Secure TCP — providing security functions in TCP layer

Toshiyuki Tsutsumi[1]
Suguru Yamaguchi[2]
Nara Institute of Science and Technology

Abstract

Currently the Internet communication is under threat of attacks, such as unauthorized accesses, message modifications and eavesdroppings. In order to guard it against these attacks, several mechanisms and protocols have been proposed in recent years. They were designed to perform their security functions in network and application layers. However, providing security functions in the transport layer give us many useful features, thus we propose a new security mechanism called "Secure TCP" in the transport layer. This mechanism can be one of the basic units to form trusted network applications. In this paper, we discuss design and implementation of the Secure TCP.

1 Introduction

In the Internet, some of application protocols handle important and/or sensitive data. However, because of lack of security functions in the current TCP/IP protocols, they are under threat of attacks, such as unauthorized accesses, message modifications and eavesdroppings. Several mechanisms and protocols to guard the Internet communications have been proposed against these attacks, such as Privacy Enhanced Mail(PEM)[1][2][3][4], Privacy Enhanced Telnet(PET), Kerberos authentication service[5], swIPe[6] and IP/Secure[7].

In order to reduce overhead for adding security functions to current existing application protocols and systems, various kinds of security services should be available. Currently, adding security functions onto the host/host communication are performed by IP level security such as swIPe or IP/Secure. However, for process/process communication, enhancement of each application protocol has been the only way to add security functions. Therefore, because of the overhead to enhance application protocols, only few of applications are now providing security functions such as data encryption or user authentication.

In order to reduce this overhead and accelerate the integration of security functions to currently available network applications, security functions should be provided in the transport layer. So, we propose and have developed a new mechanism called "Secure TCP" — a security enhanced TCP[8] to guard communication against the attacks. Since TCP is widely used for various applications, the Secure TCP can be applied to many of current existing network applications. The Secure TCP can be a common platform for the secure network applications to achieve secure process/process communication.

Furthermore, the Secure TCP provides a solution for the smooth migration to secure services for existing applications. The Secure TCP is interoperable with the classical TCP since it uses a natural extension of TCP three-way handshake to exchange session keys for TCP segment encryption, while swIPe requires a complicated swIPe neighbor discovery mechanism. This feature is important to apply it to the existing network environment.

This paper is organized as follows: Section 2 shows problems of a current version of TCP. Section 3 gives our goal and assumption of Secure TCP. Section 4 describes a design of Secure TCP protocol. Section 5 discusses the protocol format in our implementation. Section 6 shows our implementation on BSD UNIX System and a simple performance measurement. Section 7 argues future works of Secure TCP development.

2 Security problems of TCP

In this section, we address problems of the current version of TCP.

TCP provides fundamental communication service and is used by many application protocols. Therefore, TCP becomes one of the most popular protocol. But TCP has several security problems as follows.

- TCP can not guard a segment against the message modification attacks.

  TCP has a checksum field. This field is used in order to identify a modification of a segment. However, since this field is not protected against the message modification attacks, it is possible to modify any TCP

[1] tosiyu-t@is.aist-nara.ac.jp. He also works for Hitachi Software Engineering Corp. and can be reached at tutumi@ori.hitachi-sk.co.jp.
[2] suguru@is.aist-nara.ac.jp.

segments. Moreover, there is no ways for peer entities to find out the message modification attacks.

- TCP can not keep segment data secure against the message eavesdropping attacks.

  TCP transports stream data used in the application layer. Since TCP does not provide any data encryption functions, anyone can gain any valuable information.

- TCP can not protect connections against the unauthorized access attacks.

  TCP certifies a peer entity by a source IP address and a port number. However, it is possible to modify the source address and port number.

3 Goals

The goals of the Secure TCP are shown as follows.

- Provide data integrity for the TCP segment. The data integrity is a security service where right entities can identify data modified by unauthorized entities.

- Provide data confidentiality for TCP segment data. The data confidentiality is a security service where data should be unreadable by unauthorized entities.

- Provide a negotiation service for using various security functions in process/process communication.

- Keep connectivity(interoperability) with a current version of TCP.

Secure TCP does not provide any authentication services by itself. Because several third-party authentication services such as Kerberos are now available, the Secure TCP relies on the authentication services. In this sense, each peer entities in TCP layer has its certificates and the Secure TCP uses it for authentication.

4 Secure TCP Protocols

In this section, we describe design of the Secure TCP. We call the current version of TCP as "classical TCP" throughout the rest of this paper.

In the Secure TCP, its essential parts are the security service negotiation, the key exchange for encryption and TCP segment encryption. In the following subsections, we discuss the design of these parts.

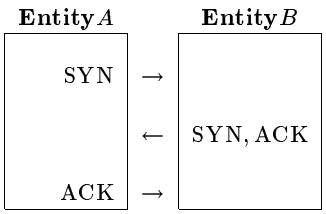

Figure 1: Three-way handshake of TCP

4.1 Extension of Three-Way Handshake

In the classical TCP, the opening phase of TCP connection establishment uses a three-way handshake to negotiate the initial sequence number for each direction. In the Secure TCP, this phase also handles negotiations of the method of TCP segment encryption and encryption key exchanges. Therefore, in the Secure TCP, the three-way handshake is extended to handle the negotiation.

The three-way handshake is a connection establishment protocol. The setup procedure of the protocol when entity A initiate shown in Figure 1. First, A sends a SYN segment to B in order to check that B gets ready for establishing TCP connection. Second, when B receives the SYN segment A sent and is ready to start the TCP session, it sends a SYN and ACK segment back to A. This ACK advertises an arrival of the first SYN segment to A. Finally, A sends ACK segment for the second SYN and ACK segment B sent.

In the Secure TCP, this three-way handshake is extended in order to deliver negotiation data and key exchange data. The procedure of the extended protocol is shown in Figure2. First, A sends a SYN segment with the negotiation data (NEGO) to B. The negotiation data is a list of encryption methods available at A used for segment protection. In other words, A advertises its encryption capability to B using this negotiation phase. If B receives the SYN+$\mathrm{NEGO_A}$ segment from A, then B checks its own encryption capability and picks one of the methods advertised by A in the segment. The method B picks ($\mathrm{NEGO_B}$) will be used for TCP segment encryption in the further segment exchanges. Then, B returns SYN+ACK segment with a public key($\mathrm{KEY_P}$) for encryption. This public key will be used for encryption of session keys applied to the stream encryption. Next, A sends a session keys ($\mathrm{KEY_S}$) for the stream encryption to B. The session keys will be applied to streams for data encryption. Finally, B sends back ACK segment for the

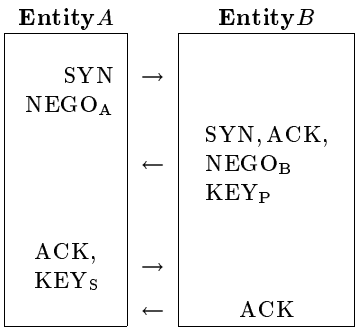

Figure 2: Extension of Three-way handshake

ACK+KEY_S segment A sent.

4.1.1 Security Service Type Negotiation

The Secure TCP can choose several kinds of cipher methods used for the data integrity and confidentiality, and select various hash algorithms for the data integrity.

In the negotiation phase, the Secure TCP peer entities exchange a security service type that indicates both cipher methods and a hash algorithms. The security service type used in the Secure TCP is listed in the Table 1. Note that this type is not an index to express the strength of these methods.

A negotiation procedure is carried out as follows. A sender entity sends a list of types to a peer entity. The list order indicates the priority the sender wants to use. For example, in the case the sender sends "421", it wants to use MD5 and DES for its data integrity and DES (CFB 1byte) for its data confidentiality, as its first priority.

The receiver entity chooses one of the methods sender advertised in the list, and sends back to it. If the receiver can't pick any methods in the list, then it can give up the connection establishment or proceed with the classical TCP. In the case that SYN+ACK segment is sent back to the sender, the sender can also select whether it resets the connection and retry with other list of security method or go ahead with the classical TCP.

A example of the security service type is shown in Table 1. A symbol "—" indicates no methods. Type 0 shows providing no security service. Type 1 shows providing data integrity method only. Type 2, 3 and 4 show providing data integrity and data confidentiality methods. In order to put data confidentiality into practice, each different cipher methods is used in type 2, 3 and 4.

	integrity		confidentiality
type	hash	cipher	cipher
0	—	—	—
1	MD5	DES	—
2	MD5	DES	DES(ECB)
3	MD5	DES	DES(CBC)
4	MD5	DES	DES(CFB 1byte)

Table 1: Security Service Type

IP_X	Network Address of a host X
WK_c	Shared Session key (confidentiality)
WK_i	Shared Session key (integrity)
SK_X	Secret Key for a host X
PK_X	Public Key for a host X
VP	Validity Period of a Public Certificate
$(M)^K$	Public Enc. of Message M with Key K

Table 2: Symbols used in key distribution

4.1.2 Key Exchange

Another objective of the negotiation phase is key exchange between peer entities. This exchange should be done in a secure manner. In this key exchange, "a public key certificate," discussed later, is introduced in order to protect this key exchange against eavesdroppings. After the key exchange, peer entities share session keys for datagram integrity and confidentiality.

Before a procedure of the key exchange is carried out, peer entities of communication have a public key certificate that is signed by a reliable certificate authority(CA).

We explain symbols and notations used throughout the rest of this paper. They are summarized in Table 2.

Public Key Certificate Issue The procedure to issue a public key certificated by CA to a host is shown in Figure 3(a).

CA makes PK_{CA} available to the public and maintains its own secret key SK_{CA}. The procedure for issuing a public key certificated is put into practice as follows.

1. A host J sends a public key PK_J and its network address IP_J to CA.

2. The CA adds VP which indicates a term of validation to (PK_J,IP_J), and encrypts it with CA's secret key. Resulting certificate $(PK_J,IP_J,VP)^{SK_{CA}}$ is sent to the host J.

Procedure of Key Exchange The procedure of shared session keys exchange is illustrated in Figure 3(b).

The procedure is carried out as follows.

Host J		CA
PK_J, IP_J	$\rightarrow$	$(PK_J, IP_J, VP)^{SK_{CA}}(Sign)$
	$\leftarrow$	$(PK_J, IP_J, VP)^{SK_{CA}}$

(a) Procedure for Public Key Certificate Issue

Host A		**Host** B
$(PK_B, IP_B, VP)^{SK_{CA}PK_{CA}}(dec)$ $\Rightarrow PK_B, IP_B, VP$	$\leftarrow$	$(PK_B, IP_B, VP)^{SK_{CA}}$
$\left\{ \begin{array}{l} (PK_A, IP_A, VP)^{SK_{CA}} \\ (WK_c, WK_i)^{SK_A PK_B} \end{array} \right\}$	$\rightarrow$	$\left\{ \begin{array}{l} (PK_A IP_A, VP)^{SK_{CA}PK_{CA}} \\ (WK_c, WK_i)^{SK_A PK_B SK_B PK_A} \end{array} \right\}(dec)$ $\Rightarrow \left\{ \begin{array}{l} PK_A, IP_A, VP \\ WK_c, WK_i \end{array} \right\}$

(b) Procedure for Key Exchange

Figure 3: Key Issue and Exchange Procedures

1. Host B sends $(IP_B, PK_B, VP)^{SK_{CA}}$ to Host A.

2. Host A decrypts $(IP_B, PK_B, VP)^{SK_{CA}PK_{CA}} = (IP_B, PK_B, VP)$ and verifies the public key for the IP_B is PK_B.

3. Host A encrypts shared keys WK_c, WK_i with its own secret key SK_A and PK_B, and sends resulting $(WK_c, WK_i)^{SK_A PK_B}$ together with the A's public key certificate $(IP_A, PK_A, VP)^{SK_{CA}}$ to Host B.

4. Host B decrypts $(IP_A, PK_A, VP)^{SK_{CA}PK_{CA}} = (IP_A, PK_A, VP)$ and verifies the public key for the IP_A is PK_A. Then, Host B decrypts $(WK_c, WK_i)^{SK_A PK_B}$ to obtain the conventional shared keys.

4.2 State Transition

The operation of the Secure TCP can be explained with "state transition machine". The Secure TCP state transition machine is shown in Figure 4. In "ESTABLISHED" state, peer entities should deal with the Secure TCP segment. Exceptionally, in the state "RECV ESTABLISHED", the entity deals with a received TCP segment as the Secure TCP and a TCP segment will be sent as the classical TCP.

The entity A in the Figure 2 makes a following state transmission. The entity A begins in the state "CLOSED". First, A sends a $SYN+NEGO_S$ segment to B actively and move to the state "SYN SENT". Next, A receives a $SYN+ACK+NEGO_R + KEY_S$ segment, sends $ACK+KEY_R$ and traverses to the state "RECV ESTABLISHED". In this state, A can receive only a Secure TCP segment. Finally, A receives

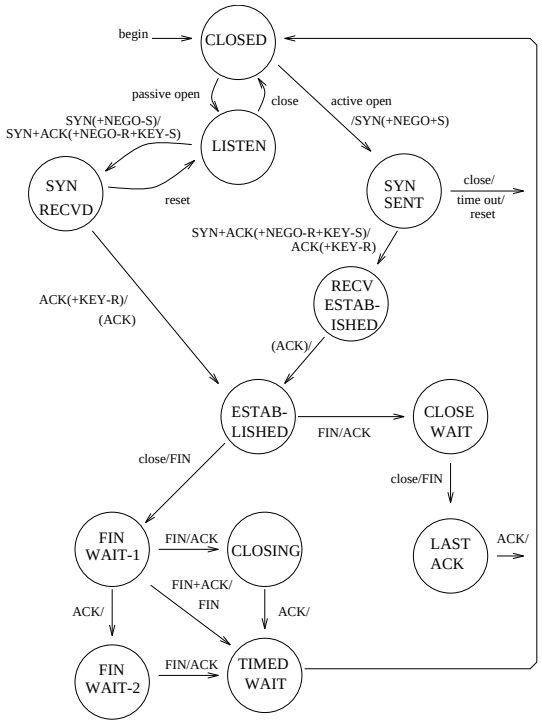

Figure 4: State Transition Machine

a ACK segment and moves to the state "ESTABLISHED", and then begins data transfer with the Secure TCP segment.

4.3 Security Service

4.3.1 Data Integrity

The Secure TCP must protect following parts against message modification attacks: TCP header and TCP segment data.

The TCP header has TCP control informa-

tion, such as checksum, acknowledgment number, sequence number and control flags. These information are essential for TCP protocol, and thus they must be protected. Furthermore, since the TCP segment data includes real application data that may be potential target for attackers, its integrity must be maintained.

We introduced MAC (Message Authentication Code) in the Secure TCP segment to identify if it is modified by attackers. MAC is made by following procedures.

1. The Message Digest (MD) is calculated by applying a hash function as follows;

$$\begin{aligned} \text{MD} = \quad & \text{hash}(\text{TCP pseudo header} + \\ & \text{TCP header} + \\ & \text{TCP segment data}) \end{aligned}$$

   The TCP pseudo header consists of a source IP address, a destination IP address, a TCP segment length and a protocol number derived from the IP header. The TCP header includes the TCP options, but all the bits in its checksum field are filled up with zeros. Both headers are the network byte order, as specified in the TCP protocol specification. If there are no payload data, the TCP segment data will be zero.

2. MAC is derived from the encryption of the MD. This encryption process is achieved by the shared key WK_i.

   $\text{MAC} = (\text{MD})^{\text{WK}_\text{i}}$

4.3.2 Data Confidentiality

The Secure TCP segment data is encrypted to keep it secure from the message eavesdropping attacks. However, the scope of the Secure TCP does not include the header confidentiality, and thus the TCP header is not encrypted and is delivered in the plain text.

The data confidentiality must be carried out with the data integrity. For the receiver to know if received segments are modified quickly, the data confidentiality is carried out before the data integrity, i.e.,

$$\begin{aligned} \text{Segment} = \quad & \text{Plaintext header} + \\ & (\text{Segment data})^{\text{WK}_\text{c}} + \\ & \text{MAC}(\text{Encrypted segment}) \end{aligned}$$

In the case the block cipher method is used for encryption, the padding may be required to align the segment to the block size employed by the ciphers. The padding data is appended to the end of the TCP segment data, and the whole segment is encrypted. Therefore, the decryption procedure at the receiver entity must know its original size of the segment data. Accordingly, in the Secure TCP, the original data size is send to the receiver entity by means of the TOE mechanism, which details are described in the next section.

5 Secure TCP Format

In this section, we discuss the protocol format in our implementation of the Secure TCP.

5.1 TCP Option Extension

The negotiation procedure in the Secure TCP must provide functions to exchange keys for data encryption. However, as described in section 4, the procedure takes place in the three-way handshake phase. Thus, the SYN, SYN+ACK and the ACK segment in the phase must carry the extra data for the Secure TCP functions.

Furthermore, the sender must send the original segment data size to its receiver to use the data confidentiality discussed above. In order to implement these requirements with the minimum changes in the classical TCP, we propose TCP option extension called TOE. The details of the TOE are as follows,

- In the negotiation phase of the Secure TCP, carrying the negotiation data, authentication information and the session keys are performed. To achieve this, we add 4 new TCP options; "Negotiation Send" option, "Negotiation Reply" option "Key Exchange Send "option and "Key Exchange Reply "option.

- We assign a new bit field called TOE at the top of the TOS field in the TCP header. If this bit is set, the TOE data is located in the top of the TCP segment data. This mechanism is employed to carry large size of data as a TCP option.

  In the classical TCP specification, the maximum length of the TCP option is only 40 octets. However, since the typical size of encryption keys in the public-key cryptography is more than 512 bits (64 octets), these keys cannot be delivered in the TCP option field in the classical TCP. In this case, the sender sets the TOE bit in the TCP header, and appends the TOE data to the top of the TCP segment data. To indicate the length of the TOE data in the segment, the urgent pointer field in the TCP header is used. Therefore, the receiver checks if the the TOE bit is set. In the case the bit set, the receiver gets TOE data from the

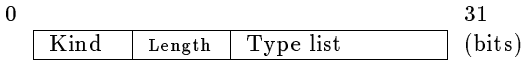

Figure 6: Negotiation Send/Reply Option

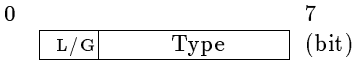

Figure 7: Security Service Type Format

top of the TCP segment data with using the length information in the urgent pointer field, and proceed for further processing. The format of the TOE data is described in the later section.

- We add 1 more new TCP option called "TCP length." As discussed in the section 4, the sender must advertise the original length of the TCP segment data to the receiver for its data confidentiality. This option carries the original length of the TCP segment data.

5.2 Secure TCP Header

A protocol format of the Secure TCP segment is shown in Figure 5. In the Secure TCP, the TOE flag is added to a classical TCP header and the urgent pointer field is used as a TOE pointer field. Two shadings are the TOE flag and the TOE pointer field.

5.3 Negotiation Send/Reply Option

The Negotiation Send/Reply Option is shown in Figure 6. The value of the Kind is 20 in the Negotiation Send Option and 21 in the Negotiation Reply Option. These options can send a list of security type. A maximum number of the security service type that the option can contain is 38, because the length of the TCP option is limited in 40 octets.

The format of security type data is shown in Figure 7. The security type data is 1 octet long. The L/G field(1bit) determines whether local or global security type is set up. The Type field (7bit) identifies whether type of security service uses.

5.4 Key Exchange Send/Reply Option

The Key Exchange Send option is shown in Figure 8 and the Key Exchange Reply option is shown in Figure 9. The value of the Kind is 22 in the Key Exchange Send Option and the Kind is 23 in the Key Exchange Reply Option.

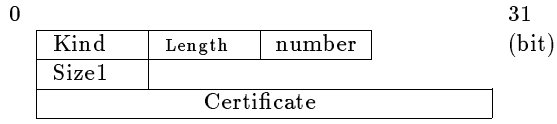

Figure 8: Key Exchange Send Option

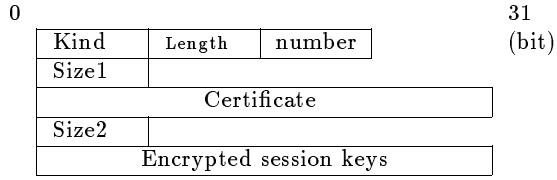

Figure 9: Key Exchange Reply Option

A number field shows a number of key exchange data elements. Each element is constructed by a size field and a data field. The size field shows a length of the element and the data field shows data related key exchange.

The Key Exchange Send option contains only the public-key certificate. The certificate is composed of a RSA public-key, a IP address, a date data that indicates in the term of validation period and a padding date. The RSA public-key is 64 octets (512 bits), the IP address is 4 octets (32 bits), the date data is 4 octets (32 bits) and padding data is 56 octets (448 bits). The total length of the certificate is 128 octets (1024 bits).

The Key Exchange Reply option contains the public-key certificate and session keys data. The certificate is the same in the Key Exchange Send option. The session keys data is composed of a DES session keys and padding data. The session keys are 16 octets (128 bit) and padding data is 48 octets (384bits).

5.5 TCP Length Option

This option format is shown in Figure 10. The value of kind is 24. The TCP Length is 2 octets and indicates the length of a plaintext segment data in a segment.

5.6 Trailer Format

In order to deal with large size, MAC described in the section 5 is appended to the end of the TCP segment as a trailer. If the TCP option sends MAC, the maximum size of MAC is limited to 38 octets. In the current implementation, the length of MAC is 16 octets.

6 Implementation

We have implemented the Secure TCP on a BSD/386 UNIX system[3] for the IBM PC/AT

[3] BSD/386 version 1.1

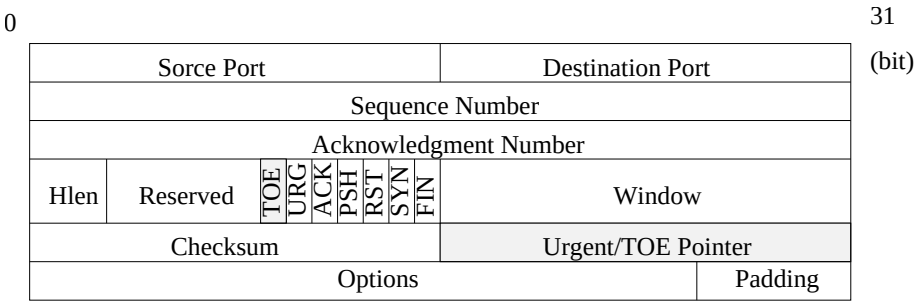

Figure 5: Secure TCP Header

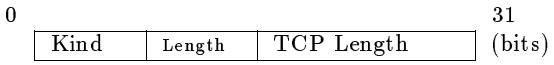

Figure 10: TCP Length Option

platforms. We implemented only security service type 2.

In our implementation, all the files in `tcp_input.c`, `tcp_output.c`, `tcp_subr.c` , `tcp_usrreq.c` , `tcp.h`and `in_pcb.h` are modified to perform negotiation, data integrity and data confidentiality processes. Moreover, we added the re-entrant cipher module (DES and MD5) to the UNIX kernel.

In order to maintain the Secure TCP state transition and session keys for encryption/decryption processes, we define the STCPCB (Secure TCP Control Block) data structure. This data structure is prepared for each Secure TCP connections. In this STCPCB, the transmitted TCP segments are kept for segment retransmissions, while the plaintext is kept in the original TCPCB. This implementation helps to retransmit the TCP segment quickly, because the recalculation of MAC and the encryption of the original TCP segment data is eliminated in the retransmission procedure.

6.1 Differences from the classical TCP implementations

The significant changes in the negotiation phase (three-way handshake phase) from the classical TCP are;

- Any SYN segments cannot include real data because the connection is not securely established. T/TCP (Transaction TCP) in RFC1644[9] promotes real data transmissions in the SYN segment for performance reasons, however, this idea is conflicted with the Secure TCP.

- The cross-active open cannot be allowed. In the TCP protocol specification, both peer entities can send the SYN segment simultaneously. In this case, the peer entities merely return ACK segments, and then the TCP connection is established[4]. This connection establishment is called "cross-active open." However, this cross-active open does not exchange encryption keys in secure manner in the Secure TCP. Therefore, if the entity receives the SYN segment in the state "SYN SENT," the state is immediately moved to the state "CLOSED". This state transmission avoids insecure cross-active open in the Secure TCP.

6.2 Interoperability with the classical TCP

Our implementation is interoperable with the classical TCP.

In a case a sender uses the Secure TCP and its receiver uses the classical TCP, the response for the SYN segment sent by the sender is a normal SYN+ACK segment that does not include any negotiation data. Using this characteristic, the sender can determine the receiver uses the classical TCP. Hence, the sender can easily decide whether it goes ahead to establish a connection in the classical TCP or shuts down the negotiation.

In the current implementation, if the receiver uses the classical TCP, the sender will try to establish a connection using the classical TCP. This may be a drawback in some cases. We discuss this problem in the later section.

6.3 Performance

The current version is implemented on BSD/386 running on i486DX2 platform with

[4] In this case, both entities traverse from the state "CLOSED" to "SYN SENT", onto "SYN RECEIVED" and then "ESTABLISHED". Thus, the TCP connection can be established.

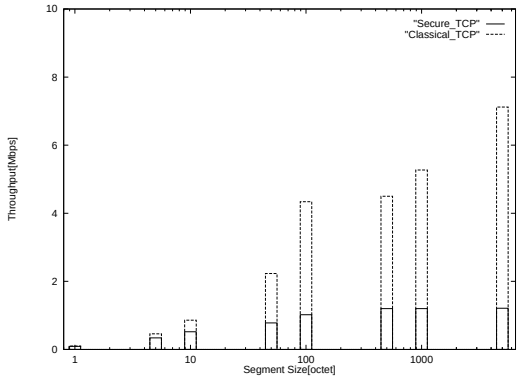

Figure 11: Throughput against various Segment Sizes

16MB memory connected to the Ethernet[5]. Its simple throughput performance is shown in Figure 11. We measured throughputs of the classical TCP and the Secure TCP against various segment data sizes.

When the segment data size is less than 5 octets, the throughputs of the classical TCP and the Secure TCP are a little different. However, when the segment data size is more than 10 octets, the throughput of the Secure TCP is lower than the throughput of the classical TCP. In the Secure TCP, a limitation of the throughput gets approximately 1Mbps. We assume one reason of the limitation is overhead of encryption/decryption processes achieved by software.

7 Discussions

In this section, we discuss future works of the Secure TCP development.

7.1 Periodic Key Exchange

In the current implementation of the Secure TCP, the session key negotiation takes place at the connection establishment phase, and session keys for the connection are not changed until the connection is closed. However, using the same key in the long period may give chances for eavesdroppers to decode data stream on the connection by using brute-force methods. In order to strengthen the Secure TCP, it is better to change session keys periodically.

There are some possible solutions to achieve this periodical session key change. One solution we are considering is to add a new TCP option to advertise new session key to the peer entity. If the peer entity receives the segment with this TCP option, then it uses the key in the option as a new session key. We are now integrating this option to the current version of the Secure TCP.

7.2 Application Interface Library

In the current version of the Secure TCP, all the controls from the application layer are practiced by using the UNIX socket I/O control mechanism. Since the socket I/O mechanism is a raw interface to the Secure TCP, it is better to provide the Secure TCP interface library to use the Secure TCP features in simple manner. This library will be included in next version of the Secure TCP.

7.3 Interoperability with the classical TCP

As mentioned in the previous section, the current implementation of the Secure TCP provides the interoperability with the classical TCP. In the current implementation, when a sender finds out a receiver uses the classical TCP, it will try to establish the connection in the classical TCP, not the Secure TCP. However, in some cases, this interoperability may cause some troubles, because the classical TCP transfers all the data in the plaintext. Thus, there must be a notification mechanism for application programs to indicate if the receiver uses the classical TCP.

7.4 Hardware Cipher

In the session 6, we provide the performance of the current version of the Secure TCP. The bottle neck of the Secure TCP is our software cipher. In order to adapt the Secure TCP to the high-performance networking system such as FDDI or ATM, we should introduce hardware implemented ciphers into the Secure TCP implementation.

8 Conclusion

We propose the Secure TCP as a transport layer protocol as one of the basic units to realize secure communication mechanisms in the Internet. It brings security services for data integrity and confidentiality. Moreover, it keeps connectivity (interoperability) with the classical TCP and provides a negotiation service of security service for process/process communication.

This paper discusses the design and implementation of the Secure TCP.

The current implementation running on the BSD/386 UNIX systems (on IBM PC architectures) provides approximately 1Mbps as its throughput.

This performance is not enough when we apply the Secure TCP to high speed network environment. However, it can be improved by intro-

[5] Gateway 2000/466VL. The Ethernet card is SMC's Elite16C on ISA bus.

ducing the fast cipher technologies such as DES chips or other hardware implemented ciphers, since the overhead of the encryption/decryption processes limits its performance.

Acknowledgment

Authors would like to thank Noritoshi Demizu (OMRON), Nobuhiko Tada (Panasonic/Matsushita), Youki Kadabayashi (Osaka University) and the members of the Information Network laboratory at the Graduate School of Information Science, Nara Institute of Science and Technology for their active discussion and valuable suggestions. The Authors also thank Yoshiki Sameshima and Hideki Fujioka (Hitachi Software Engineering) for their great cooperation as proofreaders.

References

[1] J. Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Messag Encryption and Authentication Procedures, RFC1421," February 1993.

[2] S. Kent, "Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management, RFC1422," February 1993.

[3] D. Balenson, "Privacy Enhancement for Internet Electronic Mail: Part III: Algorithms, Modes, and Identifiers, RFC1423," February 1993.

[4] J. Burton S. Kaliski, "Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services, RFC1424," February 1993.

[5] J. Kohl and B. Neuman., "The Kerberos Network Authentication Service(V5), RFC1510," September 1993.

[6] J. Ioannidis and M. Blaze., "The architecture and implementation of network-layer security under unix," in *Proceedings of the Fourth Usenix UNIX Security Symposium*, October 1993.

[7] T. Tanida and Y. Shinoda., "IP/Secure : Providing Security on Datagram Delivery for Mobile Host Environment," in *Internet Society: Proceedings of INET'94/JENC5*, June 1994.

[8] J. Postel., "The Kerberos Network Authentication Service(V5), RFC1510," September 1993.

[9] R. Braden., "T/TCP – TCP Extensions for Transactions Functional Specification, RFC1644," July 1994.

Author Information

Toshiyuki Tsutsumi has been working for Hitachi Software Engineering Corp. since 1991, where he has developed network management software, maintained the network environment, and researched the computer networks. He has also been a student of the Graduate School of Information Science, Nara Institute of Science and Technology, Japan since 1993. He received his B.S. from the University of Electro Communications in 1991. He has been a member of the WIDE Project since 1994.

Suguru Yamaguchi received the M.E. and D.E. degrees in computer science from Osaka University, Osaka, Japan, in 1988 and 1991, respectively. From 1990 to 1992 he was an Assistant Professor in Education Center for Information Processing, Osaka University. From 1992 to 1993, he was with the Information Technology Center, Nara Institute of Science and Technology, Nara, Japan, as an Associate Professor. Currently, he is an Associate Professor with the Graduate School of Information Science, Nara Institute of Science and Technology, Nara, Japan. He has been also a member of WIDE Project, since its creation in 1987, where he has been conducting research on network security system for wide area distributed computing environment. His research interests include technologies for information sharing, multimedia communication, network security and network management for the Internet.