Transport of Intercepted IP Traffic (TIIT)
Version 0.2.0

Nederlands Forensisch Instituut, Ministerie van Justitie
Binnenlandse Veiligheidsdienst, Ministerie van BZK
Korps Landelijke Politiediensten, Ministerie van BZK
Editor: EJ van Eijk, NFI

Date: February 2001
Last change: January 31, 2001

Status of this memo

This memo is a standard.

Changes to version 0.1.2:

- The number of different structures has been reduced to facilitate implementation, and to minimize the chance of miscommunication in the S1–S2–T1–T2 communication. This has also resulted in a increment of the majorVersionNumber in PDU2.
- Session startup and teardown had been specified ambiguously, that has been rectified.
- Definition of the TIIT-keepalive and TIIT-alive packets has been edited, to iron out any uncertainties.
- The length fields in the `PDU2` and `DPDU` messages were updated to give uniform semantics. The length describes the length of the encapsulated content at all time.
- A stricter definition is created of the SHA packets in Section 6.1.1.
- Type opaque is used instead of string to give way to more freeform identifiers in Program 11.1. The intention still is to have human readable strings in these identifiers.
- Section 13.2 is updated to reflect that the AES has been chosen. The cipher mode and IV have been specified.
- Section 6.1.1, Table 9.1 and Program 9.2: The DPDU should be padded to an even 128bit (16byte) boundary before encryption.
- WarrantPeriod in Section 8.2 is dropped, as it was not used.
- `HI2_Init` and `HI2_End` are renamed to StartSession and EndSession respectively.

Copyright notice

Abstract

This document describes the way the result of interception (CC) should be delivered to a LEMF. It's basis is that each funcional unit should not contain all the necessary information to identify the target of interception and the interested LEA.

Contents

List of Tables

List of Programs

Chapter 1

Introduction

This memo describes in detail the Internet LI interface required for LI based upon the requirements described in the Functional Specifications [CLE+00].

It provides the specification of the interface from an Interception Function within an IP network to a Law Enforcement Monitoring Facility for the purpose of providing data to Law Enforcement Agencies (LEAs) in the area of Lawful Interception (LI) of communications.

This memo also describes a conceptual architecture that can perform LI in a distributed manner. The architecture also serves as a reference how to secure the delivery of the intercepted data to the LEA.

This is a technical document, and as such data structures used in transport channels are readily inserted into the document where needed, instead of in an Annex. This will minimize the number of lookups necessary while reading this document.

This memo does not describe how the data should be intercepted.

The keywords “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in [Bra97].

Chapter 2

Scope of this document

This document does not describe the means by which data should be intercepted from the network. Those means rely on the precise architecture of the network on which LI should take place.

Chapter 3

Definitions and abbreviations

CC Content of Communication. Identical to Target Traffic.

HI Handover Interface

IA Interception Authorization

IRI Intercept Related Information

LEA Law Enforcement Agency

LEMF Law Enforcement Monitoring Facility

LI Legal Interception

RI Results of Interception, CC and IRI

Target Traffic Network traffic originating at, or being routed to the target identity

TLS Transport Layer Security Protocol, defined in [DA99].

Chapter 4

User requirements for transport

This protocol should be fulfilling the following goals, a requirement stated in [CLE+00].

1. Protect information which can be gathered from the protocol traffic that allows an outsider to observe:
   (a) How many target identities are subject to interception.
   (b) Which target identities are subject to interception.
   (c) Which LEAs requested the LI of any target identity.
2. Require a minimum number of personnel to be involved in the LI requirement.
3. Allow an authenticated and secure transmission of confidential data over a (possibly) hostile IP network, e.g. the Internet.
4. Allow the transmission to be resilient to communication errors at a lower layer, including malicious tampering with the lower level communication.
5. Provide standard logging entries which can be used to prevent or trace misuse of the technical functions integrated in the IP network to intercept data from the network.
6. Allow implementation using only open standards.

Chapter 5

Description of Handover Interface

Logically the Handover Interface (HI) can be divided in three parts:

HI1 concerned only with the administrative protocol involved with LI.

HI2 provides a transport for Interception Related Information (IRI).

HI3 provides a transport for Communication Content (CC)

5.1 HI1

Secure communication SHALL be used to transport relevant documents requesting to initiate, prolong or terminate LI. Which type of secure communication is used depends on the LEA requesting the LI.

The HI1 will also be used for transport of cryptographic key material where appropriate:

- public keys for asymmetric cryptography where this is used for authentication
- symmetric case keys associated with a particular Interception Authorization.

5.2 HI2

Data transported through the HI2 are:

1. Authentication events, e.g. login/logout information provided by a RADIUS server.
2. Log events related to a target identity, e.g. POP box access events. Which log events are to be transferred is defined in [RV00].

5.3 HI3

Data transported through the HI3 are:

1. Content of Communication
2. Information generated from the CC, e.g. hash results.

5.4 HI bundling

For practical purposes HI2 and HI3 MAY be bundled together on one communication channel, if that provides economical or technical advantages.

HI1 MAY NOT use the same communication channel as either HI2 or HI3 at present (pending security and threat analysis of actual implementations of the interfaces.) [1]

For HI2 and HI3 traffic, bundling is REQUIRED in a secure tunnel if the communication channel used is routed on “the Internet”.

[1] This may be subject to change in a future revisions of this document.

Chapter 6

Transport Implementation

Figure 6.1 describes the general architecture necessary to create a reliable transport for HI2 and HI3.

$S1$ and $S2$ are two functional entities that implement two functions on the premises of the ISP:

$S1$ Interception of traffic directed at or originating from the target identity.

$S2$ Gathering of traffic from S1 and transport it to $T1$ by a secure means.

The following items are true for $S1$ and $S2$:

1. One $S2$ functional entity MAY serve multiple S1 functional entities.
2. $S1$ and $S2$ MAY be mapped onto multiple physically separated machines. If $S1$ and $S2$ are not mapped onto the same physical machine $A1$ denotes the route from $S1$ to $S2$. $A1$ MUST be a secure channel.

$T1$ and $T2$ are the functional entities on the LEA side that implement the receiving side of the protocol. C1 is a secure channel from $T1$ to $T2$. C1 SHALL NOT be "the Internet" or any other public network. The security measures necessary to secure C1 are defined by the receiving LEA.

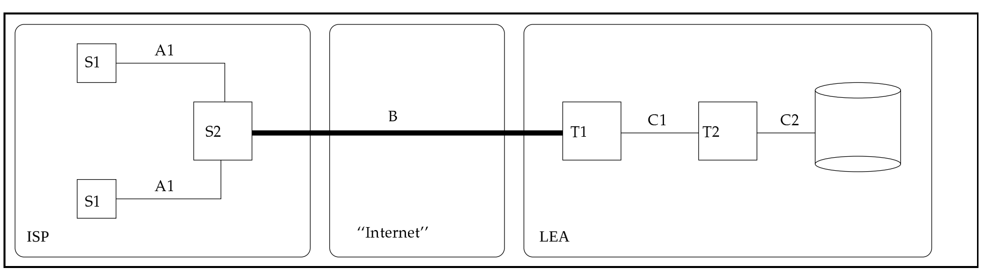

Figure 6.1: Architecture overview

6.1 Functional descriptions

6.1.1 $S1$

These are the functional requirements of $S1$:

1. $S1$ MUST intercept all the target traffic as described in [CLE$^+$00]. $S1$ should be dimentioned adequately to the original stream the traffic is gathered from.
2. $S1$ MAY use or offer SNMP functionality for day to day maintenance monitoring of the correct operation of the unit. It MAY NOT be configurable by SNMP. CC or IRI MAY NOT be monitored with SNMP.
3. CC or IRI MAY NOT be monitored remotely.
4. $S1$ MAY be configured from an $S2$ functional entity if a client-server relationship between those functional entities exists.
5. $S1$ MUST generate a correct timestamp at the time of interception of each packet. The timestamp is derived from the system time.
6. $S1$ MUST maintain a correct system time, by using the NTP protocol [Mil92]. $S2$ MAY operate as a stratum 2 NTP server.
7. $S1$ MUST generate a SHA hash for every 64 packets intercepted from one stream of targettraffic. The SHA hash is computed over the plaintext of the DPDU part[1] in a PDU2 packet (see Program 9.1. TIIT-Alive, TIIT-Keepalive, HI2:StartSession and HI2:EndSession packets are not included in the SHA hash.

   The DPDUs MUST be logically concatenated into a buffer, the SHA hash will be computed over this buffer. If less than 64 packets come available in a 300 second period, the SHA hash MUST be computed over the logical buffer containing only the available packets.
8. $S1$ MUST encrypt the target traffic with a known cryptographic key. This key is specified for the target identity in the IA. Other traffic from $S1$ to $S2$ and vice versa SHOULD be encrypted, i.e. SNMP status traffic.
9. $S1$ SHALL NOT have an IP stack operating on the intercepting side.

[1] including padding

6.1.2 $S2$

These are the functional requirements of $S2$:

1. $S2$ MUST open an TLS/SSLv3 tunnel to every point of delivery defined in the legal authorization. The keys are negotiated via the HI1. If a point of delivery cannot be reached the $S2$ MUST try to connect every 300 seconds. If a point of delivery cannot be reached for more than 3600 seconds, authorized personnel and receiving side (named in the warrant) MUST be notified by other means than HI2.

   Instead of a TLS/SSLv3 tunnel it is also possible to establish a TCP/IPSec connection with IKE doing the authentication between $S2$ and $T1$. For the remainder of this RFC where TLS/SSLv3 is written, also TCP/IPSec with IKE can be read.

2. $S2$ MUST accept traffic from every $S1$ functional entity with which it has a authenticated client-server relation. The accepted traffic MUST be forwarded on one of the open TLS/SSLv3 tunnels. Which tunnel is used to forward the traffic SHOULD be decided randomly.

3. $S2$ SHALL authenticate the $S1$ client before accepting traffic. Authentication MAY be based on IP address if, and only if, network infrastructure does not allow other traffic to be directed at $S2$.

4. The TLS/SSLv3 tunnel SHALL only accept allowed cryptosuites. Provision MUST be taken that the negotiation of other cryptosuites than the allowed set MUST result in disconnection of the tunnel. An alarm MUST be raised to authorized personnel in this case.

5. $S2$ MAY use or offer SNMP functionality for monitoring the operation of the unit. It MAY NOT be configurable by SNMP.

6. $S2$ MAY operate as a stratum 2 NTP server.

6.1.3 $T1$

These are the functional requirements of $T1$:

1. $T1$ MUST accept incoming TLS/SSLv3 tunnels from every known $S2$ functional unit. Known means that both the IP address(range) and public key of the $S2$ are available to the $T1$. The keys are negotiated via the HI1.

2. Incoming traffic from unknown IP addresses MAY be discarded.

3. $T1$ MUST accept traffic from every $S2$ functional entity with which it has a authenticated client-server relation. The accepted traffic MUST be forwarded to a $T2$. Which $T2$ will be chosen depends on the target identity.

4. $T1$ MUST authenticate the $S2$ client before accepting traffic.

5. Connections with unauthorized keys MAY be discarded.

6. The TLS/SSLv3 tunnel should only accept allowed cryptosuites. Provision MUST be taken that the negotiation of other cryptosuites than the allowed set MUST result in disconnection of the tunnel. An alarm SHOULD be raised to authorized personnel in this case.

7. $T1$ MAY use or offer SNMP functionality for monitoring the operation of the unit. It MAY NOT be configurable by SNMP.

8. $T1$ MAY deliver incoming packets to more than 1 $T2$.

9. $T1$ MAY operate as a stratum 1 NTP server.

6.1.4 $T2$

These are the functional requirements of $T2$:

1. $T2$ MUST unalterably store the plaintext of the incoming messages from $T1$ without delay.

2. Incoming SHA lists SHOULD be compared to the computed SHA of the incoming intercepted traffic. $T2$ operators SHOULD be notified if any mismatches occur.

Chapter 7

Notation

All description and encoding of data structures mentioned in this document are following the External Data Representation standard, as described in [Sri95]. However, two small derivations are used:

1. Bitfields are used, as in C, bigendian order.
2. All structures are not padded to an even 4byte size, unless stated otherwise explicitly.
3. All non-byte or opaque values should be interpreted as an integer of that size in network order. Example: `opaque length[3]` should be interpreted as a 24bit integer, in network order.

Chapter 8

Global data structures

Several data items will be shared by both the Provider and the LEMF. This section describes these items.

8.1 Provider identifier

The identifier `provider_identifier` is a unique number that identifies a specific provider, it is defined as an unsigned 16 bit integer. This identifier is to be used in communications over any of the interfaces HI1, HI2 or HI3 to identify the provider.

8.2 Time information

All information regarding time will be stored as defined in Program 8.1:

```
struct {
    unsigned int seconds;  /* seconds since 1-1-1970 */
    unsigned int useconds; /* microsecond timer */
} Timestamp;
```

Program 8.1: Timestamp definition

This is the standard UNIX format for storing timestamps.

8.3 Sequence number

Sequence numbers should be strictly monotonically increasing for every packet sent to any LEMF. The initial value of seqNum MUST be larger than

0x10. If wrap around of the value of seqNum should occur, the wrapped value MUST be smaller than 0x10.

```
struct {
    opaque snifferID[1];
    opaque seqNum[3];
} SequenceNumber;
```

Program 8.2: Sequence number description

8.4 Target Identifier

The target identifiers are structured as described in Program 8.3. The LEMFID is the MD5 hash generated by the LEMF. In this way a LEMF can generate its own identifiers, without compromising the interested LEA if the packet is intercepted en route.

```
struct {
    opaque md5hash[16];
} TargetID;
```

Program 8.3: Target ID structure

Chapter 9

$S1$ – $T2$ Traffic definition

CC and IRI packets transported from $S1$ to $T2$ are encapsulated at $S1$ in a PDU2 packet as defined in Program 9.1 and Table 9.1. The CC and IRI information itself is encoded in a DPDU packet according to Program 9.2, Table 9.2 and Table 9.4.

Transport of traffic between $S1$ and $S2$ should use TCP on port 1903. IPSec or TLSv1.0/SSLv3 MAY be used to enhance the confidentiality of the data en route if the data does not leave a building. IPSec or TLSv1.0/SSLv3 MUST be used in all other cases.

Traffic between $T1$ and $T2$ is transported over a secured channel. Under certain conditions physical security and total administrative control of the route from $T1$ and $T2$ can be enough.

After every packet sent on a TLS link a Keep Alive counter is restarted. When that counter reaches 120 seconds a TIIT-KeepAlive packet is sent. $T1$ MUST respond within 30 seconds with a TIIT-Alive packet. If this packet is not received, it must be assumed that $T1$ is not reachable, and the corresponding TLS connection MUST be torn down. When the link has been torn down, general rules for connecting $S2$ to $T1$ apply as defined in Section 6.1.2.

TIIT-Keepalive and TIIT-Alive packets are PDU2 packets with `versionMajor` set to `0x0`, and `versionMinor` set to `0xf` and `0xe` respectively.

TIIT-Keepalive and TIIT-Alive packets are transmitted without any case-related encryption. Also, the sequencenumbers they use are sequencenumbers only relevant to the communication between an $S2$ and a $T1$. The targetID is set to `0x00`. TIIT-Alive messages are also not used in the SHA packet calculations.

$T1$ MUST NOT accept traffic from any $S2$ if all the $T2$'s cannot be reached, or when traffic cannot be buffered in the mean time.

$S2$ MUST NOT accapt traffic from any $S1$ if it cannot create a secure TLS tunnel with at least one $T1$.

```
struct {
    unsigned int versionMajor:4;
    unsigned int versionMinor:4;
    opaque length[3];
    TargetID targetid;
    SequenceNumber sequenceNumber;
    opaque encrypedDPDU<length>;
} PDU2;
```

Program 9.1: Description of a PDU2 packet

Attribute	Description	Default Value
versionMajor	Major version number of the protocol	0x2
versionMajor	TIIT-keepalive or Alive	0x0
versionMinor	RC4	0x0
versionMinor	Rijndael	0x1
versionMinor	RC6	0x2
versionMinor	Mars	0x3
versionMinor	Serpent	0x4
versionMinor	Twofish	0x5
versionMinor	TIIT-keepalive	0xf
versionMinor	TIIT-alive	0xe
length	The transmitted length of the DPDU, including padding	None
targetID	Unique target identification	None
sequenceNumber	Assigned by $S1$	None
encryptedDPDU	DPDU encrypted with the algorithm denoted by the version minor number	None

Table 9.1: PDU2 description

```
struct {
    unsigned int providerID:16;
    unsigned int direction:2;
    unsigned int payLoadID:14;
    unsigned int length;
    Timestamp timestamp;
    opaque payload<length>;
    opaque padding<>;
} DPDU;
```

Program 9.2: Description DPDU packet

Attribute	Description	Default Value
providerID	Unique provider identification. Assigned by the ministry of V&W	None
direction	Direction of the payload	Table 9.3
payLoadID	Identification of the payload	Table 9.4
length	Total size of the payload	
timestamp	See definition of Timestamp	None
payload	defined by payloadID	None
padding	extra bytes to create an even 16byte sized DPDU	0x00

Table 9.2: DPDU description

Value	Description
00b	Unknown (cannot be determined)
01b	From user to network
10b	From network to user

Table 9.3: Payload Direction

Value	Description of Payload	Program
0x0000	NULL packet, can be used for traffic generation	
0x0001	HI3: IPv4 packet	
0x0003	HI3: IPv6 packet	
0x0006	HI3: Ethernet packet	
0x0100	HI3: Email packet	
0x1000	HI3: SHA Packet	12.1
0x2000	Fake: generated packet	
0x3000	HI2: Start Session	11.1
0x3001	HI2: End Session	11.1
0x3002	HI2: SuccessfulDialupLoginIPv4	11.1
0x3003	HI2: SuccessfulDailupLoginIPv6	11.1
0x3004	HI2: FailedLogin	11.1
0x3005	HI2: SuccessfulIPv4DHCP-Registration	11.1
0x3006	HI2: Logout	11.1
0x3100	HI2: Generic LogFile lines such as produced by `syslogd`	
0x3F00	HI2: Generic ISP – LEA message	
0x3FFC	HI2: $S1$ malfunction	
0x3FFD	HI2: Attempt to connect TLS to $S2$	
0x3FFF	HI2: Undefined Error	

Table 9.4: Payload Identification

Chapter 10

Handover Interface 1

Through the HI1 the following items are negotiated or sent:

1. IP addresses of five (5) $T1$ points and their corresponding X.509 certificates.
2. Target identification as defined in [CLE$^+$00].
3. A 192bit key belonging to a target. If RC4 is used to encrypt the DPDU packets, the lower 128bits are used.

Official media for HI1 are paper and WORM discs, e.g. CD-Recordable. Other media types MAY be negotiated between LEA and ISP at a later time.

The electronic format for HI1 information is in XML format. An example for a warrant is described in Appendix A.

Chapter 11

Handover Interface 2

This section describes the format of messages that can be exchanged over HI2. Messages transmitted via HI2 originate at functional entity $S1$ and terminate at $T2$. Functional entity $T2$ MAY discard any message that does not comply to the given definitions. A log event MUST be generated if this occurs.

Messages on the HI2 are encapsulated in DPDU packets, described in Program 9.2.

11.1 HI2 – Session establishment

After a TLS tunnel has been established between $S2$ and $T1$, the first message on HI2 MUST be a HI2:StartSession stated in the Program 11.1. This message MUST have DPDU::payLoadID == 0x3000. The targetID in this message will be the target for which a valid warrant has been received.

If the TLS tunnel is already established, and a new warrant is activated in the $S1$ also a HI2:StartSession MUST be sent. The packet is encrypted with the casekey, but not included in the SHA calculations.

11.2 HI2 – Session termination

Similarly to session establishment, a terminated session – indicated by the event that the warrant has expired – will give rise to HI2:EndSession message (see Program 11.1). This message MUST have `DPDU::payLoadID` set to `0x3001`. The packet is encrypted with the casekey, but not included in the SHA calculations.

If there are no more warrants left active on $S1$, the TLS tunnel SHOULD be torn down after this message.

11.3 Operational message flows

Messages on the HI2 are encapsulated in DPDU packets, described in Program 9.2. The payloadID field in the DPDU packets defines the structure in the payload attribute, see Table 9.4.

Value	**Description of attribute**
dialedNumberFromPOP	The number dialed by the target. Filled with ascii 0x20 if not available
cLIFromTarget	The number used by the target. Filled with ascii 0x20 if not available
location	X and Y coordinates of the target. Filled with ascii 0x20 if not available
popID	An identifier assigned by the ISP to a POP location.
assignedAddress	The IPv4 or IPv6 address assigned to the target.
leaseTime	The maximum time a target has this address as a valid IP address.
MACAddress	IEEE 802.3 (Ethernet) host identifier

Table 11.1: Message attribute description

```
struct {
        Timestamp beginOfSession;
} StartSession;

struct {
        Timestamp endOfSession;
} EndSession;

struct {
        telephoneNumber dialedNumberFromPOP;
        telephoneNumber cLIFromTarget;
        opaque location[6];
        opaque popID[8];
        IPv4Number assignedAddress;
} SuccessfulDialupLoginIPv4;

struct {
        telephoneNumber dialedNumberFromPOP;
        telephoneNumber cLIFromTarget;
        opaque location[6];
        opaque popID[8];
        IPv6Number assignedAddress;
} SuccessfulDailupLoginIPv6;

struct {
        IPv4Number assignedAddress;
        Timestamp leaseTime;
        MACAddress referenceMACAddress;
        opaque location[6];
        opaque popID[8];
} SuccessfulIPv4DHCPRegistration;

struct {
        telephoneNumber dialedNumberFromPOP;
        telephoneNumber cLIFromTarget;
        opaque location[6];
        opaque popID[8];
} FailedLogin;

struct {
        opaque location[6];
        opaque popID[8];
} Logout;
```

Program 11.1: HI2: Messages

```
typedef opaque telephonenumber[20];
        // Telephone number in Ascii, padded with 0x20

typedef int IPv4Number;    /* 32 bits */

typedef int IPv6Number[4]; /* 128 bits */

typedef opaque MACAdress[6];
```

Program 11.2: HI2 Structures

Chapter 12

Handover Interface 3

This section describes all necessary data structures and message flows to get a proper instance of the HI3.

12.1 Session establishment

Assumptions:

1. An IP route exists from $S1$ to $S2$.
2. An IP route exists from $S2$ to $T1$.
3. A route exists from $T1$ to $T2$.

This the message sequence to get the system running.

1. $S1$ SHOULD establish a secure path to $S2$ when it is put in operational mode. $S1$ SHOULD authenticate to $S2$.
2. $T1$ establishes a path to $T2$
3. $S2$ establishes a TLS connection to every $T1$ that is named in the warrant. See [DA99] for the correct message flows.
4. A TLS connection MUST be initiated by functional entity $S2$. $T1$ and $T2$ MAY NEVER initiate a session. Any deviation from this behavior MUST generate a log event that MUST be reported immediately to the LEMF via HI2 and in a timely fashion via the communication channel used for HI1.
5. A `StartSession` packet (Program 11.1) is sent from $S1$ to $T2$.

At this point $S2$ is connected to $T1$, and the system is ready to transport intercepted traffic[1].

12.2 Operational message flows

This section describes the message flows occurring between S1 and $T2$. Message flows will be through $S2$ and $T1$.

Interception IPv4 or ICMPv4 packet An intercepted IPv4 packet will be encapsulated in the payload of a DPDU packet (See Program 9.2. The `payloadID` is set to `0x0001`.

Interception IPv6 or ICMPv6 packet An intercepted IPv6 packet will be encapsulated in the payload of a DPDU packet (See Program 9.2. The `payloadID` is set to `0x0003`.

SHA Description packet Every 64 packets intercepted generate a SHA description packet described in Program 12.1. A SHA description packet is encapsulated in a DPDU packet. The SHA hash is computed over the plaintext of the DPDU part in a PDU2 packet.

Traffic Shaping The output rate of packets at $S1$ MAY be traffic shaped to alleviate worst case bandwidth behavior and make traffic analysis more difficult. Care must be taken that a packet SHALL be sent within 30 seconds after being intercepted.

Fake traffic Fake Packets MAY be output at $S1$ to make traffic analysis more difficult. These packets have their payloadID set to `0x2000`. The size of the packets SHALL be pseudo random between 64 and 1500 bytes. The content of the packets SHALL be pseudo random.

```
struct {
    unsigned int n;
    SequenceNumber snList<n>;
    unsigned char SHAhash[20];
} SHAPDU;
```

Program 12.1: SHA information PDU

The DPDUs are encrypted, and the encrypted content is inserted in a PDU2, as described in Program 9.1. PDU2 packets are inserted in the TLS tunnel from $S2$ to $T1$. Which TLS tunnel is used to transport the PDU2 packet SHOULD be randomly on all available tunnels.

[1] Deviations are possible in case of non-regular LI

Attribute	Description	Default
n	Number of sequence numbers in `snList`	64
snList	An array of `n` SequenceNumbers. It holds the sequence numbers of the DPDU packets for which this SHA hash was computed	
SHAhash	The computed SHA hash	

Table 12.1: SHA Information PDU Attributes

Chapter 13

Use of cryptography

Cryptography is used as a tool to safeguard the intercepted data during transport against traffic analysis as well as to ensure confidentiality of the contents. All keys used within the framework of this protocol MUST be kept from all parties outside of the protocol. Only authorized personnel MUST be able to access the keys or operate the equipment using the keys. See also [CLE+00].

13.1 Cryptographic key representation

Keys for asymmetric algorithms – as used for authentication of the TLS tunnel – MUST be distributed as X.509 certificates on the HI1. X.509 certificates used in LI expire after one year.

Keys for symmetric algorithms – i.e. the DPDU encryption – MUST be specified as hexadecimal strings by the LEA over the HI1. The strings will be in big-endian notation.

13.2 PDU encryption

All PDUs as defined in Program 9.2 MUST be encrypted with a symmetric algorithm using a key that is specified in the corresponding IA. The following cryptography is defined:

1. the symmetric algorithm to be used will be the AES at a medium key size (192 bits) in Cipher-Block-Chaining (CBC) mode. The initial vector will be all zero (0x00).
2. in the mean time RC4 with a key length of 128 bits will be used.

Both options MUST be implemented.

13.3 TLS Tunnel specifications

The path from $S2$ to $T1$ is an encrypted tunnel, based on SSLv3 or TLS [DA99]. The path is encrypted using a session key that is negotiated based on the key material specified in the X.509 certificates available at the ISP and the LEA. Care should be taken that cryptosuites specified here are eligible for use. Fallback to plaintext or another cryptosuite than the predefined ones SHALL NOT occur.

Possible cryptosuites are:

1. RSA 2048 with the AES at a medium key size (192 bits).
2. RSA 2048 with RC4 with a 128 bit key.

Every PDU2 that is to be transported MUST be encapsulated in one TLS Record [DA99].

13.4 Versions and cryptography

The AES candidate SHOULD be used for the symmetric encryption used in the TIIT protocol. Use of the AES for encryption MUST be signaled by a minor version number 1. Until the AES is widely available, RC4 128 MAY be used. Use of RC4 with a 128 bit key MUST be signaled by the use of a minor version number of 0.

Bibliography

[Bra97] S. Bradner. Key words for use in rfcs to indicate requirement levels. Request for comment 2119, IETF, March 1997.

[CLE+00] CO, LM, EJ, EL, and SJ. Functional specifications li in the netherlands, May 2000.

[DA99] T. Dierks and C. Allen. The tls protocol version 1.0. Request for comment 2246, IETF, Januari 1999.

[Mil92] David L. Mills. Network time protocol (version 3). Request for comment 1305, IETF, March 1992.

[RV00] RV. Algemene maatregel van bestuur voor ip taps, May 2000.

[Sri95] R. Srinivasan. Xdr: External data representation standard. Request for comment 1832, IETF, August 1995.

Appendix A

HI1 XML example

This appendix sketches an example of a warrant as it can be in a digital format.

```
<?xml version=''1.0''?>
<hi1:warrant>
  <hi1:delivery>
```

In the delivery section a number of delivery points are described. The number of delivery points is given up front.

```
    <hi1:attribute name=''Number of Delivery points''>
      <hi1:real val=''1''>
    </hi1:attribute>
    <hi1:address>
      <hi1:attribute name=''IP address''>
        <hi1:ipv4addr val=''192.168.18.1''>
      </hi1:attribute>
      <hi1:attribute name=''Certificate''>
        <hi1:x509cert val=''hex string of
                            binary X.509 cert''>
      </hi1:attribute>
    </hi1:address>
    <hi1:address>
    <hi1:targetKey>
      <hi1:attribute name=''targetKey''>
         <hi1:targetKey val=''hex string of tar-
getKey used by S1 and T2''>
      </hi1:attribute>
    </hi1:targetKey>
  </hi1:delivery>
```

At least one (1) hi1:address should exist within the delivery section, and a maximum of five (5).

```
  <hi1:targetIdentity>
```

A target can be identified by name and address data, account name or login name or by a list of times and ip addresses.

```
    <hi1:person>
      <hi1:attribute name=''Name''>
        <hi1:string val= ''The target's name''>
      </hi1:attribute>
      <hi1:attribute name=''Address''>
        <hi1:string val= ''The target's address''>
      </hi1:attribute>
    </hi1:person>
```

or

```
    <hi1:account>
      <hi1:attribute name=''Account name''>
        <hi1:string val=''someaccount@
                          somedomain.com''>
      </hi1:attribute>
    </hi1:account>
```

or

```
    <hi1:attribute name=''Number of locationInforma-
tion''>
      <hi1:real val=''1''>
    </hi1:attribute>
    <hi1:locationInformation>
      <hi1:attribute name=''IP address''>
        <hi1:ipaddress val=''176.16.23.22''>
      </hi1:attribute>
      <hi1:attribute name=''Start Time''>
        <hi1:time val=''A date/time in UTC''>
      </hi1:attribute>
      <hi1:attribute name=''End Time''>
        <hi1:time val=''A date/time in UTC''>
      </hi1:attribute>
    </hi1:locationInformation>
```

The number of `hi1:locationInformation` sections that are neccessary to uniquely identify a target is provider dependent. Start and end time MAY be the same.

```
  </hi1:targetIdentity>
```

The start and end time and date of the warrant.

```
  <hi1:legalPeriod>
    <hi1:attribute name=‘‘Start time’’>
      <hi1:time val=‘‘A date/time in UTC’’>
    <hi1:attribute>
    <hi1:attribute name=‘‘End time’’>
      <hi1:time val=‘‘A date/time in UTC’’>
    <hi1:attribute>
  </hi1:legalPeriod>
```

And the electronic signature of the judge signing the warrant.

```
  <hi1:judge>
    <hi1:attribute name=‘‘Name’’>
      <hi1:string val= ‘‘The judge’s name’’>
    </hi1:attribute>
    <hi1:attribute name=‘‘Certificate’’>
      <hi1:x509cert val=‘‘hex string of
                          binary X.509 cert’’>
    </hi1:attribute>
  </hi1:judge>
</hi1:warrant>
```