.

Problems also arise when the secured application uses multiple TCP connections, but *Serve single connection* mode has been selected for the session. In this case, only the first operation will be successful and the client will show the partial result, with the error message. Try the *Serve multiple connections* option.

Check the time and data in the client computer. All certificates contain an expiration date. If the computer clock is set very inaccurately, the certificate can erroneously be considered as invalid.

### *The SSA Server configuration is incorrect*

This section is meant for the server administrator. The SSA Server must connect to the application server. The application host address and port are stored in the *Server hostname* and the *Server port* fields of the session correspondingly. Check to be sure that these data are entered correctly.

Check with telnet whether the server answers on this address:

```
telnet servername portnumber
```

If you get from telnet an error message saying “*Connect failed*”, or “*Connection refused*”, then you probably have made a mistake by configuring the application server.

Check the consistency of the session security attributes (the key, the certificate, and the Certification Authority public key). To do it, view the session (see chapter “Session viewing”, p. 51) and find the *Session status:* field. If the value of this field is not *OK*, you have made a mistake when selecting the key, the certificate, or the Certification Authority public key.

Make sure that the SSA Server uses the *Serve multiple connections* mode.

Check the time and date in the server computer. All certificates contain an expiration date. If the computer clock is set very inaccurately, the certificate can erroneously be considered as invalid.

## Securing the Windows with the SSA

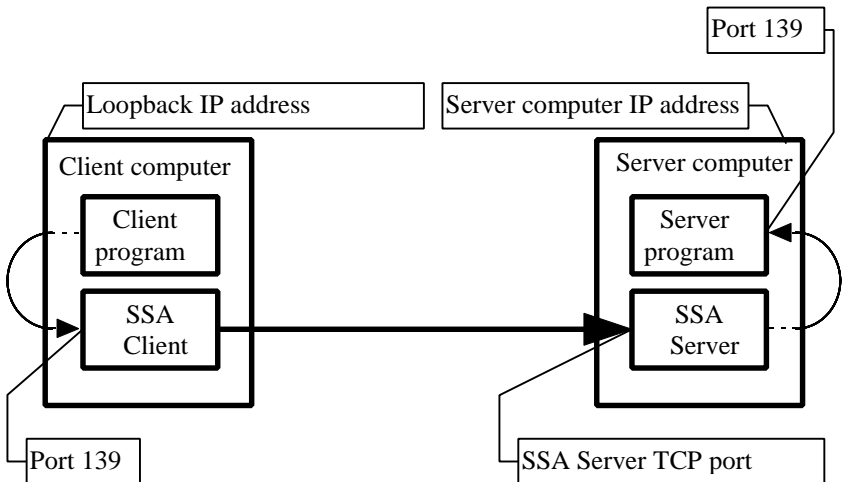
With the SSA, the Windows-specific network applications can also be protected, for example, disk sharing with the SMB protocol, communication between MS Exchange client and server etc.

### Securing the SMB network file system

The NetBIOS protocol (not to be confused with the lower-level NetBEUI protocol!) can work on the TCP/IP transport layer. The corresponding protocol is specified in the RFC1001 and RFC1002 standards. To implement all the services provided by the NetBIOS, the following ports are needed:

- name service: TCP port 137 and UDP port 137
- session service: TCP port 139
- datagram service: UDP port 138

The datagram service is used in the Windows environment, for example, to implement an interprocess communication method known as mailslots.



**Figure 52. Securing the SMB protocol**

With the session service, file sharing and named pipes can be implemented. The name service allows converting the NetBIOS names to the IP addresses. The

corresponding relations can also be described in the *LMHOSTS* file, without using the name service.

To secure the network file system and the named pipes a special SSA session has to be created, connecting the client computer port 139 with the server computer port 139.

In addition, you have to tell to the client computer that it has to access the server located on the IP address 127.0.0.1. For this purpose, add to the *LMHOSTS* file the following line:

```
127.0.0.1          SERVER #PRE
```

*SERVER* is the NetBIOS name of the server computer. The *LMHOSTS* file is located in different places depending on the operating system. For example, under the Windows NT operating system its right location is: %SystemRoot%\system32\drivers\etc\lmhosts. In addition, in the TCP/IP settings the use of the *LMHOSTS* file has to be enabled. To do it, open the *Control Panel* and select *Network*. Find the TCP/IP protocol and open its settings. Check the *Enable LMHOSTS Lookup* selection.

After editing the *LMHOSTS* file start the SSA Client and Server; from the client computer command line enter the following command:

```
NET VIEW \\SERVER
```

As a result, the list of the resources allocated by the server will be shown. To mount the disks use the command

```
NET USE * \\SERVER\RESSURSS
```

## Securing the MS Exchange

Securing the communication between the MS Exchange client and server is rather similar to the procedure described in previous section. In addition to the port 139 also the port 135 (used by the *RPC Locator*) must be secured.

In the Exchange server, you have to change the sequence of the protocols used, bringing the named pipes to the first place. It can be done during or after the installation of the server. During the installation of the server, in the *Setup Editor* program select the *Binding Order* page and move the named pipes to the first place. After the installation you can change the protocol order by changing the **RPC\_Binding\_Order** value of the register key **HKEY\_LOCAL\_MACHINE\Microsoft\Exchange\Exchange Provider**. Move the **ncacn\_np** protocol to the first place.

## Command line switches

The following is the alphabetical list of the SSA Client command line switches. It is important to note that they are subject to the **case-sensitive** processing.

### SSA Client with graphical interface

Switch	Comment
-f <filename>	The name of the autoloadable configuration file. The SSA Client asks from the user the passphrase for opening the file and then opens it.
-s <session>	The name of the session to be automatically activated. Use this switch together with -f switch only. The switch can be repeated, with different session names; all the specified sessions will be activated in this case.

### SSA configuration file editor

Switch	Comment
-a	Makes the SSA Client terminate on deactivating the related session. Can be used in the -session -insert operation.
-A <key>	Name of the private key in the configuration file. A mandatory parameter of the -session -insert operation.
-B <certificate>	Name of the certificate in the configuration file. A mandatory parameter of the -session -insert operation.
-b <partner IP address>	For creating the SSA Server configuration file only. Can be used in -partner -insert operation. Allows binding the user with particular IP address, or network. This user may only connect from the hosts of the sub-network defined by the switches -b and -q.
-C <cipher>	The cipher used. A mandatory parameter of the -session -insert operation.

-cert	Specifies the next subcommand to be of the certificate operation type. The -list, -delete, -insert and -view subcommands are allowed.
-D <dtype>	The distinguished name for the certificate request. A mandatory parameter in the -key -Req operation. If the distinguished name contains spaces or special characters, it must be in double or single quotes.
-delete	This subcommand deletes an object of selected type from the configuration file; it is used with -key, -cert, -root, -session, or -partner main command. Always requires the parameter -N.
-E <command>	Specifies the command to be executed on session activation. Can be used in the -session -insert operation. If the command contains spaces or special characters, it must be in double or single quotes.
-e	Allows reusing sessions.
-f <filename>	Defines the name of the configuration file for an operation. \${HOME}/.ssarc is used by default.
-G <bindaddr>	Specifies the interface for inbound connections to be transferred by the client. By default, inbound connections will be accepted only from loopback interface. Can be used in the -session -insert operation.
-gen	Subcommand for generating a new key; is used with the -key main command. Always requires the parameters -N and -K.
-H <server>	Name or IP address of the server to which the SSA Client or Server connects. A mandatory parameter of the -session -insert operation.
-I <certfile>	Name of the file containing the certificate. A mandatory parameter in the -cert -insert and -root -insert operations.



-insert	Subcommand for adding an object of selected type to the configuration file, is used with the -cert, -root, -session, or -partner main command. Always requires the parameter -N.
-J <partner>	Name of the communication partner. A mandatory parameter of the -partner -insert operation. If the partner's name contains spaces or special characters, it must be in double or single quotes.
-j	Enables immediate connections. Required for the applications running in the interactive mode (Progress). Can be used for the -session -insert operation.
-K <keysize>	The bit length of the key to be generated. A mandatory parameter of the -key -gen operation.
-key	Specifies the next subcommand to be of the private key operation type; -list, -delete, -gen, and -Req are the allowed subcommands.
-L <port>	The port where the SSA Client or Server waits for inbound TCP connections. A mandatory parameter of the -session -insert operation.
-license	Specifies the next subcommand to be of the license operation type. The -list, -delete, -insert and -view subcommands are allowed.
-list	Subcommand for viewing a list of objects of selected type; is used with main commands -key, -cert, -root, -session, -partner.
-listall	Command for viewing a list of all objects.
-M	Sets multiple TCP connection service mode for the session. Can be used in the -session -insert operation.
-m	Makes the SSA Client with graphical interface to iconize on the activation of the session. The SSA Client window will be restored on the deactivation of the session. Can be used in the -session -insert operation.

-N <objname>	The name of the operation object of the selected type in the configuration file.
-o <logintype>	For creating the SSA Server configuration file only. The SSA Server can itself log in the application server for the user, with a log-in name derived from the distinguished name. The switch specifies the name of the protocol. Two protocols are supported at the moment: the MS SQL Server protocol (server type <code>sqlserver</code> ), and HTTP (server type <code>http</code> ). Can be used in the <code>-session -insert</code> operation.
-O <reqfile>	The file where the created certification request will be stored. A mandatory parameter of the <code>-key -Req</code> operation.
-P <port>	Server port, where the SSA Client or Server connects. A mandatory parameter of the <code>-session -insert</code> operation.
-partner	Specifies the next subcommand as operating with the list of the session partners. The <code>-list</code> , <code>-delete</code> , and <code>-insert</code> subcommands are allowed. Requires the session name parameter <code>-N</code> .
-Q <licensename>	Specifies the licence name in the configuration file. The licence must be stored in the configuration file prior to giving the command! Can be used in the <code>-session -insert</code> operation.
-q <partner netmask>	For creating the SSA Server configuration file only. Can be used in the <code>-partner -insert</code> operation. Allows to link the user with a specific IP address or network. This user is allowed only to connect from the hosts belonging to the subnetwork defined by the switches <code>-b</code> and <code>-q</code> .
-Req	Subcommand for creating a certificate request is used with the <code>-key</code> main command. Requires: the <code>-N</code> switch specifying the key to be certified; the <code>-D</code> switch defining the wanted distinguished name; and the <code>-O</code> switch specifying a file for storing the request.

-root	Specifies the next subcommand as operating with Certification Authorities public keys. The -list, -delete, and -insert subcommands are allowed.
-S <filename>	Name of the file which content and attributes are added to the seed of the random number generator before generating a private key. Can be used in the -key -gen operation.
-session	Specifies the next subcommand as operating with sessions. The -list, -delete, and -insert subcommands are allowed.
-T <accept timeout>	Defines the secondary connection timeout. By default, the timeout is infinite. Required only for the protocols using multiple TCP connections (FTP, Progress, SQL*Net). Can be used for the -session -insert operation.
-U <loginname>	For creating the SSA Server configuration file only. The login name of the communication partner. Can be used in the -partner -insert operation, if the session was created with the -o switch.
-u <timeout>	Server timeout in seconds. If no communication took place between client and server during this interval, the connection will be cancelled. This is necessary for preventing "leakage" of server resources. Where possible, the use of the timeout should be avoided; instead it, the embedded timeout mechanisms of the application server should be used. To switch off the timeout function, you have to zero the corresponding parameter.
-view	Subcommand for viewing a selected object. Can be used with the -cert, -root, and -session main commands. Requires the name-N of the object.
-viewall	Command for viewing the contents of the configuration file.
-X <cacert>	The name of the Certification Authority public key in the configuration file. A mandatory parameter of the -session -insert operation.

-x	Switches the partner authentication off. Can be used for the <code>-session -insert</code> operation.
-y	Command for changing the configuration file passphrase.
-Z <ssl mode>	Specifies the SSL protocol version. Valid options are: 0 = SSLv2; 1 = SSLv2 and SSLv3; 2 = SSLv3; by default: SSLv2. Can be used for the <code>-session -insert</code> operation.

### **SSA Client with command-line interface**

-d	Daemon mode (in Unix version only). The program will run in the background and log through the <i>syslog</i> .
-f <filename>	Name of the configuration file
-p <password>	For defining the configuration file password from the command line. From security considerations, using this switch cannot be recommended: to enter the passphrase interactively would be safer.

# **Cybernetica Software End User License Agreement**

**IMPORTANT - READ CAREFULLY!**

**WHEREAS**

- this product contains computer software (Software), of which Küberneetika AS (Cybernetica) with its principal place of business at Akadeemia tee 21, 12618 Tallinn, Estonia, is the sole proprietor and that is protected by the laws of the Republic of Estonia and international copyright conventions,
- Cybernetica only grants you the right to use, copy and distribute the Software according to the terms and conditions set forth in this Cybernetica Software End User License Agreement (CS-EULA),
- you have been informed by Cybernetica about the foregoing and
- by using, copying or distributing the Software, you are expressing your free will;

NOW, THEREFORE, by using, copying or distributing the Software you are consenting to be bound by this CS-EULA and the legal Agreement between Cybernetica and you as the Customer or Customer's legal representative be entered into from the moment of your performing of any such act.

**IF YOU DO NOT AGREE WITH THE TERMS AND CONDITIONS SET FORTH IN THIS AGREEMENT, CYBERNETICA GRANTS YOU NO RIGHT WHATSOEVER TO USE, COPY OR DISTRIBUTE THE SOFTWARE.**

## **1. LICENSE**

Cybernetica hereby grants the Customer a personal, permanent, nonexclusive and nontransferable license to use, copy and distribute the Software as follows:

### **(a) SSA Client, SSA Server, SSA User Manager (SSA Evaluation License)**

Customer may use the Software during a 30-day evaluation period for the purpose of evaluating the Software and its suitability to Customer's further purposes. Upon the expiration of the evaluation period, Customer shall terminate the use of the Software unless he obtains an appropriate license from Cybernetica.

Customer may produce an unlimited number of identical copies of the Software and distribute such copies to any third parties provided that (a) no fee is charged for such copies or (b) the recipient has been informed of his right to obtain a free copy.

The object code of the Software carries the license.

**(b) SSA Client, SSA Server, SSA User Manager (SSA Non-commercial / Academic License)**

Customer may produce an unlimited number of copies of the Software and take such copies into non-commercial or academic use.

The license information file carries the license.

**(c) SSA Server (SSA Public Server License)**

Customer may use one copy of the Software for providing services to unlimited number of users without possibility of authenticating the users. Customer may also produce one additional backup copy of the Software.

The license information file carries the license.

**(d) SSA Server (SSA Commercial Server License, N users)**

Customer may use one copy of the Software for providing services to N users. Customer may also produce one additional backup copy of the Software.

Customer may and shall provide all users with a license information file that carries an appropriate SSA Connection License.

The license information file carries the license.

**(e) SSA Client (SSA Connection License)**

Customer may use one copy of the Software for establishing connections with the SSA Server specified in the license information file. Customer may also produce one additional backup copy of the Software.

The license information file carries the license.

**(f) SSA Client (SSA Client License, N users)**

Customer may produce and use N copies of the Software and license carrier. Customer may also produce one additional backup copy of the Software.

The license information file carries the license.

**(g) SSA User Manager (SSA User Manager License)**

Customer may use one copy of the Software. Customer may also produce one additional backup copy of the Software.

The license information file carries the license.

## **2. USING THE SOFTWARE**

The Software is considered to be “in use” or “used” when its object code is loaded into the random access memory of any computing device. When used with multi-user operating systems, such number of copies are considered to be in use as there are “users”, according to the specifications of the operating system, who may directly interact with the Software.

**2.1. NON-COMMERCIAL USE** means using the Software by a physical person for his/her own private purposes that are not related to work or business.

**2.2. ACADEMIC USE** means using the Software by the members or students of an academic institution within the context of education process, and using the Software for providing publicly available services by a public library.

**2.3. COMMERCIAL USE** means using the Software in any situation not described in p.2.1 or p.2.2 above.

## **3. RESPONSIBILITY OF PARTIES**

Because by entering into this agreement, Customer has accepted no obligation to pay license fees, Cybernetica disclaims and waives all warranties and liabilities, including, but not limited to, any liabilities for any indirect damages. **IN NO CASE SHALL CYBERNETICA'S ENTIRE LIABILITY EXCEED THE TOTAL AMOUNT OF LICENSE FEES PAID BY THE CUSTOMER TO CYBERNETICA OR ITS AGENTS.**

Should the Customer produce, use or distribute any copies of the Software notwithstanding with the terms and conditions set forth in this CS-EULA, he shall be held responsible to the maximum extent allowed by applicable law. **IN NO CASE SHALL CUSTOMER'S ENTIRE LIABILITY BE LESS THAN THE TOTAL AMOUNT OF LICENSE FEES UNPAID TO THE CYBERNETICA BECAUSE OF THE CUSTOMER'S BREACH OF THIS AGREEMENT.**

## **4. NEW VERSIONS OF SOFTWARE**

Cybernetica may license new versions of the Software as “patches” or “updates”.

Any software licensed as a patch becomes an integral part of the original software. The original CS-EULA shall also govern the use of the “patch”.

If Cybernetica believes that the Customer holds at least one valid “free” license of “updateable software”, Cybernetica may license the Software to the Customer as an “update”. For the purposes of this paragraph, a license is considered “free” if it has never been used for obtaining “update” licenses. The original software becomes an integral part of the “update” and the new license agreement shall

govern the use of the resulting software. For the purposes of this paragraph, such “update” licenses are considered “free”.

Cybernetica may, at its own option, modify the list of “updateable” software. Cybernetica may require the Customer to prove that he holds a valid free license of “updateable” software.

## **5. RESERVED RIGHTS**

Except as expressly specified in this agreement or required by applicable law, Customer shall not (a) separate the Software from the carrier of license, (b) rent, lease, sell or otherwise transfer the Software or his rights under this agreement to any third parties, or (c) decompile, disassemble, decipher or reverse engineer the Software. This paragraph survives the termination of this agreement.

## **6. TERM AND TERMINATION**

This agreement is effective until termination. Customer may terminate the agreement by destroying all copies of the Software and license carriers.

## **7. JURISDICTION**

This agreement shall be construed and governed in accordance with the laws of the Republic of Estonia and is considered to have been entered into on the territory of the Republic of Estonia. Customer shall attorn to the jurisdiction of the courts in the Republic of Estonia. This paragraph survives the termination of this agreement.



## SSLeay copyright notice

Copyright (C) 1997 Eric Young (eay@cryptsoft.com)

All rights reserved.

This package is an SSL implementation written by Eric Young (eay@cryptsoft.com). The implementation was written so as to conform with Netscapes SSL.

This library is free for commercial and non-commercial use as long as the following conditions are aheared to. The following conditions apply to all code found in this distribution, be it the RC4, RSA, lhash, DES, etc., code; not just the SSL code. The SSL documentation included with this distribution is covered by the same copyright terms except that the holder is Tim Hudson (tjh@cryptsoft.com).

Please note that MD2, MD5 and IDEA are publically available standards that contain sample implementations, I have re-coded them in my own way but there is nothing special about those implementations. The DES library is another mater :-).

Copyright remains Eric Young's, and as such any Copyright notices in the code are not to be removed. If this package is used in a product, Eric Young should be given attribution as the author of the parts of the library used. This can be in the form of a textual message at program startup or in documentation (online or textual) provided with the package.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement: "This product includes cryptographic software written by Eric Young (eay@cryptsoft.com)" The word 'cryptographic' can be left out if the rouines from the library being used are not cryptographic related :-).

4. If you include any Windows specific code (or a derivative thereof) from the apps directory (application code) you must include an acknowledgement:  
"This product includes software written by Tim Hudson  
(tjh@cryptsoft.com)"

THIS SOFTWARE IS PROVIDED BY ERIC YOUNG "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The licence and distribution terms for any publically available version or derivative of this code cannot be changed. i.e. this code cannot simply be copied and put under another distribution licence [including the GNU Public Licence.]

The reason behind this being stated in this direct manner is past experience in code simply being copied and the attribution removed from it and then being distributed as part of other packages. This implementation was a non-trivial and unpaid effort.