SICS/T-89/8913

SPION
Secure Protocols in OSI Networks

Bengt Ahlgren Per Lindgren Teet Sirotkin

May 1989

SICS technical report
T89013
ISSN 1100-3154

SICS/T-89/8913

SPION
Secure Protocols in OSI Networks

Bengt Ahlgren Per Lindgren Teet Sirotkin

May 1989

SICS technical report T89013
ISSN 1100-3154

Swedish Institute of Computer Science
Box 1263
S-164 28 KISTA
Sweden

Abstract

This report describes how security services can be realized in a computer network using the protocols of the Open Systems Interconnection (OSI) reference model for communication.

The report starts with defining security requirements for a "typical" local area network in a company, university or similar organization. It is assumed that the organization does not use the network for transfer of extremely sensitive information, such as military secrets.

A set of security services, as specified in the OSI security architecture, are selected in order to satisfy the requirements. The selected services are then placed in suitable layers of the OSI model according to the criteria in the security architecture, and to the taste of the authors.

The report concentrates on the transport layer. An extension of the OSI transport protocol, class 4, including security services is described in detail. The protocol is a fully compatible extension of the standard transport protocol.

Key management is another topic which is included in the report. A key management system for handling public keys and digital signatures based on an article by Dorothy E. Denning is described. The system includes functions for distributing and validating public keys, and registering and later verifying digital signatures. A key management protocol supporting these functions is defined for communication between ordinary open systems and special key server systems.

Contents

Preface

Over the past ten years or so, the use of computer networks has increased dramatically. The need for secret and secure (in some sense) transfer of data is not new, it has been around for a very long time—maybe for the greater part of history of modern man.

Traditionally, security mechanisms such as encryption has mainly been used by the military community to prevent spying. Today most of the transfer of funds between banks (and other similar institutions), which previously was made by sending pieces of paper around, is being done over electrical wires—a medium more easily penetrated and manipulated than the postal service.

Use of networks to connect people to computers also makes it more difficult to control exactly *who* it is that sits behind a keyboard. Previously, when terminals were placed in certain rooms etc., access control was much easier.

This work is done as a half-semester project at the Swedish Institute of Computer Science (SICS) in Kista, Stockholm, Sweden.

Stockholm, August 1988

BENGT AHLGREN
PER LINDGREN
TEET SIROTKIN

1 Security Requirements

Many areas of computer communication have been subject for international standardization. Secure communication has not been one of those, at least not until 1988, when ISO standardized a security architecture within the framework of the OSI reference model [5]. The OSI security architecture outlines a security framework with possible (or desirable) security services and mechanisms to implement the services.

The aim of our SPION[1] project is therefore to specify—in detail—services and mechanisms for security in the OSI model (International Standard 7498 [4]). Our work is concentrated to the transport layer, because we think that many security services should be placed there.

Information in itself is often valuable—take for instance a magnetic tape with money transfer orders. It is fairly easy to protect information *inside* a computer system, but when information is transferred between computers it becomes more difficult. One reason for this being the fact that a network is seldom completely *physically* protected.

1.1 Basic requirements

Data transferred over a network should be protected against the following threats (both accidental and intentional):

- Disclosure of secret data
- Destruction and insertion of data

1.2 Detailed requirements

There are several special cases of the threats above:

Disclosure of secret data: Data sent should not be disclosed to anyone but the intended recipient.

Destruction of data: It should not be possible to destroy data sent over the network without detection. If retransmission fails persistently, then the connection should be closed.

Modification of data: It should not be possible to change any of the data sent without detection.

Insertion of synthesized data: It should not be possible to insert any data into the networks' datastream.

Replay of earlier transferred data: It should not be possible to replay data sent earlier on the same connection or data from another connection.

[1]Secure Protocols In OSI Networks, pronounced "spy-on".

Reordering of data: It should not be possible to reorder data packages sent over the network.

Masquerading: It should not be possible to masquerade as another entity at either side of a connection.

Unauthorized accessing: It should not be possible for an entity to access a resource that it has no access rights for.

1.3 Other requirements

There are some requirements that are not associated with any specific threat:

- Attempts to violate security should be logged.
- Error recovery should be attempted when data is destroyed, altered or inserted.
- It should be possible to only encrypt a desired part of the data sent to increase the throughput.
- Peer entities should be able to negotiate about the security that will be used on a connection.

1.4 Assumptions

SPION is meant for implementation on 'typical' LANs such as the ones used in universities, large companies and other similar institutions.

When constructing secure protocols there always has to be a compromise between security and efficiency/compatibility/usability. We have made the assumption that there is no transfer of extremely sensitive information (such as military secrets) on a typical LAN, nor that it is the target for extremely powerful cryptanalysis.

We regard as a most relevant issue, that the user doesn't get annoyed using security and thus make transfers without security, as could be the case if the use of the security functions is bothersome or slow.

2 Security Services

2.1 Service overview

The OSI security architecture allows the security services to be placed in different layers of the OSI reference model. Table 1 shows the possible service placement together with the service selected in SPION (indicated by filled dots).

	Layer						
Service	1	2	3	4	5	6	7
Peer entity authentication			○	●			●
Data origin authentication			○	○			○
Access control service			●	●			●
Connection confidentiality	○	○	○	●			○
Connectionless confidentiality		○	○	○			○
Selective field confidentiality							●
Traffic flow confidentiality	○		○				○
Connection integrity with recovery				●			○
Connection integrity without recovery			○	○			○
Selective field connection integrity							●
Connectionless integrity			○	○			○
Selective field connectionless integrity							○
Non-repudiation, origin							●
Non-repudiation, delivery							●

Table 1: Placement of services.

The requirements of section 1 are mapped to services in the following way:

- To transfer data in a way that its content cannot be intelligible to an intruder, **confidentiality** services are provided in the transport and application layers.
- To prevent data packets from disappearing, or being altered, added or repeated without detection, **integrity** services are provided in the transport and application layers.
- To prevent masquerading, **authentication** services are provided in the transport and application layers.
- To prevent unauthorized accesses to network resources, **access control** services are provided in the network, transport and application layers.
- To make it possible to administer the security services in the network from any node, security management services are provided.
- Logging of security relevant events are provided by the security management services.

2.1.1 Confidentiality

The confidentiality service provides for the protection of data from unauthorized disclosure by passive monitoring ('wiretapping'). The data is protected between end systems (i.e., data is not decrypted at relay nodes).

2.1.2 Integrity

The integrity service consists of several different parts. They should ensure that:

- data is not lost between sender and recipient.
- the data received is the data sent.
- the data originates at the sender.
- false data is discarded.

Denial of service can only be detected, not rectified, by the protocol. Corrective actions—which can be taken by a network supervisor—can be suspension of processes which cause the network to congest. Another cause for loss of data packets is damage of the transmission media. This can only be prevented by installing cables and other network hardware in safe places, unreachable by all but network supervising personnel.

2.1.3 Authentication

The peer entity authentication service provides for a correct identification of peer entities at the transport layer. An authentication service can also be provided in the application layer, but this is optional and application dependent.

The peer entity authentication service must be used in conjunction with an integrity service to provide a data origin authentication service on a connection.

2.1.4 Access control

The access control service provides access control between peer entities at the network, transport and application layers. (Actually, it is access to NSAPs, TSAPs and ASAPs that is controlled.)

2.1.5 Management

All of the above security services need management in some form. Examples are key distribution, public key verification, and handling of access control information. Another management activity is to report information about the use of the net, with alarms 'going off' when unauthorized actions are taken, including physical manipulation thereof.

Some management issues are already in the hands of the host computer; access control at login, access control to files, programs and peripheral equipment connected directly to the host.

2.2 Security service specification

2.2.1 Application layer service

Although the security services provided by an application are individual for each application, we propose the following services:

Selective field confidentiality: An application should be able to encrypt only the sensitive part of the data sent for efficiency.

Selective field integrity: An application should be able to use the integrity service on only the sensitive part of data sent for efficiency.

Peer-entity authentication: An application entity should be able to authenticate its peer.

Non-repudiation: This service is easy to provide when public-key cryptography is used for authentication, and is therefore included.

Access control: It is desirable to control access separately for different applications, which may have different security requirements.

2.2.2 Transport layer service

Service	MAC	DES-CBC	RSA
Peer entity authentication			×
Connection confidentiality		×	×
Connection integrity with recovery	×		×

Table 2: Mapping of services to mechanisms in the transport layer.

The transport layer in SPION offers the following security services (see also table 2):

Peer entity authentication of transport entities. Uses the RSA cryptosystem by Rivest, Shamir and Adleman to make digital signatures, which are exchanged during connection initiation. (For a description of RSA, see for example Denning [2].)

Connection confidentiality of all transport user data. Uses DES encryption in CBC (cipher block chaining) mode with a session key which is transferred RSA encrypted at connection initiation. (For a description of DES and CBC mode see for example Denning [2].)

Connection integrity with recovery as to insertion, deletion and replay of TPDUs. Uses a message authentication code (MAC) with an initialization seed which is transferred RSA encrypted at connection initiation.

Access control is performed on incoming and outgoing connections. Whether to do the access control or not is a matter of system configuration. The access controlled is that between TSAP addresses. The access control service uses an access control list in the security management information base (see section 2.3.4). The user has no influence over the use of this service.

Section 10.9 "TC protection" in International Standard 8072 [6], the OSI transport service definition, is in SPION extended as follows. ("TC" is an abbreviation of "transport connection".)

> TC protection is specified qualitatively by selecting a combination of three TC protection options:
>
> a) peer-entity authentication
>
> b) connection confidentiality (protection against passive monitoring)
>
> c) connection integrity with recovery (protection against modification, replay, addition or deletion)

2.2.3 Network layer service

The only security service in the network layer is **access control.** Access control is performed on incoming and outgoing connections. Whether to do the access control or not is a matter of system configuration. The access controlled is that between NSAPs. The access control uses an access control list in the SMIB. The network user has no influence over the use of this service.

2.3 Security management services

The OSI Security Architecture [5] defines three categories of security management activities:

System security management is the management of security aspects of the overall OSI environment.

Security service management is the management of particular security services.

Security mechanism management is the management of particular security mechanisms.

The security management functions in OSI are not yet very well defined. Because of this, we will not try to penetrate the area completely in order to avoid incompatibilities with the OSI-model.

One important part of security management is the distribution of cryptographic keys. The key distribution method in SPION is basically a method described by Denning [3]. It requires at least one key server which should reside on a dedicated host. The key server certifies the public keys in the network and keeps a log of security events.

2.3.1 System security management

The system security management functions needed in SPION are:

- Communication with the key server and other open systems.
- Security Management Information Base (SMIB) handling.
- Event handling, security audit and security recovery management.
- Access control policy management.

2.3.2 Security service management

The security service management functions needed in SPION are:

- Determination and assignment of the target security protection for the service.

  The target security protection for the services of the transport layer is specified in the SMIB for TSAP-addresses.

- Selection of the specific security mechanism to be employed to provide the requested service.

  The term *mechanism configuration set* is used in SPION to denote a set of mechanisms which together can provide for the security service at some layer. The reasons for defining sets of mechanisms, instead of selecting mechanisms for single services, are that the application of a specific mechanism in many instances must be coordinated with the other mechanisms to provide the desired service and that the performance of the mechanisms can be optimized better.

2.3.3 Security mechanism management

The security mechanism management functions needed in SPION are:

- *Key management*—Supports encipherment and authentication management. This function generates and distributes keys with support from the system security management functions. It uses the method by Denning mentioned above. A key server (or several servers) keeps a log with public keys and distributes public keys in certificates when requested. When a key disclosure is detected, new keys are generated and sent to the key server. The key server logs the old keys as disclosed, and the new keys as current.
- *Encipherment management*
- *Access control management*
- *Data integrity management*

- *Authentication management*—Depends on key management for key generation and distribution. SPION uses the RSA cryptosystem (the public-key system by Rivest, Shamir and Adleman) to generate signatures for authentication purpose.

2.3.4 Security management information base – SMIB

The security management information base (SMIB) is a logical part of the management information base (MIB), but not necessarily a physical part. The SMIB contains the following information:

- Outgoing and incoming access control lists for NSAPs
- Outgoing and incoming access control lists for TSAPs
- Minimum transport security level table for source and destination TSAPs. (The different security levels are all combinations of the three services confidentiality, integrity and peer-entity authentication.)
- Maximum transport security level table for source and destination TSAPs.
- A table listing the available mechanism configuration sets for each layer.
- Authentication information for transport entities. In SPION this information consists of:
  - Public RSA key table for transport entities of the network.
  - Private RSA key table for the transport entities on the open system.
- Authentication information for application entities. In SPION this information consists of:
  - Public RSA key table for application entities of the network.
  - Private RSA key table for the application entities on the open system.

3 The Secure Transport Layer Protocol

This section presents SPION's secure transport protocol. A SPION implementation can communicate not only with other SPION implementations, but with *all* ISO transport protocol implementations supporting class 4. Thus, SPION is an extension of the ISO transport protocol, not a modification of it. We also consider it important to be able to multiplex SPION connections with normal class 4 transport connections on the network.

Only our extensions to the transport protocol are described here. For a complete description see also the ISO transport protocol specification [7] (or the equivalent CCITT Recommendation X.224).

3.1 Specification of security in class 4

To be able to communicate with non-SPION transport entities (security permitting...), standard CC and CR TPDUs are used. The entity which sends the CR is called the *initiator* and the other entity is called the *responder.*

3.1.1 General procedures for integrity

All TPDUs must include the signature parameter if the integrity service is requested.

3.1.2 Connection establishment

Not all cryptographic algorithms are possible to use with a normal class 4 connection initiation. This is due to the limit of 128 octets on the size of the CR TPDU. The length of a RSA digital signature, for example, can be too long to fit in an 128 octet TPDU together with the other necessary information. In order not to limit the choice of mechanisms, SPION offers an extended way to initiate a connection. The extensions do not affect connection initiations without security.

The modification extends the connection initiation into a five-way handshake. In the first exchange of messages, the security level and mechanisms are determined, then follows a three-way handshake similar to the one used for normal secure connection establishment.

A new TPDU is defined for this purpose: the *Security Connect Confirm* (SCC) TPDU.

3.1.2.1 TPDUs and parameters used

The following is a list of the *extra* parameters used.

1. CR TPDU
   - Security parameter
   - Signature parameter

2. CC TPDU
   - Security parameter
   - Signature parameter
3. AK TPDU
   - Signature parameter
4. SCC TPDU
   - Signature parameter

3.1.2.2 Procedure

The connection can be established either by a three-way handshake or a five-way handshake.

Normally, the TPDU maximum size can be negotiated down to 128 octets. SPION raises this to a maximum size of 256 octets, in order to allow long signature parameters.

The following extra negotiations take place:

1. *Security service.* Currently SPION offers three negotiable security services: peer entity authentication, connection confidentiality and connection integrity with recovery. The initiator proposes a preferred service and a minimum acceptable service. Neither may be lower than the minimum level stated in the SMIB. The response may not be lower than the given minimum and the responder's own lowest level, according to the SMIB.

2. *Mechanism configuration.* SPION must know which set of mechanisms is used to provide the security services. It is possible to specify any number of mechanism sets. Mechanism configuration set 1 (DES/RSA) is the default set and must be supported by all SPION entities.

3. *Extended connection initiation.* If the responder changes either mechanism set or raises the security service, an extended connection initiation must take place. The omission of the signature parameter in either the CR or CC TPDU signals the need for an extended connection initiation because of long signatures.

3.1.2.3 Extended connection initiation

The extended connection initiation is a five-way handshake. The first CR and CC TPDU exchange decides which security services to use and the mechanism set to use.

After the CC TPDU follows an exchange of SCC TPDUs followed by an AK (or DT) TPDU from the initiator. The fields of the signature parameter in the SCC and final AK TPDUs are set and used in the same way as in the three-way handshake.

3.1.2.4 Errors

If any errors are detected during the connection initiation, the normal retransmission procedure of class 4 is used. The extended connection initiation presents a problem though, because the integrity of the CR and CC TPDUs can't be checked until the respective SCC TPDUs have been received. Therefore, the connection is disconnected if a SCC TPDU has an integrity error.

3.1.2.5 Mechanism configuration set 1 (DES/RSA)

This is the default mechanism set. It uses the DES algorithm with cipher block chaining (CBC) and the RSA cryptosystem.

Integrity

Integrity is achieved by computing a cryptographic checksum (MAC) over the whole TPDU; the MAC field and the checksum parameter (if any) should be set to all zeros. When the MAC is computed it is entered in the signature parameter, and then the checksum parameter is computed. In case of a five-way handshake, the MAC in the SCC TPDUs is computed over the concatenation of the SCC TPDU and the previous CR or CC TPDU sent by the entity.

Confidentiality

Confidentiality of the connection establishment is achieved by encrypting the signature parameter (excluding the parameter code and length indicator) with the RSA algorithm using the other entity's public key. The signature parameter is padded at the end up to the size of a RSA block.

Authentication

Authentication is achieved by encrypting the signature parameter with the entity's own private RSA key as well as the other entity's public RSA key.

3.1.3 Data transfer

The additions and changes to the procedures for data transfer are described in the following sections.

3.1.3.1 TPDUs and parameters used

The following is a list of the *extra* parameters used.

1. DT TPDU
   - Signature parameter
   - Pad parameter

2. ED TPDU
    - Signature parameter
    - Pad parameter
3. AK TPDU
    - Signature parameter

3.1.3.2 General procedure

Data transfer is affected in two ways: an extra checksum (MAC) may have to be computed, and the user data may have to be encrypted. If the user data is not aligned with the block length of the encryption algorithm, a pad parameter must be inserted in the header immediately before the user data field.

3.1.3.3 Forced disconnect

If the TPDU-NR, ED-TPDU-NR or the AK sub-sequence number is about to cycle, the connection must have new encryption keys. This is done by forcing a new connection. Currently, the transport user has to reestablish the connection—the transport entity disconnects the connection with the reason set to "key expiration".

3.1.3.4 Procedures for transmission of AK TPDUs

The function of the sub-sequence parameter for the AK TPDU is extended to provide detection of some forms of denial of service. The AK TPDU retransmitted when the window timer expires shall always contain the sub-sequence parameter with a value one higher than the previous AK TPDU. If the sub-sequence number space is exhausted, the connection is terminated (see previous section).

3.1.3.5 Mechanism configuration set 1 (DES/RSA)

This is the default mechanism configuration set. It uses the DES algorithm with cipher block chaining (CBC) and the RSA cryptosystem.

Integrity

Integrity is achieved by computing a cryptographic checksum (MAC) over the whole TPDU. The value field of the MAC and checksum parameters (if any) should be set to all zeros when the MAC is computed. The MAC value is inserted in the TPDU before the normal checksum is computed.

Confidentiality

Confidentiality of the data transfer is achieved by encrypting from the pad field of the Pad parameter (if present) to the end of the user data field. The encryption algorithm is DES in cipher block chaining (CBC) mode. DES has a block length of 8 octets, so the padding may be from 1 to 7 octets long.

The initiator uses K_1 as key and K_2 as seed for IV, the initialization vector for the cipher block chaining, and the responder uses K_2 as key and K_1 as seed for IV. (See section 3.2.2.1 for explanation of K_1 and K_2.) The IV actually used is computed by XOR-ing the seed with the header of the TPDU to be sent. If the header is shorter than 8 octets, it is repeated before XOR-ing.

The same key and seed for the IV is used for encryption of both normal user data and expedited data.

3.1.4 Normal connection release

3.1.4.1 TPDUs and parameters used

The following is a list of the *extra* parameters used.

1. DR TPDU
   - Signature parameter
   - Final TPDU number parameter
2. DC TPDU
   - Signature parameter
   - Final TPDU number parameter

3.1.4.2 Procedure

In the DR and DC TPDUs the sequence numbers of the last TPDUs sent and received are transmitted to the peer entity in the final TPDU number parameter. This detects any attempt to delete TPDUs at the end of a connection.

3.2 Structure and encoding of TPDUs

3.2.1 Security parameter

C5	len	Preferred service	Minimum service	Mechanism config. #	...

Code: 1100 0101

Length: $2 + n$ octets

Value: The first two octets specify the preferred service and the minimum acceptable service, respectively. Bits 3-1 of these octets encodes the security service as indicated by table 3. The bits 8 to 4 shall

Bit	Service	
3	= 1	use of peer-entity authentication
	= 0	non-use of peer-entity authentication
2	= 1	use of connection confidentiality
	= 0	non-use of connection confidentiality
1	= 1	use of connection integrity
	= 0	non-use of connection integrity

Table 3: Encoding of the security service.

be set to zero when sending the TPDU and ignored upon receipt.

The following (optional) octet, octet number 3, is a binary number specifying the preferred *mechanism configuration* to use, that is, the set of particular security mechanisms to use. The mechanism configuration number for SPION is 1, which is the default if a configuration isn't specified. The following zero or more octets specifies alternative mechanism configurations acceptable by the initiating transport entity.

The security parameter may be present in the CR, CC and SCC TPDUs.

3.2.2 Signature parameter

?	len	Signature

Code: (not assigned)

Length: n octets

Value: Depending on the requested security service in the security parameter this value contains:

Authentication: a digital signature used to corroborate the identity of the sending transport entity.

Integrity: (a) a message authentication code computed over the complete TPDU with this signature value and the checksum value of the checksum parameter (if present) set to zero, and (b), if the TPDU is the first TPDU of the connection where this parameter is present, a key to be used for computing the message authentication code for this TPDU and the following TPDUs on the connection.

Confidentiality: a session key (in case the confidentiality service requires one).

The parameter value is encrypted as appropriate for the mechanism configuration.

The signature parameter shall be present in all TPDUs when the integrity service is selected. An exception to this rule is the CR and CC TPDUs when a five-way handshake including SCC TPDUs are performed. All fields of the signature parameter are present only in the TPDUs during connection establishment. In all other TPDUs of the connection, the only field that may be present is the message authentication code.

3.2.2.1 Mechanism configuration set 1 (DES/RSA)

When using the mechanism configuration set 1, the signature parameter value can contain four fields:

?	len	*MAC*	*A*	*B*	*Pad*

MAC: a 64 bit message authentication code, which is present when the integrity service is requested.

A and B: two 64 bit random numbers, which are used as indicated by table 4 depending on the selected service.

Pad: an n-bit field to pad the above fields to the block size of the RSA keys. The padding is generated by duplicating the above fields the appropriate number of times.

TPDU type	field	Integrity	Confidentiality	Authentication
CR	A	I		
	B		K_1	X_1
CC	A			X_1
	B		K_2	X_2
first AK	A			X_2
	B	(not present)		

Table 4: Usage of the random numbers in the signature parameter.

Table 4 specifies the usage of the fields A and B in the parameter. The variables are interpreted as follows:

I	is an initialization seed for the MAC.
K_1	is the session key for the DES-CBC encryption of the data TPDUs in the direction from the initiator, and the initialization vector for the opposite direction.
K_2	is the initialization vector for the DES-CBC encryption of the data TPDUs in the direction from the initiator, and the session key for the opposite direction.
X_1	is a number used by the initiator to corroborate the identity of the responder.
X_2	is a number used by the responder to corroborate the identity of the initiator.

The signature parameter is sent in SCC TPDUs (see section 3.2.5) when a five-way handshake is performed. The interpretation of the parameter in this case is the same as in the corresponding CR and CC TPDUs.

During connection establishment, the four fields are signed and/or encrypted with the RSA algorithm to produce the parameter value as follows (assuming that A is the sending transport entity and B is the receiving transport entity):

- If the integrity service is used, the parameter is encrypted with E_B.
- If the confidentiality service is used, the parameter is encrypted with E_B.
- If the authentication service is used, the parameter is signed with D_A.

When both transformations are applied, the order of application,

$$E_B(D_A(MAC, A, B, Pad)) \quad \text{or} \quad D_A(E_B(MAC, A, B, Pad)),$$

depends on which key modulus are the greater. The first formula is used when the modulus of D_A is *less than or equal* to the modulus of E_B, and the second formula is used otherwise.

As stated above, the only field of the signature parameter present in the TPDUs following the connection establishment phase is the MAC. In these TPDUs, the signature parameter is not encrypted with RSA.

3.2.3 Pad parameter

?	len	Pad

The purpose of the pad parameter is to pad data that is to be encrypted to evenly match the blocksize of the encryption algorithm. For mechanism configuration set 1, the pad parameter may be present in data and expedited data TPDUs when the confidentiality service is used. The parameter is always the *last* parameter in the variable part of the header.

Code: (not assigned)

Length: n octets

Value: Zero or more octets with random contents.

For mechanism configuration set 1, the pad parameter contains from zero to seven padding octets, since the block size of DES-CBC is eight octets.

3.2.4 Final TPDU number parameter

?	len	Final DT TPDU sequence numbers	Final ED TPDU sequence numbers

The purpose of the final TPDU number parameter is to detect deletion of TPDUs at the end of the connection.

Code: (not assigned)

Length: 4 or 16 octets

Value: The first 2 or 8 octets (depending on whether the extended sequence number format has been selected) is the sequence numbers of the last DT TPDU sent and received, respectively. The second 2 or 8 octets are the corresponding sequence numbers for the ED TPDUs.

The DR and DC TPDUs shall contain the final TPDU number parameter when the integrity service is selected.

3.2.5 Security connection confirm (SCC) TPDU

3.2.5.1 Structure

The structure of the SCC TPDU shall be as follows:

LI	SCC	DST-REF	Variable part

3.2.5.2 LI—Length indicator field

This field, octet number 1, indicates the length of the TPDU header in octets excluding the length indicator field. The maximum value is 254.

3.2.5.3 Fixed part (octets 2 to 4)

The fixed part of the SCC TPDU contains:

1. SCC: Extended connection request code: (not assigned). Octet number 2.
2. DST-REF: Reference identifying the transport connection at the remote transport entity. Octets number 3 and 4.

3.2.5.4 Variable part (octets 5 to the end)

The variable part shall contain the signature parameter and may contain the checksum parameter.

3.2.6 Disconnect request (DR) TPDU

The following additional reason codes can be used:

1. $128 + 11$ – Key expiration
2. $128 + 12$ – Authentication error
3. $128 + 13$ – Integrity error

4 Key Management in SPION

The purposes of key management in SPION are to distribute public keys in a safe way, and to act as a notary public for electronic signatures. The SPION key management thus assumes that the cryptosystem used is a public key system, such as the RSA cryptosystem by Rivest, Shamir and Adleman. (See for example Denning [2]).

The SPION key management system consists of cooperating *security management application entities* (SMAEs) in ordinary open systems and in *key server* systems. The first section below is an overview of key management in SPION. The second section describes the key management service provided by the security management application entity. The third section describes the key management protocol used by SMAEs for communication both between an ordinary open system and a key server system, and between key server systems themselves. The last section describes the functions performed internally in a key server in order to manage the certificate log.

4.1 Key management overview

The key management in SPION is based on the article "Protecting Public Keys and Signature Keys" by Denning [3]. SPION divides the network in *key management areas* to allow different organizations to handle their own keys, and to improve performance in large networks. A key management area has one or more *key servers*, which handle the public keys for open systems (hosts) in the area. An open system in the network is located in exactly one key management area. If more than one key server is serving an area, some protection from key server failure is gained, in addition to shorter key request times.

4.1.1 Key server functions

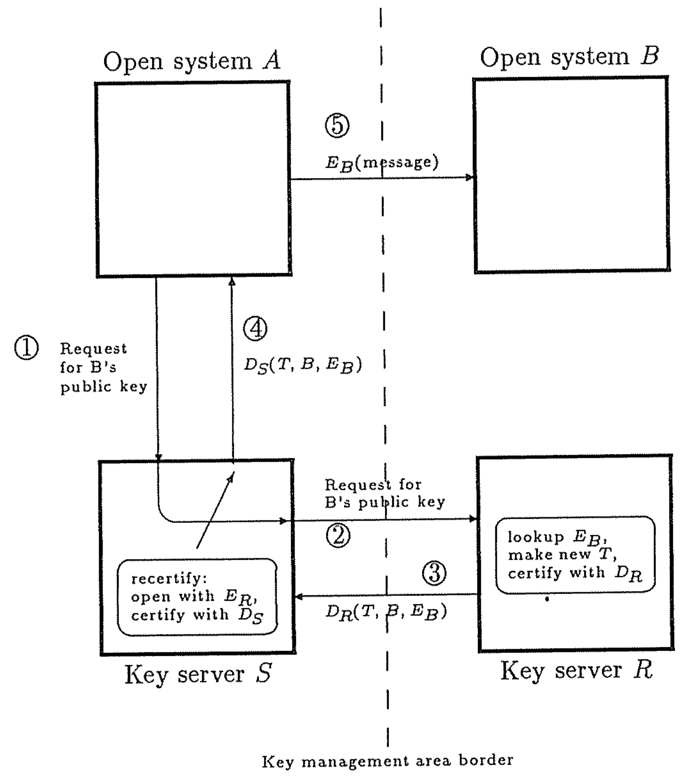
A key server issues *certificates* for public keys and digital signatures. A certificate is a message signed by a key server with its private key. The message contains a timestamp together with the certified data to provide a means for the receiver to verify the time integrity of the certificate.

The keys and signatures are stored in a write-once *certificate log*. The purpose of the log is to protect against forgeries and compromises of all keys (including the server's). See Denning [3] for motivation and discussion.

The key server performs the following functions:

- Registration of public keys, signatures, and key compromises. (This results in new entries in the certificate log.)
- Distribution of public key certificates.
- Validation of public keys, signatures and certificates.

Key servers should be physically protected hosts dedicated to the task of handling keys. The key servers in a key management area all have the



Open system A
Open system B
⑤
E_B(message)
① Request for B's public key
④
$D_S(T, B, E_B)$
Request for B's public key
②
③
lookup E_B, make new T, certify with D_R
recertify: open with E_R, certify with D_S
$D_R(T, B, E_B)$
Key server S
Key server R
Key management area border

Figure 1: Remote key request.

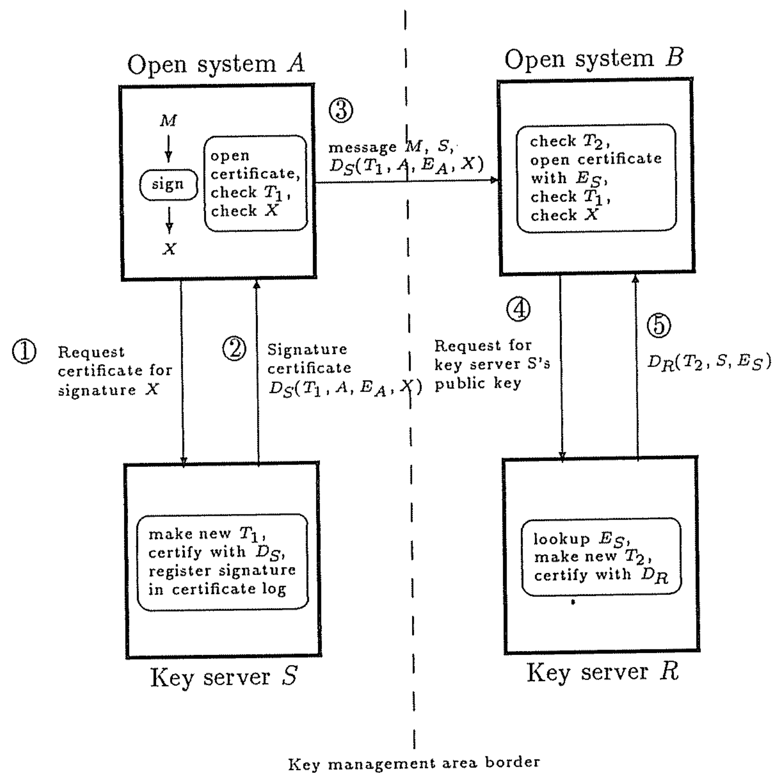
same key pairs, but servers in different areas have different keys. Thus, the key servers must know the public keys of servers in other areas to be able to communicate with them. The public keys of the servers should be distributed through a medium different from the communication medium which they serve, which usually means manual distribution.

4.1.2 Distribution of public keys

When an open system needs the public key of an entity, it requests a public key certificate from a key server in the key management area. If the entity is located in another key management area, then the key server, in turn, has to request a certificate for the key from a key server in that area. This means that the open system only needs to know the public key of the local key server.

Figure 1 shows the procedure used to process a request from open system A for the public key of open system B[2]. System A first sends a request message to the local key server S (1). Since B is located in another key management area, S forwards the request to key server R (2). R looks up the public key of B, and returns a new certificate $D_R(T, B, E_B)$, where T is a new timestamp and E_B is the public key of B (3). The certificate can't be directly sent to A, because A only knows the public key of the local key server S.

[2]The keys are actually for specific *entities* in the open systems.



Open system A
Open system B
M
sign
X
open certificate, check T_1, check X
③
message M, S, $D_S(T_1, A, E_A, X)$
check T_2, open certificate with E_S, check T_1, check X
① Request certificate for signature X
② Signature certificate $D_S(T_1, A, E_A, X)$
④ Request for key server S's public key
⑤ $D_R(T_2, S, E_S)$
make new T_1, certify with D_S, register signature in certificate log
lookup E_S, make new T_2, certify with D_R
Key server S
Key server R
Key management area border

Figure 2: Signature registration.

Therefore S recertifies the certificate from R, which yields $D_S(T, B, E_B)$, and returns it to A (4). In this way, the open system A doesn't have to worry about whether B is in the same key management area or not.

4.1.3 Signature certificates

A signature has to be registered with a key server in order to be valid. An open system always registers signatures with a local key server, resulting in a signature certificate issued by that key server. This certificate is only useful for entities knowing the public key of the issuing key server. This is not the case when the certificate is sent to another key management area. Therefore, the receiver of a signature certificate from another key management area has to ask a local key server for the public key of the issuing key server.

Figure 2 illustrates the registration procedure for a signature, and the handling by the receiver of a signature certificate. The open system A first signs its message resulting in the signature X. A then registers X by requesting a signature certificate from the local key server S (1). S creates the certificate $D_S(T_1, A, E_A, X)$, where T_1 is a current timestamp. The certificate is then logged in the certificate log and returned to A (2). A checks that the timestamp in the certificate isn't too old and then forwards the certificate with the message M and the name of the local key server S to the recipient open system B (3). B sees that S is not its local key server R, and requests the public key of S from R (4 and 5). Note that this request is specifically for a key server's public key. Since B receives the name of the issuing key server from A, A has the possibility to fake the certificate. B must be sure that the certificate is issued by a key server, and therefore makes the specific key request. Finally, B opens the certificate and checks the signature.

Thus, the only complication with multiple key management areas is that the receiver has to request the public key of the sender's key server.

4.2 Key management service

The SPION security management application entity provides the following key management services:

- Get public key: Gets a certificate for the public key for a network entity.
- Get server public key: Gets a certificate for the public key for a key server.
- Register key: Registers a new public key for an entity. Returns a public key certificate for the key.
- Register key disclosure: Registers that the private key of an entity is disclosed or in some other way not usable, and informs the security management that the corresponding public key is invalid. Returns a certificate for the disclosure registration.
- Validate public key: Searches the certificate log for a public key, and checks if the key is/was valid at a specific time or period of time.
- Register signature: Registers a signature and returns a certificate for the signature.
- Validate signature: Searches the certificate log for a signature. If it is found and if the key used for signing is valid, the signature is valid.
- Validate certificate: Searches the certificate log for a certificate. If it is found and if the key used for certifying is valid, the certificate is valid.

4.3 Key management protocol

In order to provide the key management service, the SMAE on an open (non key server) system must communicate with a SMAE on a key server in the key management area. The SMAEs on key servers have to communicate with each other to exchange key management information across key management area borders. This section describes the SPION Key Management Protocol which is used for both of these two types of communication.

The protocol supports the following functions:

- **Distribution of public key certificates.**

  This function provides the Get public key and Get server public key services.

- **Registration of public keys.**

  This function provides the Register key service.

- **Registration of key disclosures.**

  This function provides the Register key disclosure service.

- **Broadcasting of key compromise registrations.**

  The purpose of this function is to notify all possible users of the key that it is no longer valid.

- **Validation of public keys.**

  This function provides the Validate public key service.

- **Registration of signatures.**

  This function provides the Register signature service.

- **Validation of signatures.**

  This function provides the Validate signature service.

- **Validation of certificates.**

  This function provides the Validate certificate service.

4.3.1 Protocol data units

KREQ	Key Request
KREG	Key Registration
KDEL	Key Deliver
KCRR	Key Compromise Registration Request
KCRC	Key Compromise Registration Confirm
KCRN	Key Compromise Registration Notification
KVR	Key Validation Request
KVC	Key Validation Confirm
SRR	Signature Registration Request
SRC	Signature Registration Confirm
SVR	Signature Validation Request
SVC	Signature Validation Confirm
CVR	Certificate Validation Request
CVC	Certificate Validation Confirm
ERR	Error Notification

Table 5: The PDUs used by the key management protocol.

The key management protocol uses the data units (PDUs) defined in table 5. They are described in more detail in the following sections.

The ERR (Error Notification) PDU can be used when an error condition occurs. An example of an error condition is when a received PDU can't be decoded. The format of the ERR PDU is:

ERR	Error code	Additional info.

The field "Additional info." can be used to return a PDU that couldn't be decoded.

4.3.2 General protocol procedures

The key management protocol is basically a connectionless datagram protocol. The connectionless presentation service is therefore best suited for the protocol, but the connection-oriented can also be used if no connectionless is available.

All functions of the protocol, except the "Broadcasting of key compromise registration" function, consist of a request followed by a reply. For each request PDU there is a corresponding reply (or confirm) PDU. To be able to match a reply with the corresponding request, the PDUs have an identification (ID) field, that is set by the requesting SMAE and copied to the reply PDU by the replying SMAE.

Since the connectionless presentation service doesn't guarantee delivery (as any datagram service), request PDUs are retransmitted if no reply is received in a specified time. However, retransmission is not required if the connection-oriented service is used.

Figure 3 shows a state transition diagram of the behavior of the protocol on an ordinary (non key server) open system. The labels on the transitions

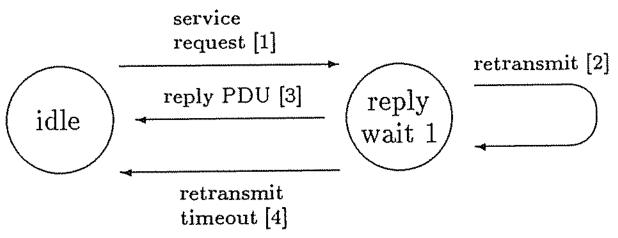


Figure 3: Non key server protocol state diagram.

state the event which causes the transition to take place. The transitions are described in detail in table 6. The "idle" state is the initial state of the protocol machine. A transition to the "reply wait 1" state occurs and a request PDU is sent to the key server when a service request primitive is received from the security management application service access point. In this state the protocol waits for a reply PDU from a key server. If a reply PDU is received, a service indication primitive is produced at the service access point, and the protocol returns to the idle state.

Figure 4 shows a state transition diagram of the behavior of the protocol on a key server system. The transitions of this state diagram are also described by table 6. The "idle" state is the initial state of the protocol machine. The transition loop on this state occurs when a request PDU is

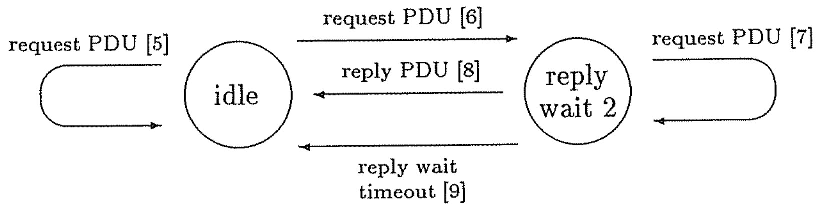


Figure 4: Key server protocol state diagram.

	Event	Action
[1]	Service request received from service access point	Send request PDU to key server
[2]	Retransmit timer timeout and retransmit count less than maximum	Send request PDU to key server
[3]	Reply PDU received from key server	Return a service indication with the reply to service access point
[4]	Retransmit timer timeout and maximum retransmit count	Return a service indication with an error status to service access point
[5]	Request PDU received from a peer entity and requested information is locally available	Retrieve requested information and send back a reply PDU
[6]	Request PDU received from a peer entity and requested information isn't locally available	Determine which key server to request from and send a request PDU to that key server
[7]	Duplicate request PDU received from a peer entity	Send request PDU to the other key server
[8]	Reply PDU received from the other key server	Do local processing, if necessary, and send a reply PDU back to the original requester
[9]	Reply wait timer timeout	(none)

Table 6: Protocol state transitions.

received, and that request can be satisfied locally. The transition to the "reply wait 2" state occurs when a request can't be satisfied locally. In the "reply wait 2" state the protocol waits for a reply PDU from another key server. If the reply is received, it is processed and forwarded to the original requester. Note that no retransmission occurs here—it is only performed by the original requester.

A SMAE may act both as a non key server SMAE and as a key server SMAE at the same time. The entity then performs the functions of both state diagrams.

4.3.3 Distribution of public key certificates

A request for a public key certificate is sent to a key server in a KREQ (Key Request) PDU:

KREQ	ID	Entity name

The "Entity name" field is the name (entity title or SAP-address) of the entity for which the public key is requested.

The public key certificate is returned in a KDEL (Key Deliver) PDU:

KDEL	ID	Public key certificate

4.3.4 Registration of public keys

A new public key is registered at a key server with a KREG (Key Registration) PDU:

KREG	ID	Entity name	Public key

The KREG PDU contains the new public key and the name of the key's owner.

A certificate for the new key is returned in a KDEL (Key Deliver) PDU (described above).

4.3.5 Registration of key disclosures

Registration of key disclosures are initiated by the owner of the compromised key. The requesting SMAE sends a KCRR (Key Compromise Registration Request) PDU to the key server SMAE:

KCRR	ID	Entity name	Signature

The KCRR PDU contains the name and signature of an entity.

When the key server has registered the compromise it replies with a KCRC (Key Compromise Registration Confirm) PDU:

KCRC	ID	Compromise certificate

The compromise certificate is a certificate for the signature in the KCRR PDU. The key server may also invoke the "Broadcasting of key compromise registrations" function.

4.3.6 Broadcasting of key compromise registrations

A key server may broadcast a notification of a key compromise registration to all or some open systems in the key management area. It may also send notifications to open systems in other key management areas. This is done with a KCRN (Key Compromise Registration Notification) PDU:

KCRN	ID	Compromise certificate

The KCRN PDU contains the same data as the KCRC PDU.

4.3.7 Validation of public keys

A key server may be asked to validate a public key for a given period of time with a KVR (Key Validation Request) PDU:

KVR	ID	Entity name	Public key	Time-period

The KVR PDU contains the name and key of a network entity, and a timeperiod.

The reply from the key server is sent in a KVC (Key Validation Confirm) PDU:

KVC	ID	$D_S(T, A, E_A, \text{Timeperiod}, \text{Result})$

where "Result" is one of *Invalid*, *Valid* or *Compromised*.

4.3.8 Registration of signatures

A signature is entered into the log with a SRR (Signature Registration Request) PDU:

SRR	ID	Entity name	Signature

The SRR PDU contains the name and signature of a network entity.

The key server confirms the registration with a SRC (Signature Registration Confirm) PDU:

SRC	ID	Signature certificate

The SRC PDU contains the certificate of the signature.

4.3.9 Validation of signatures

A key server may be asked to validate a signature with a SVR (Signature Validation Request) PDU:

SVR	ID	Entity name	Signature

The SVR PDU contains the name and signature of a network entity. The key server receiving a SVR PDU uses the entity name to determine which key server registered the signature.

The key server sends back the result in a SVC (Signature Validation Confirm) PDU:

SVC	ID	$D_S(T, A, X, \text{Result})$

where "Result" is one of *Valid* or *Invalid*.

4.3.10 Validation of certificates

A key server may be asked to validate a certificate with a CVR (Certificate Validation Request) PDU:

CVR	ID	Server name	Certificate

The CVR PDU contains the certificate and the name of the key server that made the certificate.

The result is returned in a CVC (Certificate Validation Confirm) PDU:

CVC	ID	$D_S(T, S, \text{Certificate, Result})$

where "Result" is one of *Valid* or *Invalid*.

4.4 Internal key server functions

A key management area is served by one or more key servers. The servers maintain a certificate log, which is distributed and fully replicated. If an area only has one key server, the functions for distribution are, of course, not needed for that area.

New log entries are always added to the end of the log, and old entries are not allowed to be changed. The certificate log is preferably stored on a write once device for maximum security. The log contains an audit trail of the following events:

- Registration of public keys.
- Registration of signatures.
- Notification of key compromises.

4.4.1 Structure of log entries

Registration of	Resulting certificate
public key for the server	T, S, E_S
public key for A	$D_S(T, A, E_A)$
A's signature X	$D_S(T, A, E_A, X)$
key compromise for A	$D_S(T, A, E_A, XCOMP)$

Table 7: Structure of certificates in the log.

An entry in the log consists of two parts: a certificate, that is, data signed by the key server, and the same data in cleartext to simplify searches. The formats of the different certificates are shown in table 7. The variables are interpreted as follows:

T	A timestamp created by the server
S	The name of the server
E_S	The public key of the server(s)
$D_S(\ldots)$	The server's signature transformation
A	The name of an entity
E_A	The public key of entity A
X	A's signature for a message
$XCOMP$	A's signature for the message "compromise"

4.4.2 Synchronization

When multiple key servers are present in a key management area, the operations on the log have to be synchronized, as for all distributed databases, in order to maintain consistency. An optimistic algorithm for concurrency control is used in SPION, because the prerequisites is very good. For example, key server recovery after failure is very simple, since any other running server's database is itself a complete event history.

With an optimistic synchronization method, timestamps are used to detect conflicting operations. Each key server maintains a read time (T_r) and a write time (T_w) for the local copy of the certificate log. This is also a simplification compared to a "real" distributed database, where read and write times are usually needed for smaller objects in the database.

In this environment, it isn't possible for read operations to conflict, since every operation is independent of all other operations. Therefore, reads can be performed without synchronization—the read time of the local log (T_r) is only updated to the current time. However, there is one situation which needs care: when a write operation is in progress. If the write operation affects the result of the read in any way, the read operation has to wait until the write is completed.

A write operation is assigned a timestamp (T) when it is initiated. The procedure used to synchronize write operations is shown in figure 5 for the initiating server and in figure 6 for the other (remote) servers. Conflicts shouldn't occur very often since the transactions are short (only one operation). Note that a write operation can't conflict with another operation on the local key server, since the timestamp assigned to the transaction is new and thus larger than the local T_r.

```
$T$ := current-time

write locally and block dependent reads

send "prepare" message

wait for replies (*ok* or *conflict*)

**if** no *conflict* reply **then**
    send "commit" message
    $T_w := \max(T, T_w)$
    $T_r := T_w$
    allow reads
**else**
    send "abort" message
    undo the local write
    allow reads
    restart the write operation
**endif**
```

Figure 5: Write synchronization procedure for initiating server.

```
**case** message **of**

"prepare":
    **if** $T \geq T_r$ **then**
        reply "ok"
        write locally and block dependent reads
        $T_w := \max(T, T_w)$
        $T_r := T_w$
    **else**
        reply "conflict"
    **endif**

"commit":
    allow reads

"abort":
    undo the local write
    allow reads

**endcase**
```

Figure 6: Write synchronization procedure for non-initiating servers.

A Definitions

The following selected definitions are mainly taken from the OSI Security Architecture [5]. Additional definitions can be found in the references.

For more information on computer and network security we recommend the tutorial by Marshall D. Abrams and Harold J. Podell [1].

Access control: The prevention of unauthorized use of a resource, including the prevention of use of a resource in an unauthorized manner.

Access control list: A list of entities, together with their access rights, which are authorized to have access to a resource.

Active threat: The threat of a deliberate unauthorized change to the state of the system.

Audit trail: Data collected and potentially used to facilitate a security audit.

Authentication: The corroboration, that a peer entity in an association is the one claimed, or in the context of connectionless service, that the source of data received is as claimed.

Confidentiality: The property that information is not made available or disclosed to individuals, entities, or processes.

Digital signature: Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery.

End-to-end encipherment: Encipherment of data within or at the source end system, with the corresponding decipherment occurring only within or at the destination end system.

Event-handling: Detection and reporting of security-relevant events.

Integrity: The property that data has not been altered or destroyed in an unauthorized manner.

Key management: The generation, storage, secure distribution and application of keys in accordance with a security policy.

Link-by-link encipherment: The individual application of encipherment to data on each link of a communications systems.

Masquerade: A type of attack where an entity pretends to be another entity.

Message Authentication Code (MAC): An electronic data integrity seal usually in the form of a cryptographic checksum.

Notarization: The registration of data with a trusted third party that provides for future recourse to the data and assures accuracy concerning its characteristics such as content, origin, time and delivery of the data.

Padding: The generation of spurious instances of communication, spurious data units and/or spurious data within data units.

Passive threat: The threat of unauthorized disclosure of information without changing the state of the system.

Peer entities: Entities within the same layer.

Protocol data unit (PDU): A unit of data specified in a protocol and consisting of protocol information and possibly user data.

Privacy: The right of individuals to control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

Repudiation: Denial by one of the entities involved in a communication of having participated in all or part of the communication.

Security audit: An independent review and examination of system records and activities in order to test for adequacy of system controls, to ensure compliance with established policy and operational procedures and to recommend any indicated changes in control, policy and procedures.

Security Management Information Base (SMIB): The storage place for security-relevant information, such as public keys and access rights.

Selective field protection: The protection of specific fields within a message which is to be transmitted.

Service data unit (SDU): An amount of interface data whose identity is preserved from one end of a connection to the other.

Traffic analysis: The inference of information from observation of traffic flows (presence, absence, amount, direction and frequency).

Traffic flow confidentiality: A confidentiality service to protect against traffic analysis.

The following symbols are used in the report:

Symbol	Definition
A	A communicating entity
D_A	The private (decrypting) key of A
E_A	The public (encrypting) key of A
$D_A(\ldots)$	The decrypting transformation of A
$E_A(\ldots)$	The encrypting transformation of A

References

[1] Marshall D. Abrams and Harold J. Podell. *Tutorial—Computer and network security.* IEEE Computer Society Press, 1987. IEEE Computer Society order number 756.

[2] Dorothy E. Denning. *Cryptography and Data Security.* Addison-Wesley Publishing Company, Inc, 1982.

[3] Dorothy E. Denning. Protecting public keys and signature keys. *Computer*, 16(2):27–35, February 1983.

[4] ISO, International Organization for Standardization. International Standard 7498-1, *Information processing systems - Open Systems Interconnection - Basic Reference Model*, 1984. Ref. No. ISO 7498-1984.

[5] ISO, International Organization for Standardization. International Standard 7498-2, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security architecture*, 1988. Ref. No. ISO 7498-2-1988 (E).

[6] ISO, International Organization for Standardization. International Standard 8072, *Information processing systems - Open Systems Interconnection - Transport service definition*, first edition, July 1986. Ref. No. ISO 8072-1986 (E).

[7] ISO, International Organization for Standardization. International Standard 8073, *Information processing systems - Open Systems Interconnection - Connection oriented transport protocol specification*, first edition, July 1986. Ref. No. ISO 8073-1986 (E).