Design of a VPN Software Solution Integrating TCP and UDP Services

Javier Lopez[1], Jose A. Montenegro[1], Rodrigo Roman[1], and Jorge Davila[2]

[1] Computer Science Department, E.T.S. Ingenieria Informatica
Universidad de Malaga, Spain
{jlm,monte,roman}@lcc.uma.es

[2] Computer Science Department,
Universidad Politecnica de Madrid, Spain
jdavila@fi.upm.es

Abstract. Virtual Private Network (VPN) solutions mainly focus on security aspects. Their main aims are to isolate a distributed network from outsiders and to protect the confidentiality and integrity of sensitive information traversing a non-trusted network such as the Internet. But when security is considered the unique problem, some collateral ones arise. VPN users suffer from restrictions in their access to the network. They are not free to use traditional Internet services such as electronic mail exchange with non-VPN users, and to access Web and FTP servers external to the organization. In this paper we present a new solution, located at the TCP/IP transport layer that, while maintaining strong security features, allows the open use of traditional network services running over TCP and UDP. The solution does not require the addition of new hardware because it is an exclusively software solution. As a consequence, the application is totally portable. Moreover, the implementation is located at the transport layer; thus, there is no need to modify any software previously installed, like FTP, Telnet, HTTP, electronic mail or other network applications.

1 Introduction

The needs of digital communications for most of organizations have grown very much during last years because of different reasons. For instance, globalization of economy increases the demand of telecommunication among branch offices, and inter-companies agreements impose that resources are shared. Thus, decreasing the cost of telecommunication infrastructures is imperative.

Traditionally, companies have used leased lines with that purpose. The most representative example is Frame-Relay service [2],[4], which is based on the transfer of information frames between intermediate switching offices. The service, that uses permanent virtual circuits (PVCs) through telephone network routers, presents some drawbacks:

- It becomes expensive because connections remain open permanently.

- The architecture creates large latency periods because of the poor connectivity between intermediate routers.
- Full connectivity requires the increment of PVCs and, hence, of intermediate network routers; but the cost of avoiding routing problems in this way is high.
- The number of companies that offer Frame-Relay services is small compared to the number of Internet Service Providers (ISPs), so competitiveness is more limited.

On the other hand, open networks offer a more profitable solution than leased lines. Thus, for example, *Virtual Private Networks*(VPNs) use relatively low-cost, widely available access to public networks to connect remote sites together safely. Network architectures defined by VPNs are inherently more scalable and flexible than classical WANs, and they allow organizations to add and remove branch offices into their systems in an easy way.

However, and as shown later, the study of the different TCP/IP stack layers reveals that the different solutions that enable establishing a VPN essentially focus on security aspects. Their main aims are to isolate a distributed network from outsiders and to protect the privacy and integrity of sensitive information traversing the non-trusted open networks, as the Internet. These approaches fail to be complete.

The main drawback in conceiving the security problem as the unique target is that VPN users suffer from restrictions in accessing the Internet. That is, they cannot freely use traditional services such as electronic mail exchange with non-VPN users, and cannot freely access Web and FTP servers external to the organization. Actually, within the same application, it is a difficult task to enable generic Internet access to VPN users and, at the same time, to provide a strong enough security model for the organization.

This work presents an approach to this problem that is an extension of a previous approach of the same authors. We have developed a new security integrated solution for VPNs that, while maintaining strong security features, allows the open use of traditional Internet services that run not only over TCP (the case of our previous solution), but also over UDP. The solution does not require the addition of new hardware because it is an exclusively software solution. As a consequence, the application is totally portable. Moreover, the implementation is located at the transport layer; thus, it allows using those services running over TCP (like FTP, Telnet, WWW, etc.) and also those running over UDP (that is, real-time applications like audio and video-conferences), which are increasingly being used in the Internet. Additionally, and as we will show, there is no need to modify any software previously installed.

The paper is organized in the following way: Section 2 introduces the different cryptographic solutions used in the TCP/IP stack layers to establish private network communications over the Internet, focusing on their advantages and disadvantages. Section 3 explains the two-module architecture of the new system. Section 4 describes the operation of the first module, called *Secsockets*, an extension of the traditional socket interface to which security features have been added. Section 5 outlines the operation of *VPN-Insel*, the second module. This is the part of the system that really establishes the VPN, and resides on the interface of the previous module. Section 6 shows a real example of how the new solution works, and section 7 presents concluding remarks.

2 Security solutions at the TCP/IP Layer

There are different security solutions in the TCP/IP stack layers that can be used to establish confidential and authenticated communications over Internet. In this section we briefly review the most relevant solutions in those layers, pointing out their advantages and disadvantages.

It is necessary to point out that all the solutions that have been developed to work above the network layer are oriented to TCP services. To the best of our knowledge there is no widely used standard mechanism that provides security to UDP services.

2.1 Data Link Layer

The method used in the lowest layer of the stack is point-to-point encryption. The architecture of link level encryption defines a completely isolated connection between two systems. On the one hand, physical access is restricted because every connected host belongs to the organization; on the other hand, logical access is restricted too because information is encrypted throughout the transmission. This solution does not necessarily require the use of the TCP/IP stack protocols because these ones work at higher levels.

Point-to-point encryption is a simple concept that makes it, from a security standpoint, a good VPN solution. However, it has certain properties that make it hard to use in every application. Firstly, system users have no choice regarding encryption. They can either link to a host using encryption or they cannot communicate at all. Therefore, it does not provide generic Internet access because it protects from hostile outsiders, but also blocks access to beneficial outsiders. Secondly, point-to-point encryption scales very poorly. It becomes complex and expensive to establish links to new hosts that are installed in the network system.

2.2 Network Layer

The solution used at the network layer is the encryption of the data fields in the IP packets. The set of *IPSEC protocols* (IP security protocols) [1], is a part of IPv6, the IPs future version and it is designed to provide privacy and/or data forgery detection.

For this purpose, IPSEC defines two optional packet headers: *Authentication Header* (AH), and *Encapsulating Security Payload* (ESP). Both headers contain a numeric value called the *Security Parameter Index* (SPI), which is used by a host to identify the cryptographic keys and the security procedures to be used.

In order to communicate, each pair of hosts using IPSEC must negotiate a security association. The security association establishes what types of protection to apply, how to encrypt or authenticate, and which keys are needed. The packet IP address and the packet SPI in the packet header determine the security association applied to an IPSEC header.

The combination of IPSEC protocols with routers or firewalls constitutes a VPN solution, and it enables the use of traditional Internet services. Nevertheless, and because of its location, it is difficult to establish a security relationship between two applications, which makes difficult to provide the same granularity and control over authentication and encryption. Subsequently, and important disadvantage of this solution is that the system automatically applies protection according to the security associations between the host; users cannot decide when to apply security mechanisms, which is inefficient for most cases. An additional disadvantage is that architecture blocks communications to other non-trusted computer systems and networks, so it does not enable generic Internet access.

2.3 Transport Layer

There are some standardized solutions at the transport layer for TCP, but as stated, not for UDP. The implementation of Netscapes *Secure Socket Layer* (SSL) [11] protocol stands out from other solutions. It is widely used in conjunction with World Wide Web service and includes capabilities for authentication., encryption and key exchange. Some similar alternatives to this solution are *Transport Layer Security* (TLS) [3], proposed by the IETF TLS working group, and the Microsoft compatible protocol Private Communication Technology (PCT) [8].

The SSL protocol is integrated in both Web client and server software, and protects data transfer between them. Client and server negotiate the type of protection by setting up a group of cryptographic parameters that includes a strong encryption algorithm and a shared secret key. A public key cryptosystem is used to facilitate authentication and distribution of the secret key.

Applying security at the transport level in this way provides good control over the mechanisms to be used, because a web browser can choose whether a particular connection will be secure or not.

However, a serious disadvantage of this scheme is that it requires the integration of SSL protocol software into the application itself; that is, it is necessary to modify every network application that needs to use these security mechanisms. Therefore, this is not a general solution for a VPN because it only solves specific network services.

2.4 Application Layer

There are specific solutions at the application layer, but again, only for TCP-based services, not for UDP-based ones. These solutions are, for instance, electronic mail (PGP, PEM, S-MIME) [17] [7] [12], the file transfer protocol (Secure-FTP) [5], the hypertext transfer protocol (S-HTTP) [15], etc. Regarding UDP, each application based on this protocol must provide its own security solution.

The lack of a homogeneous solution becomes a serious problem because each application provides its particular encryption and key management mechanisms. Moreover, even if only solution existed, providing a VPN solution at this level would make necessary to control that every single user updates his/her non-secure applications with the selected security mechanisms. There is no doubt that this management would be hard, not only for the establishment of the VPN but also for any little change in the configuration of the VPN.

3 Architecture of the New Scheme

The scheme we introduce is an entirely software solution that works at the transport level. The solution adopts the advantages of IPSEC because it facilitates the use of traditional Internet services. However, the new design presents notable differences with IPSEC too because it allows users to decide when to apply security mechanisms. It also enables access to any remote node; hence, any VPN user can connect to other VPN and non-VPN users. Additionally, it does not require the modification of the software of traditionally insecure applications such as FTP, HTTP, Telnet, electronic mail, etc. As a result, the scheme is a simple and cheap solution for those organizations that want to install a VPN.

A generic organization scenario that has been used for the development of the new solution is depicted in figure 1. There is a main entity that centralizes control of the organization and represents the Headquarters. There are other entities that represent the branch offices of the organization, the secondary entities. Every secondary entity is located in a different LAN.

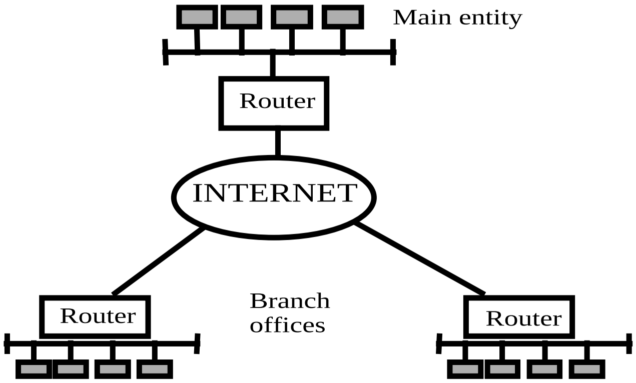

Fig. 1. Scenario of a VPN

The new design is divided into two sublevels, Secsockets y RPV-Insel. The module Secsockets is the lower sublevel. It is an interface that works on top of the traditional socket interface that provides access to TCP. The service offered by Secsockets is the transmission of data from an origin process to a target one through a secure and authenticated channel.

The module VPN-Insel is the upper sublevel. It uses the services provided by Secsockets and, at the same time, it supports the applications with the VPN services that these ones require. Next sections show the operation of both modules in detail.

4 SecSockets

Secsockets is an extension of the traditional socket interface, and enables authenticated and confidential communications. That is, Secsockets provides the same functions as the socket interface and, in addition, provides authentication, confidentiality, and integrity.

The goal of this interface is to provide a secure delivery service for both TCP and UDP, while facilitating its use. This interface is based on a client/server architecture, in such a way that the process that starts the communication plays the role of the client, and the process that attends to the connection at the other end plays the role of the server. The operation of Secsockets is divided into two phases: firstly a connection phase, and secondly, the communication phase itself.

4.1 Connection Phase

Firstly, during the connection phase, mutual authentic ation is carried out. Secondly, there is a negotiation of the parameters (hash function, encryption algorithm, and optional compression algorithm) that will be used during the communication phase.

In this phase the two interacting parts have no secret key to share as yet. So, the use of public key certificates provides mutual authentication and confidentiality during parameter negotiation. This enables communicating the secret key, so avoiding the risk of a third party interception, and facilitating the identification of VPN users because it provides a method to implement digital signatures.

We must point out that during this phase the TCP protocol is used regardless the type of service that needs the creation of a secure channel. The reason of using TCP is that we need a reliable protocol for parameters interchange, and is widely known that UDP is not reliable at all.

The following are the steps that are done:

1. The client and the server interchange the digital certificates.
2. The client sends, encrypted with the servers public key, a connection request. This message contains the security parameters (hash function, symmetric key algorithm, a random value and, optionally, data compression). The hash functions that can be used are MD5 [13] and SHA [10]. The symmetric key

encryption algorithms that can be used are DES [9], IDEA [6], Blowfish [16] and RC4 [14]. In case data compression is selected, the GZIP algorithm is used.

3. Once the request is processed, the server sends a message to the client indicating acceptance or rejection, encrypted with the clients public key. In case negotiation is accepted, it includes a random number that will be used for the calculation of the key. If the channel to be opened is and UDP one, then the message will include the port number where the server will listen messages from the client.
4. Both client and server calculate the secret key that will be used during the next phase: the communication phase. Random values exchanged in step 2 and 3 and the hash function negotiated in these steps are used to calculate a bit string. The string is obtained from the following expression:

   ```
   H( H(aleatorio_Cliente) XOR H(aleatorio_Servidor) )
   ```

   Additionally, and in the case the channel to be opened is UDP, the server closes the TCP connection with the client, and opens a UDP connection to wait messages from this one.

4.2 Communication Phase

The operations performed during the communication phase (or transmission phase) are, for each message: a) fragmentation of the message into frames; b) data field compression, if this was decided during the initial negotiation; c) calculation of hash value for each frame; and, d) frame encryption and MAC calculation.

Figure 2 shows the composition of each frame. Maximum size of the message depends on the length of packets accepted by the network; thus it is a value that can be configured. Obviously, after the reception of each packet, the receiver has to undo all previous operations.

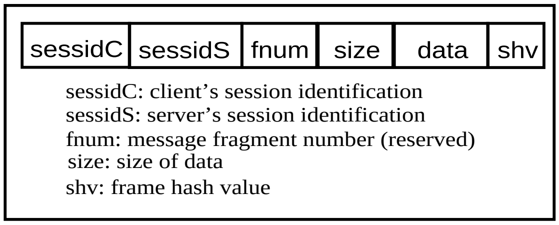
sessidC
sessidS
fnum
size
data
shv
sessidC: client's session identification
sessidS: server's session identification
fnum: message fragment number (reserved)
size: size of data
shv: frame hash value

Fig. 2. Format of communication frame

4.3 Functions of SecSockets

The service offered by Secsockets is to send data from a source process to a target one through a secure and authenticated channel. Secsockets works as a connection oriented communication protocol based on the client/server paradigm.

Secksockets provides a group of functions. From the programming interface point of view these functions work in a client/server mode. As we will see later, the reason is that levels above Secksockets need an interface that helps in the creation of parallel client/server systems. Also, working on this way, it is possible to use either TCP or UDP for the communication by changing only one parameter.

The functions are the following ones:

- *sec_init( )*: This function creates a connection point at the server's end. The point is initialized and the function assigns it to a local communication port. Then the server remains in an idle state waiting for a client connection-request through that port.
- *sec_accept( )*: This function is invoked by the server to positively acknowledge an incoming connection and immediately provides it with a new socket. After accepting the connection the negotiation phase, under TCP, starts at server's end. A socket address, either TCP or UDP, according to the mentioned parameter, is returned. Security parameters are returned too.
- *sec_connect( )*: The client invokes this function in order to create a final connection point. After a negotiation phase under TCP starts at the clients end, authenticating the other end and agreeing on the security parameters. A TCP or UDP socket address and security parameters are returned.

From this point on, and following the parameters settled on during the negotiation phase, it will be possible to perform securely the communication phase. The functions of this phase are described next:

- *sec_recv( )*: This function enables data reception through a secure socket connection, decrypts them, checks their authenticity, and decompresses them if necessary
- *sec_send( )*: This function is symmetrical to the previous one. It compresses data if this was negotiated, calculates the hash value, encrypts data, and finally, sends them through the socket. These operations are performed according to the negotiations of the client and server.
- *sec_close( )*: This function erases the socket.

5 RPV-Insel

The module VPN-Insel uses the services provided by the interface Secsockets. At the same time it supports the communication applications that run at the upper level and provides them with the characteristic services of a VPN. So, by making use of Secsockets security mechanisms, VPN-Insel offers users the possibility of registration into the VPN, participating in on-line connections, like HTTP, Telnet, or FTP, and off-line connections, such as ele ctronic mail. Moreover, this module manages the VPN. That is, VPN-Insel initializes the private network, setting the configuration parameters. Once the private network is working, VPN-Insel controls which users belong to it and can use its services, and which new users and secondary entities can register. It also allows system supervisors to install servers for HTTP, Telnet, FTP, SMTP and videoconference which are of exclusive use for users belonging to the VPN.

Basically, the VPN-Insel processes and their interrelations are those shown in figure 3. Now we detail the operation of these processes:

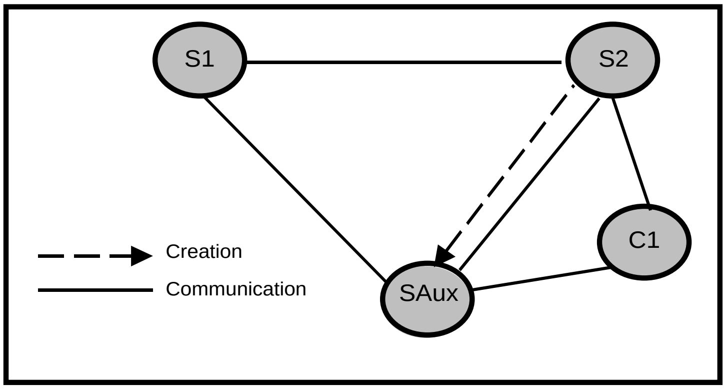

Fig. 3. RPV-Insel scheme

1. *Clients* (C1): represents the client software for those users who communicate from a LAN.
2. *Main server* (S1). It runs in the main entity's computer system (main LAN in the organization). This type of process centralizes the VPN management, controls other types of server processes, maintains a database with the information related to these servers (IP addresses), and provides information about the VPN. There is only one of these processes in each VPN.
3. *Secondary Server* (S2): This server controls the local users (the users inside its LAN) and decides which of them can log in to the VPN. It also attends to their requests in the same way as the main server does. One secondary server exists in each entity, i.e., in each LAN of the organization.
4. *Auxiliary Servers* (SAux): Auxiliary servers are created by the main server or by a secondary one to attend to users during a particular connection. They are intermediate processes within secure communications and make use of the functions that the SecSockets interface provides, in such a way that security mechanisms are transparent to client processes.

5.1 Detailed Operation of VPN-Insel

RPV-Insel starts working when the VPN administrator starts-up the main server (S1). After this, the following sequence of events takes place:

1. S1 initializes the secondary servers database.
2. S1 starts a secondary server (S2) for its own LAN (main entitys LAN).
3. S1 remains open waiting for requests. The requests come from other LANs secondary servers or remote users.
4. Each secondary server, started by the corresponding LAN administrator, is registered in the main server, in order to integrate the LAN into the VPN.
5. Each secondary server stays open for: a) requests from users in its own LAN; or b) requests from remote auxiliary servers that want to use its LANs resources.

Then clients can start using the VPN to communicate securely with other members of the private network. The communication among VPN members follows these steps:

1. The user decides, by using the application interface of VPN-Insel, which network application to use (e.g. FTP) and the name of the computer (or IP address) where the real server (e.g. FTP server) is running.
2. The application interface of VPN-Insel sends a communication request to its secondary server. This one creates an auxiliary server that exclusively manages that secure connection. Afterwards, the secondary server remains open for new requests
3. The auxiliary server connects with the secondary server in the other end, and a secure channel is established.
4. At this moment, VPN-Insel runs the client program, but it does not provide it with the computer address where the real server is running. VPN-Insel provides the client with the address where the auxiliary server is running inside itis own client LAN.
5. From this point on, the communication between the client and the server processes takes place in a secure way through the channel that connects both auxiliary servers.

6 Examples

In this section we show two scenarios that clarify how our solution works for TCP and UDP services. In the first scenario we show how an FTP session is secured inside the VPN. The second scenario describes an audio-conference between two users.

The example scenario for FTP service is shown in figure 4. A client C1 wants to connect to one FTP server in the VPN. The sequence of events is the following one:

1. C1, started by the user, request a connection to S2, its secondary server.

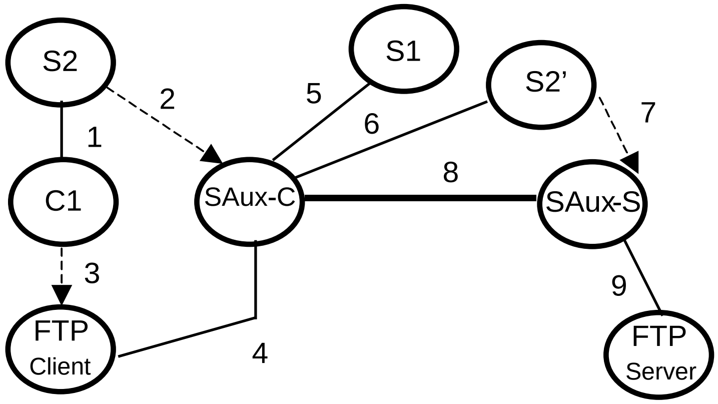

Fig. 4. Scheme of processes for a FTP service

2. If the user is a legitimate VPN user, S2 creates an auxiliary server, SAux-C, to manage the connection.
3. At the same time, process C1 runs the FTP client software.
4. FTP client software connects to SAux-C.
5. SAux-C request S1 for the number of the port and the host where process S2 is running.
6. SAux-C request S2 to create SAux-S, the remote auxiliary server.
7. S2 creates SAux-S.
8. SAux-C connects to SAux-S, and they set up the secure channel.
9. SAux-S connects to the FTP server.

Regarding the audio-conference service, the example scenario is shown in figure 5. Two clients, C1 and C2 want to establish a secure communication. The steps performed are the following ones (we suppose that C! is the one that starts the communication):

Steps 1 and 2 are the same one as in the previous scenario.

3. SAux-S request S1 for the number of the port and the host where process S2 is running.
4. SAux-S notifies S2, y S2 notifies user C2. SAux-S adopts the role of server, and waits for the request to start the communication.
5. C2, started by the user, request a connection to S2, its secondary server.
6. If the user belongs to the VPN, S2 creates an auxiliary server SAux-C, that will communicate with to SAux-S.
7. SAux-C connects to SAux-S, and they set up the secure channel.
8. The audioconference program runs in both ends.

In both cases, the secure communication is established between the auxiliary servers (thick line). They work as gateways, receiving the frames, decrypting the information and sending them to the specific receiver.

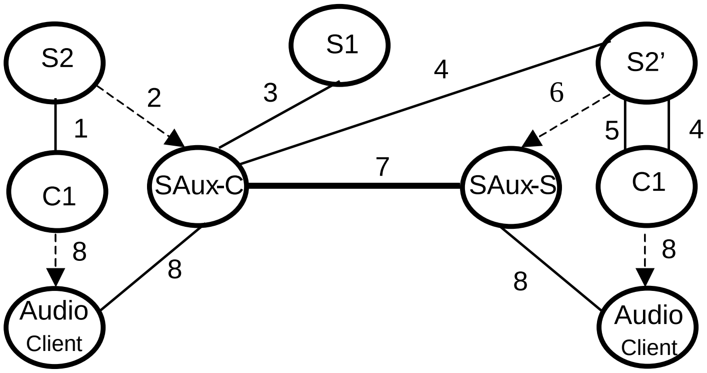

Fig. 5. Scheme for a videoconference service

7 Conclusions

The study of the different TCP/IP stack layers reveals that the different solutions that enable establishing a VPN pay most of their attention to security aspects. These solutions focus on the isolation of a distributed network from outsiders and on the protection of sensitive information traversing Internet. But the users of these systems cannot freely use traditional services such as electronic mail exchange with non-VPN users, and cannot freely access Web and FTP servers external to the organization.

In this paper we have presented a new solution for the implementation of VPNs at the transport layer that, while maintaining strong security features, allows the open use of traditional Internet services that run over TCP and UDP layers. As a consequence, it enables access to any remote node in the VPN and outside the VPN. The solution is exclusively software, so its deployment does not require the addition of new hardware or the modification of any existing one.

The solution allows an homogeneous control of all network services, which is necessary to gain higher user confidence regarding the entire system. Moreover, it does not require the modification of the software of traditionally insecure applications such as FTP, HTTP, Telnet, electronic mail, etc. As a result, the scheme is a simple and cheap solution for those organizations that want to install a VPN.

References

1. Atkinson, R. Security Architecture for the Internet Protocol. RFC 1825, August 1995.
2. Black, U. Frame-Relay: Specifications and Implementations, McGraw-Hill, 1994.
3. Dierks, T. ; Allern, C. The TLS Protocol Version 1.0. RFC2246, January 1999.

4. Harbison, R. Frame-Relay: Technology for our Time, LAN Technology, December 1992.
5. Horowitz, M.; Lunt, S. FTP Security Extensions. RFC 2228, October 1997.
6. Lai, X.; Massey, J. Hash Functions Based on Block Ciphers. Advances in Cryptology. EUROCRYPT '92, Springer-Verlag, 1992, pp. 55-70.
7. Linn, J. Privacy Enhancement for Internet Electronic Mail: Part I Message Encipherment and Authentication Procedures. RFC 989, February 1987.
8. Microsoft Corporation. The Private Communication Technology, 1997.
9. National Bureau of Standards. Data Encryption Standard. U.S. Department of Commerce, FIPS pub. 46, January 1977.
10. National Institute of Standards and Technology, NIST FIPS PUB 180. Secure Hash Standard. U.S. Department of Commerce, May 1993.
11. Netscape Communications. SSL 3.0 Specification.
12. Ramsdell, B. S/MIME Version 3.1 Message Specification, Internet Draft, February 2002.
13. Rivest, R. The MD5 Message Digest Algorithm. RFC 1321, April 1992.
14. Rivest, R. The RC4 Encryption Algorithm, RSA Data Security, Mar 1992.
15. Schiffman, A.; Rescorla, E. The Secure Hypertext Transfer Protocol. Internet Draft, June 1998.
16. Schneier, B. Description of a New Variable-Lenght Key, 64-Bit Block Cipher (Blowfish). Fast Security Workshop Proceedings, Springer-Verlag, 1994, pp. 191-204.
17. Zimmermann, P.R. The Official PGP Users Guide. MIT Press, 1995.