

Transport of Intercepted IP Traffic

This specification was produced by
"Working group TIIT Bake Off phase 2"
and is owned by
"Directorate General for Telecommunication and Post
of the Ministry of Economic Affairs (EZ)"

Reference

TIIT V1.0.0 (2002-09)

Ministry of Economic Affairs (EZ)

**Directorate-General for Telecommunications
and Post**

Willem Witsenplein 6
's Gravenhage
THE NETHERLANDS

Important notice

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF).

Users of the present document should be aware that the document may be subject to revision or change of status.

Copyright Notification

The copyright and the foregoing restrictions extend to reproduction in all media.

© Ministry of Economic Affairs 2002.
All rights reserved.

Contents

Introduction	4
1 Scope	5
2 References	5
3 Definitions and abbreviations	6
3.1 Definitions	6
3.2 Abbreviations	6
4 User requirements for transport	8
5 Description of Handover Interface	9
5.1 HI1	9
5.2 HI2	9
5.3 HI3	9
5.4 HI bundling	9
6 Transport Implementation	10
6.1 Functional descriptions	10
6.1.1 S1	10
6.1.2 S2	11
6.1.3 T1	11
6.1.4 T2	12
7 Notation	13
8 Global Data Structures	14
8.1 Provider Identifier	14
8.2 Time information	14
8.3 Sequence Number	14
8.4 Target Identifier	14
9 S1 — T2 Traffic definition	15
9.1 S2 – T1 connection monitoring	15
9.2 Optional packet buffering	15
9.3 E-mail intercepts	15
9.4 Generic packet formats and attribute values	16
10 Handover Interface 1	19
11 Handover Interface 2	20
11.1 HI2 — Session establishment	20
11.2 HI2 — Session termination	20
11.3 IRI Data messages	21
11.4 IRI E-mail messages	22
12 Handover Interface 3	24
12.1 Session establishment	24
12.2 Operational message flows	24
13 Use of Cryptography	26
13.1 Cryptographic key representation	26
13.2 PDU encryption	26
13.3 TLS Tunnel specifications	26
Annex A (informative): HI1 XML example	27
Annex B (informative): Packet layout examples	29
History	32

Introduction

This document describes in detail the Internet Lawful Interception (LI) interface required for LI based upon the requirements described in the Functional Specifications [2].

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

This document 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 minimise the number of lookups necessary while reading this document.

This document does not describe how the data should be intercepted or analysed.

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 [1].

1 Scope

This document describes the way the result of interception, the Content of Communication (CC) should be delivered to a LEMF. Its basis is that no functional unit should contain complete information needed to identify the target of interception and the interested LEA.

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

2 References

For the purposes of this Technical Report, the following references apply:

- [1] S. Bradner. Key words for use in RFC's to Indicate Requirement Levels. Request For Comment 2119, IETF, March 1997.
- [2] CO, LM, EE, EL and SJ. WAI/GT/FuncSpecs, Functional specifications lawful interception of Internet traffic in The Netherlands, V1.0.1, June 2000.
- [3] T. Dierks and C. Allen. The TLS Protocol Version 1.0. Request For Comment 2246, IETF, January 1999.
- [4] David L. Mills. Network Time Protocol (Version 3) Specification, Implementation. Request For Comment 1305, IETF, March 1992.
- [5] Void.
- [6] R. Srinivasan. XDR: External Data Representation Standard. Request For Comment 1832, IETF, August 1995.
- [7] Federal Information Processing Standards Publication 197, 26 November 2001
- [8] D. Eastlake and P. Jones. US Secure Hash Algorithm 1 (SHA1). Request For Comment 3174, IETF, September 2001.
- [9] D. Harkins, D. Carrel. The Internet Key Exchange (IKE). Request For Comment 2409, November 1998
- [10] S. Kent, R. Atkinson. Security Architecture for the Internet Protocol . Request For Comment 2401, November 1998

3 Definitions and abbreviations

3.1 Definitions

For the purposes of the present document, the following terms and definitions apply:

Buffering: temporary storing of information in case the necessary connection to transport information to the LEMF is temporarily unavailable.

Law Enforcement Agency (LEA): An organisation authorised by a lawful authorisation based on a national law to receive the results of interceptions.

Law Enforcement Monitoring Facility (LEMF): A law enforcement facility designated as the transmission destination for the results of interception relating to a particular interception subject.

Provider: Any service provider that may be summoned to handover intercepted user data to the LEA. This may be an ISP (i.e. access providers), hosting provider or any other service provider.

Rijndael: The symmetric algorithm of choice for the AES.

S1: The data gathering function at the Provider side in the TIIT architecture, typically a sniffer or a function of a mail server.

S2: The data forwarding function at the Provider side in the TIIT architecture, responsible for setting up, maintaining and tearing down the connection to the LEMF.

Secure buffering: buffering in such a way that access to the buffered data by unauthorised parties is prohibited.

Secure channel: A communications path that prohibits information retrieval by unauthorised parties.

Secure tunnel: TCP/IPsec or TLS/SSLv3 connection.

T1: The data collection function at the LEMF side in the TIIT architecture.

T2: Data storage function at the LEMF side in the TIIT architecture.

Target identification: The identity that relates to a specific lawful authorisation as such. This might be an account name or similar identifier.

Target identity: The identity associated with a Provider for the Target of interception.

Target of interception: The subject (typically a person) of interception.

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

3.2 Abbreviations

For the purposes of the present document, the following abbreviations apply:

AES	Advanced Encryption Standard. Uses Rijndael as symmetric encryption algorithm. See [7]
CC	Content of Communication. Identical to Target Traffic.
DPDU	Data Compression Payload Data Unit
EmailPDU	PDU format for transporting intercepted E-mail messages
HI	Handover Interface
HI1	Handover Interface Port 1 (for Administrative Information)
HI2	Handover Interface Port 2 (for Intercept Related Information)
HI3	Handover Interface Port 3 (for Content of Communication)
IA	Interception Authorisation
IKE	Internet Key Exchange. See [9].
IPPDU	IP Protocol Data Unit. PDU format for transporting data from intercepted IP sessions
IPSec	IP Secure. See [10].

IRI	Intercept Related Information
ISP	Internet Service Provider
LEA	Law Enforcement Agency
LEMF	Law Enforcement Monitoring Facility
LI	Lawful Interception
NTP	Network Time Protocol, defined in [4]
PIPPDU	PDU format for transporting data from intercepted application specific parts of IP sessions
PDU2	Protocol Data Unit
SHA	Secure Hash Algorithm, defined in [8]
SNMP	Simple Network Management Protocol
SSL	Secure Socket Layer
TIIT	Transport of Intercepted IP Traffic
TLS	Transport Layer Security Protocol, defined in [3]

4 User requirements for transport

This protocol is designed to transport intercepted IP traffic from the point of interception to the required LEMF, while fulfilling the following goals, as required in [2].

- 1) Protect the protocol traffic so as to hide the information an outsider would use to try 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 a specific target identity.
- 2) Minimise the 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, like 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.

5 Description of Handover Interface

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

5.1 HI1

HI1 is concerned only with the administrative protocol involved with LI. This is used to handle communications between Provider and LEA concerning interceptions. E.g. warrants will be issued through this channel, extension and termination of LI will also be communicated through HI1. Secure communication SHALL be used to transport relevant documents. Which type of secure communication is used depends on the LEA requesting the LI. It is typically an offline communication channel.

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 Authorisation.

5.2 HI2

HI2 is used to send data about the communication by a target of interception, Interception Related Information (IRI) to the LEMF. Data transported through the HI2 typically are:

- Authentication events, e.g. login/logout information provided by a RADIUS server.
- Log events related to a target identity, e.g. POP box access events.

5.3 HI3

HI3 is used to send the actual intercepted data to the LEMF. These data will contain the actual content of the communication the LEA is interested in.

- Content of Communication, i.e. all intercepted IP traffic destined for or originated by the target of interception, or alternatively, all E-mail traffic to the target of interception.
- 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.

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

HI1 MUST NOT use the same communication channel as either HI2 or HI3.

6 Transport Implementation

Figure 1 describes the general architecture necessary to create a reliable transport for HI2 and HI3.

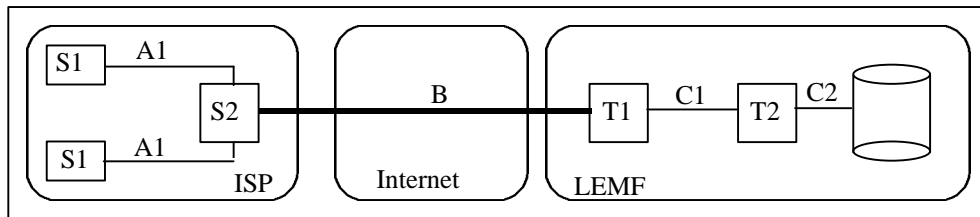


Figure 1: Architecture overview

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

S1 is the actual point of interception of traffic directed at or originating from the target identity. This can be a packet sniffer or an interception function in a mail server.

S2 is the function that gathers traffic from one or more S1's at the Provider location and transports it to the LEMF by a secure means.

A1 denotes the route from S1 to S2.

The following items are true for S1, S2 and A1:

- 1) One S2 functional entity *MAY* serve multiple S1 functional entities.
- 2) S1 and S2 *MAY* be mapped onto multiple physically separated machines.
- 3) If S1 and S2 are not mapped onto the same physical machine A1 *MUST* be a secure channel.

T1 and T2 are the functional entities on the LEMF side that implement the receiving side of the protocol.

T1 accepts the connections from one or more S2's at the Provider sites.

T2 stores the intercepted data for further analysis and other usage.

B is a secure channel on top of a non-secure infrastructure. Therefore encryption will be used to adequately secure the channel.

C1 is a secure channel from T1 to T2. C1 *SHALL NOT* be the Internet or any other public network. The receiving LEA defines the security measures necessary to secure C1.

C2 is not specified within the scope of this document.

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 [2]. S1 should be dimensioned 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 *MUST NOT* be configurable by SNMP. CC or IRI *MUST NOT* be monitored with SNMP.
- 3) CC or IRI *MUST 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 [4]. S2 MAY operate as a stratum 2 NTP server.
- 7) S1 MUST generate a SHA-1 hash for every 64 packets intercepted from one stream of target traffic. The SHA-1 hash is computed over the plaintext of the DPDU part including padding. Table 7 lists which packets should be included in the SHA-1 hash.

The DPDU's MUST be logically concatenated into a buffer, the SHA-1 hash will be computed over this buffer. If less than 64 packets come available in a 300 second period, the SHA-1 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 (e.g. SNMP status traffic) SHOULD travel on a secure channel.
- 9) S1 SHALL NOT have an IP stack operating on the intercepting side.

6.1.2 S2

These are the functional requirements of S2:

- 1) S2 MUST open a TLS/SSLv3 tunnel to every point of delivery defined in the legal authorisation. The keys are negotiated via the HI1. If a point of delivery cannot be reached the S2 MUST try to connect every 300 seconds. 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 specification 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 an 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 for each HI2 or HI3 message sent.
- 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. Otherwise, the mechanism by which S2 authenticates S1 is up to the provider.
- 4) The TLS/SSLv3 tunnel SHALL only accept allowed cryptosuites. Provisions 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 authorised personnel in this case.
- 5) S2 MAY use or offer SNMP functionality for monitoring the operation of the unit. It MUST NOT be configurable by SNMP.
- 6) S2 MAY operate as a stratum 2 NTP server.
- 7) S2 shall connect to TCP-port 1904 when using TLS/SSLv3. S2 shall connect to TCP-port 1903 when using TCP/IPSec.
- 8) S2 SHALL never act as a server towards the Internet.

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 an 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 unauthorised 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 authorised personnel in this case.
- 7) T1 MAY use or offer SNMP functionality for monitoring the operation of the unit. It MUST NOT be configurable by SNMP.
- 8) T1 MAY deliver incoming packets to more than one T2.
- 9) T1 MAY operate as a stratum 1 NTP server.
- 10) T1 SHALL listen for incoming TLS/SSLv3 based TIIT service connections on TCP-port 1904. T1 SHALL listen for incoming TCP/IPSec based TIIT service connections on TCP-port 1903.

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-1 lists SHOULD be compared to the computed SHA-1 of the incoming intercepted traffic. T2 operators SHOULD be notified if any mismatches occur.

7 Notation

All description and encoding of data structures mentioned in this document are following the External Data Representation standard, as described in [6]. However, some small derivations are used:

- 1) Bitfields are used, as in C, bigendian order.
- 2) All structures are not padded to an even 4 byte size, unless explicitly stated otherwise.
- 3) Variable length opaque data doesn't include the length part.
- 4) 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.
- 5) Unless specified otherwise, the type `int` shall mean a 32 bit integer.

8 Global Data Structures

Several data items will be shared by the HI2 and the HI3. 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 **MUST** be used to identify the provider by all S1 and S2 units deployed by this provider in communications over any of the interfaces HI1, HI2 or HI3.

8.2 Time information

All information regarding time will be stored as defined in Program 1:

```
struct {
    unsigned int seconds; /* seconds since 1-1-1970 0:00 hours UTC */
    unsigned int useconds; /* microsecond timer */
} Timestamp;
```

Program 1: Timestamp definition

This is the standard UNIX format for storing timestamps.

8.3 Sequence Number

The `sequenceNumber` is defined in program 2.

`snifferID` is a value determined by the Provider in order to distinguish between the different S1's connecting to an S2.

The `seqNum` value will be incremented for every packet sent, whether the LEMF channel is available or not. 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`.

Whenever it is not known what the next `seqNum` should be (e.g. when an S1 is rebooted without the possibility to store the last `seqNum` used), a wrap around **SHOULD** be forced by reinitialising to a number smaller than `0x10`.

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

Program 2: Sequence number description

8.4 Target Identifier

The target identifiers are structured as described in Program 3. The `TargetID` 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 3: Target ID structure

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 4 and Tables 1 through 4. The CC and IRI information itself is encoded in a DPDU packet according to Program 5, Tables 5 through 7.

All communication parts of the end-to-end connection S1-T2 SHALL be secure channels.

9.1 S2 – T1 connection monitoring

After every packet sent on a TLS/SSLv3 link a Keep Alive counter is restarted on S2. When that counter reaches 120 seconds S2 sends a TIIT-KeepAlive packet. T1 MUST respond within 30 seconds with a TIIT-Alive packet. If S2 does not receive this packet, it must assume that T1 is not reachable, and the corresponding TLS/SSLv3 connection MUST be torn down. When the link has been torn down, general rules for connecting S2 to T1 apply as defined in 6.1.2. When no data or TIIT-KeepAlive packets are received by T1 for 240 seconds, it MUST tear down the connection.

TIIT-KeepAlive and TIIT-Alive packets are PDU2 packets with `versionMajor` and `versionMinor` set according to Tables 2 and 3 respectively. TIIT-KeepAlive and TIIT-Alive packets contain zero-length encryptedDPDU. The `targetID` is set to all zeroes. The `sequenceNumber` is distinct and independent from sequence numbers used for other HI2/HI3 messages, and is only relevant to the communication between an S2 and a T1.

9.2 Optional packet buffering

Because there is no way for the LEMF to acknowledge received packets or to request retransmission at application level, and therefore significant parts of a stream of packets may be lost in transport, T1 MUST be able to pick up a stream of HI3 packets in mid-session by looking for a certain minimum number of contiguous `seqNum` values. S1 and/or S2 MUST therefore take care that they send the packets in an orderly fashion.

When the connection to the LEMF is temporarily unavailable, S1 and/or S2 MAY buffer HI3 encapsulated in PDU2 packets. Buffering MUST be done in a secure way. When the connection is re-established, the buffering device SHALL choose one of two options:

- Send buffered HI3 packets in correct order before resuming realtime forwarding.
- Discard the buffered HI3 packets, or equivalently, not buffer packets in the first place.

The above provisions for HI3 packets do not hold for HI2 packets. S1 and S2 SHALL always try to send HI2 packets and SHALL therefore buffer HI2 packets as needed.

NOTE 1: the exact number of contiguous `seqNum` values at which T1 will recognise a pattern is implementation specific.

If T1 cannot reach any of the T2's, and T1 cannot buffer packets in the mean time, T1 MUST NOT accept traffic from any S2.

9.3 E-mail intercepts

E-mail intercepts are partially intercepted TCP/IP streams. A partially intercepted TCP/IP stream will be handed over via HI3 in a special packet format, while the accompanying IRI will be transported over HI2 in messages specific to the application used.

The LEA requires the complete E-mail message (from the SMTP DATA statement up to and including the SMTP message closing sequence `<cr/lf> . <cr/lf>`) as well as the associated signalling information (RCPT TO, MAIL FROM, sequence number, IP addresses). All E-mail to the target must be delivered in this way. The number of RCPT TO addresses MAY be limited to a bilaterally agreed number.

The E-mail message can be delivered to the LEMF via the TIIT in two ways:

- As the (partial) TCP/IP stream containing the SMTP connection. This will typically be used if the E-mail is intercepted by a sniffer-like device.
- As the E-mail message itself. This will typically be the case if the interception function is part of the mail server software.

In either case the message, or the TCP/IP stream, **MUST** be delivered via the HI3 interface, while the associated signalling information **MUST** be delivered via HI2.

9.4 Generic packet formats and attribute values

```

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

```

Program 4: Description of a PDU2 packet

Attribute	Description	Default Value
versionMajor	Packet type	See Table 2
versionMinor	Subtype/Encryption Algorithm	See Table 3 & 4
length	Length of the total PDU2 packet	None
targetid	Unique target identification	None
sequenceNumber	Assigned by S1	None
encryptedPDU	DPDU encrypted with the algorithm denoted by the version minor number	None

Table 1: PDU2 description

Packet Type	VersionMajor Value
TIIT-(Keep)Alive	0x0
Data	0x1

Table 2: Packet types in versionMajor field

TIIT-KeepAlive subtype	VersionMinor Value
TIIT-KeepAlive	0xf
TIIT-Alive	0xe

Table 3: KeepAlive subtypes in versionMinor field

Encryption Algorithm	VersionMinor Value
RC4	0x0
Mars	0x1
RC6	0x2
Rijndael	0x3
Serpent	0x4
Twofish	0x5
unencrypted DPDU (for testing & debugging only)	0xf

Table 4: Algorithm values in versionMinor field

NOTE 2: Since Rijndael has become the algorithm of choice for the AES, this will be the algorithm used. For compatibility reasons also RC4 will need to be supported. The other values are listed for historical reasons and will not be used in this version of the protocol.

```

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

```

Program 5: Description of a DPDU packet

Attribute	Description	Default Value
providerID	Unique provider identification. Assigned by the Ministry of Economic Affairs.	None
direction	Direction of the payload	Table 6
payLoadID	Identification of the payload	Table 7
timestamp	See definition of timestamp	None
payload	defined by payloadID	None
padding	Extra bytes to create an even 16-byte DPDU	0x00

Table 5: DPDU description

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

Table 6: Payload Direction

Attribute	Description	Program	Included in SHA-1 hash
0x0000	reserved		
0x0001	HI3: IPv4 packet in IPPDU	12	Yes
0x0003	HI3: IPv6 packet in IPPDU	12	Yes
0x0006	HI3: Ethernet packet in IPPDU (future use)	12	Yes
0x0100	HI3: Email packet in EmailPDU	13	Yes
0x0201	HI3: Application specific partial IPv4 stream in PIPDDU	14	Yes
0x0203	HI3: Application specific partial IPv6 stream in PIPDDU	14	Yes
0x1000	SHA-1 Packet	15	No
0x2000	Fake: generated packet		No
0x3000	HI2: Start Session	7	Optional
0x3001	HI2: End Session	8	Optional
0x3002	HI2: SuccessfulDialupLoginIPv4	10	Yes
0x3003	HI2: SuccessfulDialupLoginIPv6	10	Yes
0x3004	HI2: FailedLogin	10	Yes
0x3005	HI2: SuccessfulIPv4DHCPRegistration	10	Yes
0x3006	HI2: Logout	10	Yes
0x3007	HI2: InterceptedEmailIPv4	11	Yes
0x3008	HI2: InterceptedEmailIPv6	11	Yes
0x3100	HI2: GenericLogFileLines	10	Yes
0x3F00	HI2: Generic ISP – LEA message (future use)		
0x3FFC	HI2: S1 malfunction (future use)		
0x3FFD	HI2: Attempt to connect TLS to S2 (future use)		
0x3FFF	HI2: Undefined Error (future use)		

Table 7: Payload Identification

10 Handover Interface 1

HI1 is used to communicate administrative information, such as issuing of warrants, extensions and terminations. The following items are negotiated or sent through the HI1:

1) IP addresses of up to five (5) T1 points and their corresponding X.509 certificates.

NOTE 3: More than one T1's may be used in order to make it harder for outsiders to analyse traffic. However the number will be kept small to limit the administrative burden on provider staff.

2) Target identification as defined in [2].

3) A 192bit key belonging to a target. If RC4 is used to encrypt the DPDU packets, the lower 128 bits are used.

EXAMPLE 1: DPDU key is : b3d689afb42a452214894abb0f7fc8f5bbea57c25ac2aef
For RC4 use : 14894abb0f7fc8f5bbea57c25ac2aef

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

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

11 Handover Interface 2

HI2 is concerned with the transport of information about the communications by the target of interception. This information will consist of login and logout events, log entries recording mailbox access and so on.

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.

All messages on the HI2 are encapsulated in DPDU packets, described in Program 5. All HI2 messages are encapsulated in HI2_Message structures (see Program 6).

```
struct {
    opaque    len[2];
    opaque    message<len>;
} HI2_Message;
```

Program 6: HI2: general payload type

11.1 HI2 — Session establishment

HI2 Start Session message SHOULD be sent once for each warrant, when the warrant has been activated.

However, because the channel from S2 to T1 may be unreliable and therefore the HI2 Start Session Message can get lost without any means of detecting this loss, T1 and T2 SHALL NOT rely on this message to initiate HI2/HI3 message forwarding, storage, or analysis. The Start Session message is purely informational and MAY be used for packet correlation purpose.

This message MUST have `payloadID` as defined in Table 7. The `targetID` corresponds to the warrant that has become active. Program 7 defines the message structure. Timestamp in the `sessionIdentifier` corresponds to the message generation time on the originating S1 or S2.

```
struct {
    ProviderID    providerIdentifier;
    Timestamp     begin_of_session;
} SessionID;

struct {
    SessionID     sessionIdentifier;
} HI2_init;
```

Program 7: HI2_init code

11.2 HI2 — Session termination

HI2 End Session SHOULD be sent once for each warrant, when the warrant has been deactivated (expired).

This message MUST have `payloadID` as defined in Table 7. The `targetID` corresponds to the warrant that has become inactive. Program 8 defines the message structure. The `sessionIdentifier` SHALL match the one contained in the last generated Start Session message. When the `SessionID` is unavailable, a `SessionID` SHALL be generated with the `begin_of_session` filled with zeroes. `End_of_session` timestamp corresponds to the message generation time on the originating S1 or S2.

The TLS/SSLv3 tunnel MAY be torn down after the HI2 End Session for the last warrant on this particular message. However tunnel setup and teardown SHOULD NOT rely solely on these packets since they may get lost in transfer. Other (manual) means of tunnel management MUST be available.

```
struct {
    SessionID      sessionIdentifier;
    Timestamp      end_of_session;
} HI2_end;
```

Program 8: HI2_end code

11.3 IRI Data messages

The structure in the payload attribute of the DPDU packet is defined by the `payloadID` field, see Table 7. The message itself is defined using the type definitions listed in Program 9 and the message structures in Program 10. A description of the fields used in the message structures is given in Table 8.

```
typedef string telephoneNumber[20];
    // Telephone number in Ascii, padded with 0x20
    // ITU supports 3 different formats. Either one may be used

typedef unsigned int IPv4Number;

typedef unsigned int IPv6Number[4];

typedef opaque MACAddress[6];
```

Program 9: HI2 types

```

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

struct {
    telephoneNumber dialledNumberFromPOP;
    telephoneNumber cLIFromTarget;
    string location[6];
    string popID[8];
    IPv6Number assignedAddress;
} SuccessfulDialupLoginIPv6;

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

struct {
    telephoneNumber dialledNumberFromPOP;
    telephoneNumber cLIFromTarget;
    string location[6];
    string popID[8];
} FailedLogin;

struct {
    string location[6];
    string popID[8];
} Logout;

struct {
    opaque logLines<>;
} GenericLogFileLines;

```

Program 10: IRI Data messages

Value	Description
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 Coordinate system to be determined.
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
logLines	One or more concatenated, null terminated, strings representing log file lines as produced by <code>syslogd</code>

Table 8: Message attribute description

11.4 IRI E-mail messages

When E-mail messages are transported through HI3, the corresponding HI2 information must be encapsulated in a HI2:InterceptedEmailIPv4 or HI2:InterceptedEmailIPv6 message. These structures are defined in Program 11. The fields used in these structures are described in Table 9.

The payload identifications for these HI2 messages are defined in Table 7.

```

struct {
    unsigned int    msgIdentifier;
    IPv4Number     source;
    IPv4Number     destination;
    unsigned int   numberOfRecipients;
    opaque         mailAddresses<>
} InterceptedEmailIPv4;

struct {
    unsigned int    msgIdentifier;
    IPv6Number     source;
    IPv6Number     destination;
    unsigned int   numberOfRecipients;
    opaque         mailAddresses<>
} InterceptedEmailIPv6;

```

Program 11: IRI E-mail messages

Value	Description
msgIdentifier	corresponds to the MsgIdentifier in the EmailPDU or PIPPDU, in order to link the HI2 information with the HI3 messages.
source	IP address of the machine that initiated the intercepted SMTP session.
destination	IP address at which the intercepted SMTP session was targeted.
numberOfRecipients	The actual number of recipients included in the mailAddresses field. This MAY be limited to a bilateral agreed number.
mailAddresses	a series of null terminated mail addresses, the first being the originator, the rest being recipients.

Table 9: Message attribute description for HI2 E-mail messages

If the list of recipient addresses becomes too large to fit in a single HI2 message, multiple InterceptedEmail message MUST be sent. The originator's Email address MUST be used repeatedly as the first address in the list of mailAddresses in each packet.

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

This describes the message sequence to get the system running.

- 1) S1 SHOULD establish a connection on a secure channel to S2 when it is put in operational mode. S1 SHOULD authenticate to S2.
- 2) T1 establishes a connection to T2
- 3) S2 establishes a secure tunnel to every T1 known to it. See [3, 10] for the correct message flows.
- 4) A secure tunnel MUST be initiated by functional entity S2. T1 and T2 MAY NEVER initiate a session to a S1 or S2.
- 5) A HI2_init packet (Program 7) is sent from S1 to T2.

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

12.2 Operational message flows

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

IPv4 packet: An intercepted IPv4 packet will be encapsulated in a IPPDU packet (See Program 12). This packet is stored in a DPDU packet with `payloadID` set according to table 7.

IPv6 packet: An intercepted IPv6 packet will be encapsulated in a IPPDU packet (See Program 12). This packet is stored in a DPDU packet with `payloadID` set according to table 7.

Email packet: When an intercepted E-mail message is handed over as a complete message, it will be fitted in one or more EmailPDU messages (See Program 13 and Table 10). These packets are stored in DPDU packets with `payloadID` set according to table 7. To signal the end of an E-mail message a EmailPDU message with zero message length is sent after the EmailPDU's containing data.

Application specific partial IPv4 stream: When not all traffic to a certain target, but only a specific application is intercepted, the intercepted data will be encapsulated in one ore more PIPPDU packets (See Program 14). These packets are stored in DPDU packets with `payloadID` set according to table 7.

Application specific partial IPv6 stream: When not all traffic to a certain target, but only a specific application is intercepted, the intercepted data will be encapsulated in one ore more PIPPDU packets (See Program 14). These packets are stored in DPDU packets with `payloadID` set according to table 7.

SHA-1 packet: Every 64 packets intercepted generate an SHA-1 description packet described in Program 15 and Table 11. A SHA-1 description packet is encapsulated in a DPDU packet.

Traffic Shaping: The output rate of packets at S2 MAY be traffic shaped to alleviate worst case bandwidth behaviour and make traffic analysis more difficult. Care must be taken that a packet SHALL be sent within 30 seconds after being intercepted.

Fake traffic: S2 MAY output Fake Packets to make traffic analysis more difficult. These packets have their `payloadID` set according to table 7. `TargetID` SHOULD be all zeroes.

The DPDU content PDUs and the DPDU's are encrypted, and the encrypted content is inserted in a PDU2, as described in Program 4. PDU2 packets are inserted in the TLS/SSLv3 tunnel from S2 to T1. Which tunnel is used to transport the PDU2 packet SHOULD be decided randomly among all available tunnels. Available capacity on the tunnels MUST be taken into account.

```

struct {
    opaque len[2];
    opaque payload<len>;
} IPPDU;

```

Program 12: IPPDU code

```

struct {
    opaque len[2];
    unsigned int msgIdentifier
    unsigned int msgBlockSequence
    opaque message<len>;
} EmailPDU;

```

Program 13: EmailPDU code

Attribute	Description
Len	Number of bytes in the message field
msgIdentifier	A (pseudo-)randomly chosen number to uniquely identify an E-mail message
msgBlockSequence	Sequence number used to put the parts of an E-mail message in the right order
message	(part of) the actual E-mail message

Table 10: Email PDU attributes

EXAMPLE 2: an E-mail message with size 128000 bytes is intercepted. It will be split up in two EmailPDU's of size 65535 and 62465 bytes respectively. A msgIdentifier is randomly chosen (say 4321). This msgIdentifier is used in the EmailPDU's. The following three EmailPDUs are created:

```

65535 4321 0 (first 65535 bytes of the message)
62465 4321 1 (last 62465bytes of message)
0      4321 2 (0 bytes end of message)

```

```

struct {
    opaque len[2];
    unsigned int msgIdentifier
    opaque message<len>;
} PIPDDU;

```

Program 14: PIPDDU code

```

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

```

Program 15: SHA-1 information PDU

Attribute	Description	Default Value
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-1 hash was computed	
SHAAhash	The computed SHA-1 hash	

Table 11: SHA-1 information PDU attributes

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 authorised personnel **MUST** be able to access the keys or operate the equipment using the keys. See also [2].

13.1 Cryptographic key representation

Keys for asymmetric algorithms — as used for authentication of the TLS/SSLv3 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. See also chapter 10.

13.2 PDU encryption

All PDUs as defined in Program 5 **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 zeroes (0x00). Use of the AES for encryption **MUST** be signalled by a minor version number as specified in table 4.
- 2) RC4 with a key length of 128 bits. The cipher should be initialised for every packet sent. Use of RC4 with a 128 bit key **MUST** be signalled by the use of a minor version number of as specified in table 4.

To be compatible with existing implementations, the second option **MUST** be implemented. Support for Rijndael AES **MUST** be possible in all implementations supporting the TIIT protocol as specified in this document.

13.3 TLS Tunnel specifications

The path from S2 to T1 is an encrypted tunnel, based on TLS/SSLv3 [3]. 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 Rijndael AES at a medium key size (192 bits).
- 2) RSA 2048 with RC4 with a 128 bit key.

Annex A (informative): HI1 XML example

The appendix sketches an example of how a warrant can be implemented in digital format.

```
<!ELEMENT warrant (signeddata, signature)>
<!ELEMENT signeddata (target+, lemfm_data)>
<!ELEMENT target (warrantnumber, start, end, targetid, dpdukey?, contact)>
<!ELEMENT warrantnumber (#PCDATA)>
<!ELEMENT start (#PCDATA)>
<!ELEMENT end (#PCDATA)>
<!ELEMENT targetid (#PCDATA)>
<!ELEMENT dpdukey (#PCDATA)>
<!ELEMENT contact (#PCDATA)>
<!ELEMENT lemfm_data (number_of_t1, t1+, root_certificate)>
<!ELEMENT number_of_t1 (#PCDATA)>
<!ELEMENT t1 (hostaddr, contact)>
<!ELEMENT hostaddr (#PCDATA)>
<!ELEMENT root_certificate (#PCDATA)>
<!ELEMENT signature (#PCDATA)>
<!ATTLIST dpdukey type (RC4 | AES) #REQUIRED>
<!ATTLIST hostaddr type (ipv4 | ipv6 | x25) #REQUIRED>
<!ATTLIST root_certificate type (X.509) #REQUIRED
      encoding (BASE64 | Base64 | base64 ) #REQUIRED>
<!ATTLIST signature algorithm (MD5 | SHA-1 | RIPEMD) #REQUIRED>
```

An example of a warrant.xml file that satisfies this DTD:

```
<?xml version="1.0" encoding="iso-8859-15" ?>
<!DOCTYPE warrant SYSTEM "tiitwarrant.dtd">
<warrant>
  <signeddata>
    <target>
      <!-- Unique warrant number (free format). This number
           is the relation between the warrant itself, (the
           paper document) and this digital XML part -->
      <warrantnumber>"Number of the warrant"</warrantnumber>
      <!-- Start and end date/time for this warrant. The
           times are given in local time -->
      <start>"Example: 06-03-2002 10:28 GMT+1"</start>
      <end>"Example: 06-04-2002 10:28 GMT+2"</end>
      <!-- TargetID and DPDU key assigned by the LEMF -->
      <targetid>
        "Example: 00112233445566778899aabbccddeeff"
      </targetid>
      <dpdukey type="RC4">
        <!-- This key is 192 bit (AES Rijndael). In case of
             using RC4, which uses only 128 bits, use the
             RIGHTmost 32 characters. -->
        "Example: e3d2003cba95458f37e1ead98b9a531f1d22b8ff6640cb1"
      </dpdukey>
      <!-- Contact for this particular tap. free format -->
      <contact>"Example: J. Detective 070-1234567"</contact>
    </target>
    <lemfm_data>
      <!-- T1 address(es) -->
      <number_of_t1>3</number_of_t1>
      <!-- For each T1, list IP address and contact info
           for that T1. The contact field is free format -->
      <t1>
        <hostaddr type="ipv4">"Example: 192.168.1.1"</hostaddr>
        <contact>"Example: name name@url.nl"</contact>
      </t1>
      <t1>
        <hostaddr type="ipv4">"Example: 192.168.1.2"</hostaddr>
        <contact>"Example: someone@url.nl"</contact>
      </t1>
      <t1>
        <hostaddr type="ipv4">"Example: 192.168.1.3"</hostaddr>
        <contact>"Example: P. Beheerder 010-7654321"</contact>
      </t1>
    </lemfm_data>
  </signeddata>
  <signature>
    "Example: Signature data"
  </signature>
</warrant>
```

```
</t1>
<!-- root_certificate gives the root certificate of
      the CA in PEM format. In this case Diginotar -->
<root_certificate type="X.509" encoding="Base64">
  "-----BEGIN CERTIFICATE-----
    Base-64 certificate block
  -----END CERTIFICATE-----"
</root_certificate>
</lemf_data>
</signeddata>
<!-- The hash of the block marked signeddata, is calculated with
      the MD5 algorithm and added in the next field.
      The signature is calculated over the text between the <signeddata>
      tags with all whitespace removed except for whitespace within quoted strings
-->
<signature algorithm="MD5">"Example: af2f0d60368a0d0e88766e63803592a9"</signature>
<!-- The signature and the warrantnumber MUST be added to the
      actual warrant (the paper document). -->
</warrant>
```

Annex B (informative): Packet layout examples

This Annex lists several packet layouts to be used as illustration. It is not a complete list of possible packets.

```

0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
|0 0 0 0|1 1 1 1| length (24) |
+-----+-----+-----+-----+
| targetid md5hash (0x00) |
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
| seq.snifferID | seq.seqNum |
+-----+-----+-----+-----+

```

Figure 2: PDU2: Keepalive packet (versionMajor==0x0; versionMinor==0xf; zero length payload)

```

0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
|0 0 0 1|0 0 1 1| length |
+-----+-----+-----+-----+
| targetid md5hash |
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
| seq.snifferID | seq.seqNum |
+-----+-----+-----+-----+
| encryptedDPDU (length as in length field) |
| . |
| . |
| . |
+-----+-----+-----+-----+

```

Figure 3: PDU2: Data encrypted with Rijndael (versionMajor==0x1; versionMinor==0x3)

```

0           1           2           3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
| providerID | 0 1|1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 |
+-----+-----+-----+-----+
| timestamp.seconds |
+-----+-----+-----+-----+
| timestamp.useconds |
+-----+-----+-----+-----+
| message length (10) | providerIdentifier |
+-----+-----+-----+-----+
| begin_of_session.seconds |
+-----+-----+-----+-----+
| begin_of_session.useconds |
+-----+-----+-----+-----+
| 0x00 (padding) |
+-----+-----+-----+-----+
| 0x00 (padding) |
+-----+-----+-----+-----+

```

Figure 4: DPDU: HI2_init (dir==01b; payloadID == 0x3000)


```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
| providerID          | 0 1|1 1 0 0 0 0 0 0 0 0 0 1 1 0|
+-----+-----+-----+-----+
| timestamp.seconds   |
+-----+-----+-----+-----+
| timestamp.useconds  |
+-----+-----+-----+-----+
| message length (14) | location
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
| popID
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
| 0x00 (padding)
+-----+-----+-----+-----+

```

Figure 7: DPDU: HI2_Logout (dir==01b; payloadID == 0x3006)

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
| providerID          | 0 1|0 0 0 0 0 0 0 0 0 0 0 0 0 1|
+-----+-----+-----+-----+
| timestamp.seconds   |
+-----+-----+-----+-----+
| timestamp.useconds  |
+-----+-----+-----+-----+
| len                 | payload (len bytes)
+-----+-----+-----+-----+
| .....
+-----+-----+-----+-----+

```

Figure 8: DPDU: IPv4 packet in IPPDU (dir==01b; payloadID == 0x0001)

```

0          1          2          3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+
| providerID          | 0 1|0 1 0 0 0 0 0 0 0 0 0 0 0 0|
+-----+-----+-----+-----+
| timestamp.seconds   |
+-----+-----+-----+-----+
| timestamp.useconds  |
+-----+-----+-----+-----+
| n
+-----+-----+-----+-----+
| snifferID          | seqNum
+-----+-----+-----+-----+
| snifferID          | seqNum
+-----+-----+-----+-----+
| including the two above, a total of n SequenceNumbers
|
|
|
+-----+-----+-----+-----+
| SHAnhash
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+
|
+-----+-----+-----+-----+

```

Figure 9: DPDU: SHA packet (dir==01b; payloadID == 0x1000)

History

Document history		
0.1.2	October 19, 2000	Working Document
1.0.0	September 2002	Result of the Bake Off working group agreed in OPT-daf